## CONTENTS

### CHAPTER 1

**debug aaa accounting through debug auto-config** 1
- debug aaa accounting through debug auto-config 1
- debug aaa accounting 2
- debug aaa authentication 3
- debug aaa authorization 4
- debug aaa cache filterserver 7
- debug aaa cache group 9
- debug aaa common-criteria 10
- debug aaa dead-criteria transaction 11
- debug aaa per-user 13
- debug aaa pod 14
- debug aaa redundancy 16
- debug aaa sg-server selection 22
- debug aaa test 25
- debug acircuit 27
- debug acircuit checkpoint 30
- debug adjacency 32
- debug adjacency (vasi) 35
- debug alarm-interface 36
- debug alps ascu 38
- debug alps circuit event 42
- debug alps peer 44
- debug alps peer event 46
- debug alps snmp 47
- debug ancp 48
- debug appfw 52
- debug apple arp 54
- debug apple domain 55
debug atm ha-error 130
debug atm ha-events 132
debug atm ha-state 133
debug atm l2transport 134
debug atm lfi 136
debug atm native 138
debug atm nbma 140
debug atm oam cc 141
debug atm oc3 pom 143
debug atm t3e3 149
debug audit 155
debug authentication 157
debug auto-config 159
debug auto-ip-ring 161
debug autoupgrade 163

c h a p t e r  2

debug backhaul-session-manager session through debug channel packets 165

d e b u g  b a c k h a u l - s e s s i o n - m a n a g e r  s e s s i o n  t h r o u g h  d e b u g  c h a n n e l  p a c k e t s  

165
d e b u g  b a c k h a u l - s e s s i o n - m a n a g e r  s e s s i o n  166
d e b u g  b a c k h a u l - s e s s i o n - m a n a g e r  s e t  169
d e b u g  b a c k u p  171
d e b u g  b e r t  172
d e b u g  b f d  173
d e b u g  b g p  i p v 6  d a m p e n i n g  179
d e b u g  b g p  i p v 6  u p d a t e s  182
d e b u g  b g p  l 2 v p n  e v p n  u p d a t e s  185
d e b u g  b g p  l 2 v p n  v p l s  u p d a t e s  187
d e b u g  b g p  n s a p  189
d e b u g  b g p  n s a p  d a m p e n i n g  191
d e b u g  b g p  n s a p  u p d a t e s  194
d e b u g  b g p  v p n v 6  u n i c a s t  196
d e b u g  b r i - i n t e r f a c e  197
d e b u g  b s c  e v e n t  199
d e b u g  b s c  p a c k e t  200
d e b u g  b s t u n  e v e n t s  201
debug bstun packet 203
debug bundle errors 204
debug bundle events 205
debug call-home diagnostic-signature 206
debug call-mgmt 209
debug call fallback detail 212
debug call fallback probe 214
debug call filter detail 216
debug call filter inout 218
debug call rsvp-sync events 221
debug call rsvp-sync func-trace 223
debug call threshold 225
debug call treatment action 226
d debug callback 227
d debug capf-server 228
d debug cas 230
d debug ccaal2 session 233
d debug cce dp named-db urlfilter 235
d debug ccf rf11 session 237
d debug cch323 239
d debug cch323 capacity 252
d debug cch323 h225 255
d debug cch323 h245 258
d debug cch323 preauth 261
d debug cch323 ras 263
d debug cch323 video 266
d debug ccm-manager 268
d debug ccsip all 274
d debug ccsip calls 282
d debug ccsip dhcp 285
d debug ccsip error 289
d debug ccsip events 292
d debug ccsip feature 294
d debug ccsip info 296
d debug ccsip level 299
debug ccsip media 300
debug ccsip messages 302
debug ccsip non-call 307
debug ccsip preauth 308
debug ccsip states 310
debug ccsip transport 312
ddebug ccswvoice vo-debug 315
ddebug ccswvoice vofr-debug 317
ddebug ccswvoice vofr-session 319
ddebug ccswvoice vo-session 321
ddebug cdapi 322
ddebug cdma pdsn a10 ahdlc 326
ddebug cdma pdsn a10 gre 327
ddebug cdma pdsn a10 ppp 329
ddebug cdma pdsn a11 331
ddebug cdma pdsn accounting 334
ddebug cdma pdsn accounting flow 336
ddebug cdma pdsn accounting time-of-day 337
ddebug cdma pdsn cluster 338
ddebug cdma pdsn ipv6 339
ddebug cdma pdsn prepaid 340
ddebug cdma pdsn qos 341
ddebug cdma pdsn resource-manager 342
ddebug cdma pdsn selection 343
ddebug cdma pdsn service-selection 345
ddebug cdma pdsn session 346
ddebug cdp 347
ddebug cdp ip 349
ddebug cef 351
ddebug cell-hwic driver 355
ddebug cell-hwic firmware 357
ddebug cellular messages all 358
ddebug cellular messages async 360
ddebug cellular messages data 362
ddebug cellular messages dm 364
debug cellular messages management 366
debug cellular messages 367
debug cellular messages sms 369
debug cell-hwic virt-con 370
d debug cem ls errors 372
d debug cem ls events 373
d debug ces-conn 374
d debug cfm 375
d debug channel events 377
d debug channel ilan 379
d debug channel love 381
d debug channel packets 382

CHAPTER 3 debug clns esis events through debug dbconn tcp 385
d debug clns esis events through debug dbconn tcp 385
d debug clns esis events 386
d debug clns esis packets 387
d debug clns events 389
d debug clns igrp packets 390
d debug clns packet 392
d debug clns routing 393
d debug cls message 394
d debug cls vdlc 395
d debug cme-xml 397
d debug cns config 398
d debug cns events 400
d debug cns exec 402
d debug cns image 404
d debug cns management 405
d debug cns xml 407
d debug cns xml-parser 409
d debug compress 411
d debug condition 413
d debug condition application voice 417
d debug condition glbp 419
debug condition interface 421
debug condition match-list 425
debug condition standby 427
debug condition voice-port 429
debug condition vrf 431
debug condition xconnect 432
debug configuration lock 435
debug confmodem 437
debug conn 438
debug content-scan 439
debug control-plane 442
debug cops 445
debug cot 448
debug cpp event 451
debug cpp negotiation 453
debug cpp packet 455
debug credentials 457
debug crm 459
debug crypto ace b2b 462
debug crypto ace boot 463
debug crypto ace cfgmon 464
debug crypto ace congestion-mgr 465
debug crypto ace dpd 466
debug crypto ace error 467
debug crypto ace hapi 468
debug crypto ace ikea 469
debug crypto ace invspi 470
debug crypto ace ipd 471
debug crypto ace mace 472
debug crypto ace pcp 473
debug crypto ace polo 474
debug crypto ace propcfg 475
debug crypto ace qos 476
debug crypto ace rcon 477
debug crypto ace redundancy 478
debug crypto ace spi  479
debug crypto ace stats  480
debug crypto ace syslog  481
debug crypto ace tftp  482
debug crypto ace tpn  483
debug crypto ace warning  484
debug crypto condition unmatched  485
d debug crypto ctcp  487
d debug crypto engine  488
d debug crypto engine accelerator logs  490
d debug crypto engine ism-vpn  492
d debug crypto engine ism-vpn ssl  494
d debug crypto engine ism-vpn traffic  496
d debug crypto engine ism-vpn traffic selector  498
d debug crypto error  500
d debug crypto gdoi  502
d debug crypto gdoi condition  506
d debug crypto ha  508
d debug crypto ipv6 ipsec  510
d debug crypto ipv6 packet  512
d debug crypto ikev2  514
d debug crypto ipsec  516
d debug crypto ipsec client ezvpn  519
d debug crypto ipsec ha  522
d debug crypto isakmp  525
d debug crypto isakmp ha  528
d debug crypto key-exchange  530
d debug crypto mib  531
d debug crypto pki messages  533
d debug crypto pki server  535
d debug crypto pki transactions  537
d debug crypto provisioning  539
d debug crypto sesmgmt  541
d debug csm neat  543
d debug csm tgrm  550
debug csm voice 552
debug ct1-client 558
debug ctunnel 559
debug custom-queue 560
debug cwmp 561
debug cws 562
debug dampening 564
debug data-store 566
debug data-store detail 567
debug dbconn all 569
debug dbconn appc 570
debug dbconn config 572
debug dbconn drda 574
debug dbconn event 576
debug dbconn tcp 578

CHAPTER 4

debug decnet adj through debug dss ipx event 581

data-decnet adj through debug dss ipx event 581

data-decnet adj 582

data-decnet connects 584

data-decnet events 586

data-decnet packet 587

data-decnet routing 588

data-device-sensor 589

data-dhcp 591

data-dhcp redundancy 595

data-dialer events 596

data-dialer forwarding 598

data-dialer map 600

data-dialpeer 602

data-diameter 605

data-d1sw 610

data-dmosp doc-to-fax 619

data-dmosp fax-to-doc 621

data-dmvpn 623
debug aaa accounting through debug auto-config

- debug aaa accounting through debug auto-config, page 1
debug aaa accounting

To display information on accountable events as they occur, use the `debug aaa accounting` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug aaa accounting
no debug aaa accounting
```

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

**Command Modes**
Privileged EXEC

**Usage Guidelines**
The information displayed by the `debug aaa accounting` command is independent of the accounting protocol used to transfer the accounting information to a server. Use the `debug tacacs` and `debug radius` protocol-specific commands to get more detailed information about protocol-level issues.

You can also use the `show accounting` command to step through all active sessions and to print all the accounting records for actively accounted functions. The `show accounting` command allows you to display the active "accountable events" on the system. It provides system administrators a quick look at what is happening, and may also be useful for collecting information in the event of a data loss of some kind on the accounting server. The `show accounting` command displays additional data on the internal state of the authentication, authorization, and accounting (AAA) security system if `debug aaa accounting` is turned on as well.

**Examples**
The following is sample output from the `debug aaa accounting` command:

```
Router# debug aaa accounting
16:49:21: AAA/ACCT: EXEC acct start, line 10
16:49:32: AAA/ACCT: Connect start, line 10, glare
16:49:47: AAA/ACCT: Connection acct stop:
task_id=70 service=exec port=10 protocol=telnet address=172.31.3.78 cmd=glare bytes_in=308
bytes_out=76 paks_in=45 paks_out=54 elapsed_time=14
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug aaa authentication</code></td>
<td>Displays information on accountable events as they occur.</td>
</tr>
<tr>
<td><code>debug aaa authorization</code></td>
<td>Displays information on AAA/TACACS+ authorization.</td>
</tr>
<tr>
<td><code>debug radius</code></td>
<td>Displays information associated with the RADIUS.</td>
</tr>
<tr>
<td><code>debug tacacs</code></td>
<td>Displays information associated with the TACACS.</td>
</tr>
</tbody>
</table>
**debug aaa authentication**

To display information on authentication, authorization, and accounting (AAA) TACACS+ authentication, use the **debug aaa authentication** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug aaa authentication**

**no debug aaa authentication**

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.0(1)M</td>
<td>This command was introduced in a release earlier than Cisco IOS Release 15.0(1)M.</td>
</tr>
<tr>
<td>12.2(33)SRC</td>
<td>This command was integrated into a release earlier than Cisco IOS Release 12.2(33)SRC.</td>
</tr>
<tr>
<td>12.2(33)SXI</td>
<td>This command was integrated into a release earlier than Cisco IOS Release 12.2(33)SXI.</td>
</tr>
<tr>
<td>Cisco IOS XE Release 2.1</td>
<td>This command was integrated into a release earlier than Cisco IOS XE Release 2.1.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use the **debug aaa authentication** command to learn the methods of authentication being used.

**Examples**

The following is sample output from the **debug aaa authentication** command. A single EXEC login that uses the "default" method list and the first method, TACACS+, is displayed.

```
Router# debug aaa authentication
Nov 17 03:06:40.805 PST: AAA/BIND(0000000F): Bind i/f
Nov 17 03:06:40.805 PST: AAA/AUTHEN/LOGIN (0000000F): Pick method list 'default'
```
debug aaa authorization

To display information on authentication, authorization, and accounting (AAA) TACACS+ authorization, use the `debug aaa authorization` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

debug aaa authorization

no debug aaa authorization

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

Use this command to learn the methods of authorization being used and the results of these methods.

Examples

The following is sample output from the `debug aaa authorization` command. In this display, an EXEC authorization for user "carrel" is performed. On the first line, the username is authorized. On the second and third lines, the attribute value (AV) pairs are authorized. The debug output displays a line for each AV pair that is authenticated. Next, the display indicates the authorization method used. The final line in the display indicates the status of the authorization process, which, in this case, has failed.

```
Router# debug aaa authorization
2:23:21: AAA/AUTHOR (0): user='carrel'
2:23:21: AAA/AUTHOR (0): send AV service=shell
2:23:21: AAA/AUTHOR (0): send AV cmd*
2:23:21: AAA/AUTHOR {342885561}: Method=TACACS+
2:23:21: AAA/AUTHOR/TAC+ {342885561}: user=carrel
2:23:21: AAA/AUTHOR/TAC+ {342885561}: send AV service=shell
2:23:21: AAA/AUTHOR/TAC+ {342885561}: send AV cmd*
2:23:21: AAA/AUTHOR {342885561}: Post authorization status = FAIL
```

The `aaa authorization` command causes a request packet containing a series of AV pairs to be sent to the TACACS+ daemon as part of the authorization process. The daemon responds in one of the following three ways:

- Accepts the request as is
- Makes changes to the request
- Refuses the request, thereby refusing authorization

The table below describes AV pairs associated with the `debug aaa authorization` command that may appear in the debug output.

**Table 1: Attribute Value Pairs for Authorization**

<table>
<thead>
<tr>
<th>Attribute Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service=arap</td>
<td>Authorization for the AppleTalk remote access (ARA) protocol is being requested.</td>
</tr>
<tr>
<td>Attribute Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>service=shell</td>
<td>Authorization for EXEC startup and command authorization is being requested.</td>
</tr>
<tr>
<td>service=ppp</td>
<td>Authorization for PPP is being requested.</td>
</tr>
<tr>
<td>service=slip</td>
<td>Authorization for SLIP is being requested.</td>
</tr>
<tr>
<td>protocol=lcp</td>
<td>Authorization for LCP is being requested (lower layer of PPP).</td>
</tr>
<tr>
<td>protocol=ip</td>
<td>Used with service=slip to indicate which protocol layer is being authorized.</td>
</tr>
<tr>
<td>protocol=ipx</td>
<td>Used with service=ppp to indicate which protocol layer is being authorized.</td>
</tr>
<tr>
<td>protocol=atalk</td>
<td>Used with service=ppp or service=arap to indicate which protocol layer is being authorized.</td>
</tr>
<tr>
<td>protocol=vines</td>
<td>Used with service=ppp for VINES over PPP.</td>
</tr>
<tr>
<td>protocol=unknown</td>
<td>Used for undefined or unsupported conditions.</td>
</tr>
<tr>
<td>cmd=x</td>
<td>Used with service=shell, if cmd=NULL, this is an authorization request to start an EXEC. If cmd is not NULL, this is a command authorization request and will contain the name of the command being authorized. For example, cmd=telnet.</td>
</tr>
<tr>
<td>cmd-arg=x</td>
<td>Used with service=shell. When performing command authorization, the name of the command is given by a cmd=x pair for each argument listed. For example, cmd-arg=archie.sura.net.</td>
</tr>
<tr>
<td>acl=x</td>
<td>Used with service=shell and service=arap. For ARA, this pair contains an access list number. For service=shell, this pair contains an access class number. For example, acl=2.</td>
</tr>
<tr>
<td>inacl=x</td>
<td>Used with service=ppp and protocol=ip. Contains an IP input access list for SLIP or PPP/IP. For example, inacl=2.</td>
</tr>
<tr>
<td>outacl=x</td>
<td>Used with service=ppp and protocol=ip. Contains an IP output access list for SLIP or PPP/IP. For example, outacl=4.</td>
</tr>
<tr>
<td>Attribute Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>addr=x</td>
<td>Used with service=slip, service=ppp, and protocol=ip. Contains the IP address that the remote host should use when connecting via SLIP or PPP/IP. For example, addr=172.30.23.11.</td>
</tr>
<tr>
<td>routing=x</td>
<td>Used with service=slip, service=ppp, and protocol=ip. Equivalent in function to the /routing flag in SLIP and PPP commands. Can either be true or false. For example, routing=true.</td>
</tr>
<tr>
<td>timeout=x</td>
<td>Used with service=arap. The number of minutes before an ARA session disconnects. For example, timeout=60.</td>
</tr>
<tr>
<td>autocmd=x</td>
<td>Used with service=shell and cmd=NULL. Specifies an autocmd to be executed at EXEC startup. For example, autocmd=telnetyxz.com.</td>
</tr>
<tr>
<td>noescape=x</td>
<td>Used with service=shell and cmd=NULL. Specifies a noescape option to the username configuration command. Can be either true or false. For example, noescape=true.</td>
</tr>
<tr>
<td>nohangup=x</td>
<td>Used with service=shell and cmd=NULL. Specifies a nohangup option to the username configuration command. Can be either true or false. For example, nohangup=false.</td>
</tr>
<tr>
<td>priv-lvl=x</td>
<td>Used with service=shell and cmd=NULL. Specifies the current privilege level for command authorization as a number from 0 to 15. For example, priv-lvl=15.</td>
</tr>
<tr>
<td>zonelist=x</td>
<td>Used with service=arap. Specifies an AppleTalk zonelist for ARA. For example, zonelist=5.</td>
</tr>
<tr>
<td>addr-pool=x</td>
<td>Used with service=ppp and protocol=ip. Specifies the name of a local pool from which to get the address of the remote host.</td>
</tr>
</tbody>
</table>
**debug aaa cache filterserver**

To help troubleshoot your filter cache configurations, use the `debug aaa cache filterserver` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

`debug aaa cache filterserver`

`no debug aaa cache filterserver`

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(13)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(28)SB.</td>
</tr>
<tr>
<td>12.2(33)SRC</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRC.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug aaa cache filterserver` command:

```
Router# debug aaa cache filterserver
AAA/FLTSV: need "myfilter" (fetch), call 0x612DAC64
AAA/FLTSV: send req, call 0x612DAC50
AAA/FLTSV: method SERVER_GROUP myradius
AAA/FLTSV: recv reply, call 0x612DAC50 (PASS)
AAA/FLTSV: create cache
AAA/FLTSV: add attr "call-inacl"
AAA/FLTSV: add attr "call-inacl"
AAA/FLTSV: add attr "call-inacl"
AAA/FLTSV: skip attr "filter-cache-refresh"
AAA/FLTSV: skip attr "filter-cache-time"
AAA/CACHE: set "AAA filtserv cache" entry "myfilter" refresh? no
AAA/CACHE: set "AAA filtserv cache" entry "myfilter" cachetime 15
AAA/FLTSV: add attr to list "call-inacl" call 0x612DAC64
AAA/FLTSV: add attr to list "call-inacl" call 0x612DAC64
AAA/FLTSV: add attr to list "call-inacl" call 0x612DAC64
AAA/FLTSV: PASS call 0x612DAC64
AAA/CACHE: timer "AAA filtserv cache", next in 10 secs (0 entries)
AAA/CACHE: timer "AAA filtserv cache", next in 10 secs (1 entry)
AAA/CACHE: destroy "AAA filtserv cache" entry "myfilter"
AAA/CACHE: timer "AAA filtserv cache", next in 10 secs (0 entries)
```
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>aaa authorization cache filterserver</code></td>
<td>Enables AAA authorization caches and the downloading of ACL configurations from a RADIUS filter server.</td>
</tr>
</tbody>
</table>
debug aaa cache group

To debug the caching mechanism and ensure that entries are cached from authentication, authorization, and accounting (AAA) server responses and found when queried, use the `debug aaa cache group` command in privileged EXEC mode.

**debug aaa cache group**

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

**Command Default**
Debug information for all cached entries is displayed.

**Command Modes**
Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(28)SB</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRC</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRC.</td>
</tr>
<tr>
<td>15.0(1)M</td>
<td>This command was integrated into Cisco IOS Release 15.0(1)M.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
Use this command to display debug information about cached entries.

**Examples**
The following example displays the debug information about all cached entries:

```
Router# debug aaa cache group
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear aaa cache group</td>
<td>Clears an individual entry or all entries in the cache.</td>
</tr>
<tr>
<td>show aaa cache group</td>
<td>Displays cache entries stored by the AAA cache.</td>
</tr>
</tbody>
</table>
debug aaa common-criteria

To troubleshoot authentication, authorization, and accounting (AAA) common criteria password security policy information, use the debug aaa common-criteria command in privileged EXEC mode. To disable debugging output, use the no form of this command.

debug aaa common-criteria
no debug aaa common-criteria

Syntax Description
This command has no arguments or keywords.

Command Default
Conditional debugging for this command is disabled.

Command Modes
Privileged EXEC (#)

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.0(2)SE</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

Usage Guidelines
Use this command to display debug information about AAA common criteria policies.

Examples
The following example displays the debug information about AAA common criteria:

Device> enable
Device# debug aaa common-criteria
AAA common-criteria debugs debugging is on
*Aug 6 08:21:06.554: AAA CC: Increment ref count to 1 for test
*Aug 6 08:21:06.554: AAA CC: User test linked to CC policy test

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaa common-criteria policy</td>
<td>Configures a AAA common criteria security policy.</td>
</tr>
<tr>
<td>show aaa common-criteria policy</td>
<td>Displays common criteria security policy details.</td>
</tr>
</tbody>
</table>
**debug aaa dead-criteria transaction**

To display authentication, authorization, and accounting (AAA) dead-criteria transaction values, use the `debug aaa dead-criteria transaction` command in privileged EXEC mode. To disable dead-criteria debugging, use the `no` form of this command.

```
debug aaa dead-criteria transaction
no debug aaa dead-criteria transaction
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

If the command is not configured, debugging is not turned on.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(6)</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(7)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(7)T. The command output includes two new fields: Current Tries and Elapsed Time.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Dead-criteria transaction values may change with every AAA transaction. Some of the values that can be displayed are estimated outstanding transaction, retransmit tries, and dead-detect intervals. These values are explained in the table below.

**Examples**

The following example shows dead-criteria transaction information for a particular server group:

```
Router# debug aaa dead-criteria transaction
AAA Transaction debugging is on
* Nov 14 23:44:17.403: AAA/SG/TRANSAC: Computed Dead Detect Interval: 10s, Elapsed Time: 317s, Current Max Interval: 10s
```

The table below describes the significant fields shown in the display.

**Table 2: debug aaa dead-criteria transaction Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA/SG/TRANSAC</td>
<td>AAA server-group transaction.</td>
</tr>
</tbody>
</table>
### Field | Description
---|---
Computed Retransmit Tries | Currently computed number of retransmissions before the server is marked as dead.
Current Tries | Number of successive failures since the last valid response.
Current Max Tries | Maximum number of tries since the last successful transaction.
Computed Dead Detect Interval | Period of inactivity (the number of seconds since the last successful transaction) that can elapse before the server is marked as dead. The period of inactivity starts when a transaction is sent to a server that is considered live. The dead-detect interval is the period that the router waits for responses from the server before the router marks the server as dead.
Elapsed Time | Amount of time that has elapsed since the last valid response.
Current Max Interval | Maximum period of inactivity since the last successful transaction.
Estimated Outstanding Transaction | Estimated number of transaction that are associated with the server.
Current Max Transaction | Maximum transaction since the last successful transaction.

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>radius-server dead-criteria</td>
<td>Forces one or both of the criteria--used to mark a RADIUS server as dead--to be the indicated constant.</td>
</tr>
<tr>
<td>show aaa dead-criteria</td>
<td>Displays dead-criteria detection information for an AAA server.</td>
</tr>
</tbody>
</table>
**debug aaa per-user**

To display debugging information about PPP session per-user activities, use the `debug aaa per-user` command in privileged EXEC mode. To turn off debugging, use the `no` form of this command.

`debug aaa per-user`
```
no debug aaa per-user
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3 T</td>
<td>This command has existed since Cisco IOS Release 11.3 T.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The per-user module is responsible for installing per-user attributes for PPP sessions.

**Examples**

The following example displays the configuration commands that were generated by the per-user process:

```
Router# debug aaa per-user
AAA/PER-USER: line=[ip access-list standard Virtual-Access2#31]
AAA/PER-USER: line=[deny 10.0.0.2 0.0.0.0]
AAA/PER-USER: line=[permit any]
```

The fields in the display are self-explanatory.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug aaa authorization</td>
<td>Displays information on AAA TACACS+ authorization.</td>
</tr>
<tr>
<td>debug ppp</td>
<td>Displays information on traffic and exchanges in an internetwork implementing the PPP.</td>
</tr>
<tr>
<td>debug radius</td>
<td>Displays information associated with RADIUS.</td>
</tr>
<tr>
<td>debug tacacs</td>
<td>Displays information associated with TACACS.</td>
</tr>
</tbody>
</table>
debug aaa pod

To display debug messages related to packet of disconnect (POD) packets, use the **debug aaa pod** command in privileged EXEC mode. To disable debugging output, use the no form of this command.

```
debug aaa pod
no debug aaa pod
```

**Syntax Description**

This command has no keywords or arguments.

**Command Default**

Debugging for POD packets is not enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)XB</td>
<td>Support for the voice applications as well as support for the Cisco AS5350, Cisco AS5400 and the Cisco 3600 series was added.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>Support for the Cisco AS5800 was added.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>Support for the Cisco AS5850 was added. This command was integrated into Cisco IOS Release 12.2(11)T.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Examples**

The following example shows output from a successful POD request when using the **show debug** command:

```
Router# debug aaa pod
AAA POD packet processing debugging is on
Router# show debug
General OS:
    AAA POD packet processing debugging is on
Router#
Apr 25 17:15:59.318:POD:172.19.139.206 request queued
Apr 25 17:15:59.318:voice_pod_request:
Apr 25 17:15:59.318:voip_populate_pod_attr_list:
Apr 25 17:15:59.318:voip_pod_get_guid:
Apr 25 17:15:59.318:voip_pod_get_vsa_attr_val:
Apr 25 17:15:59.318:voip_pod_get_vsa_attr_val:attr_len=50
Apr 25 17:15:59.318:voip_pod_get_vsa_attr_val:attr=h323-conf-id
Apr 25 17:15:59.318:voip_pod_get_vsa_attr_val:attr_len=50 value_len=35
Apr 25 17:15:59.318:voip_pod_get_guid:conf-id=FFA7785F F7F607BB 00000000 993FB1F4 n_bytes=35
Apr 25 17:15:59.318:voip_pod_get_guid:GUID = FFA7785F F7F607BB 00000000 993FB1F4
Apr 25 17:15:59.318:voip_populate_pod_attr_list:
```
Apr 25 17:15:59.318:voip_pod_get_vsa_attr_val:
Apr 25 17:15:59.318:voip_pod_get_vsa_attr_val:attr_len=23
Apr 25 17:15:59.318:voip_pod_get_vsa_attr_val:attr=h323-originate
Apr 25 17:15:59.318:voip_pod_get_vsa_attr_val:attr_len=23 value_len=6
Apr 25 17:15:59.318:voip_get_call_direction:
Apr 25 17:15:59.318:voip_get_call_direction:returning answer
Apr 25 17:15:59.318:voip_eval_pod_attr:
Apr 25 17:15:59.318:cc_api_trigger_disconnect:
Apr 25 17:15:59.322:POD: Sending ACK to 172.19.139.206/1700
Apr 25 17:15:59.322:voip_pod_clean:

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaa pod server</td>
<td>Enables the POD feature.</td>
</tr>
</tbody>
</table>
debug aaa redundancy

To display debug output that displays authentication, authorization, and accounting (AAA) redundancy events during session activation, session synchronization to the standby device, and dynamic session updates to the standby device, use the **debug aaa redundancy** command in privileged EXEC mode. To disable debugging for AAA redundancy, use the **no** form of this command.

**debug aaa redundancy**

**no debug aaa redundancy**

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

**Command Modes**
Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(31)SB2</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRC</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRC.</td>
</tr>
<tr>
<td>12.2(33)SRE</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRE.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

This command displays the AAA synchronization data for the session synchronization to the standby device. This information might be useful for diagnosing any AAA problems related to the session synchronization.

**Examples**

The following example shows sample output from the **debug aaa redundancy** command collected while a session is activated and synchronized to the standby device:

```
Router# debug aaa redundancy
Logs on Active
------------------
01:31:55: CCM: PPPoE Required
01:31:55: CCM: PPP Required
01:31:55: CCM: LTERM Required
01:31:55: CCM: PPPoE is Initiator
01:31:55: AAA/BIND(0000000B): Bind i/f Virtual-Template1
01:31:55: CCM: AAA Ready
01:31:55: AAA/CM/(0000000B): AAA sso init completed successfully
01:31:55: SSS INFO: Element type is Access-Type = 3 (PPPoE)
01:31:55: SSS INFO: Element type is Protocol-Type = 0 (PPP)
01:31:55: SSS INFO: Element type is Media-Type = 1 (Ethernet)
01:31:55: SSS INFO: Element type is Switch-Id = 4105 (00000009)
01:31:55: SSS INFO: Element type is Media-Hdl = 4114 (00000012)
01:31:55: SSS INFO: Element type is AAA-Id = 11 (0000000B)
01:31:55: SSS INFO: Element type is AccIe-Hdl = 33554441 (02000009)
01:31:55: SSS INFO: Element type is SHDB-Handle = 1476395017 (58000009)
01:31:55: SSS INFO: Element type is Input Interface = "GigabitEthernet6/0/0"
01:31:55: SSS MGR [uid:10]: Sending a Session Assert ID Mgr event
```
01:31:55: SSS MGR [uid:10]: Updating ID Mgr with the following keys:
    username "user1"
    aaa-unique-id 11 (0xB)
01:31:55: SSS MGR [uid:10]: Handling Policy Service Authorize action (1 pending sessions)
01:31:55: SSS PM [uid:10] [63D5D594]: RM/VPDN disabled: RM/VPDN author not needed
01:31:55: SSS MGR [uid:10]: Handling Need More Keys action
01:31:55: SSS INFO: Element type is AccIe-Hdl = 33554441 (02000009)
01:31:55: SSS INFO: Element type is Access-Type = 0 (PPP)
01:31:55: SSS MGR [uid:10]: Got reply Need More Keys from PM
01:31:55: SSS INFO: Element type is Resource-Determined = 1 (YES)
01:31:55: SSS MGR [uid:10]: Handling Need More Keys action
01:31:55: SSS MGR [uid:10]: Got reply Local Terminate from PM
01:31:55: SSS MGR [uid:10]: Not yet ready to start the Local service
01:31:57: SSS INFO: Element type is Unauth-User = "user1"
01:31:57: SSS INFO: Element type is Access-Type = 0 (PPP)
01:31:57: SSS MGR [uid:10]: Sending a Session Update ID Mgr event
01:31:57: SSS MGR [uid:10]: Sending a Session Update ID Mgr event
01:31:57: SSS MGR [uid:10]: Sending a Session Update ID Mgr event
debug aaa redundancy

01:31:57: SSS AAA AUTHOR [uid:10]: Active Handle present
01:31:57: SSS AAA AUTHOR [uid:10]: Freeing Active Handle; SSS Policy Context Handle = 63D5D594
01:31:57: SSS AAA AUTHOR [uid:10]: Free request
01:31:57: SSS MGR [uid:10]: Got reply Apply Config from PM
01:31:57: SSS MGR [uid:10]: Successfully applied policy config
01:31:57: SSS LTERM [uid:10]: Processing Local termination request
01:31:57: SSS LTERM [uid:10]: Sent create-clone request to vtemplate manager
01:31:57: SSS MGR [uid:10]: Handling Connect Local Service action
01:31:57: SSS LTERM [uid:10]: Creating vaccess interface Vi3
01:31:57: SSS MGR [uid:10]: Handling Local Service Connected action
01:31:57: SSS MGR [uid:10]: Apply for Vi3: segment 4114, owner 3825205277
01:31:57: SSS MGR [uid:10]: Interface config 212C27B8
01:31:57: SSS MGR [uid:10]: Per-user config 2146BD48
01:31:57: SSS LTERM [uid:10]: Switching session provisioned
01:31:57: SSS MGR [uid:10]: Handling Local Service Connected, Features Applied action
01:31:57: %LINK-3-UPDOWN: Interface Virtual-Access3, changed state to up
01:31:57: SSS LTERM [uid:10]: Installed Vi3 process path switching vector
01:31:57: SSS LTERM [uid:10]: Installed Vi3 fastsend path switching vector
01:31:57: CCM: PPPoE Ready
01:31:57: CCM: PPP Ready
01:31:57: CCM: New State[Ready]
01:31:57: AAA/CCM/(0000000B): No of sync avps = 4 Total sync data len = 94
01:31:57: AAA/CCM/(0000000B): Adding sync avp [AAA Unique Id] Length = 4
01:31:57: CCM: AAA Adding Data Type[4] Subtype[0] Length[84]
01:31:57: AAA/CCM/(0000000B): Adding sync avp [AAA Author Data] Length = 84
01:31:57: AAA/CCM: Adding author data entry 32
01:31:57: SSS LTERM [uid:10]: LTERM segment handle synced
01:31:57: Client[PPP] Type[7] Subtype[0] Length[0]
01:31:57: Client[PPP] Type[1] Subtype[0] Length[54]
01:31:57: Client[PPP] Type[0] Length[2]
01:31:57: Client[PPP] Type[0] Length[2]
01:31:57: Client[PPP] Type[0] Length[2]
01:31:57: Client[PPP] Type[0] Length[2]
01:31:57: Client[PPP] Type[0] Length[2]
01:31:57: Client[PPP] Type[0] Length[2]
01:31:57: Client[PPP] Type[0] Length[2]
01:31:57: Client[PPP] Type[0] Length[2]
01:31:57: Client[PPP] Type[0] Length[2]
01:31:57: Client[PPP] Type[0] Length[2]
01:31:57: Client[PPP] Type[0] Length[2]
01:31:57: Client[PPP] Type[0] Length[2]
01:31:57: Client[PPP] Type[0] Length[2]
01:31:57: 00 00 10 12
01:31:57: 00 02 06 00 00 00 A6 B8 00 00 00 00 00 00 00 2A
01:31:57: 00 00 FF FF
01:31:57: 00 00 03 00 00 00 00 00 00 00 00 19 00 00 00 1D
01:31:57: 00 00 00 0B
01:31:57: 00 00
01:31:57: 00 00 00 0B
01:31:57: 00 00 00 00 00 00 00 00 63 E8 73 D0 00 00 00 0B
01:31:57: 64 02 FE 71 00 00 00 00 00 00 00 00 00 00 00 00
01:31:57: 00 00 00 04 00 00 00 01 00 00 00 20 00 00 00 00
01:31:57: 58 00 00 09 02 0A 00 20 E4 68 43 2C 2F E7 B4 57
01:31:57: 05 70 FF B1 22 13 E8 0F 1C 06 00 00 00 C8 06 06
01:31:57: 00 00 00 02
01:31:57: CCM: New State[Dyn Sync]
01:31:58: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3, changed state to up
Logs on Standby

01:21:16: CCM ISSU: Received negotiation message type [ISSU_RC_USER_MESSAGE_COMP]
01:21:16: 01 04 05 D4 03 04 C0 23 05 06 03 F4 37 79
01:21:16: 01 04 05 D4 05 06 9A 6B 68 FE
01:21:16: 03 06 07 01 01 01
01:21:16: Client[PPP] Type[7] Subtype[0] Length[0]
01:21:16: 73 75 6D 61 6E 74 68 00
01:21:16: 00 00 00 02
01:21:16: Client[PPPoE] Type[1] Subtype[0] Length[54]
01:21:16: 00 03 A0 10 22 90 00 0A 42 7D EC 38 88 63 11 19
01:21:16: 00 00 00 22 01 02 00 06 61 61 61 5F 68 61 01 04
01:21:16: 00 10 98 99 BB 6D 59 B8 35 33 0B FB 14 B9 07 EB
01:21:16: 83 B4 01 01 00 00
01:21:16: 00 0A
01:21:16: 00 00 10 09
01:21:16: 00 00 10 12
01:21:16: 00 02 06 00 00 00 A6 B8 00 00 00 00 00 00 00 2A
01:21:16: 00 00 FF FF
01:21:16: 00 00 00 03 00 00 00 00 00 00 00 19 00 00 00 1D
01:21:16: 00 00 00 0B
01:21:16: 00 00
01:21:16: 00 00 00 0B
01:21:16: 00 00 00 00 00 00 00 00 63 E8 73 D0 00 00 00 0B
01:21:16: 64 02 FE 71 00 00 00 00 00 00 00 00 00 00 00 00
01:21:16: 00 00 00 04 00 00 00 01 00 00 00 20 00 00 00 00
01:21:16: 58 00 00 09 02 0A 00 20 E4 68 43 2C 2F E7 B4 57
01:21:16: 05 70 FF B1 22 13 E8 0F 1C 06 00 00 00 C8 06 06
01:21:16: 00 00 00 02
01:21:16: 00 00 20 13
01:21:16: CCM:PPPoE Recreate Session Active[0x58000009] Standby[0x96000009]
01:21:16: CCM: PPPoE Required
01:21:16: CCM: LTERM Required
01:21:16: CCM: PPPoE is Initiator
01:21:16: AAA/CCM/: return checkpointed aaa id = 0000000B
01:21:16: Adding cache entry for id B
01:21:16: AAA/CCM/(0000000B):return acct_sess_id = 11
01:21:16: CCM: AAA Ready
01:21:16: AAA/CCM/(0000000B): AAA sso init completed successfully
01:21:16: SSS INFO: Element type is Access-Type = 3 (PPPoE)
01:21:16: SSS INFO: Element type is Protocol-Type = 0 (PPP)
01:21:16: SSS INFO: Element type is Media-Type = 1 (Ethernet)
01:21:16: SSS INFO: Element type is Switch-Id = 4105 (00001009)
01:21:16: SSS INFO: Element type is Segment-Hdl = 4114 (00001012)
01:21:16: SSS INFO: Element type is AAA-Id = 11 (0000000B)
01:21:16: SSS INFO: Element type is AccIe-Hdl = 4127195145 (F6000009)
01:21:16: SSS INFO: Element type is SHDB-Handle = 2550136841 (98000009)
01:21:16: SSS INFO: Element type is Input Interface = "GigabitEthernet6/0/0"
01:21:16: SSS MGR [uid:10]: Sending a Session Assert ID Mgr event
01:21:16: SSS MGR [uid:10]: Updating ID Mgr with the following keys:
  
  username "user1"
  aaa-unique-id 11 (0xB)

01:21:16: SSS MGR [uid:10]: Handling Need More Keys action
01:21:16: SSS MGR [uid:10]: Handling Policy Send More Keys action
01:21:16: SSS MGR [uid:10]: Not yet ready to start the Local service
01:21:16: SSS MGR [uid:10]: Not ready to start the Local service
01:21:16: SSS MGR [uid:10]: Got reply Need More Keys from PM
01:21:16: SSS MGR [uid:10]: Handling Policy Service Authorize action (1 pending sessions)
01:21:16: SSS MGR [uid:10]: Got reply More Keys from PM
01:21:16: SSS MGR [uid:10]: Handling Need More Keys action
01:21:16: SSS MGR [uid:10]: Got reply More Keys from PM
01:21:16: SSS MGR [uid:10]: Got reply More Keys from PM
01:21:16: AAA/CCM/(0000000B): return authen_method_index = 0
01:21:16: RADIUS/ENCODE(0000000B):Orig. component type = PPoE
01:21:16: RADIUS(0000000B): Config NAS IP: 0.0.0.0
01:21:16: RADIUS(0000000B): sending
01:21:16: RADIUS/ENCODE: Best Local IP-Address 2.1.1.1 for Radius-Server 9.2.36.253
01:21:16: RADIUS: authenticator 14 48 25 90 A5 7B 53 02 - 11 05 01 13 6D 34 E2 04
01:21:16: RADIUS: User-Name [1] 9 "user1"
01:21:16: RADIUS: NAS-Port-Id [87] 9 "6/0/0/0"
01:21:16: RADIUS: Caching response
01:21:16: RADIUS: authenticator E4 68 43 2C 2F E7 B4 57 - 05 70 FF B1 22 13 E8 0F
01:21:16: RADIUS(0000000B): Received from id 1645/10
01:21:16: SSS INFO: Element type is Auth-User = "user1"
01:21:16: SSS INFO: Element type is AAA-Attr-List = 20000100
01:21:16: SSS INFO: Element type is idletime 200 (0xC8)
The table below describes the significant fields shown in the display.

Table 3: debug aaa redundancy Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0000000B)</td>
<td>AAA unique ID for the session.</td>
</tr>
<tr>
<td>Adding sync avp</td>
<td>Adding synchronization attribute-value pair.</td>
</tr>
<tr>
<td>[AAA Unique ID]</td>
<td>The AAA synchronization data type.</td>
</tr>
</tbody>
</table>

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccm-manager</td>
<td>Displays debugging information about Cisco CallManager.</td>
</tr>
</tbody>
</table>
**debug aaa sg-server selection**

To obtain information about why the RADIUS and TACACS+ server group system in a router is choosing a particular server, use the `debug aaa sg-server selection` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Debugging is not turned on.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(1)</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was extended for RADIUS server load balancing to show which server is selected on the basis of a load balancing algorithm.</td>
</tr>
<tr>
<td>12.4(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(11)T.</td>
</tr>
<tr>
<td>12.2(33)SRC</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRC.</td>
</tr>
</tbody>
</table>

**Examples**

The following example shows that debugging has been set to display information about server selection:

```
Router# debug aaa sg-server selection
The following two debug outputs display the behavior of RADIUS transactions within a server group with the server-reorder-on-failure feature configured.
```

```
In the following sample output, the RADIUS server-reorder-on-failure feature is configured. The server retransmits are set to 0 (so each server is attempted only once before failover to the next configured server), and the transmissions per transaction are set to 4 (the transmissions will stop on the third failover). The third server in the server group (192.0.2.118) has accepted the transaction on the third transmission (second failover).
```

```
00:38:35: %SYS-5-CONFIG-I: Configured from console by console
00:38:53: RADIUS/ENCODE (OOOOOOOF) : ask "Username: ">
00:38:53: RADIUS/ENCODE (0000000F) : send packet; GET-USER
00:38:58: RADIUS/ENCODE (0000000F) : ask "Password: ">
00:38:58: RADIUS/ENCODE (0000000F) : send packet; GET-PASSWORD
00:38:59: RADIUS: AAA Unsupported [152] 4
00:38:59: RADIUS: 7474 [tt]
00:38:59: RADIUS (0000000F) : Storing nasport 2 in rad-db
```
In the following sample output, the RADIUS server-reorder-on-failure feature is configured. The server retransmits are set to 0, and the transmissions per transaction are set to 8. In this transaction, the transmission to server 192.0.2.1 has failed on the eighth transmission.

```
00:42:30: RADIUS(00000011): Received from id 21645/13
00:43:34: RADIUS: Unsupported [152] 4
00:43:34: RADIUS: 7474 [tt]
00:43:34: RADIUS(00000012) : Storing nasport 2 in rad-db
00:43:40: RADIUS: AAA Unsupported [152] 4
00:43:40: RADIUS: 7474 [tt]
00:43:40: RADIUS(00000012) : Storing nasport 2 in rad-db
00:43:40: RADIUS(00000012) : Co-fig NAS IP: 192.0.2.4
00:43:40: RADIUS(00000012) : acct-session-id: 18
00:43:40: RADIUS: NAS-Port-Type [61] 6 Virtual [5]
00:43:40: RADIUS: Calling-Station-Id [31] 15 "192.0.2.23"
00:43:40: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645 id 21645/11
00:43:44: RADIUS: NAS-IP-Address [4] 6 192.0.2.130
00:43:46: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:43:48: RADIUS: Fail-over to (192.0.2.1:1645,1646) for server 192.0.2.1
00:43:50: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:43:52: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:43:54: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:43:56: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:43:58: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:00: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:02: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:04: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:06: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:08: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:10: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:12: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:14: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:16: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:18: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:20: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:22: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:24: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:26: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:28: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:30: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:32: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:34: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:36: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:38: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:40: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:42: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:44: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:46: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:48: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:50: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:52: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:54: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
00:44:56: RADIUS(00000012) : Send Access-Request to 192.0.2.118:1645,1646 for id 21645/11
```

The field descriptions are self-explanatory.
Examples

In the following sample output, the RADIUS server load balancing feature is enabled with a batch size of 3. The server selection, based on the load balancing algorithm, is shown as five access-requests that are being sent to the server group.

```
Router# debug aaa sg-server selection
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Obtaining least loaded server.
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Obtaining least loaded server.
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Obtaining least loaded server.
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Obtaining least loaded server.
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Obtaining a new least loaded server.
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Server[0] load: 3
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Selected Server[1] with load 0
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Obtaining least loaded server.
```

The field descriptions are self-explanatory.

## Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>load-balance</td>
<td>Enables RADIUS server load balancing for named RADIUS server groups.</td>
</tr>
<tr>
<td>radius-server load-balance</td>
<td>Enables RADIUS server load balancing for the global RADIUS server group.</td>
</tr>
<tr>
<td>radius-server retry method reorder</td>
<td>Specifies the reordering of RADIUS traffic retries among a server group.</td>
</tr>
<tr>
<td>radius-server transaction max-tries</td>
<td>Specifies the maximum number of transmissions per transaction that may be retried on a RADIUS server.</td>
</tr>
<tr>
<td>test aaa group</td>
<td>Tests RADIUS load balancing server response manually.</td>
</tr>
</tbody>
</table>
**debug aaa test**

To show when the idle timer or dead timer has expired, when test packets are being sent, server response status, and the server state for RADIUS server load balancing, use the `debug aaa test` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug aaa test
no debug aaa test
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Debugging is not enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(28)SB</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(11)T.</td>
</tr>
<tr>
<td>12.2(33)SRC</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRC.</td>
</tr>
</tbody>
</table>

**Examples**

In the following sample output, the RADIUS server load balancing feature is enabled. The idle timer has expired.

```
Router# debug aaa test
Router#
Jul 16 00:07:01: AAA/SG/TEST: Server (192.0.2.245:1700,1701) quarantined.
Jul 16 00:07:01: AAA/SG/TEST: Sending test request(s) to server (192.0.2.245:1700,1701)
Jul 16 00:07:01: AAA/SG/TEST: Sending 1 Access-Requests, 1 Accounting-Requests in current batch.
Jul 16 00:07:01: AAA/SG/TEST(Req#: 1): Sending test AAA Accounting-Request.
Jul 16 00:07:01: AAA/SG/TEST: Obtained Test response from server (192.0.2.245:1700,1701)
Jul 16 00:07:01: AAA/SG/TEST: Obtained Test response from server (192.0.2.245:1700,1701)
Jul 16 00:07:01: AAA/SG/TEST: Necessary responses received from server (192.0.2.245:1700,1701)
Jul 16 00:07:01: AAA/SG/TEST: Server (192.0.2.245:1700,1701) marked ALIVE. Idle timer set for 60 sec(s).
Jul 16 00:07:01: AAA/SG/TEST: Server (192.0.2.245:1700,1701) removed from quarantine.
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>load-balance</td>
<td>Enables RADIUS server load balancing for named RADIUS server groups.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>radius-server host</code></td>
<td>Enables RADIUS automated testing for load balancing.</td>
</tr>
<tr>
<td><code>radius-server load-balance</code></td>
<td>Enables RADIUS server load balancing for the global RADIUS server group.</td>
</tr>
<tr>
<td><code>test aaa group</code></td>
<td>Tests RADIUS load balancing server response manually.</td>
</tr>
</tbody>
</table>
debug acircuit

To display errors and events that occur on the attachment circuits (the circuits between the provider edge (PE) and customer edge (CE) routers), use the `debug acircuit` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug acircuit {error| event}
no debug acircuit {error| event}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>Displays errors that occur in attachment circuits.</td>
</tr>
<tr>
<td>event</td>
<td>Displays events that occur in attachment circuits.</td>
</tr>
</tbody>
</table>

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(23)S</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(14)S</td>
<td>This command was integrated into Cisco IOS Release 12.2(14)S.</td>
</tr>
<tr>
<td>12.2(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(15)T.</td>
</tr>
<tr>
<td>12.2(25)S</td>
<td>This command was integrated into Cisco IOS Release 12.2(25)S.</td>
</tr>
<tr>
<td>12.2(27)SBC</td>
<td>Support for this command was integrated into Cisco IOS Release 12.2(27)SBC.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

### Usage Guidelines

Use the `debug acircuit` command to identify provisioning events, setup failures, circuit up and down events, and configuration failures on attachment circuits.

An attachment circuit connects a PE router to a CE router. A router can have many attachment circuits. The attachment circuit manager controls all the attachment circuits from one central location. Therefore, when you enable the debug messages for the attachment circuit, you receive information about all the attachment circuits.

### Examples

The following is sample output from the `debug acircuit event` command when you enable an interface:

```
Router# debug acircuit event
*Jan 28 15:19:03.070: ACLIB: ac_cstate() Handling circuit UP for interface Se2/0
```
The following is sample output from the debugacircuitevent command when you disable an interface:

```
Router# debug acircuitevent
*Jan 28 15:25:57.014: ACLIB: SW AC interface UP for interface Se2/0
*Jan 28 15:25:57.014: ACLIB [11.0.1.1, 200]: Setting new AC state to Ac-Idle
*Jan 28 15:25:57.014: Se2/0 ACMGR: Receive <Circuit Down> msg
*Jan 28 15:25:59.014: %LINK-5-CHANGED: Interface Serial2/0, changed state to administratively down
*Jan 28 15:25:59.014: ACLIB: ac_cstate() Handling circuit DOWN for interface Se2/0
*Jan 28 15:26:00.014:%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed state to down
```

The following example shows output from the debugacircuitcommand for an xconnect session on an Ethernet interface:

```
Router# debug acircuit
23:28:35: ACLIB [10.0.3.201, 5]: SW AC interface UP for Ethernet interface Et2/1
23:28:35: Et2/1 ACMGR: circuit up event, SIP state chg down to connecting, action is service request
23:28:37: %LINK-3-UPDOWN: Interface Ethernet2/1, changed state to up
23:28:38: %LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet2/1, changed state to up
23:28:53: Et2/1 ACMGR: Rcv SIP msg: resp connect forwarded, hdl D6000002, sss_hdl 9E00000F
23:28:53: Et2/1 ACMGR: service connected event, SIP state chg connecting to connected, action is respond forwarded
23:28:53: ACLIB: pthru_intf_response hdl is D6000002, response is 1
23:28:53: ACLIB [10.0.3.201, 5]: Setting new AC state to Ac-Connected
```

The command output is self-explanatory.

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug vpdn</td>
<td>Displays errors and events relating to L2TP configuration and the surrounding Layer 2 tunneling infrastructure.</td>
</tr>
<tr>
<td>debug xconnect</td>
<td>Displays errors and events related to an xconnect configuration.</td>
</tr>
</tbody>
</table>
debug aaa accounting through debug auto-config

d debug acircuit
debug acircuit checkpoint

To enable the display of attachment circuit checkpoints, use the debug acircuit checkpoint command in privileged EXEC mode. To disable the display of these messages, use the no form of this command.

```
debug acircuit checkpoint
no debug acircuit checkpoint
```

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

**Command Default**
Debugging of attachment circuit checkpoints is disabled.

**Command Modes**
Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(25)S</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(28)SB.</td>
</tr>
<tr>
<td>12.2(33)SRC</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRC.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
Use this command when Any Transport over MPLS (AToM) is configured for nonstop forwarding/stateful switchover (NSF/SSO) and Graceful Restart.

Use debug commands with caution. They use a significant amount of CPU resources and can affect system performance.

**Examples**
The `debug acircuit checkpoint` command is issued on the active route processor (RP):

```
Router# debug mpls 12transport checkpoint
Router# debug acircuit checkpoint
Router# show debug

AToM HA:
  AToM checkpointing events and errors debugging is on
AC HA:
  Attachment Circuit Checkpoint debugging is on
Router# conf terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# interface Fa5/1/1.2
Router(config-subif)# xconnect 10.55.55.2 1002 pw-class mpls
AToM HA [10.55.55.2, 1002]: Build provision msg, SSM sw/seg 8192/8194 [0x2000/0x2002] PW id 9216 [0x2400] local label 21
AToM HA: CF sync send complete
AC HA CF: Sync send complete. Code:0
```
On the standby RP, the following messages indicate that it receives checkpointing data:

AC HA [10.55.55.2, 1002]: Add to WaitQ. Flags:1
AToM HA [10.55.55.2, 1002]: Received 32-byte provision version 1 CF message
AC HA CF: ClientId:89, Entity:0 Length:40
AToM HA [10.55.55.2, 1002]: Process chkpt msg provision [1], ver 1
AToM HA [10.55.55.2, 1002]: Reserved.SGM sw/seg 8192/8194 [0x2000/0x2002] FW id 9216 [0x2400]
AC HA [10.55.55.2, 1002]: Remove from WaitQ. Flags:1[OK][OK]

During a switchover from an active RP to a backup RP, the debug messages look similar to the following:

%HA-5-MODE: Operating mode is hsa, configured mode is sso.
AC HA RF: CId:83, Seq:710, Sta:RF_STATUS_OPER_REDUNDANCY_MODE_CHANGE, Opr:5, St:STANDBY HOT, PST:ACTIVE
AToM HA: CId 84, Seq 715, Status RF_STATUS_OPER_REDUNDANCY_MODE_CHANGE, Op 5, State STANDBY HOT, Peer ACTIVE
AC HA RF: CId:83, Seq:710, Sta:RF_STATUS_PEER_PRESENCE, Opr:0, St:STANDBY HOT, PST:ACTIVE
AToM HA: CId 84, Seq 715, Status RF_STATUS_PEER_PRESENCE, Op 0, State STANDBY HOT, Peer ACTIVE
AC HA RF: CId:83, Seq:710, Sta:RF_STATUS_PEER_COMM, Opr:0, St:STANDBY HOT, PST:DISABLED
AToM HA: CId 84, Seq 715, Status RF_STATUS_PEER_COMM, Op 0, State STANDBY HOT, Peer DISABLED
%HA-2-CUTOVER_NOTICE: Cutover initiated. Cease all console activity until system restarts.
%HA-2-CUTOVER_NOTICE: Do not add/remove RSPs or line cards until switchover completes.
%OIR-6-REMCARD: Card removed from slot 4, interfaces disabled
%OIR-6-REMCARD: Card removed from slot 5, interfaces disabled
%OIR-6-REMCARD: Card removed from slot 9, interfaces disabled
%HA-5-NOTICE: Resuming initialization...
AC HA RF: CId:83, Seq:710, Sta:RF_STATUS_REDUNDANCY_MODE_CHANGE, Opr:7, St:STANDBY HOT, PST:DISABLED

%LDP-5-GR: LDP restarting gracefully. Preserving forwarding state for 250 seconds.
AC HA RF: CId:83, Seq:710, Sta:RF_PROG_ACTIVE, Opr:0, St:ACTIVE, PST:DISABLED
AToM HA: CId 84, Seq 715, Event RF_PROG_ACTIVE, Op 0, State ACTIVE, Peer DISABLED
AC HA: Switchover: Standby->Active
AC HA RF: Reconciling

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>debug mpls l2transport checkpoint</strong></td>
<td>Enables the display of AToM events when AToM is configured for NSF/SSO and Graceful Restart.</td>
</tr>
</tbody>
</table>
To enable the display of information about the adjacency database, use the `debug adjacency` command in privileged EXEC mode. To disable the display of these events, use the `no` form of this command.

```
depbug adjacency [epoch| ipc| state| table] [ prefix ] [ interface ] [ connectionid id ] [ link { ipv4 | ipv6 | mpls } ]
```

```
no debug adjacency [epoch| ipc| state| table] [ prefix ] [ interface ] [ connectionid id ] [ link { ipv4 | ipv6 | mpls } ]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>epoch</code></td>
<td>(Optional) Displays adjacency epoch events.</td>
</tr>
<tr>
<td><code>ipc</code></td>
<td>(Optional) Displays interprocess communication (IPC) events for adjacencies.</td>
</tr>
<tr>
<td><code>state</code></td>
<td>(Optional) Displays adjacency system state machine events.</td>
</tr>
<tr>
<td><code>table</code></td>
<td>(Optional) Displays adjacency table operations.</td>
</tr>
<tr>
<td><code>prefix</code></td>
<td>(Optional) Displays debugging events for the specified IP address or IPv6 address.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> On the Cisco 10000 series routers, IPv6 is supported in Cisco IOS Release 12.2(28)SB and later releases.</td>
</tr>
<tr>
<td><code>interface</code></td>
<td>(Optional) Displays debugging events for the specified interface. For line cards, you must specify the line card <code>if_number</code> (interface number). Use the <code>show cef interface</code> command to obtain line card <code>if_numbers</code>.</td>
</tr>
<tr>
<td><code>connectionid id</code></td>
<td>(Optional) Displays debugging events for the specified client connection identification number.</td>
</tr>
<tr>
<td>`link {ipv4</td>
<td>ipv6</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> On the Cisco 10000 series routers, IPv6 is supported in Cisco IOS Release 12.2(28)SB and later releases.</td>
</tr>
</tbody>
</table>

**Command Default**

Debugging events are not displayed.

**Command Modes**

Privileged EXEC
Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(7)XE</td>
<td>This command was introduced on the Cisco 7600 series routers.</td>
</tr>
<tr>
<td>12.1(1)E</td>
<td>This command was implemented on the Cisco 7600 series routers.</td>
</tr>
<tr>
<td>12.2(14)SX</td>
<td>This command was implemented on the Supervisor Engine 720.</td>
</tr>
<tr>
<td>12.2(25)S</td>
<td>This command was integrated into Cisco IOS Release 12.2(25)S, and the</td>
</tr>
<tr>
<td></td>
<td><code>prefix</code>, <code>interface</code>, <code>connectionid</code>, and `link {ipv4</td>
</tr>
<tr>
<td></td>
<td>and arguments were added.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(28)SB and</td>
</tr>
<tr>
<td></td>
<td>implemented on the Cisco 10000 series routers.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

Usage Guidelines

Because debugging output is assigned high priority in the CPU process, you should use debug commands only to troubleshoot specific problems or during troubleshooting sessions with Cisco technical support staff. Also, you should use debug commands during periods of lower network traffic and fewer users. Debugging during these periods decreases the likelihood that increased debug command processing overhead will affect system use.

You can use any combination of the `prefix`, `interface`, `connectionid`, and `link {ipv4 | ipv6 | mpls}` keywords and arguments (in any order) as a filter to enable debugging for a specified subset of adjacencies.

Note

On the Cisco 10000 series routers, IPv6 is supported in Cisco IOS Release 12.2(28)SB and later releases.

Examples

The following example shows how to display information on the adjacency database:

```
Router# debug adjacency
*Jan 27 06:22:50.543: ADJ-ios_mgr: repopulate adjs on up event for Ethernet3/0
  (incomplete) no src set: init/update from interface
  (incomplete) no src set: set bundle to IPV6 adjacency oce
  (incomplete) no src set: allocated, setup and inserted OK
  (incomplete) src IPv6 ND: source IPv6 ND added OK
  (incomplete) src IPv6 ND: computed macstring (len 14): OK
  IPv6 ND: made complete (macstring len 0 to 14/0 octets)
  IPv6 ND: changed state to up
00:04:40: %LINEPROTO-5-UPDOWN: Interface Ethernet3/0, changed state to up
00:04:41: %LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet3/0, changed
```
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear adjacency</td>
<td>Clears the Cisco Express Forwarding adjacency table.</td>
</tr>
<tr>
<td>clear arp-cache</td>
<td>Deletes all dynamic entries from the ARP cache.</td>
</tr>
<tr>
<td>show adjacency</td>
<td>Displays Cisco Express Forwarding adjacency table information.</td>
</tr>
<tr>
<td>show mls cef adjacency</td>
<td>Displays information about the hardware Layer 3 switching adjacency node.</td>
</tr>
</tbody>
</table>
**debug adjacency (vasi)**

To display debugging information for the VRF-Aware Service Infrastructure (VASI) adjacency, use the `debug adjacency` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
deploy adjacency {vasileft| vasiright} number
no deploy interface {vasileft| vasiright} number
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vasileft</td>
<td>Displays information about the vasileft interface.</td>
</tr>
<tr>
<td>vasiright</td>
<td>Displays information about the vasiright interface.</td>
</tr>
<tr>
<td>number</td>
<td>Identifier of the VASI interface. The range is from 1 to 256.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS XE Release 2.6</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug adjacency` command:

```
Router# debug adjacency vasileft 1
Condition 1 set
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug vasi</td>
<td>Displays debugging information for the VASI.</td>
</tr>
<tr>
<td>debug interface (vasi)</td>
<td>Displays debugging information for the VASI interface descriptor block.</td>
</tr>
<tr>
<td>interface (vasi)</td>
<td>Configures VASI virtual interface.</td>
</tr>
<tr>
<td>show vasi pair</td>
<td>Displays the status of a VASI pair.</td>
</tr>
</tbody>
</table>
**debug alarm-interface**

To show real-time activities in the data channel or the management channel of the Alarm Interface Controller (AIC), use the `debug alarm-interface` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug alarm-interface slot-number {data|management}
noc debug alarm-interface slot-number {data|management}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>slot-number</code></td>
<td>Router chassis slot where the AIC network module is installed.</td>
</tr>
<tr>
<td><code>data</code></td>
<td>Displays AIC serial data channel and asynchronous craft port communication activity.</td>
</tr>
<tr>
<td><code>management</code></td>
<td>Displays IOS-to-AIC communication activity.</td>
</tr>
</tbody>
</table>

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(2)XG</td>
<td>This command was introduced for the Cisco 2600 series and the Cisco 3600 series.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(8)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

This command allows you to observe the management channel activity from the AIC in the specified slot. Such activity shows that the software running on the AIC CPU has reached a minimum level of working order.

**Examples**

The following is sample output from the `debug alarm-interface` command:

```
Router# debug alarm-interface
AIC Slot 1:STATUS received
```

The following is sample output from the `debug alarm-interface 1 data` command:

```
Router# debug alarm-interface 1 data
AIC Slot 1:STATUS received
aic_fastsend:particle count=1, len=1504
aic_pak_to_txring:scattered particle count=1, tx bytes=1504, leftover=0
aic_interrupt:# 30419 gstar=0x1000000
```
Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alarm-interface</td>
<td>Enters the alarm interface mode and configures the AIC.</td>
</tr>
<tr>
<td>reset</td>
<td>Resets the AIC CPU.</td>
</tr>
</tbody>
</table>
debug alps ascu

To enable debugging for airline product set (ALPS) agent set control units (ASCUs) use the `debug alps ascu` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
deploy alps ascu {event| packet| detail| all| format {ipars| router| both}} [interface [ascu id]]
no deploy alps ascu {event| packet| detail| all| format {ipars| router| both}} [interface [ascu id]]
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>Displays ASCU events or protocol errors.</td>
</tr>
<tr>
<td>packet</td>
<td>Displays sent or received packets.</td>
</tr>
<tr>
<td>detail</td>
<td>Displays all ASCU protocol events.</td>
</tr>
<tr>
<td>all</td>
<td>Enables event, packet, and detail debugging.</td>
</tr>
<tr>
<td>format ipars router both</td>
<td>Specifies how to display ASCU addresses and the hexadecimal data in the debug output:</td>
</tr>
<tr>
<td></td>
<td>• ipars -- Displays only the IPARS hexadecimal output.</td>
</tr>
<tr>
<td></td>
<td>• router -- Displays only the router hexadecimal output.</td>
</tr>
<tr>
<td></td>
<td>• both -- Displays both the IPARS and router hexadecimal output.</td>
</tr>
<tr>
<td></td>
<td>The only difference between the IPARS output and the router output is the format of the hexadecimal data.</td>
</tr>
<tr>
<td>interface</td>
<td>(Optional) Enables debugging on a specified interface. Applies only to the <code>event</code>, <code>packet</code>, <code>detail</code>, and <code>all</code> keywords.</td>
</tr>
<tr>
<td>ascu id</td>
<td>(Optional) Enables debugging for a specified ASCU.</td>
</tr>
</tbody>
</table>

### Command Default

Debugging is off.

### Command Modes

Privileged EXEC
Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(6)T</td>
<td>This command was introduced for limited availability.</td>
</tr>
<tr>
<td>12.0(1)</td>
<td>This command was available for general release.</td>
</tr>
<tr>
<td>12.0(5)T</td>
<td>This command was modified.</td>
</tr>
<tr>
<td>12.1(2)T</td>
<td>The <strong>format</strong>, <strong>ipars</strong>, <strong>router</strong>, and <strong>both</strong> keywords were added. The output for this command was modified to include IPARS and router formats.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

Usage Guidelines

To enable debugging for a group of ASCUs, enter a separate command for each ASCU interface and IA combination.

The **interface** option applies only to the **event**, **packet**, **detail**, and **all** keywords.

>Note: To specify the particular debug tracing level (**event**, **packet**, **detail** or **all**) and the format (router, pairs or both), you must configure the **debugalpsascu** command two times: once to configure the debug tracing level and once to configure the format.

Examples

The following output is from the **debugalpsascu** command, showing events or protocol errors in **router** format for ASCU 42 on interface Serial7:

```
Router# debug alps ascu format router
Router# debug alps ascu event Serial7 42
ALPS ASCU: T1 expired for ascu 42 on i/f Serial7
ALPS ASCU: DOWN event while UP for ascu 42 on i/f Serial7 : C1 count = 1
```

>Note: If you specify the **ipars** or **both** format for the **event** or **detail** tracing level, both the IPARS and router formats will be displayed.

The following output is from the **debugalpsascu** command, showing events or protocol errors in **ipars** format for ASCU 42 on interface Serial7:

```
Router# debug alps ascu format ipars
Router# debug alps ascu event Serial7 42
ALPS ASCU: T1 expired for ascu 42/2F on i/f Serial7
ALPS ASCU: DOWN event while UP for ascu 42/2F on i/f Serial7 : C1 count = 1
```
The following output is from the `debug alps ascu detail` command, showing all protocol events in `router` format for ASCU 42 on interface Serial6:

```plaintext
Router# debug alps ascu format router
Router# debug alps ascu detail Serial6 42
```

**Note**
If you specify the `ipars` or `both` format for the `event` or `detail` tracing level, both the IPARS and router formats will be displayed.

The following output is from the `debug alps ascu detail` command, showing all protocol events in `both` format for ASCU 42 on interface Serial6:

```plaintext
Router# debug alps ascu format both
```

The following output is from the `debug alps ascu packet` command, showing all packets sent or received in `router` format for ASCU 42 on interface Serial6:

```plaintext
Router# debug alps ascu packet format router Serial6 42
```

The following output is from the `debug alps ascu packet` command, showing all packets sent or received in `ipars` format for ASCU 42 on interface Serial6:

```plaintext
Router# debug alps ascu packet format ipars Serial6 42
```
ALPS IPARS Format:
2F3E38 26161C38 161C1826 141D24
ALPS ASCU: Tx ALC DATA MSG (14 bytes + CCC + 0 pad bytes) to ascu 42/2F on i/f Serial6
ALPS IPARS Format:
2F3E38 26161C38 161C1826 141D24
ALPS ASCU: Rx ALC DATA MSG (14 bytes + CCC) from ascu 42/2F on i/f Serial6, fwd ckt RTP_MATIP
ALPS Router Format:
02321D26 0C261616 140C0D18 26163135 0611C6
ALPS IPARS Format:
2F2C1126 33262525 35331339 26251C14 271DC6
ALPS ASCU: Tx ALC SERVICE MSG (18 bytes + CCC + 0 pad bytes) to ascu 42/2F on i/f Serial6
ALPS Router Format:
02321D26 0C261616 140C0D18 26163135 0611C6
ALPS IPARS Format:
2F2C1126 33262525 35331339 26251C14 271DC6
ALPS ASCU: Rx ALC DATA MSG (14 bytes + CCC) from ascu 42/2F on i/f Serial6, fwd ckt RTP_MATIP
ALPS Router Format:
42607866 65717866
ALPS IPARS Format:
65717966 755124
ALPS ASCU: Tx ALC DATA MSG (14 bytes + CCC + 0 pad bytes) to ascu 42/2F on i/f Serial6
ALPS Router Format:
022038 26253138 26253139 263511E4
ALPS IPARS Format:
2F3E38 26161C38 26161C18 26141DE4

The following output is from the debug alpsascupacket command, showing all packets sent or received in both formats for ASCU 42 on interface Serial6:

Router# debug alps ascu packet format both Serial6 42

ALPS ASCU: Tx ALC SERVICE MSG (18 bytes + CCC + 0 pad bytes) to ascu 42/2F on i/f Serial6
ALPS Router Format:
02321D26 0C261616 140C0D18 26163135 0611C6
ALPS IPARS Format:
2F2C1126 33262525 35331339 26251C14 271DC6
ALPS ASCU: Rx ALC DATA MSG (14 bytes + CCC) from ascu 42/2F on i/f Serial6, fwd ckt RTP_MATIP
ALPS Router Format:
42607866 65717866 65717966 755124
ALPS IPARS Format:
2F3E38 26161C38 161C1826 141D24
ALPS ASCU: Tx ALC DATA MSG (14 bytes + CCC + 0 pad bytes) to ascu 42/2F on i/f Serial6
ALPS Router Format:
022038 26253138 26253139 263511E4
ALPS IPARS Format:
2F3E38 26161C38 26161C18 26141DE4
debug alps circuit event

To enable event debugging for airline product set (ALPS) circuits, use the `debug alps circuit event` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
depalp circuit event [ name ]
nodepalp circuit event [ name ]
```

**Syntax Description**

| name | (Optional) Name given to identify an ALPS circuit on the remote customer premises equipment (CPE). |

**Command Default**

If no circuit name is specified, then debugging is enabled for every ALPS circuit.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3 T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

To enable debugging for a single ALPS circuit, specify the name of the circuit.

To enable debugging for a group of circuits, enter a separate command for each circuit name.

**Examples**

The following is sample output from the `debug alps circuit event` command for circuit RTP_AX25:

```
alps-rcpe# debug alps circuit event RTP_AX25
ALPS P1024 CKT: FSM - Ckt= RTP_AX25, State= OPEN, Event= DISABLE: (CloseAndDisable)->DISC
ALPS P1024 CKT: FSM - Ckt= RTP_AX25, State= DISC, Event= ENABLE: (TmrStartNullRetry)->INOP
ALPS P1024 CKT: Ckt= RTP_AX25, Open - peer set to 200.100.40.2
ALPS P1024 CKT: Ckt= RTP_AX25, Open - peer open.
ALPS P1024 CKT: FSM - Ckt= RTP_AX25, State= INOP, Event= RETRY_TIMEOUT: (Open)->OPEN
ALPS P1024 CKT: FSM - Ckt= RTP_AX25, State= OPNG, Event= Ckt_OPEN_CFM: (CacheAndFwdAscuData)->OPEN
alps-ccpe# debug alps circuit event RTP_AX25
ALPS AX.25 FSNM: Ckt= RTP_AX25, State= OPEN, Event= CktClose, Rsn= 12: (PvcKill,CktRemove,TmrStartClose)->INOP
ALPS AX.25 FSNM: Ckt= RTP_AX25, State= INOP, Event= X25PvcInact, Rsn= 0: (-,-,-)->INOP
ALPS AX.25 FSNM: Ckt= RTP_AX25, State= INOP, Event= X25VcDeleted, Rsn= 0: |
```
(-,CktDestroy,TmrStop)->INOP
ALPS AX.25 FSM: Ckt= RTP_AX25, State= INOP, Event= CktOpReq, Rsn= 4:
(PvcMake,CktAdd,TmrStartOpen)->OPNG
ALPS AX.25 FSM: Ckt= RTP_AX25, State= OPNG, Event= X25ResetTx, Rsn= 0:
(-,-,-)->OPNG
ALPS AX.25 FSM: Ckt= RTP_AX25, State= OPNG, Event= X25VcUp, Rsn= 0:
(-,OpnCfm,TmrStop)->OPEN
debug alps peer

To enable event or packet debugging for airline product set (ALPS) peers, use the `debug alps peer` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
depalps peer {event| packet} [ ip-address ]
no debug alps peer {event| packet} [ ip-address ]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>Specifies debugging for an event.</td>
</tr>
<tr>
<td>packet</td>
<td>Specifies debugging for a packet.</td>
</tr>
<tr>
<td>ip-address</td>
<td>(Optional) Remote peer IP address.</td>
</tr>
</tbody>
</table>

**Command Default**

If no IP address is specified, then debugging is enabled for every peer connection.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(6)T</td>
<td>This command was introduced for limited availability.</td>
</tr>
<tr>
<td>12.0(1)</td>
<td>This command was available for general release.</td>
</tr>
<tr>
<td>12.0(5)T</td>
<td>The <code>packet</code> keyword was added. The format for the output was modified for consistency.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

To enable debugging for a single remote ALPS peer, specify the peer IP address.

To enable debugging for a set of remote peers, enter the command for each peer IP address.

**Examples**

The following is sample output from the `debug alps peer` command:

```
Router# debug alps peer packet
ALPS PEER:Peer  (10.227.50.106, MATIP_A_CKT-1) - TX Peer Data Msg (18 bytes) 040A5320: 01 00001241 040A5330: 45546B5F 6F4F7757 67477B5B 51
ALPS PEER:Peer  (10.227.50.106, MATIP_A_CKT-1) - RX Peer Data Msg (18 bytes) 04000550: 01000012 45546B5F 6F4F7757 67477B5B 51
```
debug alps peer

ALPS PEER:Peer (10.227.50.106, MATIP_A_CKT-1) - TX Peer Data Msg (18 bytes)
0409F6E0: 01 00001241 45546B5F
0409F6F0: 6F4F7757 67477B5B 51
ALPS PEER:Peer (10.227.50.106, MATIP_A_CKT-1) - RX Peer Data Msg (18 bytes)
04000680: 01 000012 4145546B
04000690: 5F6E6F77 57677B5B 51

debug aaa accounting through debug auto-config
debug alps peer event

To enable event debugging for airline product set (ALPS) peers, use the `debug alps peer event` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug alps peer event ipaddr
no debug alps peer event ipaddr
```

**Syntax Description**

<table>
<thead>
<tr>
<th>ipaddr</th>
<th>Peer IP address.</th>
</tr>
</thead>
</table>

**Command Default**

If no IP address is specified, debugging is enabled for every peer connection.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3 T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

To enable debugging for a single remote ALPS peer, specify the peer IP address.

To enable debugging for a set of remote peers, enter the command for each peer IP address.

**Examples**

The following is sample output from the `debug alps peer event` command:

```
Router# debug alps peer event
ALPS PEER: FSM - Peer 200.100.25.2, Event ALPS_CLOSED_IND, State OPENED
ALPS PEER: peer 200.100.25.2 closed - closing peer circuits.
ALPS PEER: Fromiscuous peer created for 200.100.25.2
ALPS PEER: TCP Listen - passive open 200.100.25.2(11003) -> 10000
ALPS PEER: FSM - Peer 200.100.25.2, Event ALPS_OPEN_IND, State DISCONNN
ALPS PEER: peer 200.100.25.2 opened OK.
```
debug alps snmp

To enable debugging for airline product set (ALPS) Simple Network Management Protocol (SNMP) agents, use the `debug alps snmp` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug alps snmp
no debug alps snmp
```

### Syntax Description

This command has no arguments or keywords.

### Command Default

Debugging for SNMP agents is not enabled.

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(6)T</td>
<td>This command was introduced for limited availability.</td>
</tr>
<tr>
<td>12.0(1)T</td>
<td>This command was available for general release.</td>
</tr>
<tr>
<td>12.0(5)T</td>
<td>This command was added to the documentation.</td>
</tr>
<tr>
<td>12.1(2)T</td>
<td>The output for this command was modified to reflect MIB and SNMP changes.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

### Examples

The following output is from the `debug alps snmp` command. The first line shows a circuit event status change. The second line shows an ASCU status change. The third line shows a peer connection status change.

```
Router# debug alps snmp
ALPS CktStatusChange Notification for circuit CKT-1
ALPS AscuStatusChange Notification for ascu (Serial3, 41)
PeerConnStatusChange Notification for peer (10.227.50.106, MATIP_A_CKT-1)
The following output shows that an open failure has occurred on circuit 1:
ALPS CktOpenFailure Notification for circuit CKT1
The following output shows that a partial rejection to an ALPS circuit peer open request has occurred on circuit 1:
ALPS CktPartialReject Notification for ascu (Serial2, 41) on circuit CKT1
```
debug ancp

To enable the display of debugging information related to the Access Node Control Protocol (ANCP), use the `debug ancp` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug ancp {adjacency| details| errors| events| neighbor | packets [brief]| port-event {details| events}| port-management {details| events}}
no debug ancp {adjacency| details| errors| events| neighbor | packets [brief]| port-event {details| events}| port-management {details| events}}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adjacency</td>
<td>Displays information about adjacency messages on an ANCP server.</td>
</tr>
<tr>
<td>details</td>
<td>Displays detailed static configuration information relating to ANCP and dynamic line conditions.</td>
</tr>
<tr>
<td>errors</td>
<td>Displays information about ANCP protocol errors.</td>
</tr>
<tr>
<td>events</td>
<td>Displays information about ANCP protocol events.</td>
</tr>
<tr>
<td>neighbor</td>
<td>Displays information about ANCP neighbors.</td>
</tr>
<tr>
<td>packets</td>
<td>Displays information about ANCP control packets.</td>
</tr>
<tr>
<td>brief</td>
<td>(Optional) Displays static configuration information for ANCP control packets.</td>
</tr>
<tr>
<td>port-event</td>
<td>Displays ANCP event messages (port up and port down) related to data ports.</td>
</tr>
<tr>
<td>port-management</td>
<td>Displays ANCP operations, administration, and maintenance (OAM) messages.</td>
</tr>
</tbody>
</table>

**Command Modes**

- Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(28)ZV</td>
<td>This command was introduced on the Cisco 10000 series router.</td>
</tr>
<tr>
<td>12.2(31)ZV1</td>
<td>This command was modified. L2CP was replaced with ANCP.</td>
</tr>
<tr>
<td>Cisco IOS XE Release 2.4</td>
<td>This command was integrated into Cisco IOS XE Release 2.4.</td>
</tr>
</tbody>
</table>
Modification

Cisco IOS XE Release 3.2S

This command was modified. The `port-event` and `port-management` keywords were added.

Usage Guidelines

You can use the `debug condition interface` command to conditionalize all the `debug` commands where the ANCP TCP connections are terminated.

Examples

The following is sample output from the `debug ancp adjacency` command. The output fields are self-explanatory.

Router# debug ancp adjacency
*Sep 20 08:23:53.833: Sending adj (SYN) msg nbr_info 0 nbr 10.1.1.1/46459 tcb 50C2050
*Sep 20 08:23:53.834: ANCP: Using Mac address aabb.cc00.7a00
*Sep 20 08:23:53.834: ANCP : Sending adj (SYN) msg to writeQ, msg len 48 interface
Ethernet0/0.1 TCB 50C2050
*Sep 20 08:23:53.836: ANCP: Received adjacency (SYN) message, from 10.1.1.1 port 46459 tcb 50C2050
*Sep 20 08:23:53.836: ANCP: 2 capability tlv(s) received
*Sep 20 08:23:53.837: Sending adj (SYNACK) msg nbr_info 6368300 nbr 10.1.1.1/46459 tcb 50C2050
*Sep 20 08:23:53.837: ANCP: Using Mac address aabb.cc00.7a00
*Sep 20 08:23:53.837: ANCP: Sending adj (SYNACK) msg to writeQ, msg len 48 interface
Ethernet0/0.1 TCB 50C2050
*Sep 20 08:23:53.837: ANCP: Using Mac address aabb.cc00.7a00
*Sep 20 08:23:53.837: ANCP ADJ: received SYN from nbr, state SYNSENT Sending SYNACK to nbr. State transitions to SYNRCVD
*Sep 20 08:23:53.841: ANCP: 2 capability tlv(s) received
*Sep 20 08:23:53.842: ANCP ADJ: received SYNACK from nbr, state SYNRCVD Sending ACK to nbr. State transitions to ESTAB

The following is sample output from the `debug ancp events` command:

Router# debug ancp events
ANCP protocol events debugging is on
ANCP TCP accepted on address 10.1.1.1 for remote peer 10.1.1.1
Sending adj msg, code 1 nbr_info 0 nbr 10.1.1.1/40224 tcb 549D930
TCP read_notify called, calling ancp_io_read_request. ANCP IO should service the read request
ANCP: Received adjacency message, code 1 Src 10.1.1.1 Src port 40224 tcb 549D930
ANCP: 2 capability tlv(s) received
Sending adj msg, code 3 nbr_info 5FA92F0 nbr 10.1.1.1/40224 tcb 549D930
ANCP ADJ: received SYN from nbr, state SYNSENT Sending SYNACK to nbr. State transitions to SYNRCVD
TCP read_notify called, calling ancp_io_read_request. ANCP IO should service the read request
ANCP: Received adjacency message, code 2 Src 10.1.1.1 Src port 40224 tcb 549D930
ANCP: 2 capability tlv(s) received
Sending adj msg, code 3 nbr_info 5FA92F0 nbr 10.1.1.1/40224 tcb 549D930
ANCP ADJ: received SYNACK from nbr, state SYNRCVD Sending ACK to nbr. State transitions to ESTAB
TCP read_notify called, calling ancp_io_read_request. ANCP IO should service the read request
ANCP: Received adjacency message, code 3 Src 10.1.1.1 Src port 40224 tcb 549D930
ANCP: 2 capability tlv(s) received
ANCP ADJ: received ACK from nbr, state ESTAB Sending ACK to nbr thou timer. State stays ESTAB
TCP read_notify called, calling ancp_io_read_request. ANCP IO should service the read request
Sending adj msg, code 3 nbr_info 5FA92F0 nbr 10.1.1.1/40224 tcb 549D930
ANCP: Received adjacency message, code 3 Src 10.1.1.1 Src port 40224 tcb 549D930
ANCP: 2 capability tlv(s) received
ANCP ADJ: received ACK from nbr, state ESTAB Sending ACK to nbr thou timer. State stays ESTAB
Sending adj msg, code 3 nbr_info 5FA92F0 nbr 10.1.1.1/40224 tcb 549D930
TCP read_notify called, calling ancp_io_read_request. ANCP IO should service the read request.
ANCP: Received adjacency message, code 3 Src 10.1.1.1 Src port 40224 tcb 549D930
ANCP: 2 capability tlv(s) received
ANCP ADJ: received ACK from nbr, state ESTAB Sending ACK to nbr thou timer. State stays ESTAB.

The table below describes the significant fields shown in the display.

**Table 4: debug ancp events Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Received adjacency message</td>
<td>Adjacency message received from the ANCP neighbor. Adjacency protocol messages synchronize the Network Access Server and access nodes.</td>
</tr>
<tr>
<td>2 capability tlv(s) received</td>
<td>Supported capability type-length-values are included in the adjacency message.</td>
</tr>
</tbody>
</table>

The following is sample output from the `debug ancp port-event details` command. The output fields are self-explanatory.

```
Router# debug ancp port-event details
*Sep 20 08:24:55.608: ANCP: Found db entry based on access-loop-circuit-id alcid1, received in Port UP message
*Sep 20 08:24:55.608: Advance 12 bytes (aligned) to next TLV.
*Sep 20 08:24:55.609: ANCP: Processing DSL LINE_ATTRIBUTES sub-TLVs: PORT UP
*Sep 20 08:24:55.609: ANCP: Port UP: received DSL Line Attrs, tlv length in msg 32 aligned len 32
*Sep 20 08:24:55.609: DSL_TYPE received. 4
*Sep 20 08:24:55.609: Advance 8 bytes (aligned) to next sub-TLV.
*Sep 20 08:24:55.609: ACTUAL_DATA_RATE_UP received. 7878
*Sep 20 08:24:55.609: Advance 8 bytes (aligned) to next sub-TLV.
*Sep 20 08:24:55.609: ACTUAL_DATA_RATE_DN received. 9090
*Sep 20 08:24:55.610: Advance 8 bytes (aligned) to next sub-TLV.
*Sep 20 08:24:55.610: DSL_LINE_STATE received. 1
*Sep 20 08:24:55.610: Advance 8 bytes (aligned) to next sub-TLV.
```

The following is sample output from the `debug ancp port-management details` command. The output fields are self-explanatory.

```
Router# debug ancp port-management details
*Sep 20 08:29:22.263: 3120 3500
0000 0000
8001 0034
0000 0000
0000 0000
0000 0000
0000 0900
0000 0000
0020 0500
0001 0010
0001 0006
616C 6369
6431 0000
*Sep 20 08:29:22.266: ANCP_PORT_MGMT: Ethernet0/0.1 Port Mgmt message, Total TLVs : 1
*Sep 20 08:29:22.266: ANCP_PORT_MGMT: Received Access-Loop-Cir-ID alcid1
*Sep 20 08:29:22.266: ANCP_PORT_MGMT: Received Result: success(0x3) Code: Specified access line does not exist(0x500).
*Sep 20 08:29:22.266: Advance 12 bytes (aligned) to next TLV.
```
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug condition interface</code></td>
<td>Limits the output for some <code>debug</code> commands on the basis of interface, VC, or VLAN.</td>
</tr>
<tr>
<td><code>show ancp neighbor</code></td>
<td>Displays statistics of ANCP neighbor information and neighborship information with local ANCP ports.</td>
</tr>
<tr>
<td><code>show ancp status</code></td>
<td>Displays ANCP-related information for the ANCP endpoints configured on a BRAS interface.</td>
</tr>
</tbody>
</table>
debug appfw

To display debug messages about Cisco IOS Firewall events, use the `debug appfw` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug appfw {application protocol| function-trace| object-creation| object-deletion| events| timers| detailed}
no debug appfw {application protocol| function-trace| object-creation| object-deletion| events| timers| detailed}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>application protocol</code></td>
<td>Displays messages about protocol events of firewall-inspected applications, including details about the protocol’s packets. Currently, the only supported protocol is HTTP. (Issue the <code>http</code> keyword.)</td>
</tr>
<tr>
<td><code>function-trace</code></td>
<td>Displays messages about software functions called by Cisco IOS Firewall.</td>
</tr>
<tr>
<td><code>object-creation</code></td>
<td>Displays messages about software objects that are being created by Cisco IOS Firewall. Cisco IOS Firewall-inspected sessions begin when the object is created.</td>
</tr>
<tr>
<td><code>object-deletion</code></td>
<td>Displays messages about software objects that are being deleted by Cisco IOS Firewall. Cisco IOS Firewall-inspected sessions close when the object is deleted.</td>
</tr>
<tr>
<td><code>events</code></td>
<td>Displays messages about Cisco IOS software events, including Cisco IOS Firewall packet processing.</td>
</tr>
<tr>
<td><code>timers</code></td>
<td>Displays messages about Cisco IOS Firewall timer events, such as an idle timeout by the Cisco IOS Firewall.</td>
</tr>
<tr>
<td><code>detailed</code></td>
<td>Detailed information for all other enabled Cisco IOS Firewall debugging is displayed.</td>
</tr>
</tbody>
</table>

**Note**: This keyword should be used in conjunction with other Cisco IOS Firewall debugging commands.

### Command Modes

Privileged EXEC
This command was introduced.

Examples

The following sample configuration shows how to configure an HTTP policy with application firewall debugging enabled:

```
Router(config)# appfw policy-name myPolicy
APPFW FUNC:appfw_policy_find -- Policy myPolicy is not found
APPFW FUNC:appfw_policy_alloc
APPFW FUNC:appfw_policy_alloc -- policy_alloc 0x65727278
APPFW FUNC:appfw_policy_alloc -- Policy 0x65727278 is set to valid
APPFW FUNC:appfw_policy_alloc -- Policy myPolicy has been created
APPFW FUNC:appfw_policy_command -- memlock policy 0x65727278

! Debugging sample for application (HTTP) creation
```

```
Router(cfg-appfw-policy)# application http
APPFW FUNC:appfw_http_command
APPFW FUNC:appfw_http_appl_find
APPFW FUNC:appfw_http_appl_find -- Application not found
APPFW FUNC:appfw_http_appl_alloc
APPFW FUNC:appfw_http_appl_alloc -- appl_http 0x64D7A25C
APPFW FUNC:appfw_http_appl_alloc -- Application HTTP parser structure 64D7A25C created

! Debugging sample for HTTP-specific application inspection
```

```
Router(cfg-appfw-policy-http)# strict-http action reset alarm
APPFW FUNC:appfw_http_subcommand
```

```
Router# debug appfw detailed
APPFW Detailed Debug debugging is on
```

```
fw7-7206a#debug appfw object-creation
APPFW Object Creations debugging is on
```

```
fw7-7206a#debug appfw object-deletion
APPFW Object Deletions debugging is on
```
debug apple arp

To enable debugging of the AppleTalk Address Resolution Protocol (AARP), use the `debug apple arp` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug apple arp [type number]
no debug apple arp [type number]
```

**Syntax Description**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>type</code></td>
<td>(Optional) Interface type.</td>
</tr>
<tr>
<td><code>number</code></td>
<td>(Optional) Interface number.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC

**Usage Guidelines**

This command is helpful when you experience problems communicating with a node on the network you control (a neighbor). If the `debug apple arp` display indicates that the router is receiving AARP probes, you can assume that the problem does not reside at the physical layer.

**Examples**

The following is sample output from the `debug apple arp` command:

```
Router# debug apple arp
Ether0: AARP: Sent resolve for 4160.26
Ether0: AARP: Reply from 4160.26(0000.0c00.0453) for 4160.154(0000.0c00.8ea9)
Ether0: AARP: Resolved waiting request for 4160.26(0000.0c00.0453)
Ether0: AARP: Reply from 4160.19(0000.0c00.0082) for 4160.154(0000.0c00.8ea9)
Ether0: AARP: Resolved waiting request for 4160.19(0000.0c00.0082)
Ether0: AARP: Reply from 4160.19(0000.0c00.0082) for 4160.154(0000.0c00.8ea9)
```

Explanations for representative lines of output follow.

The following line indicates that the router has requested the hardware MAC address of the host at network address 4160.26:

```
Ether0: AARP: Sent resolve for 4160.26
```

The following line indicates that the host at network address 4160.26 has replied, giving its MAC address (0000.0c00.0453). For completeness, the message also shows the network address to which the reply was sent and its hardware MAC address (also in parentheses).

```
Ether0: AARP: Reply from 4160.26(0000.0c00.0453) for 4160.154(0000.0c00.8ea9)
```

The following line indicates that the MAC address request is complete:

```
Ether0: AARP: Resolved waiting request for 4160.26(0000.0c00.0453)
```
### debug apple domain

To enable debugging of the AppleTalk domain activities, use the `debug appledomain` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

**debug apple domain**

**no debug apple domain**

#### Syntax Description

This command has no arguments or keywords.

#### Command Modes

Privileged EXEC

#### Usage Guidelines

Use the `debug appledomain` command to observe activity for domains and subdomains. Use this command in conjunction with the `debug appleremap` command to observe interaction between remapping and domain activity. Messages are displayed when the state of a domain changes, such as creating a new domain, deleting a domain, and updating a domain.

#### Examples

The following is sample output from the `debug appledomain` command intermixed with output from the `debug appleremap` command; the two commands show related events:

```
Router# debug apple domain
Router# debug apple remap
AT-REMAP: RemapProcess for net 30000 domain AURP Domain 1
AT-REMAP: Could not find a remap for cable 3000-3001
AT-DOMAIN: atdomain_DisablePort for Tunnel0
AT-DOMAIN: CleanUpDomain for domain 1 [AURP Domain 1]
AT-DOMAIN: Disabling interface Ethernet1
AT-DOMAIN: atdomain_DisablePort for Ethernet1
AT-DOMAIN: CleanUpDomain for domain 1 [AURP Domain 1]
AT-DOMAIN: CleanSubDomain for inbound subdomain 1
AT-REMAP: Remap for net 70 inbound subdomain 1 has been deleted
AT-DOMAIN: DeleteAvRemapList for inbound subdomain 1
AT-DOMAIN: DeleteAvRemapList for inbound subdomain 1
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-DOMAIN: CleanSubDomain for outbound subdomain 1
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-REMAP: RemapProcess for net 30000 domain AURP Domain 1 Remapped Net 10000
AT-REMAP: Remap for net 50 outbound subdomain 1 has been deleted
AT-DOMAIN: DeleteAvRemapList for outbound subdomain 1
AT-DOMAIN: DeleteAvRemapList for outbound subdomain 1
AT-DOMAIN: CleanUpDomain for domain 1 [AURP Domain 1]
AT-DOMAIN: CleanSubDomain for inbound subdomain 1
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-DOMAIN: DeleteAvRemapList for inbound subdomain 1
AT-DOMAIN: CleanSubDomain for outbound subdomain 1
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-DOMAIN: DeleteAvRemapList for outbound subdomain 1
```

#### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug apple remap</code></td>
<td>Enables debugging of the AppleTalk remap activities.</td>
</tr>
</tbody>
</table>
debug aaa accounting through debug auto-config

d debug apple domain
debug apple eigrp-all

To enable debugging output from the Enhanced IGRP routines, use the `debug apple eigrp-all` privileged EXEC command. The `no` form of this command disables debugging output.

```
debug apple eigrp-all
no debug apple eigrp-all
```

**Syntax Description**

This command has no arguments or keywords.

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(13)T</td>
<td>This command is no longer supported in Cisco IOS Mainline releases or in Technology-based (T-train) releases. It might continue to appear in 12.2S-family releases.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The `debug apple eigrp-all` command can be used to monitor acquisition of routes, aging route table entries, and advertisement of known routes through Enhanced IGRP.

**Caution**

Because the `debug apple eigrp-all` command can generate many messages, use it only when the CPU utilization of the router is less than 50 percent.

**Examples**

The following is sample output from the `debug apple eigrp-all` command:

```
Router# debug apple eigrp-all
3:54:34: atigrp2_router: peer is 83.195
3:54:37: AT: atigrp2_write: about to send packet
3:54:37: Ethernet2: output AT packet: enctype UNKNOWN, size 65
3:54:37: 07FFFFFF0000FFFFFFFFFFFF0000C1485B00046|0041ACD100000053F8F58585802059110
3:54:37: 000000000000000000000000000000010001000C01000100000000000F0204000C0053005300
3:54:39: AT: atigrp2, src=Ethernet2:83.143, dst=83-83, size=65, EIGRP pkt sent
3:54:39: atigrp2_router: peer is 83.195
3:54:42: AT: atigrp2_write: about to send packet
3:54:42: Ethernet2: output AT packet: enctype UNKNOWN, size 65
3:54:42: 07FFFFFF0000FFFFFFFFFFFF0000C1485B00046|0041ACD100000053F8F58585802059110
3:54:42: 000000000000000000000000000000010001000C01000100000000000F0204000C0053005300
3:54:42: AT: atigrp2, src=Ethernet2:83.143, dst=83-83, size=65, EIGRP pkt sent
```

The table below describes the significant fields shown in the display.

**Table 5: debug apple eigrp Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>atigrp2_router:</td>
<td>AppleTalk address of the neighbor.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>AT:</td>
<td>Indicates that this is an AppleTalk packet.</td>
</tr>
<tr>
<td>Ethernet2:</td>
<td>Name of the interface through which the router received the packet.</td>
</tr>
<tr>
<td>src=</td>
<td>Name of the interface sending the Enhanced IGRP packet, as well at its AppleTalk address.</td>
</tr>
<tr>
<td>dst=</td>
<td>Cable range of the destination of the packet.</td>
</tr>
<tr>
<td>size=</td>
<td>Size of the packet (in bytes).</td>
</tr>
</tbody>
</table>
debug apple errors

To display errors occurring in the AppleTalk network, use the `debugappleerrors` command in privileged EXEC mode. To disable debugging output, use the `no debug apple errors` form of this command.

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>(Optional) Interface type.</td>
</tr>
<tr>
<td>number</td>
<td>(Optional) Interface number.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC

**Usage Guidelines**

In a stable AppleTalk network, the `debugappleerrors` command produces little output. To solve encapsulation problems, enable `debugappleerrors` and `debugapplepacket` together.

**Examples**

The following is sample output from the `debugappleerrors` command when a router is brought up with a zone that does not agree with the zone list of other routers on the network:

```
Router# debug apple errors
%AT-3-ZONEDISAGREES: Ethernet0: AppleTalk port disabled; zone list incompatible with 4160.19
%AT-3-ZONEDISAGREES: Ethernet0: AppleTalk port disabled; zone list incompatible with 4160.19
%AT-3-ZONEDISAGREES: Ethernet0: AppleTalk port disabled; zone list incompatible with 4160.19
```

As the output suggests, a single error message indicates zone list incompatibility; this message is sent out periodically until the condition is corrected or the `debugappleerrors` command is turned off.

Most of the other messages that the `debugappleerrors` command can generate are obscure or indicate a serious problem with the AppleTalk network. Some of these other messages follow.

In the following message, RTMPRsp, RTMPReq, ATP, AEP, ZIP, ADSP, or SNMP could replace NBP, and "llap dest not for us" could replace "wrong encapsulation":

```
Packet discarded, src 4160.12-254,dst 4160.19-254,NBP,wrong encapsulation
```

In the following message, in addition to an invalid echo packet error, other possible errors are unsolicited AEP echo reply, unknown echo function, invalid ping packet, unknown ping function, and bad responder packet type:

```
Ethernet0: AppleTalk packet error; no source address available
AT: pak_reply: dubious reply creation, dst 4160.19
AT: Unable to get a buffer for reply to 4160.19
Processing error, src 4160.12-254,dst 4160.19-254,AEP, invalid echo packet
```
The **debugappleerrors** command can print out additional messages when other debugging commands are also turned on. When you turn on both the **debugappleerrors** and **debugapplenbp** commands, the following message can be generated:

```
Proc err, src 4160.12-254,dst 4160.19-254,ZIP,NetInfo Reply format is invalid
```

In the preceding message, in addition to the NetInfo Reply format is invalid error, other possible errors are NetInfoReply not for me, NetInfoReply ignored, NetInfoReply for operational net ignored, NetInfoReply from invalid port, unexpected NetInfoReply ignored, cannot establish primary zone, no primary has been set up, primary zone invalid, net information mismatch, multicast mismatch, and zones disagree.

When you turn on both the **debugappleerrors** and **debugapplenbp** commands, the following message can be generated:

```
Processing error,...,NBP,NBP name invalid
```

In the preceding message, in addition to the NBP name invalid error, other possible errors are NBP type invalid, NBP zone invalid, not operational, error handling brrq, error handling proxy, NBP fwdreq unexpected, No route to srcnet, Proxy to "*" zone, Zone "*" from extended net, No zone info for "*", and NBP zone unknown.

When you turn on both the **debugappleerrors** and **debugapplerouting** commands, the following message can be generated:

```
Processing error,...,RTMPReq, unknown RTMP request
```

In the preceding message, in addition to an unknown RTMP request error, other possible errors are RTMP packet header bad, RTMP cable mismatch, routed RTMP data, RTMP bad tuple, and Not Req or Rsp.
**debug apple events**

To display information about AppleTalk special events, neighbors becoming reachable or unreachable, and interfaces going up or down, use the `debug apple events` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug apple events [type number]
no debug apple events [type number]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>type</code></td>
<td>(Optional) Interface type.</td>
</tr>
<tr>
<td><code>number</code></td>
<td>(Optional) Interface number.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC

**Usage Guidelines**

Only significant events (for example, neighbor and route changes) are logged.

The `debug apple events` command is useful for solving AppleTalk network problems because it provides an overall picture of the stability of the network. In a stable network, the `debug apple events` command does not return any information. If the command generates numerous messages, those messages can indicate possible sources of the problems.

When configuring or making changes to a router or interface for AppleTalk, enable the `debug apple events` command to alert you to the progress of the changes or to any errors that might result. Also use this command periodically when you suspect network problems.

The `debug apple events` command is also useful to determine whether network flapping (nodes toggling online and offline) is occurring. If flapping is excessive, look for routers that only support 254 networks.

When you enable the `debug apple events` command, you will see any messages that the `appleevent-logging` configuration command normally displays. Turning on the `debug apple events` command, however, does not cause the `appleevent-logging` command to be maintained in nonvolatile memory. Only turning on the `appleevent-logging` command explicitly stores it in nonvolatile memory. Furthermore, if the `appleevent-logging` command is already enabled, turning on or off the `debug apple events` command does not affect the `appleevent-logging` command.

**Examples**

The `debug apple events` command is useful in tracking the discovery mode state changes through which an interface progresses. When no problems are encountered, the state changes progress as follows:

1. Line down.
2. Restarting.
3. Probing (for its own address [node ID] using AARP).
4. Acquiring (sending out GetNetInfo requests).
5. Requesting zones (the list of zones for its cable).
6 Verifying (that the router’s configuration is correct. If not, a port configuration mismatch is declared).
7 Checking zones (to make sure its list of zones is correct).
8 Operational (participating in routing).

Explanations for individual lines of output follow.
The following message indicates that a port is set. In this case, the zone multicast address is being reset.

Ether0: AT: Resetting interface address filters
The following messages indicate that the router is changing to restarting mode:

%AT-5-INTRESTART: Ether0: AppleTalk port restarting; protocol restarted
Ether0: AppleTalk state changed; unknown -> restarting
The following message indicates that the router is probing in the startup range of network numbers (65280 to 65534) to discover its network number:

Ether0: AppleTalk state changed; restarting -> probing
The following message indicates that the router is enabled as a nonrouting node using a provisional network number within its startup range of network numbers. This type of message only appears if the network address the router will use differs from its configured address. This is always the case for a discovery-enabled router; it is rarely the case for a nondiscovery-enabled router.

%AT-6-ADDRUSED: Ether0: AppleTalk node up; using address 65401.148
The following messages indicate that the router is sending out GetNetInfo requests to discover the default zone name and the actual network number range in which its network number can be chosen:

Ether0: AppleTalk state changed; probing -> acquiring
%AT-6-ACQUIREMODE: Ether0: AT port initializing; acquiring net configuration
Now that the router has acquired the cable configuration information, the following message indicates that it restarts using that information:

Ether0: AppleTalk state changed; acquiring -> restarting
The following messages indicate that the router is probing for its actual network address:

Ether0: AppleTalk state changed; restarting -> line down
Ether0: AppleTalk state changed; line down -> restarting
Ether0: AppleTalk state changed; restarting -> probing
The following message indicates that the router has found an actual network address to use:

%AT-6-ADDRUSED: Ether0: AppleTalk node up; using address 4160.148
The following messages indicate that the router is sending out GetNetInfo requests to verify the default zone name and the actual network number range from which its network number can be chosen:

Ether0: AppleTalk state changed; probing -> acquiring
%AT-6-ACQUIREMODE: Ether0: AT port initializing; acquiring net configuration
The following message indicates that the router is requesting the list of zones for its cable:

Ether0: AppleTalk state changed; acquiring -> requesting zones
The following messages indicate that the router is sending out GetNetInfo requests to make sure its understanding of the configuration is correct:

Ether0: AppleTalk state changed; requesting zones -> verifying
AT: Sent GetNetInfo request broadcast on Ethernet0
The following message indicates that the router is rechecking its list of zones for its cable:

Ether0: AppleTalk state changed; verifying -> checking zones

The following message indicates that the router is now fully operational as a routing node and can begin routing:

Ether0: AppleTalk state changed; checking zones -> operational

A nondiscovery-enabled router can come up when no other router is on the wire; however, it must assume that its configuration (if accurate syntactically) is correct, because no other router can verify it. Notice that the last line indicates this situation.

The following is sample output from the `debug apple events` command that describes a discovery-enabled router coming up when there is no seed router on the wire:

```
Router# debug apple events
Ether0: AT: Resetting interface address filters
%AT-5-INTRESTART: Ether0: AppleTalk port restarting; protocol restarted
Ether0: AppleTalk state changed; unknown -> restarting
Ether0: AppleTalk state changed; restarting -> probing
%AT-6-ADDRUSED: Ether0: AppleTalk node up; using address 65401.148
Ether0: AppleTalk state changed; probing -> acquiring
AT: Sent GetNetInfo request broadcast on Ether0
AT: Sent GetNetInfo request broadcast on Ether0
AT: Sent GetNetInfo request broadcast on Ether0
AT: Sent GetNetInfo request broadcast on Ether0
AT: Sent GetNetInfo request broadcast on Ether0

When you attempt to bring up a nonseed router without a seed router on the wire, it never becomes operational; instead, it hangs in the acquiring mode and continues to send out periodic GetNetInfo requests.

When a nondiscovery-enabled router is brought up on an AppleTalk internetwork that is in compatibility mode (set up to accommodate extended as well as nonextended AppleTalk) and the router has violated internetwork compatibility:

The following three configuration command lines indicate the part of the configuration of the router that caused the configuration mismatch:

```
lestat (config)# interface ethernet 0
lestat (config-if)# apple cab 41-41
lestat (config-if)# apple zone Marketing
```

The router shown had been configured with a cable range of 41-41 instead of 40-40, which would have been accurate. Additionally, the zone name was configured incorrectly; it should have been "Marketing," rather than being misspelled as "Markting."
**debug apple nbp**

To display debugging output from the Name Binding Protocol (NBP) routines, use the `debug apple nbp` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

**debug apple nbp [type number]**

**no debug apple nbp [type number]**

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>type</code></td>
<td>(Optional) Interface type.</td>
</tr>
<tr>
<td><code>number</code></td>
<td>(Optional) Interface number.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC

**Usage Guidelines**

To determine whether the router is receiving NBP lookups from a node on the AppleTalk network, enable `debug apple nbp` at each node between the router and the node in question to determine where the problem lies.

**Caution**

Because the `debug apple nbp` command can generate many messages, use it only when the CPU utilization of the router is less than 50 percent.

**Examples**

The following is sample output from the `debug apple nbp` command:

```
Router# debug apple nbp
AT: NBP ctrl = LkUp, ntuples = 1, id = 77
AT: 4160.19, skt 2, enum 0, name: -=:ciscoRouter@Low End SW Lab
AT: NBP ctrl = LkUp->Reply, ntuples = 1, id = 77
AT: 4160.154, skt 254, enum 1, name: lestat.Ether0:ciscoRouter@Low End SW Lab
AT: NBP ctrl = LkUp, ntuples = 1, id = 78
AT: 4160.19, skt 2, enum 0, name: -=:IPADDRESS@Low End SW Lab
AT: 4160.19, skt 2, enum 0, name: -=:IPGATEWAY@Low End SW Lab
AT: 4160.19, skt 2, enum 0, name: -=:IPGATEWAY@Low End SW Lab
AT: NBP ctrl = LkUp, ntuples = 1, id = 84
AT: 4160.19, skt 2, enum 0, name: -=:IPADDRESS@Low End SW Lab
AT: NBP ctrl = LkUp, ntuples = 1, id = 85
AT: 4160.19, skt 2, enum 0, name: -=:IPGATEWAY@Low End SW Lab
AT: NBP ctrl = LkUp, ntuples = 1, id = 85
AT: 4160.19, skt 2, enum 0, name: -=:IPGATEWAY@Low End SW Lab
The first three lines describe an NBP lookup request:

AT: NBP ctrl = LkUp, ntuples = 1, id = 77
AT: 4160.19, skt 2, enum 0, name: -=:ciscoRouter@Low End SW Lab
AT: LkUp -=:ciscoRouter@Low End SW Lab
```
The table below describes the fields in the first line of output.

**Table 6: debug apple nbp Field Descriptions—First Line of Output**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT: NBP</td>
<td>Indicates that this message describes an AppleTalk NBP packet.</td>
</tr>
<tr>
<td>ctrl = LkUp</td>
<td>Identifies the type of NBP packet. Possible values are as follows:</td>
</tr>
<tr>
<td></td>
<td>• LkUp--NBP lookup request.</td>
</tr>
<tr>
<td></td>
<td>• LkUp-Reply--NBP lookup reply.</td>
</tr>
<tr>
<td>ntuples = 1</td>
<td>Indicates the number of name-address pairs in the lookup request packet.</td>
</tr>
<tr>
<td></td>
<td>Range: 1 to 31 tuples.</td>
</tr>
<tr>
<td>id = 77</td>
<td>Identifies an NBP lookup request value.</td>
</tr>
</tbody>
</table>

The table below describes the fields in the second line of output.

**Table 7: debug apple nbp Field Descriptions—Second Line of Output**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT:</td>
<td>Indicates that this message describes an AppleTalk packet.</td>
</tr>
<tr>
<td>4160.19</td>
<td>Indicates the network address of the requester.</td>
</tr>
<tr>
<td>skt 2</td>
<td>Indicates the internet socket address of the requester. The responder will send the NBP lookup reply to this socket address.</td>
</tr>
<tr>
<td>enum 0</td>
<td>Indicates the enumerator field. Used to identify multiple names registered on a single socket. Each tuple is assigned its own enumerator, incrementing from 0 for the first tuple.</td>
</tr>
</tbody>
</table>
Field | Description
--- | ---
name: =:ciscoRouter@Low End SW Lab | Indicates the entity name for which a network address has been requested. The AppleTalk entity name includes three components:
  - Object (in this case, a wildcard character [=], indicating that the requester is requesting name-address pairs for all objects of the specified type in the specified zone).
  - Type (in this case, ciscoRouter).
  - Zone (in this case, Low End SW Lab).

The third line in the output essentially reiterates the information in the two lines above it, indicating that a lookup request has been made regarding name-address pairs for all objects of the ciscoRouter type in the Low End SW Lab zone.

Because the router is defined as an object of type ciscoRouter in zone Low End SW Lab, the router sends an NBP lookup reply in response to this NBP lookup request. The following two lines of output show the response of the router:

AT: NBP ctrl = LkUp-Reply, ntuples = 1, id = 77
AT: 4160.154, skt 254, enum 1, name: lestat.Ether0:ciscoRouter@Low End SW Lab

In the first line, ctrl = LkUp-Reply identifies this NBP packet as an NBP lookup request. The same value in the id field (id = 77) associates this lookup reply with the previous lookup request. The second line indicates that the network address associated with the entity name of the router (lestat.Ether0:ciscoRouter@Low End SW Lab) is 4160.154. The fact that no other entity name/network address is listed indicates that the responder only knows about itself as an object of type ciscoRouter in zone Low End SW Lab.
debug apple packet

To display per-packet debugging output, use the `debug apple packet` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug apple packet [type number]
no debug apple packet [type number]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>(Optional) Interface type.</td>
</tr>
<tr>
<td>number</td>
<td>(Optional) Interface number.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC

**Usage Guidelines**

With this command, you can monitor the types of packets being slow switched. It displays at least one line of debugging output per AppleTalk packet processed.

The output reports information online when a packet is received or a transmission is attempted.

When invoked in conjunction with the `debug appplerouting`, `debug applezip`, and `debug applenbp` commands, the `debug apple packet` command adds protocol processing information in addition to generic packet details. It also reports successful completion or failure information.

When invoked in conjunction with the `debug apple errors` command, the `debug apple packet` command reports packet-level problems, such as those concerning encapsulation.

⚠️ **Caution**

Because the `debug apple packet` command can generate many messages, use it only when the CPU utilization of the router is less than 50 percent.

**Examples**

The following is sample output from the `debug apple packet` command:

```
Router# debug apple packet
Ether0: AppleTalk packet: enctype SNAP, size 60, encaps000000000000000000000000
AT: src=Ethernet0:4160.47, dst=4160-4160, size=10, 2 rtes, RTMP pkt sent
AT: ZIP Extended reply rcvd from 4160.19
AT: ZIP Extended reply rcvd from 4160.19
Ether0: AppleTalk packet: enctype SNAP, size 60, encaps000000000000000000000000
Ether0: AppleTalk packet: enctype SNAP, size 60, encaps000000000000000000000000
```

The table below describes the fields in the first line of output.
Table 8: debug apple packet Field Descriptions--First Line of Output

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ether0:</td>
<td>Name of the interface through which the router received the packet.</td>
</tr>
<tr>
<td>AppleTalk packet</td>
<td>Indicates that this is an AppleTalk packet.</td>
</tr>
<tr>
<td>enctype SNAP</td>
<td>Encapsulation type for the packet.</td>
</tr>
<tr>
<td>size 60</td>
<td>Size of the packet (in bytes).</td>
</tr>
<tr>
<td>encaps00000000000000000000000</td>
<td>Encapsulation.</td>
</tr>
</tbody>
</table>

The table below describes the fields in the second line of output.

Table 9: debug apple packet Field Descriptions--Second Line of Output

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT:</td>
<td>Indicates that this is an AppleTalk packet.</td>
</tr>
<tr>
<td>src=Ethernet0:4160.47</td>
<td>Name of the interface sending the packet and its AppleTalk address.</td>
</tr>
<tr>
<td>dst=4160-4160</td>
<td>Cable range of the destination of the packet.</td>
</tr>
<tr>
<td>size=10</td>
<td>Size of the packet (in bytes.)</td>
</tr>
<tr>
<td>2 rtes</td>
<td>Indicates that two routes in the routing table link these two addresses.</td>
</tr>
<tr>
<td>RTMP pkt sent</td>
<td>Type of packet sent.</td>
</tr>
</tbody>
</table>

The third line indicates the type of packet received and its source AppleTalk address. This message is repeated in the fourth line because AppleTalk hosts can send multiple replies to a given GetNetInfo request.
debug apple remap

To enable debugging of the AppleTalk remap activities, use the `debug apppleremap` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```plaintext
debug apple remap
no debug apple remap
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

Use the `debug apppleremap` command with the `debug appledomain` command to observe activity between domains and subdomains. Messages from the `debug apppleremap` command are displayed when a particular remapping function occurs, such as creating remaps or deleting remaps.

**Examples**

The following is sample output from the `debug apppleremap` command intermixed with output from the `debug appledomain` command; the two commands show related events.

```plaintext
Router# debug apple remap
Router# debug apple domain
AT-REMAP: RemapProcess for net 30000 domain AURP Domain 1
AT-REMAP: ReshuffleRemapList for subdomain 1
AT-REMAP: Could not find a remap for cable 3000-3001
AT-DOMAIN: atdomain_DisablePort for Tunnel0
AT-DOMAIN: CleanUpDomain for domain 1 [AURP Domain 1]
AT-DOMAIN: Disabling interface Ethernet1
AT-DOMAIN: atdomain_DisablePort for Ethernet1
AT-DOMAIN: CleanUpDomain for domain 1 [AURP Domain 1]
AT-DOMAIN: atdomain_DisablePort for Ethernet1
AT-DOMAIN: CleanUpDomain for domain 1 [AURP Domain 1]
AT-DOMAIN: CleanSubDomain for inbound subdomain 1
AT-REMAP: Remap for net 70 inbound subdomain 1 has been deleted
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-DOMAIN: DeleteAVRemapList for inbound subdomain 1
AT-DOMAIN: DeleteRemapTable for outbound subdomain 1
AT-DOMAIN: DeleteAVRemapList for outbound subdomain 1
AT-REMAP: RemapProcess for net 30000 domain AURP Domain 1 Remaped Net 10000
AT-REMAP: Remap for net 50 outbound subdomain 1 has been deleted
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-DOMAIN: DeleteAVRemapList for outbound subdomain 1
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-DOMAIN: DeleteAVRemapList for inbound subdomain 1
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-DOMAIN: DeleteAVRemapList for outbound subdomain 1
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-DOMAIN: DeleteAVRemapList for inbound subdomain 1
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-DOMAIN: DeleteAVRemapList for outbound subdomain 1
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug apple domain</td>
<td>Enables debugging of the AppleTalk domain activities.</td>
</tr>
</tbody>
</table>
debug apple remap
debug apple routing

To enable debugging output from the Routing Table Maintenance Protocol (RTMP) routines, use the `debugapple routing` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug apple routing [type number]
no debug apple routing [type number]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>type</code></td>
<td>(Optional) Interface type.</td>
</tr>
<tr>
<td><code>number</code></td>
<td>(Optional) Interface number.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC

**Usage Guidelines**

This command can be used to monitor acquisition of routes, aging of routing table entries, and advertisement of known routes. It also reports conflicting network numbers on the same network if the network is misconfigured.

⚠️ **Caution**

Because the `debugapple routing` command can generate many messages, use it only when router CPU utilization is less than 50 percent.

**Examples**

The following is sample output from the `debugapple routing` command:

```
Router# debug apple routing
AT: src=Ethernet0:4160.41, dst=4160-4160, size=19, 2 rtes, RTMP pkt sent
AT: src=Ethernet1:41069.25, dst=41069, size=427, 96 rtes, RTMP pkt sent
AT: src=Ethernet2:4161.23, dst=4161-4161, size=427, 96 rtes, RTMP pkt sent
AT: Route ager starting (97 routes)
AT: Route ager finished (97 routes)
AT: RTMP from 4160.19 (new 0,old 94,bad 0,ign 0, dwn 0)
AT: RTMP from 4160.250 (new 0,old 0,bad 0,ign 2, dwn 0)
AT: RTMP from 4161.236 (new 0,old 94,bad 0,ign 1, dwn 0)
AT: src=Ethernet0:4160.41, dst=4160-4160, size=19, 2 rtes, RTMP pkt sent
```

The table below describes the fields in the first line of sample `debugapple routing` output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AT:</code></td>
<td>Indicates that this is AppleTalk debugging output.</td>
</tr>
<tr>
<td><code>src=Ethernet0:4160.41</code></td>
<td>Indicates the source router interface and network address for the RTMP update packet.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>dst=4160-4160</td>
<td>Indicates the destination network address for the RTMP update packet.</td>
</tr>
<tr>
<td>size=19</td>
<td>Displays the size of this RTMP packet (in bytes).</td>
</tr>
<tr>
<td>2 rtes</td>
<td>Indicates that this RTMP update packet includes information on two routes.</td>
</tr>
<tr>
<td>RTMP pkt sent</td>
<td>Indicates that this type of message describes an RTMP update packet that the router has sent (rather than one that it has received).</td>
</tr>
</tbody>
</table>

The following two messages indicate that the ager has started and finished the aging process for the routing table and that this table contains 97 entries:

AT: Route ager starting (97 routes)
AT: Route ager finished (97 routes)

The table below describes the fields in the following line of the `debug apple routing` command output:

AT: RTMP from 4160.19 (new 0, old 94, bad 0, ign 0, dwn 0)

**Table 11: debug apple routing Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT:</td>
<td>Indicates that this is AppleTalk debugging output.</td>
</tr>
<tr>
<td>RTMP from 4160.19</td>
<td>Indicates the source address of the RTMP update the router received.</td>
</tr>
<tr>
<td>new 0</td>
<td>Displays the number of routes in this RTMP update packet that the router did not already know about.</td>
</tr>
<tr>
<td>old 94</td>
<td>Displays the number of routes in this RTMP update packet that the router already knew about.</td>
</tr>
<tr>
<td>bad 0</td>
<td>Displays the number of routes the other router indicates have gone bad.</td>
</tr>
<tr>
<td>ign 0</td>
<td>Displays the number of routes the other router ignores.</td>
</tr>
<tr>
<td>dwn 0</td>
<td>Displays the number of poisoned tuples included in this packet.</td>
</tr>
</tbody>
</table>
**debug apple zip**

To display debugging output from the Zone Information Protocol (ZIP) routines, use the `debugapplezip` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

**debug apple zip [type number]**

**no debug apple zip [type number]**

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>type</code></td>
<td>(Optional) Interface type.</td>
</tr>
<tr>
<td><code>number</code></td>
<td>(Optional) Interface number.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC

**Usage Guidelines**

This command reports significant events such as the discovery of new zones and zone list queries. It generates information similar to that generated by the `debug apple routing` command, but generates it for ZIP packets instead of Routing Table Maintenance Protocol (RTMP) packets.

You can use the `debugapplezip` command to determine whether a ZIP storm is taking place in the AppleTalk network. You can detect the existence of a ZIP storm when you see that no router on a cable has the zone name corresponding to a network number that all the routers have in their routing tables.

**Examples**

The following is sample output from the `debugapplezip` command:

```
Router# debug apple zip
AT: Sent GetNetInfo request broadcast on Ether0
AT: Recvd ZIP cmd 6 from 4160.19-6
AT: 3 query packets sent to neighbor 4160.19
AT: 1 zones for 31902, ZIP XReply, src 4160.19
AT: net 31902, zonelen 10, name US-Florida
```

The first line indicates that the router has received an RTMP update that includes a new network number and is now requesting zone information:

```
AT: Sent GetNetInfo request broadcast on Ether0
```

The second line indicates that the neighbor at address 4160.19 replies to the zone request with a default zone:

```
AT: Recvd ZIP cmd 6 from 4160.19-6
```

The third line indicates that the router responds with three queries to the neighbor at network address 4160.19 for other zones on the network:

```
AT: 3 query packets sent to neighbor 4160.19
```

The fourth line indicates that the neighbor at network address 4160.19 responds with a ZIP extended reply, indicating that one zone has been assigned to network 31902:

```
AT: 1 zones for 31902, ZIP XReply, src 4160.19
```
The fifth line indicates that the router responds that the zone name of network 31902 is US-Florida, and the zone length of that zone name is 10:

AT: net 31902, zonelen 10, name US-Florida
debug appn all

To turn on all possible debugging messages for Advanced Peer-to-Peer Networking (APPN), use the debug appn all command in privileged EXEC mode. To disable debugging output, use the no form of this command.

debug appn all
no debug appn all

Syntax Description
This command has no arguments or keywords.

Command Modes
Privileged EXEC

Usage Guidelines
This command shows all APPN events. Use other forms of the debug appn command to display specific types of events.

⚠️ Caution
Because the debug appn all command can generate many messages and alter timing in the network node, use it only when instructed by authorized support personnel.

⚠️ Caution
Debugging output takes priority over other network traffic. The debug appn all command generates more output than any other debug appn command and can alter timing in the network node. This command can severely diminish router performance or even render it unusable. In virtually all cases, it is best to use specific debug appn commands.

Examples
Refer to the documentation for specific debug appn commands for examples and explanations.

Related Commands
Refer to the other forms of the debug appn command to enable specific debug output selectively.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug appn cs</td>
<td>Displays the APPN CS component activity.</td>
</tr>
<tr>
<td>debug appn ds</td>
<td>Displays debugging information on APPN DS component activity.</td>
</tr>
<tr>
<td>debug appn hpr</td>
<td>Displays information related to HPR code execution.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>debug appn ms</td>
<td>Displays debugging information on APPN MS component activity.</td>
</tr>
<tr>
<td>debug appn nof</td>
<td>Displays information on APPN NOF component activity.</td>
</tr>
<tr>
<td>debug appn pc</td>
<td>Displays debugging information on APPN PC component activity.</td>
</tr>
<tr>
<td>debug appn ps</td>
<td>Displays debugging information on APPN PS component activity.</td>
</tr>
<tr>
<td>debug appn scm</td>
<td>Displays debugging information on APPN SCM component activity.</td>
</tr>
<tr>
<td>debug appn ss</td>
<td>Displays SS events.</td>
</tr>
<tr>
<td>debug appn ss</td>
<td>Displays debugging information on APPN TRS component activity.</td>
</tr>
</tbody>
</table>
**debug appn cs**

To display Advanced Peer-to-Peer Networking (APPN) Configuration Services (CS) component activity, use the `debug appn cs` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug appn cs
no debug appn cs
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

The CS component is responsible for defining link stations, ports, and connection networks. It is responsible for the activation and deactivation of ports and link stations and handles status queries for these resources.

**Examples**

The following is sample output from the `debug appn cs` command. In this example a link station is being stopped.

```
Router# debug appn cs
Turned on event 008000FF
APPN: ------ CS ------ Deq STOP_LS message
APPN: ------ CS ------ FSM LS: 75 17 5 8
APPN: ------ CS ------ Sending DEACTIVATE_AS - station PATTY
APPN: ------ CS ------ deactivate_as_p->ips_header.lpid = A80A60
APPN: ------ CS ------ deactivate_as_p->ips_header.lpid = A80A60
APPN: ------ CS ------ Sending DESTROY_TG to PC - station PATTY - lpid=A80A60
APPN: ------ CS ------ Deq DESTROY_TG - station PATTY
APPN: ------ CS ------ FSM LS: 22 27 8 0
APPN: ------ CS ------ Sending TG update for LS PATTY to TRS
APPN: ------ CS ------ ENTERING XID_PROCESSING: 4
%APPN-6-APPNSENDMSG: Link Station PATTY stopped
```

The table below describes the significant fields and messages shown in the display.

**Table 12: debug appn cs Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPN</td>
<td>APPN debugging output.</td>
</tr>
<tr>
<td>CS</td>
<td>CS component output.</td>
</tr>
<tr>
<td>Deq</td>
<td>CS received a message from another component.</td>
</tr>
<tr>
<td>FSM LS</td>
<td>Link station finite state machine is being referenced.</td>
</tr>
<tr>
<td>Sending</td>
<td>CS is sending a message to another component.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>debug appn all</td>
<td>Turns on all possible debugging messages for APPN.</td>
</tr>
</tbody>
</table>
debug appn ds

To display debugging information on Advanced Peer-to-Peer Networking (APPN) Directory Services (DS) component activity, use the `debug appn ds` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
default
default

Syntax Description
This command has no arguments or keywords.

Command Modes
Privileged EXEC

Usage Guidelines
The DS component manages searches for resources in the APPN network. DS is also responsible for registration of resources within the network.

Examples
The following is sample output from the `debug appn ds` command. In this example a search has been received.

```
Router# debug appn ds
turned on event 080000FF
APPN: NEWDS: pcid: DD321E895667111
APPN: NEWDS: Invoking FSM NNSolu
APPN: NEWDS: Lsfsm_NNSolu: 00A67AA0 pcid: DD321E895667111 row: 0 col: 0 inp: 80200000
APPN: NEWDS: Lsfsm_parent: 00A89940 row: 0 col: 0 inp: 80000000
APPN: NEWDS: Rcvd a LMRQ
APPN: NEWDS: Lsfsm_NNSolu: 00A67AA0 pcid: DD321E895667111 row: 12 col: 1 inp: 40000000
APPN: NEWDS: Lsfsm_parent: 00A89940 row: 8 col: 1 inp: 40000000
APPN: NEWDS: Lsfsm_child: 00A89BEB row: 0 col: 0 inp: 80000080
APPN: NEWDS: Lsfsm: QUERY_ROUTE(RQ) to TRS
APPN: NEWDS: Lsfsm_child: 00A8A1C0 row: 1 col: 0 inp: 80000008
APPN: NEWDS: Lsfsm_NNSolu: 00A67AA0 pcid: DD321E895667111 row: 5 col: 1 inp: 80C04000
APPN: NEWDS: Lsfsm_child: 00A8A1C0 row: 7 col: 1 inp: 80840008
APPN: NEWDS: Rcvd a LMY
APPN: NEWDS: Lsfsm_NNSolu: 00A67AA0 pcid: DD321E895667111 row: 16 col: 6 inp: 40800000
APPN: NEWDS: Lsfsm_parent: 00A89940 row: 3 col: 1 inp: 80000000
APPN: NEWDS: send locate to node: NETA.PATTY
```

The table below describes the significant fields shown in the display.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPN</td>
<td>APPN debugging output.</td>
</tr>
<tr>
<td>NEWDS</td>
<td>DS component output.</td>
</tr>
<tr>
<td>search from</td>
<td>Locate was received from NETA.PATTY.</td>
</tr>
<tr>
<td>Lsfsm_</td>
<td>Locate Search finite state machine is being referenced.</td>
</tr>
</tbody>
</table>
### Field and Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PQenq</td>
<td>Message was sent to another component.</td>
</tr>
<tr>
<td>Rcvd</td>
<td>Message was received from another component.</td>
</tr>
<tr>
<td>send locate</td>
<td>Locate will be sent to NETA.PATTY.</td>
</tr>
</tbody>
</table>

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug appn all</td>
<td>Turns on all possible debugging messages for APPN.</td>
</tr>
</tbody>
</table>
debug appn hpr

To display debugging information related to High Performance Routing (HPR) code execution, use the `debug appn hpr` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Examples**

The following is sample output from the `debug appn hpr` command:

```
Router# debug appn hpr
APPN: -- ncl.ncl_map_dlc_type() -- mapping TOKEN_RING(4) to NCL_TR(3)
APPN: -- ncl.ncl_port() -- called with port_type:3, cisco_idb:893A14, hpr_ssap:C8
APPN: -- ncl.process_port_change() -- port coming up
APPN: -- ncl.process_port_change() -- PORT_UP
APPN: -- ncl.ncl_port fsm -- FSM Invoked: Input:0, State:0->1, Action:0
APPN: -- ncl.ncl_port fsm -- FSM Invoked: Input:1, State:1->2, Action:1
APPN: -- ncl.ncl_unmap_dlc_type() -- mapping NCL(3) to CLS(3)
APPN: ------ ANR ---- Sending ACTIVATE_SAP.req
APPN: -- cswncsnd.main() -- received LSA_IPS ips.
APPN: -- ncl.ncl_port fsm -- FSM Invoked: Input:3, State:2->3, Action:4
APPN: -- ncl.ncl_assign_anr() -- Assigned ANR,anr:8002
APPN: -- ncl.ncl_map_dlc_type() -- mapping TOKEN_RING(4) to NCL_TR(3)
APPN: -- ncl.ncl_populate_anr() -- anr:8002, dlc_type:3, idb 893A14
APPN: -- ncl.ncl_populate_anr() -- send anr_tbl_update to owning cswncsnd
APPN: -- ncl.ncl_ls fsm -- FSM Invoked: Input:0, State:0->1, Action:0
APPN: ncl.ncl_send_reqopn_stn_req
APPN: -- ncl.ncl_ls fsm() -- send ANR_TBL_UPDATE ips.
APPN: -- cswncsnd.main() -- received LSA_IPS ips.
APPN: -- ncl.ncl_unmap_dlc_type() -- mapping NCL(3) to CLS(3)
APPN: -- ncl.ncl_ls fsm() -- send ANR_TBL_UPDATE to owning cswncsnd
APPN: -- cswncsnd.main() -- received ANR_TBL_UPDATE ips.
APPN: -- cswncsndapply_anr_table_update() -- ANR:8002
APPN: -- cswncsnd.main() -- received LSA_IPS ips.
APPN: -- ncl.ncl_ls fsm() -- send ANR_TBL_UPDATE to owning cswncsnd
APPN: -- cswncsndapply_anr_table_update() -- ANR:8002
APPN: -- cswncsnd.main() -- received ANR_TBL_UPDATE ips.
APPN: -- cswncsndapply_anr_table_update() -- ANR:8002
APPN: rtpm: rtp_send() sent data over connection B9D5E8
APPN: hpr timer: rtt start time clocked at 135952 ms
APPN: -- cswncsnd.main() -- received NCLSENDMSG ips.
APPN: -- cswncsnd.process_nlp_from_rtp() -- label: 8002, send to p_cep 00AAF638.
APPN: hpr timer: rtt end time clocked at 135972 ms
APPN: hpr timer: round trip time measured at 20 ms
```

The table below describes the significant fields shown in the display.

**Table 14: debug appn hpr Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPN</td>
<td>APPN debugging output.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NCL</td>
<td>Network control layer debugging output. Network control layer is the component that handles ANR packets.</td>
</tr>
<tr>
<td>ncl_port_fsm</td>
<td>Network control layer port finite state machine has been invoked.</td>
</tr>
<tr>
<td>ncl_assign_anr</td>
<td>ANR label has been assigned to an activating link station.</td>
</tr>
<tr>
<td>ncl_populate_anr</td>
<td>System is updating the ANR record with information specific to the link station.</td>
</tr>
<tr>
<td>ncl_is_fsm</td>
<td>Network control layer link finite state machine has been invoked.</td>
</tr>
<tr>
<td>rtp_send</td>
<td>RTP is about to send a packet.</td>
</tr>
<tr>
<td>hpr timer</td>
<td>Debugging output related to an HPR timer.</td>
</tr>
<tr>
<td>rtt start time</td>
<td>RTP is measuring the round-trip time for an HPR status request packet. This is the start time.</td>
</tr>
<tr>
<td>NCL_SND_MSG</td>
<td>Network control layer has been requested to send a packet.</td>
</tr>
<tr>
<td>process_nlp_from_rtp</td>
<td>Network control layer has been requested by RTP to send a packet.</td>
</tr>
<tr>
<td>rtt end time</td>
<td>RTP is measuring the round-trip time for an HPR status request packet. This is the time.</td>
</tr>
<tr>
<td>round trip time</td>
<td>Round-trip time for this HPR status exchange has been computed.</td>
</tr>
</tbody>
</table>

## Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug appn all</td>
<td>Turns on all possible debugging messages for APPN.</td>
</tr>
</tbody>
</table>
debug appn ms

To display debugging information on Advanced Peer-to-Peer Networking (APPN) Management Services (MS) component activity, use the `debug appn ms` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

**debug appn ms**

**no debug appn ms**

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

**Command Modes**
Privileged EXEC

**Usage Guidelines**
The MS component is responsible for generating, sending, and forwarding network management information in the form of traps and alerts to a network management focal point, such as Netview, in the APPN network.

**Examples**
The following is sample output from the `debug appn ms` command. In this example an error occurred that caused an alert to be generated.

```
Router# debug appn ms
APPN: ----- MSS00 ---- Deq ALERT MSU msg
APPN: ----- MSF70 ---- ALERT MV FROM APPN WITH VALID LGTH
APPN: ----- MSFPL ---- Find Active FP
APPN: ----- MSP30 ---- Entering Build MS Transport
APPN: ----- MSP31 ---- Entering Building Routing Info.
APPN: ----- MSP34 ---- Entering Build GDS
APPN: ----- MSP32 ---- Entering Building UOW correlator
APPN: ----- MSP34 ---- Entering Build GDS
APPN: ----- MSP30 ---- Building GDS 0x1310
APPN: ----- MSP30 ---- Building MS Transport
APPN: ----- MSF72 ---- ACTIVE FP NOT FOUND, SAVE ONLY
APPN: ----- MSUFL ---- UOW <= 60, ALL COPIED in extract_uow
APPN: ----- MSCAT ---- by enq_cached_ms QUEUE SIZE OF QUEUE after enq 4
```

The table below describes the significant fields shown in the display.

**Table 15: debug appn ms Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPN</td>
<td>Indicates that this is APPN debugging output.</td>
</tr>
<tr>
<td>MSP</td>
<td>Indicates that this is MS component output.</td>
</tr>
</tbody>
</table>

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug appn all</code></td>
<td>Turns on all possible debugging messages for APPN.</td>
</tr>
</tbody>
</table>
debug appn ms

debug aaa accounting through debug auto-config
**debug appn nof**

To display debugging information on Advanced Peer-to-Peer Networking (APPN) Node Operator Facility (NOF) component activity, use the `debug appn nof` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug appn nof
no debug appn nof
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

The NOF component is responsible for processing commands entered by the user such as start, stop, show, and configuration commands. NOF forwards these commands to the proper component and waits for the response.

**Examples**

The following is sample output from the `debug appn nof` command. In this example, an APPN connection network is being defined.

```
Router# debug appn nof
Turned on event 010000FF
Router# config term
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# appn connection-network NETA.CISCO
Router(config-appn-cn)# port TR0
Router(config-appn-cn)# complete
router(config)#
APPN: ----- NOF ----- Define Connection Network Verb Received
APPN: ----- NOF ----- send define_cn_t ips to cs
APPN: ----- NOF ----- waiting for define_cn rsp from cs
router(config)#
```

The table below describes the significant fields shown in the display.

**Table 16: debug appn nof Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPN</td>
<td>APPN debugging output.</td>
</tr>
<tr>
<td>NOF</td>
<td>NOF component output.</td>
</tr>
<tr>
<td>Received</td>
<td>Configuration command was entered.</td>
</tr>
<tr>
<td>send</td>
<td>Message was sent to CS.</td>
</tr>
<tr>
<td>waiting</td>
<td>Response was expected from CS.</td>
</tr>
</tbody>
</table>
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug appn all</td>
<td>Turns on all possible debugging messages for APPN.</td>
</tr>
</tbody>
</table>
**debug appn pc**

To display debugging information on Advanced Peer-to-Peer Networking (APPN) Path Control (PC) component activity, use the `debug appn pc` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug appn pc
no debug appn pc
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

The PC component is responsible for passing Message Units (MUs) between the Data Link Control (DLC) layer and other APPN components. PC implements transmission priority by passing higher priority MUs to the DLC before lower priority MUs.

**Examples**

The following is sample output from the `debug appn pc` command. In this example an MU is received from the network.

```
Router# debug appn pc
Turned on event 040000FF
APPN: ----- PC-----PC Deq REMOTE mag variant_name 2251
APPN: --PC-- mu received to PC lpid: A80AEC
APPN: --PC-- mu received from p_cel_id: 67C6F8
APPN: ----- PC-----PC Deq LSA_IPS from DLC
APPN: --PCX dequeued a DATA.IND
APPN: --- PC processing DL_DATA.ind
APPN: --PC-- mu_error_checker with no error, calling frr
APPN: --PC-- calling frr for packet received on LFSID: 1 2 3
APPN: ----- PC-----PC is sending MU to SC A90396
APPN: ----- SC-----send mu: A90396, rpc: 0, nws: 7, rh.bl: 90
APPN: SC: Send mu.snf: 8, th.b0: 2E, rh.bl: 90, dcf: 8
```

The table below describes the significant fields shown in the display.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPN</td>
<td>APPN debugging output.</td>
</tr>
<tr>
<td>PC</td>
<td>PC component output.</td>
</tr>
<tr>
<td>Deq REMOTE</td>
<td>Message was received from the network.</td>
</tr>
<tr>
<td>mu received</td>
<td>Message is an MU.</td>
</tr>
<tr>
<td>DATA.IND</td>
<td>MU contains data.</td>
</tr>
</tbody>
</table>
MU is session traffic for an ISR session. The MU is forwarded to the Session Connector component for routing.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug appn all</td>
<td>Turns on all possible debugging messages for APPN.</td>
</tr>
</tbody>
</table>
**debug appn ps**

To display debugging information on Advanced Peer-to-Peer Networking (APPN) Presentation Services (PS) component activity, use the `debug appn ps` command in privileged EXEC mode. To disable debugging output, use the `no debug appn ps` form of this command.

```plaintext
debug appn ps
no debug appn ps
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

The PS component is responsible for managing the Transaction Programs (TPs) used by APPN. TPs are used for sending and receiving searches, receiving resource registration, and sending and receiving topology updates.

**Examples**

The following is sample output from the `debug appn ps` command. In this example a CP capabilities exchange is in progress.

```
Router# debug appn ps
Turned on event 200000FF
APPN: ----- CCA --- CP_CAPABILITIES_TP has started
APPN: ----- CCA --- About to wait for Partner to send CP_CAP
APPN: ----- CCA --- Partner LU name: NETA.PATTY
APPN: ----- CCA --- Mode Name: CPSVCMG
APPN: ----- CCA --- CGIID: 78
APPN: ----- CCA --- About to send cp_cp_session_act to SS
APPN: ----- CCA --- Waiting for cp_cp_session_act_rsp from SS
APPN: ----- CCA --- Received cp_cp_session_act_rsp from SS
APPN: ----- CCA --- About to send CP_CAP to partner
APPN: ----- CCA --- Send to partner completed with rc=0, 0
APPN: ----- RCA --- Allocating conversation
APPN: ----- RCA --- Sending CP_CAPABILITIES
APPN: ----- RCA --- Getting conversation attributes
APPN: ----- RCA --- Waiting for partner to send CP_CAPABILITIES
APPN: ----- RCA --- Normal processing complete with cgid = 82
APPN: ----- RCA --- Deallocating CP_Capabilities conversation
```

The table below describes the significant fields shown in the display.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPN</td>
<td>APPN debugging output.</td>
</tr>
<tr>
<td>CCA</td>
<td>CP Capabilities TP output.</td>
</tr>
<tr>
<td>RCA</td>
<td>Receive CP Capabilities TP output.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>debug appn all</td>
<td>Turns on all possible debugging messages for APPN.</td>
</tr>
</tbody>
</table>
**debug appn scm**

To display debugging information on Advanced Peer-to-Peer Networking (APPN) Session Connector Manager (SCM) component activity, use the `debug appn scm` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug appn scm
no debug appn scm
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

The SCM component is responsible for the activation and deactivation of the local resources that route an intermediate session through the router.

**Examples**

The following is sample output from the `debug appn scm` command. In this example an intermediate session traffic is being routed.

```
Router# debug appn scm
Turned on event 020000FF
Router# APPN: ----- SCM-----SCM Deq a MU
APPN: ----- SCM-----SCM send ISR_INIT to SSI
APPN: ----- SCM-----SCM (i05) Enter compare_fqpcid()
APPN: ----- SCM-----SCM Deq ISR_CINIT Message
APPN: ----- SCM-----SCM (i05) Enter compare_fqpcid()
APPN: ----- SCM-----SCM sends ASSIGN_LFSID to ASM
APPN: ----- SCM-----SCM Rcvd sync ASSIGN_LFSID from ASM
APPN: ----- SCM-----SCM PQenq a MU to ASM
APPN: ----- SCM-----SCM PQenq BIND rsp to ASM
```

The table below describes the significant fields shown in the display.

**Table 19: debug appn scm Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPN</td>
<td>APPN debugging output.</td>
</tr>
<tr>
<td>SCM</td>
<td>SCM component output.</td>
</tr>
</tbody>
</table>

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug appn all</td>
<td>Turns on all possible debugging messages for APPN.</td>
</tr>
</tbody>
</table>
debug appn scm
debug appn ss

To display session services (SS) events, use the `debug appn ss` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debg appn ss
no debug appn ss
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

The SS component generates unique session identifiers, activates and deactivates control point-to-control point (CP-CP) sessions, and assists logical units (LUs) in initiating and activating LU-LU sessions.

**Examples**

The following is sample output from the `debug appn ss` command. In this example CP-CP sessions between the router and another node are being activated.

```
Router# debug appn ss
Turned on event 100000ff
APPN: ----- SS ----- Deq ADJACENT_CP>Contacted message
APPN: ----- SS ----- Sending ADJACENT_NN_1015 to SCM, adj_node_p=A6B980, cp_name=NETA.PATTY
APPN: ----- SS ----- Sending REQUEST_LAST_FRSN message to TRS
APPN: ----- SS ----- Receiving REQUEST_LAST_FRSN_RSP from TRS
APPN: ----- SS ----- Sending ACTIVE_CP_STATUS CONLOSER message to DS
APPN: ----- SS ----- Sending ACTIVE_CP_STATUS CONLOSER message to MS
APPN: ----- SS ----- Sending CP_CP_SESSION ACT_RSP message to CCA TP
APPN: ----- SS ----- Sending PENDING_ACTIVE_CP_STATUS CONWINNER message to DS
APPN: ----- SS ----- Sending REQUEST_LAST_FRSN message to TRS
APPN: ----- SS ----- Receiving REQUEST_LAST_FRSN_RSP from TRS
APPN: ----- SS ----- Sending ACTIVE_CP_STATUS CONWINNER message to DS
APPN: ----- SS ----- Sending ACTIVE_CP_STATUS CONWINNER message to MS
APPN: ----- SS ----- Sending ACTIVE_CP_STATUS CONWINNER message to TRS
APPN: ----- SS ----- Sending CP_CP_SESSION ACT_RSP message to CCA TP
APPN: ----- SS ----- Sending PENDING_ACTIVE_CP_STATUS CONWINNER message to DS
APPN: ----- SS ----- Sending REQUEST_LAST_FRSN message to TRS
APPN: ----- SS ----- Receiving REQUEST_LAST_FRSN_RSP from TRS
APPN: ----- SS ----- Sending ACTIVE_CP_STATUS CONWINNER message to DS
APPN: ----- SS ----- Sending ACTIVE_CP_STATUS CONWINNER message to MS
APPN: ----- SS ----- Sending ACTIVE_CP_STATUS CONWINNER message to TRS
APPN: ----- SS ----- Sending ASSIGN_PCID message
APPN: ----- SS ----- Sending ASSIGN_PCID_RSP message to someone
APPN: ----- SS ----- Sending INIT_SIGNAL message
APPN: ----- SS ----- Sending REQUEST_COS_TPF_VECTOR message to TRS
APPN: ----- SS ----- Receiving an REQUEST_COS_TPF_VECTOR_RSP from TRS
APPN: ----- SS ----- Sending REQUEST_SINGLE_HOP_ROUTE message to TRS
APPN: ----- SS ----- Receiving an REQUEST_SINGLE_HOP_ROUTE_RSP from TRS
APPN: ----- SS ----- Sending ACTIVATE_ROUTE message to CS
APPN: ----- SS ----- Sending ACTIVATE_ROUTE_RSP message
APPN: ----- SS ----- Sending CINIT_SIGNAL message to SM
APPN: ----- SS ----- Sending ACT_CP_CP_SESSION_RSP message
APPN: ----- SS ----- Sending CINIT_SIGNAL message to SM
APPN: ----- SS ----- Sending ACT_CP_CP_SESSION_RSP message
APPN: ----- SS ----- Sending CINIT_SIGNAL message to SM
APPN: ----- SS ----- Sending ACT_CP_CP_SESSION_RSP message
```

The table below describes the significant fields shown in the display.
Table 20: debug appn ss Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPN</td>
<td>APPN debugging output.</td>
</tr>
<tr>
<td>SS</td>
<td>SS component output.</td>
</tr>
</tbody>
</table>

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug appn all</td>
<td>Turns on all possible debugging messages for APPN.</td>
</tr>
</tbody>
</table>
debug appn trs

To display debugging information on Advanced Peer-to-Peer Networking (APPN) Topology and Routing Services (TRS) component activity, use the **debug appn trs** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug appn trs**
**no debug appn trs**

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

**Command Modes**
Privileged EXEC

**Usage Guidelines**
The TRS component is responsible for creating and maintaining the topology database, creating and maintaining the class of service database, and computing and caching optimal routes through the network.

**Examples**
The following is sample output from the **debug appn trs** command:

```
Router# debug appn trs
Turned on event 400000FF
APPN: ----- TRS ----- Received a QUERY_CPNAME
APPN: ----- TRS ----- Received a REQUEST_ROUTE
APPN: ----- TRS ----- check_node node_name=NETA.LISA
APPN: ----- TRS ----- check_node node_index=0
APPN: ----- TRS ----- check_node node_weight=60
APPN: ----- TRS ----- add index 484 to origin description list
APPN: ----- TRS ----- add index 0 to dest description list
APPN: ----- TRS ----- origin tg_vector is NULL
APPN: ----- TRS ----- weight_to_origin = 0
APPN: ----- TRS ----- weight_to_dest = 0
APPN: ----- TRS ----- u_b_s_f weight = 30
APPN: ----- TRS ----- u_b_s_f prev_weight = 2147483647
APPN: ----- TRS ----- u_b_s_f origin_index = 484
APPN: ----- TRS ----- u_b_s_f dest_index = 0
APPN: ----- TRS ----- b_r_s_f weight = 30
APPN: ----- TRS ----- b_r_s_f origin_index = 484
APPN: ----- TRS ----- b_r_s_f dest_index = 0
APPN: ----- TRS ----- Received a REQUEST_ROUTE
APPN: ----- TRS ----- check_node node_name=NETA.LISA
APPN: ----- TRS ----- check_node node_index=484
APPN: ----- TRS ----- check_node node_weight=60
APPN: ----- TRS ----- add index 484 to origin description list
APPN: ----- TRS ----- origin_tg_weight to non-VN=30
APPN: ----- TRS ----- origin_node_weight to non-VN=60
APPN: ----- TRS ----- weight_to_origin = 90
APPN: ----- TRS ----- weight_to_dest = 0
APPN: ----- TRS ----- u_b_s_f weight = 120
APPN: ----- TRS ----- u_b_s_f prev_weight = 2147483647
APPN: ----- TRS ----- u_b_s_f origin_index = 484
APPN: ----- TRS ----- u_b_s_f dest_index = 0
APPN: ----- TRS ----- b_r_s_f weight = 120
APPN: ----- TRS ----- b_r_s_f origin_index = 484
APPN: ----- TRS ----- b_r_s_f dest_index = 0
```
The table below describes the significant fields shown in the display.

**Table 21: debug appn trs Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPN</td>
<td>APPN debugging output.</td>
</tr>
<tr>
<td>TRS</td>
<td>TRS component output.</td>
</tr>
</tbody>
</table>
**debug arap**

To display AppleTalk Remote Access Protocol (ARAP) events, use the `debug arap` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug arap {internal|memory|mnp4|v42bis} [linenum [aux|console|tty|vty]]
no debug arap {internal|memory|mnp4|v42bis} [linenum [aux|console|tty|vty]]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>internal</td>
<td>Debugs internal ARA packets.</td>
</tr>
<tr>
<td>memory</td>
<td>Debugs memory allocation for ARA.</td>
</tr>
<tr>
<td>mnp4</td>
<td>Debugs low-level asynchronous serial protocol.</td>
</tr>
<tr>
<td>v42bis</td>
<td>Debugs V.42bis compression.</td>
</tr>
<tr>
<td>linenum</td>
<td>(Optional) Line number. The number ranges from 0 to 999, depending on what type of line is selected.</td>
</tr>
<tr>
<td>aux</td>
<td>(Optional) Auxiliary line.</td>
</tr>
<tr>
<td>console</td>
<td>(Optional) Primary terminal line.</td>
</tr>
<tr>
<td>tty</td>
<td>(Optional) Physical terminal asynchronous line.</td>
</tr>
<tr>
<td>vty</td>
<td>(Optional) Virtual terminal line.</td>
</tr>
</tbody>
</table>

**Command Modes** Privileged EXEC

**Usage Guidelines**

Use the `debug arap` command with the `debug callback` command on access servers to debug dial-in and callback events.

Use the `debug modem` command to help catch problems related to ARAP autodetection (that is, `autoselect arap`). These problems are very common and are most often caused by modems, which are the most common cause of failure in ARAP connection and configuration sessions.

**Examples**

The following is sample output from the `debug arap internal` command:

```
Router# debug arap internal
ARAP: --------- SRVRVERSION ---------
ARAP: --------- ACKing 0 ---------
ARAP: --------- AUTH_CHALLENGE ---------
arapsec_local_account setting up callback
ARAP: --------- ACKing 1 ---------
ARAP: --------- AUTH_RESPONSE ---------
```
arap_startup initiating callback ARAP 2.0
ARAP: -------- CALLBACK --------
TTY7 Callback process initiated, user: dialback dialstring 40
TTY7 Callback forced wait = 4 seconds
TTY7 ARAP Callback Successful - await exec/autoselect pickup
TTY7: Callback in effect
ARAP: ------------ STARTINFOFROMSERVER ------------
ARAP: ------------ ACKing 0 ------------
ARAP: ------------ ZONELISTINFO ------------
ARAP: ------------ ZONELISTINFO ------------
ARAP: ------------ ZONELISTINFO ------------
ARAP: ------------ ZONELISTINFO ------------

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug callback</td>
<td>Displays callback events when the router is using a modem and a chat script to call back on a terminal line.</td>
</tr>
<tr>
<td>debug modem</td>
<td>Observes modem line activity on an access server.</td>
</tr>
</tbody>
</table>
debug archive config timestamp

To enable debugging of the processing time for each integral step of a configuration replace operation and the size of the configuration files being handled, use the debug archive config timestamp command in privileged EXEC mode. To disable debugging output, use the no form of this command.

debug archive config timestamp
no debug archive config timestamp

Syntax Description
This command has no arguments or keywords.

Command Modes
Privileged EXEC (#)

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(7)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(25)S</td>
<td>This command was integrated into Cisco IOS Release 12.2(25)S.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(28)SB.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2(31)SB2</td>
<td>This command was implemented on the Cisco 10000 series.</td>
</tr>
<tr>
<td>12.2(33)SXH</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SXH.</td>
</tr>
<tr>
<td>12.2(33)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SB and implemented on the Cisco 10000 series.</td>
</tr>
</tbody>
</table>

Cisco IOS XE Release 3.9S | This command was integrated into Cisco IOS XE Release 3.9S.

Examples
The following is sample output from the debug archive config timestamp command:

```
Device# debug archive config timestamp
Device# configure replace disk0:myconfig force
Timing: Debug Statistics for IOS Config Replace operation:
  Time to read file slot0:sample_2.cfg = 0 msec (0 sec)
  Number of lines read:55
  Size of file :1054
Starting Pass 1
  Time to read file system:running-config = 0 msec (0 sec)
  Number of lines read:93
  Size of file :2539
  Time taken for positive rollback pass = 320 msec (0 sec)
  Time taken for negative rollback pass = 0 msec (0 sec)
  Time taken for negative incremental diffs pass = 59 msec (0 sec)
  Time taken by PI to apply changes = 0 msec (0 sec)
  Time taken for Pass 1 = 380 msec (0 sec)
```
Starting Pass 2
  Time to read file system:running-config = 0 msec (0 sec)
  Number of lines read: 55
  Size of file : 1054
  Time taken for positive rollback pass = 0 msec (0 sec)
  Time taken for negative rollback pass = 0 msec (0 sec)
  Time taken for Pass 2 = 0 msec (0 sec)
Total number of passes: 1
Rollback Done

## Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug archive versioning</td>
<td>Enables debugging of the Cisco configuration archive activities.</td>
</tr>
</tbody>
</table>
**debug archive log config persistent**

To turn on debugging of configuration logging persistent events and display the results, use the `debug archive log config persistent` command in privileged EXEC mode. To disable the debugging and display of the archive events, use the `no` form of this command.

**debug archive log config persistent**

**no debug archive log config persistent**

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

If this command is not entered, there is no debugging or display of the configuration logging persistent events in the archive.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SRA</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(11)T.</td>
</tr>
<tr>
<td>12.2(33)SXH</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SXH.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
<tr>
<td>12.2(33)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SB.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The configuration logger feature must be enabled in order for the debug capability to work.

**Examples**

The following example turns on the debugging feature and displays the configuration logging persistent events:

```
Router# debug archive log config persistent
Router# archive log config persistent save
Configuration logging persistent save triggered. Saving the config log to disk0:IOS-Config-Logger-database'.
Command 'interface eth0' saved
Command 'ip address 10.1.1.1 255.255.255.0' saved
Command 'no shut' saved
Router#
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>archive log config persistent save</td>
<td>Saves the persisted commands in the configuration log to the Cisco IOS secure file system.</td>
</tr>
</tbody>
</table>
debug archive versioning

To enable debugging of the Cisco configuration archive activities, use the `debug archive versioning` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug archive versioning
no debug archive versioning
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(7)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(25)S</td>
<td>This command was integrated into Cisco IOS Release 12.2(25)S.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(28)SB.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2(31)SB2</td>
<td>This command was implemented on the Cisco 10000 series.</td>
</tr>
<tr>
<td>12.2(33)SXH</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SXH.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
<tr>
<td>12.2(33)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SB and implemented on the Cisco 10000 series.</td>
</tr>
<tr>
<td>12.2(33)SRE</td>
<td>This command was modified. It was integrated into Cisco IOS Release 12.2(33)SRE.</td>
</tr>
<tr>
<td>Cisco IOS XE Release 3.9S</td>
<td>This command was integrated into Cisco IOS XE Release 3.9S.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug archive versioning` command:

```
Device# debug archive versioning
Jan 9 06:46:28.419:backup_running_config
Jan 9 06:46:28.419:Current = 7
Jan 9 06:46:28.443:Writing backup file disk0:myconfig-7
Jan 9 06:46:29.547: backup worked
```
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug archive config timestamp</td>
<td>Enables debugging of the processing time for each integral step of a configuration replace operation and the size of the configuration files being handled.</td>
</tr>
</tbody>
</table>
debug arp

To enable debugging output for Address Resolution Protocol (ARP) transactions, use the `debug arp` command in privileged EXEC mode. To disable debugging, use the `no` form of this command.

`debug arp [vrf vrf-name] [global] [arp-entry-event] [arp-table-event] ha [interface-interaction]`

`no debug arp [vrf vrf-name] [global] [arp-entry-event] [arp-table-event] ha [interface-interaction]`

### Syntax Description

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vrf vrf-name</td>
<td>(Optional) Enables debug trace for a specific VPN routing and forwarding (VRF) instance.</td>
</tr>
<tr>
<td>global</td>
<td>(Optional) Enables global VRF debugging.</td>
</tr>
<tr>
<td>arp-entry-event</td>
<td>(Optional) ARP entry events for which a debug trace is enabled. Keywords are as follows:</td>
</tr>
<tr>
<td></td>
<td>• dynamic—Enables debugging output for dynamic ARP entry events.</td>
</tr>
<tr>
<td></td>
<td>• interface—Enables debugging output for interface ARP entry events.</td>
</tr>
<tr>
<td></td>
<td>• static—Enables debugging output for static ARP entry events.</td>
</tr>
<tr>
<td></td>
<td>• subblocking—Enables debugging output for ARP subblocking events.</td>
</tr>
<tr>
<td>arp-table-event</td>
<td>(Optional) ARP entry events for which a debug trace is enabled. Keywords are as follows:</td>
</tr>
<tr>
<td></td>
<td>• database—Enables debugging output for ARP database operations.</td>
</tr>
<tr>
<td></td>
<td>• table—Enables debugging output for ARP table operations.</td>
</tr>
<tr>
<td></td>
<td>• timer—Enables debugging output for ARP timer operations.</td>
</tr>
<tr>
<td>ha</td>
<td>(Optional) Enables debug trace for ARP high availability (HA) events.</td>
</tr>
</tbody>
</table>

**Note**: This keyword is available only on HA-capable platforms (that is, Cisco networking devices that support dual Route Processors [RPs]).
**interface-interaction**

(Optional) ARP interface interaction for which a debug trace is enabled. Keywords are as follows:

- **adjacency**—Enables debugging output for ARP interface events and Cisco Express Forwarding adjacency interface events.
- **application**—Enables debugging output for ARP application interface events.

**Command Default**

Debugging output is disabled for ARP transactions. To enable ARP packet debugging, use this command without a keyword.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.3</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(11)T</td>
<td>The following keyword were added: adjacency, application, dynamic, ha, interface, static, subblocking, table, and timer.</td>
</tr>
<tr>
<td>12.2(33)SRE</td>
<td>This command was modified. The database keyword was added.</td>
</tr>
<tr>
<td>15.1(1)SY</td>
<td>This command was modified. The vrf vrf-name keyword-argument pair and global keyword were added.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The debugging information shows whether the device is sending ARP packets and receiving ARP packets. Use this command when only some nodes on a TCP/IP network are responding.

The amount of debug information displayed is filtered based on an interface, an access list, or both, as specified by the debug list command.

To list the debugging options enabled on this device, use the show debugging command.

**Examples**

The following example shows how to enable ARP packet debugging filtered on ARP table entries for the host at 192.0.2.10:

```
Device(config)# access-list 10 permit host 192.0.2.10
Device(config)# exit
Device# debug list 10
Device# debug arp

ARP packet debugging is on
for access list: 10
```
The following is sample output from the `debug arp` command:

IP ARP: sent req src 192.0.2.7 0000.0c01.e117, dst 192.0.2.96 0000.0000.0000
IP ARP: rcvd rep src 192.0.2.96 0800.2010.b908, dst 192.0.2.7
IP ARP: rcvd req src 172.16.6.10 0000.0c00.6fa2, dst 192.0.2.62
IP ARP: rep filtered src 192.0.2.7 aa92.1b36.a456, dst 255.255.255.255 ffff.ffff.ffff
IP ARP: rep filtered src 192.0.2.240 0000.0c00.6b31, dst 192.0.2.7 0800.2010.b908

In the output, each line of output represents an ARP packet that the device has sent or received. Explanations for the individual lines of output follow.

The first line indicates that the device at IP address 192.0.2.7 and with the MAC address 0000.0c01.e117 sent an ARP request for the MAC address of the host at 192.0.2.96. The series of zeros (0000.0000.0000) following this address indicate that the device is currently unaware of the MAC address.

IP ARP: sent req src 192.0.2.7 0000.0c01.e117, dst 192.0.2.96 0000.0000.0000

The second line indicates that the device at IP address 192.0.2.7 receives a reply from the host at 192.0.2.96 indicating that the host MAC address is 0800.2010.b908:

IP ARP: rcvd rep src 192.0.2.96 0800.2010.b908, dst 192.0.2.7

The third line indicates that the device receives an ARP request from the host at 172.16.6.10 requesting the MAC address for the host at 192.0.2.62:

IP ARP: rcvd req src 172.16.6.10 0000.0c00.6fa2, dst 192.0.2.62

The fourth line indicates that another host on the network attempted to send the device an ARP reply for its own address. The device ignores meaningless replies. Usually, meaningless replies happen if a bridge is being run in parallel with the device and is allowing ARP to be bridged. This condition indicates a network misconfiguration.

IP ARP: rep filtered src 192.0.2.7 aa92.1b36.a456, dst 255.255.255.255 ffff.ffff.ffff

The fifth line indicates that another host on the network attempted to inform the device that it is on network 192.0.2.240, but the device does not know that the network is attached to a different device interface. The remote host (probably a PC or an X terminal) is misconfigured. If the device were to install this entry, the device would deny service to the real machine on the proper cable.

IP ARP: rep filtered src 192.0.2.240 0000.0c00.6b31, dst 192.0.2.7 0800.2010.b908

The following example shows that debugging information related to vpn1 on Ethernet interface 0/0 will be logged:

Device> enable
Device# configure terminal
Device(config)# interface ethernet0/0
Device(config-if)# vrf forwarding vpn1
Device(config-if)# end
Device# debug arp vrf vpn1

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access-list (extended-ibm)</td>
<td>Configures the extended access list mechanism for filtering frames by both source and destination addresses and arbitrary bytes in the packet.</td>
</tr>
<tr>
<td>debug list</td>
<td>Enables filtering of debug trace on a per-interface or per-access list basis.</td>
</tr>
<tr>
<td>show debugging</td>
<td>Lists the debugging options enabled on this device.</td>
</tr>
</tbody>
</table>
debug arpa
**debug ase**

**Note**
Effective with Cisco IOS Release 12.4(24), the `debug ase` command is not available in Cisco IOS software.

To gather Automatic Signature Extraction (ASE) error, log, messaging, reporting, status, and timer information, use the `debug ase` command in privileged EXEC mode. To disable error, log, messaging, reporting, status, and timer information, use the `no` form of this command.

```
debug ase {errors| log| messages| reports| status| timing}
no debug ase {errors| log| messages| reports| status| timing}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>errors</code></td>
<td>Displays ASE error information.</td>
</tr>
<tr>
<td><code>log</code></td>
<td>Displays ASE logging information.</td>
</tr>
<tr>
<td><code>messages</code></td>
<td>Displays ASE messaging information.</td>
</tr>
<tr>
<td><code>reports</code></td>
<td>Displays ASE reports.</td>
</tr>
<tr>
<td><code>status</code></td>
<td>Displays ASE status information.</td>
</tr>
<tr>
<td><code>timing</code></td>
<td>Displays ASE timer information.</td>
</tr>
</tbody>
</table>

**Command Default**

Disabled

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(15)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(24)</td>
<td>This command was removed.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

This command is used on the Cisco 1800, 2800, and 7200 series routers, Cisco 7301 router, and Integrated Services Routers (ISRs) as ASE sensors.
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ase collector</td>
<td>Enters the ASE collector server IP address so that the ASE sensor has IP connectivity to the ASE collector.</td>
</tr>
<tr>
<td>ase enable</td>
<td>Enables the ASE feature on a specified interface.</td>
</tr>
<tr>
<td>ase group</td>
<td>Identifies the TIDP group number for the ASE feature.</td>
</tr>
<tr>
<td>ase signature extraction</td>
<td>Enables the ASE feature globally on the router.</td>
</tr>
<tr>
<td>clear ase signature</td>
<td>Clears ASE signatures that were detected on the router.</td>
</tr>
<tr>
<td>show ase</td>
<td>Displays the ASE run-time status, which includes the TIDP group number.</td>
</tr>
</tbody>
</table>
debug asnl events

To trace event logs in the Application Subscribe Notify Layer (ASNL), use the debug asnl events command in privileged EXEC mode. To disable debugging output, use the no form of this command.

**debug asnl events**

**no debug asnl events**

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

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(4)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
This command traces the event logs in the ASNL, which serves as the interface layer between the application and protocol stacks. Event logs are generated during normal subscription processing, when the application responds to the notification request and when the session history table is updated.

**Examples**
The following example shows the ASNL subscription table being generated and the associated subscription timers as the application responds to the subscription request. The response timer is started to determine if the application responds to the notification request. If the application that made the subscription does not respond to the notification request within 5 seconds, the system automatically removes the subscription. The session-history-record deletion timer is also started. When the timer expires, the history record is removed from the active subscription table.

```
Router# debug asnl events
Application Subscribe Notify Layer Events debugging is on
*May 4 06:26:19.091://-1//ASNL:SUB-1:/asnl_process_is_up:Creating subscription table
*May 4 06:26:19.615://5//ASNL:SUB1:/asnl_start_timer:timer (0x63146C44)starts - delay (5000)
*May 4 06:26:19.619://-1//ASNL:SUB1:/asnl_stop_timer:timer(0x63146C44) stops
*May 4 06:26:19.619://-1//ASNL:SUB1:/asnl_notify_ack:ret=0x0c5300-5#
*May 4 06:26:24.631://5//ASNL:SUB1:/asnl_start_timer:timer (0x63146C44)starts - delay (5000)
*May 4 06:26:24.633://-1//ASNL:SUB1:/asnl_stop_timer:timer(0x63146C44) stops
*May 4 06:26:24.635://-1//ASNL:SUB1:/asnl_notify_ack:ret=0x0c5300-5#
*May 4 06:26:29.647://5//ASNL:SUB1:/asnl_start_timer:timer (0x63146C44)starts - delay (5000)
*May 4 06:26:29.647://-1//ASNL:SUB1:/asnl_stop_timer:timer(0x63146C44) stops
```
Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear subscription</td>
<td>Clears all active subscriptions or a specific subscription.</td>
</tr>
<tr>
<td>show subscription</td>
<td>Displays information about ASNL-based and non-ASNL-based SIP subscriptions.</td>
</tr>
<tr>
<td>subscription asnl session history</td>
<td>Specifies how long to keep ASNL subscription history records and how many history records to keep in memory.</td>
</tr>
</tbody>
</table>
debug asp packet

To display information on all asynchronous security protocols (ASPs) operating on the router, use the `debug asp packet` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
display asp packet
no display asp packet
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

The router uses asynchronous security protocols from companies including ADT Security Systems, Inc., Adplex, and Diebold to transport alarm blocks between two devices (such as a security alarm system console and an alarm panel). The alarm blocks are transported in pass-through mode using BSTUN encapsulation.

**Examples**

The following is partial sample output from the `debug asp packet` command for asynchronous security protocols when packet debugging is enabled on an asynchronous line carrying Diebold alarm traffic. In this example, two polls are sent from the Diebold alarm console to two alarm panels that are multidropped from a single E1A/TIA-232 interface. The alarm panels have device addresses F0 and F1. The example trace indicates that F1 is responding and F0 is not responding. At this point, you need to examine the physical link and possibly use a datascop to determine why the device is not responding.

```
Router# debug asp packet
12:19:48: ASP: Serial5: ADI-Rx: Data (4 bytes): F1FF4C42
12:20:48: ASP: Serial5: ADI-Rx: Data (4 bytes): F0FFB1BE
```

The table below describes the significant fields shown in the display.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASP</td>
<td>Asynchronous security protocol packet.</td>
</tr>
<tr>
<td>Serial5</td>
<td>Interface receiving and sending the packet.</td>
</tr>
<tr>
<td>ADI-Rx</td>
<td>Packet is being received.</td>
</tr>
<tr>
<td>ADI-T</td>
<td>Packet is being sent.</td>
</tr>
</tbody>
</table>
### debug asp packet

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data ($n$ bytes)</td>
<td>Type and size of the packet.</td>
</tr>
<tr>
<td>F1FF4c42</td>
<td>Alarm panel device address.</td>
</tr>
</tbody>
</table>
debug aspp event

To display asynchronous point of sale (APOS) event debug messages, use the debug aspp event command in privileged EXEC mode. To disable debugging output, use the no form of this command.

**debug aspp event**

**no debug aspp event**

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Disabled

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(2)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The debug aspp event command should be used with the debug aspp packet command to display all available details of the APOS call flow.

**Examples**

The following is sample output from the debug aspp event command for a simple transaction:

```
Router# debug aspp event
ASPP event debugging is on
Router# aspp APOS: Serial10/1: Serial HayesAT: state = DISCONNECTED
ASPP APOS: Serial10/1: Received HayesAT DIAL: state = DISCONNECTED
ASPP APIP: Serial10/1: Serial ENABLE: state = CONNECTING
ASPP APIP: Serial10/1: Network ENABLE: state = CONNECTING
ASPP APOS: Serial10/1: Send HayesAT CONNECT 9600: state = CONNECTED
ASPP APOS: Serial10/1: Response timer expired: state = CONNECTED
ASPP APOS: Serial10/1: Response timer expired: state = CONNECTED
ASPP APOS: Serial10/1: Serial DATA: state = CONNECTED
ASPP APIP: Serial10/1: Serial DATA: state = CONNECTED
ASPP APIP: Serial10/1: Network DATA: state = CONNECTED
ASPP APOS: Serial10/1: Serial ACK: state = CONNECTED
ASPP APOS: Serial10/1: Disconnect timer expired: state = DISCONNECT WAIT
ASPP APIP: Serial10/1: Serial DISABLE: state = DISCONNECTING
ASPP APIP: Serial10/1: Network DISABLE: state = DISCONNECTING
The table below describes the significant fields shown in the display.
```

**Table 23: debug aspp event Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial ENABLE:</td>
<td>Enable event received from the serial interface.</td>
</tr>
</tbody>
</table>
### Field | Description
--- | ---
Network ENABLE: | Enable event received from the network.
Send HayesAT CONNECT | Interpreted version of the Hayes AT command that is sent to the serial interface.
Response timer expired | The response timer has expired.
Serial DATA: | Data received from the serial interface.
Network DATA: | Data received from the network.
Disconnect timer expired | Hayes AT event is received by the serial interface.
Serial ACK: | Acknowledgment received from the serial interface.
Serial DISABLE: | Disable event received from the serial interface.
Network DISABLE: | Disable event received from the network.

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug aspp packet</td>
<td>Displays APOS packet debug messages.</td>
</tr>
</tbody>
</table>
**debug aspp packet**

To display asynchronous point of sale (APOS) packet debug messages, use the `debug aspp packet` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug aspp packet
no debug aspp packet
```

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

**Command Default**
Disabled

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(2)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
The `debug aspp packet` command should be used with the `debug aspp event` command to display all available details of the APOS call flow.

**Examples**
The following is sample output from the `debug aspp packet` command for a simple transaction:

```
Router# debug aspp packet
ASPP event debugging is on
Router#
ASPP:Serial1/7:ADI-rx:Data (14 bytes): 415456302644325331313D35300D
ASPP:Serial1/7:ADI-tx:Data (2 bytes): 300D
ASPP:Serial1/7:ADI-rx:Data (27 bytes): 4154583453393D3153373D323444543138303039
ASPP:Serial1/7:ADI-tx:Data (3 bytes): 31320D
ASPP:Serial1/7:ADI-tx:Data (1 bytes): 05
ASPP:Serial1/7:ADI-rx:Data (5 bytes): 0212340325
ASPP:Serial1/7:ADI-rx:Data (1 bytes): 06
ASPP:Serial1/7:ADI-tx:Data (1 bytes): 04
```

The table below describes the significant fields shown in the display.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASPP</td>
<td>Indicates that this is an ASPP debug message.</td>
</tr>
<tr>
<td>Serial1/7:</td>
<td>The interface that received or transmitted the packet.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------</td>
</tr>
<tr>
<td>ADI-rx</td>
<td>Indicates a received packet.</td>
</tr>
<tr>
<td>ADI-tx</td>
<td>Indicates a transmitted packet.</td>
</tr>
</tbody>
</table>

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug aspp event</td>
<td>Displays APOS event debug messages.</td>
</tr>
</tbody>
</table>
debug async async-queue

To display debug messages for asynchronous rotary line queueing, use the `debug async async-queue` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug async async-queue
no debug async async-queue
```

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

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(1)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Examples**
The following example starts the asynchronous rotary line queueing debugging display:

```
Router# debug async async-queue
*Mar 2 03:50:28.377: AsyncQ: First connection to be queued - starting the AsyncQ manager
*Mar 2 03:50:28.377: AsyncQ: Enabling the AsyncQ manager
*Mar 2 03:50:28.377: AsyncQ: Started the AsyncQ manager process with pid 98
*Mar 2 03:50:30.164: WaitingTTY66: Did Authentication on waiting TTY (VTY)
*Mar 2 03:50:30.168: AsyncQ: Received ASYNCQ_MSG_ADD
*Mar 2 03:50:30.168: AsyncQ: New queue, adding this connection as the first element
*Mar 2 03:50:34.920: AsyncQ: Created a Waiting TTY on TTY67 with pid 100
*Mar 2 03:50:36.783: WaitingTTY67: Did Authentication on waiting TTY (VTY)
*Mar 2 03:50:36.787: AsyncQ: Received ASYNCQ_MSG_ADD
*Mar 2 03:50:36.787: AsyncQ: Queue exists, adding this connection to the end of the queue
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug ip tcp transactions</code></td>
<td>Enables the IP TCP transactions debugging display to observe significant transactions such as state changes, retransmissions, and duplicate packets.</td>
</tr>
<tr>
<td><code>debug modem</code></td>
<td>Enables the modem debugging display to observe modem line activity on an access server.</td>
</tr>
</tbody>
</table>
debug atm autovc

To display information about autoprovisioned ATM permanent virtual circuit (PVC) events and errors, use the `debug atm autovc` command in privileged EXEC mode. To disable the display of information about autoprovisioned ATM PVC events and errors, use the `no` form of this command.

```
debug atm autovc {event| error| all}
no debug atm autovc {event| error| all}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>Displays all autoprovisioned PVC events.</td>
</tr>
<tr>
<td>error</td>
<td>Displays all autoprovisioned PVC errors.</td>
</tr>
<tr>
<td>all</td>
<td>Displays all autoprovisioned PVC events and errors.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(15)B</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>15.0(1)M</td>
<td>This command was integrated into Cisco IOS Release 15.0(1)M.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(28)SB.</td>
</tr>
<tr>
<td>12.2(33)SRE</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRE.</td>
</tr>
</tbody>
</table>

**Examples**

The following example shows output for the `debug atm autovc` command for all autoprovisioned PVC events and errors:

```
Router# debug atm autovc all
AutoVC all debugging is on
00:09:03:AutoVC(ATM1/0):1/101 enqueued
This message indicates that there is incoming traffic on PVC 1/101 and the PVC is enqueued to be processed.

00:09:03:AutoVC(ATM1/0):process VC 1/101
This message indicates that PVC 1/101 is in the process of being autoprovisioned.

00:09:03:AutoVC(ATM1/0.1):bring up vc 1/101
This message indicates that PVC 1/101 is being brought up.

00:09:03:%ATM-5-UPDOWN:Interface ATM1/0.1, Changing autovc 1/101 to UP
This message indicates that the PVC was brought up successfully.
```
Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>create on-demand</td>
<td>Configures ATM PVC autoprovisioning, which enables a PVC or range of PVCs to be created automatically on demand.</td>
</tr>
</tbody>
</table>
debug atm bundle error

To display debug messages for switched virtual circuit (SVC) bundle errors, use the **debug atm bundle error** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug atm bundle error
no debug atm bundle error
```

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

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(4)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**
The following example provides output for the **debug atm bundle error** command:

```
Router# debug atm bundle error
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug atm bundle events</td>
<td>Displays SVC bundle events.</td>
</tr>
</tbody>
</table>
**debug atm bundle events**

To display switched virtual circuit (SVC) bundle events, use the `debugatmbundleevents` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
depend atm bundle events
no debug atm bundle events
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(4)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following example provides output for the `debugatmbundleevents` command:

```
Router# debug atm bundle events
01:14:35: BUNDLE EVENT(test): b_update_vc for four with bstate 1, vc_state4
01:14:35: BUNDLE EVENT(test): bmupdate active precedence 0x01 0
01:14:35: BUNDLE EVENT(test): bmupdate active precedence 0x02 0
01:14:35: BUNDLE EVENT(test): bmupdate active precedence 0x04 0
01:14:35: BUNDLE EVENT(test): bmupdate active precedence 0x08 0
01:14:35: BUNDLE EVENT(test): bmupdate active precedence 0x10 0
01:14:35: BUNDLE EVENT(test): bmupdate active precedence 0x20 0
01:14:35: BUNDLE EVENT(test): bmupdate active precedence 0x40 0
01:14:35: BUNDLE EVENT(test): bmupdate active precedence 0x80 0
01:14:35: BUNDLE EVENT(test): bundle precedence updated
```

The table below describes the significant fields shown in the display.

**Table 25: debug atm bundle events Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01:14:35</td>
<td>Local time on the router in hours:minutes:seconds.</td>
</tr>
<tr>
<td>BUNDLE EVENT(test)</td>
<td>Bundle event for bundle by that name.</td>
</tr>
<tr>
<td>b_update_vc for four with bstate 1, vc_state 1</td>
<td>Test describing the bundle event.</td>
</tr>
</tbody>
</table>
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug atm bundle error</td>
<td>Displays debug messages for SVC bundle errors.</td>
</tr>
</tbody>
</table>
**debug atm cell-packing**

To enable the display of ATM cell relay cell-packing debugging information, use the `debug atm cell-packing` command in privileged EXEC mode. To disable the display of debugging information, use the `no debug atm cell-packing` command.

**debug atm cell-packing**

**no debug atm cell-packing**

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(25)S</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Examples**

The following example enables debugging for ATM virtual circuits (VCs) that have been configured with cell packing:

```
Router# debug atm cell-packing
ATM Cell Packing debugging is on
00:09:04: ATM Cell Packing: vc 1/100 remote mncp 22 validated
```

The following example enables debugging for permanent virtual paths (PVPs) that have been configured with cell packing:

```
Router# debug atm cell-packing
ATM Cell Packing debugging is on
00:12:33: ATM Cell Packing: vp 1 remote mncp 22 validated
```

The output indicates that the router received the MNCP information from the remote PE router.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>atm mcpt-timers</td>
<td>Creates cell-packing timers that specify how long the PE router can wait for cells to be packed into an MPLS or L2TPv3 packet.</td>
</tr>
<tr>
<td>cell-packing</td>
<td>Enables the packing of multiple ATM cells into a single MPLS or L2TPv3 packet.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>show atm cell-packing</td>
<td>Displays information about the VCs and VPs that have ATM cell relay over MPLS or L2TPv3 cell packing enabled.</td>
</tr>
</tbody>
</table>
debug atm events

To display ATM events, use the `debug atm events` command in privileged EXEC mode. To disable event debugging output, use the `no` form of this command.

```
debug atm events
no debug atm events
```

Syntax Description

This command has no arguments or keywords.

Command Default

ATM event debugging is disabled.

Command Modes

Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XJ</td>
<td>This command was introduced on the Cisco 1700 series routers.</td>
</tr>
<tr>
<td>12.1(5)XR1</td>
<td>This command was implemented on the Cisco IAD2420 Series.</td>
</tr>
<tr>
<td>12.2(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(4)T.</td>
</tr>
</tbody>
</table>

Usage Guidelines

This command displays ATM events that occur on the ATM interface processor and is useful for diagnosing problems in an ATM network. It provides an overall picture of the stability of the network. In a stable network, the `debug atm events` command does not return any information. If the command generates numerous messages, the messages can indicate the possible source of problems.

When configuring or making changes to a router or interface for ATM, enable the `debug atm events` command. Doing so alerts you to the progress of the changes or to any errors that might result. Also use this command periodically when you suspect network problems.

Examples

The following is sample output from the `debug atm events` command:

```
Router# debug atm events
RESET(ATM4/0): PLIM type is 1, Rate is 100Mbps
aip_disable(ATM4/0): state=1
config(ATM4/0)
  aip_love_note(ATM4/0): asr=0x201
  aip_enable(ATM4/0)
  aip_love_note(ATM4/0): asr=0x4000
  aip_enable(ATM4/0): restarting VCs: 7
  aip_setup_vc(ATM4/0): vc:1 vpi:1 vci:1
  aip_love_note(ATM4/0): asr=0x200
  aip_setup_vc(ATM4/0): vc:2 vpi:2 vci:2
  aip_love_note(ATM4/0): asr=0x200
  aip_setup_vc(ATM4/0): vc:3 vpi:3 vci:3
```
The table below describes the significant fields shown in the display.

**Table 26: debug atm events Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PLIM type</strong></td>
<td>Indicates the interface rate in megabits per second (Mbps). Possible values are:</td>
</tr>
<tr>
<td></td>
<td>• 1 = TAXI(4B5B) 100 Mbps</td>
</tr>
<tr>
<td></td>
<td>• 2 = SONET 155 Mbps</td>
</tr>
<tr>
<td></td>
<td>• 3 = E3 34 Mbps</td>
</tr>
<tr>
<td><strong>state</strong></td>
<td>Indicates current state of the ATM Interface Processor (AIP). Possible values are:</td>
</tr>
<tr>
<td></td>
<td>• 1 = An ENABLE will be issued soon.</td>
</tr>
<tr>
<td></td>
<td>• 0 = The AIP will remain shut down.</td>
</tr>
<tr>
<td><strong>asr</strong></td>
<td>Defines a bitmask, which indicates actions or completions to commands. Valid bitmask values are:</td>
</tr>
<tr>
<td></td>
<td>• 0x0800 = AIP crashed, reload may be required.</td>
</tr>
<tr>
<td></td>
<td>• 0x0400 = AIP detected a carrier state change.</td>
</tr>
<tr>
<td></td>
<td>• 0x0n00 = Command completion status. Command completion status codes are:</td>
</tr>
<tr>
<td></td>
<td>• n = 8 Invalid Physical Layer Interface Module (PLIM) detected</td>
</tr>
<tr>
<td></td>
<td>• n = 4 Command failed</td>
</tr>
<tr>
<td></td>
<td>• n = 2 Command completed successfully</td>
</tr>
<tr>
<td></td>
<td>• n = 1 CONFIG request failed</td>
</tr>
<tr>
<td></td>
<td>• n = 0 Invalid value</td>
</tr>
</tbody>
</table>

The following line indicates that the AIP was reset. The PLIM TYPE detected was 1, so the maximum rate is set to 100 Mbps.

RESET(ATM4/0): PLIM type is 1, Rate is 100Mbps
The following line indicates that the AIP was given a shutdown command, but the current configuration indicates that the AIP should be up:

```plaintext
aip_disable(ATM4/0): state=1
```

The following line indicates that a configuration command has been completed by the AIP:

```plaintext
aip_love_note(ATM4/0): asr=0x201
```

The following line indicates that the AIP was given a no shutdown command to take it out of shutdown:

```plaintext
aip_enable(ATM4/0)
```

The following line indicates that the AIP detected a carrier state change. It does not indicate that the carrier is down or up, only that it has changed.

```plaintext
aip_love_note(ATM4/0): asr=0x4000
```

The following line of output indicates that the AIP enable function is restarting all permanent virtual circuits (PVCs) automatically:

```plaintext
aip_enable(ATM4/0): restarting VCs: 7
```

The following lines of output indicate that PVC 1 was set up and a successful completion code was returned:

```plaintext
aip_setup_vc(ATM4/0): vc:1 vpi:1 vci:1
aip_love_note(ATM4/0): asr=0x200
```
**debug atm ha-error**

To debug ATM high-availability (HA) errors on a networking device, use the `debug atm ha-error` command in privileged EXEC mode. To disable the display of debugging output, use the `no` form of this command.

```
debug atm ha-error
no debug atm ha-error
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Debugging is disabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(22)S</td>
<td>This command was introduced on Cisco 7500, 10000, and 12000 series Internet routers.</td>
</tr>
<tr>
<td>12.2(18)S</td>
<td>This command was integrated into Cisco IOS Release 12.2(18)S on Cisco 7500 series routers.</td>
</tr>
<tr>
<td>12.2(20)S</td>
<td>Support was added for the Cisco 7304 router. The Cisco 7500 series router is not supported in Cisco IOS Release 12.2(20)S.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(28)SB.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2(33)SXH</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SXH.</td>
</tr>
</tbody>
</table>

**Examples**

The following example displays debug messages regarding ATM HA errors on the networking device:

```
Router# debug atm ha-error
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug atm ha-events</code></td>
<td>DEBUGS ATM HA events on the networking device.</td>
</tr>
<tr>
<td><code>debug atm ha-state</code></td>
<td>DEBUGS ATM HA state information on the networking device.</td>
</tr>
</tbody>
</table>
debug aaa accounting through debug auto-config

dump atm ha-error
debug atm ha-events

To debug ATM high-availability (HA) events on the networking device, use the debug atm ha-events command in privileged EXEC mode. To disable the display of debugging output, use the no form of this command.

```
debug atm ha-events
no debug atm ha-events
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Debugging is disabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(22)S</td>
<td>This command was introduced on Cisco 7500, 10000, and 12000 series Internet routers.</td>
</tr>
<tr>
<td>12.2(18)S</td>
<td>This command was integrated into Cisco IOS Release 12.2(18)S on Cisco 7500 series routers.</td>
</tr>
<tr>
<td>12.2(20)S</td>
<td>Support was added for the Cisco 7304 router. The Cisco 7500 series router is not supported in Cisco IOS Release 12.2(20)S.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(28)SB.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2(33)SXH</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SXH.</td>
</tr>
</tbody>
</table>

**Examples**

The following example displays debug messages regarding ATM HA events on the networking device:

```
Router# debug atm ha-events
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug atm ha-error</td>
<td>Debugs ATM HA errors on the networking device.</td>
</tr>
<tr>
<td>debug atm ha-state</td>
<td>Debugs ATM HA state information on the networking device.</td>
</tr>
</tbody>
</table>
debug atm ha-state

To debug ATM high-availability (HA) state information on the networking device, use the `debug atm ha-state` command in privileged EXEC mode. To disable the display of debugging output, use the `no` form of this command.

```
debug atm ha-state
no debug atm ha-state
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(22)S</td>
<td>This command was introduced on Cisco 7500, 10000, and 12000 series Internet routers.</td>
</tr>
<tr>
<td>12.2(18)S</td>
<td>This command was integrated into Cisco IOS Release 12.2(18)S on Cisco 7500 series routers.</td>
</tr>
<tr>
<td>12.2(20)S</td>
<td>Support was added for the Cisco 7304 router. The Cisco 7500 series router is not supported in Cisco IOS Release 12.2(20)S.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(28)SB.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2(33)SXH</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SXH.</td>
</tr>
</tbody>
</table>

**Examples**

The following example displays debug messages regarding the ATM HA state on the networking device:

```
Router# debug atm ha-state
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug atm ha-error</td>
<td>Debugs ATM HA errors on the networking device.</td>
</tr>
<tr>
<td>debug atm ha-events</td>
<td>Debugs ATM HA events on the networking device.</td>
</tr>
</tbody>
</table>
debug atm l2transport

To enable the display of debugging information related to ATM over MPLS, use the debug atm l2transport command in privileged EXEC mode. To disable the display of debugging information, use the no form of this command.

debug atm l2transport
no debug atm l2transport

Syntax Description
This command has no arguments or keywords.

Command Default
Debugging of ATM over MPLS is not enabled.

Command Modes
Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(25)S</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

Examples

The following example shows the events and messages when configuring ATM Cell Relay over MPLS in VP mode.

Router# debug atm l2transport
ATM L2transport Events and Errors debugging is on
Router# show debug
ATM L2transport:
  ATM L2transport Events and Errors debugging is on
Router(config-if)# atm pvp 24 l2transport
Router(config-if-atm-l2trans-pvp)# xconnect 11.11.11.11 700 pw-class vp
Router(config-if-atm-l2trans-pvp)# end
00:14:51: ATM L2trans(ATM1/0): VP 24 is created
00:14:51: ATM L2trans(ATM1/0): ckt_type 10, ckt_id 1000024 UP
00:14:51: ATM L2trans(ATM1/0): response is connect forwarded

The following example shows the events and messages when deleting a PVP.

Router(config-if)# no atm pvp 24 l2transport
00:14:37: ATM L2trans(ATM1/0): ckt_type 10, ckt_id 1000024 DOWN
00:14:37: ATM L2trans(ATM1/0): ckt_type 10, ckt_id 1000024 DOWN
00:14:37: ATM L2trans(ATM1/0): remove xconnect circuit_type=10,
circuit_id=1000024
00:14:37: ATM L2trans(ATM1/0): ckt_type 10, ckt_id 1000024 DOWN
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>show mpls l2transport vc</code></td>
<td>Displays information about AToM circuits that have been enabled to route Layer 2 packets on a router.</td>
</tr>
</tbody>
</table>
debug atm lfi

To display multilink PPP (MLP) over ATM link fragmentation and interleaving (LFI) debug information, use the `debug atm lfi` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug atm lfi
no debug atm lfi
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(7)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(28)SB.</td>
</tr>
</tbody>
</table>

**Examples**

The following examples show output from the `debug atm lfi` command. Each example is preceded by an explanation of the output.

- The following output indicates that the packet has dequeued from the per-VC queue that is associated with the virtual circuit (VC):

```
00:17:27: MLP-ATM(Virtual-Access3) pak dequeued from per VC Q 15/200, qcount:0
```

- The following output indicates that the packet is enqueued on the per-VC queue associated with the VC:

```
00:17:27: MLP-ATM(Virtual-Access3) pak enqueued to per VC Q 15/200, qcount:0
```

- The following output indicates that the packet has dequeued from the MLP bundle queue:

```
00:17:27: MLP-ATM(Virtual-Access3) pak dequeued from MP Bundle 15/200, qcount:0
```

- The following output indicates that PPP over ATM (PPPoA) encapsulation cannot be added to the packet for some reason:

```
00:17:27: MLP-ATM(Virtual-Access3) encapsulation failure - dropping packet
```

- The following output indicates that the VC could not be found on the virtual access interface associated with the PPPoA session:

```
00:17:27: MLP-ATM(Virtual-Access3) No VC to transmit- dropping packet
```
• When a permanent virtual circuit (PVC) has been deleted, the following output indicates that MLP has been deconfigured successfully:

00:17:27: MLP-ATM(Virtual-Access3) mlp de-configured for PVC 15/200

• If the changing of any PVC parameters requires re-creation of the PVC, the following output is generated during the re-creation of the PVC:

00:17:27: MLP-ATM(Virtual-Access3) MLPoATM re-configured for PVC 15/200

• The following output indicates that the MLP over ATM structure associated with a VC has failed to allocate memory:

00:17:27: MLP-ATM(Virtual-Access3) Memory allocation error

• The following output is generated when MLP over ATM is first configured on a PVC:

00:17:27: MLP-ATM(Virtual-Access3) MLPoATM configured for PVC 15/200

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>show multilink ppp</td>
<td>Displays bundle information for MLP bundles.</td>
</tr>
</tbody>
</table>
debug atm native

To display ATM switched virtual circuit (SVC) signaling events, use the `debug atm native` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
ddebug atm native {api| conn| error| filter}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>api</td>
<td>(Optional) Native ATM application programming interface (API). Displays events that occur as a result of the exchange between the native ATM API and the signaling API.</td>
</tr>
<tr>
<td>conn</td>
<td>(Optional) Native ATM connection manager. Displays internal connection manager events for the native ATM API.</td>
</tr>
<tr>
<td>error</td>
<td>(Optional) Native ATM error. Displays errors that occur during the setup of an ATM SVC.</td>
</tr>
<tr>
<td>filter</td>
<td>(Optional) Native ATM filter. Displays the internal network service access point (NSAP) filter events of the native ATM API.</td>
</tr>
</tbody>
</table>

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(8)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Native ATM API is the layer above the signaling API. Static map and Resource Reservation Protocol (RSVP) clients use the native ATM API to interact with the signaling API to create ATM SVCs.

Use the `debug atm native` command to diagnose problems in the creation of static map and RSVP ATM SVCs.

**Examples**

The following is sample output for the `debug atm native` command with the `api` keyword:

```
Router# debug atm native api
0:24:59:NATIVE ATM :associate endpoint
```
00:24:59:NATIVE ATM :ID (3) prep outgoing call, conn_type 0
00:24:59:NATIVE ATM :ID (3) set connection attribute for 5
00:24:59:NATIVE ATM :ID (3) query connection attribute 8
00:24:59:NATIVE ATM :ID (3) set connection attribute for 8
00:24:59:NATIVE ATM :ID (3) set connection attribute for 9
00:24:59:NATIVE ATM :ID (3) set connection attribute for 10
00:24:59:NATIVE ATM :ID (3) set connection attribute for 7
00:24:59:NATIVE ATM :ID (3) set connection attribute for 6
00:24:59:NATIVE ATM :ID (3) set connection attribute for 2
00:24:59:NATIVE ATM :ID (3) set connection attribute for 0
00:24:59:NATIVE ATM :ID (3) query connection attribute 12
00:24:59:NATIVE ATM :ID (3) set connection attribute for 12
00:24:59:NATIVE ATM :ID (3) query connection attribute 13
00:24:59:NATIVE ATM :ID (3) set connection attribute for 13
00:24:59:NATIVE ATM :ID (3) connect outgoing call
00:24:59:NATIVE ATM :ID (3) callback, CONNECT received
**debug atm nbma**

To display setup and teardown events for ATM switched virtual circuits (SVCs) configured using the Resource Reservation Protocol (RSVP), use the `debug atm nbma` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug atm nbma [api]
no debug atm nbma
```

**Syntax Description**

| api | (Optional) Nonbroadcast multiaccess (NBMA) ATM application programming interface (API). Displays events that occur as a result of the exchange between RSVP and the NBMA API. |

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(8)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use the `debug atm nbma` command to diagnose problems in the creation of RSVP SVCs.

The RSVP application creates SVCs by using the NBMA API. The `debug atm nbma api` command with the `api` keyword displays events that occur as a result of the exchange between RSVP and the NBMA API.

**Examples**

The following is sample output for the `debug atm nbma api` command:

```
Router# debug atm nbma api
00:52:50:NBMA-ATM-API - atm_setup_req
00:52:50:NBMA-ATM-API - nbma_atm_fill_blli
00:52:50:NBMA-ATM-API - nbma_atm_fill_bhli
00:52:50:NBMA-ATM-API - nbma_atm_callbackMsg - NATIVE_ATM_OUTGOING_CALL_ACTIVE
00:52:50:NBMA-ATM-API - rcv_outgoing_call_active
00:52:50:NBMA-ATM-API - nbma_svc_lookup
```
debug atm oam cc

To display ATM operation, administration, and maintenance (OAM) F5 continuity check (CC) management activity, use the `debug atm oam cc` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```plaintext
debug atm oam cc [interface atm number]
no debug atm oam cc [interface atm number]
```

**Syntax Description**

| interface atm number | (Optional) Number of the ATM interface. |

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(2)XB</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(13)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(13)T.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(28)SB.</td>
</tr>
</tbody>
</table>

**Examples**

The following sample output for the `debug atm oam cc` command records activity beginning with the entry of the `oam-pvcmanagcc` command and ending with the entry of the `nooam-pvcmanagcc` command. The ATM 0 interface is specified, and the "both" segment direction is specified. The output shows an activation request sent and confirmed, a series of CC cells sent by the routers on each end of the segment, and a deactivation request and confirmation.

Router# debug atm oam cc interface atm0
Generic ATM:
    ATM OAM CC cells debugging is on
Router#
00:15:05: CC ACTIVATE MSG (ATM0) I:VCD#1 VC 1/40 OAM Cell Type:4 OAM Type:1 OAM Func:1 Direction:3 CTag:5
00:15:05: CC ACTIVATE CONFIRM MSG (ATM0) O:VCD#1 VC 1/40 OAM Cell Type:4 OAM Type:1 OAM Func:1 Direction:3 CTag:5
00:15:06: CC CELL (ATM0) O:VCD#1 VC 1/40 OAM Cell Type:4 OAM Type:1 OAM Func:1
00:15:07: CC CELL (ATM0) I:VCD#1 VC 1/40 OAM Cell Type:4 OAM Type:1 OAM Func:4
00:15:08: CC CELL (ATM0) O:VCD#1 VC 1/40 OAM Cell Type:4 OAM Type:1 OAM Func:4
00:15:09: CC CELL (ATM0) I:VCD#1 VC 1/40 OAM Cell Type:4 OAM Type:1 OAM Func:4
00:15:10: CC CELL (ATM0) O:VCD#1 VC 1/40 OAM Cell Type:4 OAM Type:1 OAM Func:4
00:15:11: CC CELL (ATM0) I:VCD#1 VC 1/40 OAM Cell Type:4 OAM Type:1 OAM Func:4
00:15:12: CC CELL (ATM0) O:VCD#1 VC 1/40 OAM Cell Type:4 OAM Type:1 OAM Func:4
00:15:13: CC CELL (ATM0) I:VCD#1 VC 1/40 OAM Cell Type:4 OAM Type:1 OAM Func:4
00:15:14: CC CELL (ATM0) O:VCD#1 VC 1/40 OAM Cell Type:4 OAM Type:1 OAM Func:4
The table below describes the significant fields shown in the display.

### Table 27: debug atm oam cc Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:15:05</td>
<td>Time stamp.</td>
</tr>
<tr>
<td>CC ACTIVATE MSG (ATM0)</td>
<td>Message type and interface.</td>
</tr>
<tr>
<td>0</td>
<td>Source.</td>
</tr>
<tr>
<td>1</td>
<td>Sink.</td>
</tr>
<tr>
<td>VC 1/40</td>
<td>Virtual circuit identifier.</td>
</tr>
<tr>
<td>Direction: 3</td>
<td>Direction in which the cells are traveling. May be one of the following values:</td>
</tr>
<tr>
<td></td>
<td>1-- local router is the sink.</td>
</tr>
<tr>
<td></td>
<td>2-- local router is the source.</td>
</tr>
<tr>
<td></td>
<td>3-- both routers operate as the source and sink.</td>
</tr>
</tbody>
</table>

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oam-pvc manage cc</td>
<td>Configures ATM OAM F5 CC management.</td>
</tr>
<tr>
<td>show atm pvc</td>
<td>Displays all ATM PVCs and traffic information.</td>
</tr>
</tbody>
</table>
debug atm oc3 pom

To display debug messages for ATM-OC3 Provisioning Object Manager (POM) network modules, use the **debug atm oc3 pom** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug atm oc3 pom {data| flow| pa| sar| sfp| trace}
no debug atm oc3 pom {data| flow| pa| sar| sfp| trace}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>data</strong></td>
<td>Displays debug messages for incoming packet indications.</td>
</tr>
<tr>
<td><strong>flow</strong></td>
<td>Displays debug messages for flow control indications.</td>
</tr>
<tr>
<td><strong>pa</strong></td>
<td>Displays debug messages for online insertion or removal (OIR) of the ATM-OC3 POM network module.</td>
</tr>
<tr>
<td><strong>sar</strong></td>
<td>Displays debug messages for blocking commands sent to the segmentation and reassembly (SAR) and their acknowledgments.</td>
</tr>
<tr>
<td><strong>sfp</strong></td>
<td>Displays debug messages for OIR of modules in the SFP port of the network module.</td>
</tr>
<tr>
<td><strong>trace</strong></td>
<td>Displays debug messages that give the hexadecimal representation of commands sent to the SAR and their acknowledgments.</td>
</tr>
</tbody>
</table>

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(2)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

### Usage Guidelines

**debug atm oc3 pom data command**

Use the **debug atm oc3 pom data** command to display the incoming packet indications. Each incoming packet transferred by direct memory access (DMA) to the host memory by the SAR will cause a packet indication.

**debug atm oc3 pom flow command**

Use the **debug atm oc3 pom flow** command to display flow control indications.
When traffic sent to the SAR exceeds the peak cell rate for a particular virtual circuit (VC), the SAR indicates this to the host by sending flow control indications. These indications inform the host that either the high watermark or the low watermark has been reached for that VC queue.

When a high watermark is received from the SAR, indicating that the VC queue is full, the host will stop sending packets to the SAR until a low watermark indication is received. A low watermark indicates that the VC queue has been drained sufficiently to receive additional packets.

**debug atm oc3 pom pa command**

Use the `debug atm oc3 pom pa` command on those platforms supporting OIR to display the indications generated when the port adapter (the ATM-OC3 POM network module) is subjected to OIR. This command is used principally during the port adapter initialization phase.

**debug atm oc3 pom sar command**

Use the `debug atm oc3 pom sar` command to display blocking commands or indications sent to or received from the SAR. This includes commands or indications of the creation or deletion of virtual circuits or virtual paths.

**debug atm oc3 pom sfp command**

Use the `debug atm oc3 pom sfp` command to display the indications generated when a module in the SFP port is subjected to OIR.

**debug atm oc3 pom trace command**

Use the `debug atm oc3 pom trace` command to display the hexadecimal representation of commands sent to or received from the SAR. To facilitate debugging, use this command in conjunction with the `debug atm oc3 pom sar` command.

**Examples**

The following is sample output from the `debug atmo3 pom data` command:

Router# debug atm oc3 pom data
DATA debugging is on
Router#
*Jun 27 22:03:17.996: Packet Indication:
   *Jun 27 22:03:17.996: word 0: 0x00007D24
   *Jun 27 22:03:17.996: word 1: 0x00002F02
   *Jun 27 22:03:17.996: word 2: 0xEE323464
   *Jun 27 22:03:17.996: word 3: 0x006C006D

The table below describes the significant fields shown in the display.

**Table 28: debug atm oc3 pom data Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jun 27 22:03:17.996:</td>
<td>Date or time stamp of packet DMA transfer.</td>
</tr>
<tr>
<td>word [0 - 3]: 0xXXXXXXX</td>
<td>Hexadecimal representation of four-word acknowledgment from the SAR when a packet is transferred by DMA to the host memory by the SAR.</td>
</tr>
</tbody>
</table>
Examples

The following example illustrates the output from the `debug atm oc3 pom flow` command:

```
Router# debug atm oc3 pom flow
FLOW CNTL INDICATION debugging is on
Router#
*Jun 27 15:14:13.123: Flow Indication:
*Jun 27 15:14:13.123: word 0: 0x00000001
*Jun 27 15:14:13.123: word 1: 0x300012C0
*Jun 27 15:14:13.123: word 2: 0x18001060
*Jun 27 15:14:13.123: word 3: 0x00080021
*Jun 27 15:14:13.456: Flow Indication:
*Jun 27 15:14:13.456: word 0: 0x00000001
*Jun 27 15:14:13.456: word 1: 0x300012C0
*Jun 27 15:14:13.456: word 2: 0x18001060
*Jun 27 15:14:13.456: word 3: 0x00090022
```

The table below describes the significant fields shown in the display.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jun 27 15:14:13.456:</td>
<td>Date or time stamp of flow indication</td>
</tr>
<tr>
<td>word [0-3]: 0xXXXXXXXX</td>
<td>Hexadecimal representation of four-word indication sent by the SAR to the host that a high watermark or low watermark event has occurred.</td>
</tr>
<tr>
<td>word 3: 0x00XXYYYY</td>
<td>When XX is 08, a high watermark has been received by the host. The host will stop queueing packets for the VC. When XX is 09, a low watermark has been received by the host. The host will resume sending packets to the VC. YYY is the running count of flow indication events sent to the host.</td>
</tr>
</tbody>
</table>

Examples

The following examples illustrate the output from the `debug atm oc3 pom pa` command.

The first example gives the output when the network module is removed:

```
Router# debug atm oc3 pom pa
PA debugging is on
*Jun 27 22:40:56.110: %OIR-6-REMCARD: Card removed from slot 2, interfaces disabled
*Jun 27 22:40:56.122: *** Freed 6146 buffers
```

The second example gives the output when the network module is inserted, and gives the values of internal registers of the module:

```
*Jun 27 22:41:08.654: %OIR-6-INSRCARD: Card inserted in slot 2, interfaces administratively shut down
*Jun 27 22:41:11.402: sar_base_addr 0x5C800000
*Jun 27 22:41:11.402: PCI_MEMBAR2_REG after configuring:0x5E000008
*Jun 27 22:41:11.402: PCI_MEMBAR3_REG after configuring:0x5F000000
*Jun 27 22:41:11.402: PCI_COMMAND_REG: Offset= 0x4; value= 0x2A00006
```
**Examples**

The following examples illustrate the output from the `debug atm oc3 pomsar` command.

The first example displays command indications for setting up a VC and opening the reassembly channel and the segmentation channel in the SAR:

```
Router# debug atm oc3 pomsar
SAR debugging is on
Router# configure terminal
Router(config)# interface atm 2/0
Router(config-if)# pvc 2/2
Router(config-if-atm-vc)# exit
Router(config-if)#
```

```
*Jun 27 22:12:28.816: ATM2/0: HI/LO watermarks: 526/263; PeakRate: 149760
```

The second example displays the commands sent to the SAR and the acknowledgements returned when the VC is deleted and the segmentation and reassembly channels are closed:

```
Router(config-if)# no pvc 2/2
Router(config-if)#
```

```
*Jun 27 22:12:59.016: ATM2/0: Sent pending EOP successfully
*Jun 27 22:12:59.016: ATM2/0: Close_Channel(RSY): Chan_ID (0x104)
*Jun 27 22:12:59.016: ATM2/0: Close_Channel(RSY): Chan_ID (0x104) CLOSE
*Jun 27 22:12:59.016: ATM2/0: Close_Channel: CLOSE_PENDING
*Jun 27 22:12:59.016: ATM2/0: Close_Channel(SEG): Chan_ID (0x105)
*Jun 27 22:12:59.016: ATM2/0: Close_Channel: CLOSE
```

**Examples**

The following examples illustrate the output from the `debug atm oc3 pomsfp` command.

The first example gives the output when the module is removed from the SFP port:

```
Router# debug atm oc3 pomsfp
SFP debugging is on
```

```
*Jun 27 22:27:40.792: SFP TX FAULT detected
*Jun 27 22:27:40.808: SFP LOS detected
*Jun 27 22:27:40.812: SFP removal detected
*Jun 27 22:27:43.464: %LINK-3-UPDOWN: Interface ATM2/0, changed state to down
*Jun 27 22:27:44.464: %LINEPROTO-5-UPDOWN: Line protocol on Interface ATM2/0, changed state to down
```

The second example gives the output when the module is inserted in the SFP port:

```
*Jun 27 22:27:47.776: SFP LOS cleared
*Jun 27 22:27:47.776: SFP TX FAULT detected
*Jun 27 22:27:48.496: Set the Container_id to 17
*Jun 27 22:27:50.496: %LINK-3-UPDOWN: Interface ATM2/0, changed state to up
*Jun 27 22:27:51.496: %LINEPROTO-5-UPDOWN: Line protocol on Interface ATM2/0, changed state to up
```

**Examples**

The first example illustrates the output from the `debug atm oc3 pombmsg` command when it is run without the `debug atmoc3sar` command being activated:

```
Router# debug atm oc3 pombmsg
SAR CMD/ACK debugging is on
```
Router# configure terminal
Router(config)# interface atm 2/0
router(config-if)# pvc 2/2
Router(config-if-atm-vc)# exit
Router(config-if)#

*Jun 27 22:15:09.284: Command Sent:
*Jun 27 22:15:09.284: word 0: 0x000000480
*Jun 27 22:15:09.284: word 1: 0x00012010
*Jun 27 22:15:09.284: word 2: 0x00000000
*Jun 27 22:15:09.284: word 3: 0x00000000
*Jun 27 22:15:09.284: word 4: 0x00200020
*Jun 27 22:15:09.284: word 5: 0x00000000
*Jun 27 22:15:09.284: word 6: 0x00000000
*Jun 27 22:15:09.284: word 7: 0x00000000
*Jun 27 22:15:09.284: word 8: 0x00000000
*Jun 27 22:15:09.284: Command Indication:
*Jun 27 22:15:09.284: word 0: 0x00000000
*Jun 27 22:15:09.284: word 1: 0x01042110
*Jun 27 22:15:09.284: word 2: 0x01050000
*Jun 27 22:15:09.284: word 3: 0x0000003B
*Jun 27 22:15:09.284: ACK received = 200 usecs
*Jun 27 22:15:09.284: Command Sent:
*Jun 27 22:15:09.284: word 0: 0x01050480
*Jun 27 22:15:09.284: word 1: 0x00011010
*Jun 27 22:15:09.284: word 2: 0x02000000
*Jun 27 22:15:09.284: word 3: 0x00001003
*Jun 27 22:15:09.284: word 4: 0x00200020
*Jun 27 22:15:09.284: word 5: 0x08830000
*Jun 27 22:15:09.284: word 6: 0x10000000
*Jun 27 22:15:09.284: word 7: 0x86850000
*Jun 27 22:15:09.284: word 8: 0x00010040
*Jun 27 22:15:09.284: word 9: 0x00000000
*Jun 27 22:15:09.284: Command Indication:
*Jun 27 22:15:09.284: word 0: 0x00010000
*Jun 27 22:15:09.284: word 1: 0x00011110
*Jun 27 22:15:09.284: word 2: 0x02000000
*Jun 27 22:15:09.284: word 3: 0x0000003D
*Jun 27 22:15:09.284: ACK received = 200 usecs

Thetablebelowdescriptionsignificantfieldsshowninthedisplay.

Table 30: debug atm oc3 pom trace Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jun 27 22:15:09.284:</td>
<td>Date or time stamp for the command dialog.</td>
</tr>
<tr>
<td>word [0 - n]: 0xXXXXXXX</td>
<td>Hexadecimal representation of the n-word command sent to the SAR (under Command Sent:) and the four-word acknowledgment returned by the SAR (under Command Indication:).</td>
</tr>
<tr>
<td>ACK received</td>
<td>Time (in microseconds) between sending the command to the SAR and receiving the acknowledgment.</td>
</tr>
</tbody>
</table>

The second example illustrates the output from the debugatmoc3pomtrace command run in conjunction with the debugatmoc3pomsar command.
In this example, each command sent to the SAR is displayed by the `debug atm oc3 pom sar` command. Then the hexadecimal representation of the command and its acknowledgement are displayed by the `debug atm oc3 pom trace` command.

Router# debug atm oc3 pom trace
SAR CMD/ACK debugging is on
Router# debug atm oc3 pom sar
SAR debugging is on
Router# configure terminal
Router(config)# interface atm 2/0
Router(config-if)# pvc 2/2
Router(config-if-atm-vc)# exit

*Jun 27 22:15:09.284: word 0: 0x00000480
*Jun 27 22:15:09.284: word 1: 0x00012010
*Jun 27 22:15:09.284: word 2: 0x00000000
*Jun 27 22:15:09.284: word 3: 0x00000000
*Jun 27 22:15:09.284: word 4: 0x00200020
*Jun 27 22:15:09.284: word 5: 0x00000000
*Jun 27 22:15:09.284: word 6: 0x00000000
*Jun 27 22:15:09.284: word 7: 0x00000000
*Jun 27 22:15:09.284: word 8: 0x00000000
*Jun 27 22:15:09.284: Command Indication:
*Jun 27 22:15:09.284: word 0: 0x00000000
*Jun 27 22:15:09.284: word 1: 0x01042110
*Jun 27 22:15:09.284: word 2: 0x01050000
*Jun 27 22:15:09.284: word 3: 0x0000003B
*Jun 27 22:15:09.284: ACK received = 200 usecs
*Jun 27 22:15:09.284: word 0: 0x00000000
*Jun 27 22:15:09.284: word 1: 0x00011010
*Jun 27 22:15:09.284: word 2: 0x01050000
*Jun 27 22:15:09.284: word 3: 0x00010003
*Jun 27 22:15:09.284: Command Indication:
*Jun 27 22:15:09.284: word 0: 0x00010000
*Jun 27 22:15:09.284: word 1: 0x00011110
*Jun 27 22:15:09.284: word 2: 0x02000000
*Jun 27 22:15:09.284: word 3: 0x00001000
*Jun 27 22:15:09.284: Command Indication:
*Jun 27 22:15:09.284: word 0: 0x00001000
*Jun 27 22:15:09.284: word 1: 0x00000000
*Jun 27 22:15:09.284: word 2: 0x00000000
*Jun 27 22:15:09.284: word 3: 0x00010000
*Jun 27 22:15:09.284: Command Indication:
*Jun 27 22:15:09.284: word 0: 0x00000000
*Jun 27 22:15:09.284: word 1: 0x00000000
*Jun 27 22:15:09.284: word 2: 0x00000000
*Jun 27 22:15:09.284: word 3: 0x00000000
*Jun 27 22:15:09.284: Command Indication:
debug atm t3e3

To display debug messages for ATM T3/E3 network modules, use the `debug atm t3e3` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug atm t3e3 {data | flow | pa | sar | trace}
no debug atm t3e3 {data | flow | pa | sar | trace}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Displays debug messages for incoming packet indications.</td>
</tr>
<tr>
<td>flow</td>
<td>Displays debug messages for flow control indications.</td>
</tr>
<tr>
<td>pa</td>
<td>Displays debug messages for online insertion or removal (OIR) of the ATM T3/E3 network module.</td>
</tr>
<tr>
<td>sar</td>
<td>Displays debug messages for blocking commands sent to the segmentation and reassembly (SAR) and their acknowledgments.</td>
</tr>
<tr>
<td>trace</td>
<td>Displays debug messages that give the hexadecimal representation of commands sent to the SAR and their acknowledgments.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(15)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

**debug atm t3e3 data command**

Use the `debug atm t3e3 data` command to display the incoming packet indications. Each incoming packet transferred by direct memory access (DMA) to the host memory by the SAR will cause a packet indication.

**debug atm t3e3 flow command**

Use the `debug atm t3e3 flow` command to display flow control indications.

When traffic sent to the SAR exceeds the peak cell rate for a particular virtual circuit (VC), the SAR indicates this to the host by sending flow control indications. These indications inform the host that either the high watermark or the low watermark has been reached for that VC queue.
When a high watermark is received from the SAR, indicating that the VC queue is full, the host will stop sending packets to the SAR until a low watermark indication is received. A low watermark indicates that the VC queue has been drained sufficiently to receive additional packets.

**debug atm t3e3 pa command**

Use the `debug atm t3e3 pa` command on those platforms supporting OIR to display the indications generated when the port adapter (the ATM T3/E3 network module) is subjected to OIR. This command is used principally during the port adapter initialization phase.

**debug atm t3e3 sar command**

Use the `debug atm t3e3 sar` command to display blocking commands or indications sent to or received from the SAR. This includes commands or indications of the creation or deletion of virtual circuits or virtual paths.

**debug atm t3e3 trace command**

Use the `debug atm t3e3 trace` command to display the hexadecimal representation of commands sent to or received from the SAR. To facilitate debugging, use this command in conjunction with the `debug atm t3e3 sar` command.

### Examples

The following is sample output from the `debug atm t3e3 data` command:

```
Router# debug atm t3e3 data
DATA debugging is on
Router# *Jun 27 22:03:17.996: Packet Indication:
*Jun 27 22:03:17.996: word 0: 0x00007D24
*Jun 27 22:03:17.996: word 1: 0x00002F02
*Jun 27 22:03:17.996: word 2: 0xEE323464
*Jun 27 22:03:17.996: word 3: 0x006C006D
```

The table below describes the significant fields shown in the display.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jun 27 22:03:17.996:</td>
<td>Date or time stamp of packet DMA transfer.</td>
</tr>
<tr>
<td>word [0 - 3]: 0xXXXXXXXX</td>
<td>Hexadecimal representation of four-word acknowledgment from the SAR when a packet is transferred by DMA to the host memory by the SAR.</td>
</tr>
</tbody>
</table>

### Examples

The following example illustrates the output from the `debug atm t3e3 flow` command:

```
Router# debug atm t3e3 flow
FLOW CNTL INDICATION debugging is on
Router# *Jun 27 15:14:13.123: Flow Indication:
*Jun 27 15:14:13.123: word 0: 0x00000001
*Jun 27 15:14:13.123: word 1: 0x300012C0
*Jun 27 15:14:13.123: word 2: 0x18001060
*Jun 27 15:14:13.123: word 3: 0x00009001
*Jun 27 15:14:13.456: Flow Indication:
*Jun 27 15:14:13.456: word 0: 0x00000001
```
The table below describes the significant fields shown in the display.

**Table 32: debug atm t3e3 flow Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jun 27 15:14:13.456:</td>
<td>Date or time stamp of flow indication</td>
</tr>
<tr>
<td>word [0 - 3]: 0xFFFFFFFF</td>
<td>Hexadecimal representation of four-word indication sent by the SAR to the host that a high watermark or low watermark event has occurred.</td>
</tr>
<tr>
<td>word 3: 0x00XXYYYY</td>
<td>When XX is 08, a high watermark has been received by the host. The host will stop queueing packets for the VC. When XX is 09, a low watermark has been received by the host. The host will resume sending packets to the VC. YYYY is the running count of flow indication events sent to the host.</td>
</tr>
</tbody>
</table>

**Examples**

The following examples illustrate the output from the `debug atm t3e3 pa` command.

The first example gives the output when the network module is removed:

```
Router# debug atm t3e3 pa
PA debugging is on
*Jun 27 22:40:56.110: %OIR-6-REMCARD: Card removed from slot 2, interfaces disabled
*Jun 27 22:40:56.122: *** Freed 6146 buffers
```

The second example gives the output when the network module is inserted, and gives the values of internal registers of the module:

```
*Jun 27 22:41:08.654: %OIR-6-INSCARD: Card inserted in slot 2, interfaces administratively shut down
*Jun 27 22:41:11.402: sar_base_addr 0x5C800000
*Jun 27 22:41:11.402: PCI_MEMBAR2_REG after configuring:0x5E000008
*Jun 27 22:41:11.402: PCI_MEMBAR3_REG after configuring:0x5F000000
*Jun 27 22:41:11.402: PCI_COMMAND_REG: Offset= 0x4; value= 0x2A00006
*Jun 27 22:41:11.402: FPGA Base address is 0x5C900000
*Jun 27 22:41:11.402: FPGA PCI config Reg is 0x02200002
```

**Examples**

The following examples illustrate the output from the `debug atm t3e3 sar` command.

The first example displays command indications for setting up a VC and opening the reassembly channel and the segmentation channel in the SAR:

```
Router# debug atm t3e3 sar
SAR debugging is on
Router# configure terminal
Router(config)# interface atm 2/0
Router(config-if)# pvc 2/2
```
The second example displays the commands sent to the SAR and the acknowledgments returned when the VC is deleted and the segmentation and reassembly channels are closed:

```
Router(config-if)# no pvc 2/2
```

Examples

The first example illustrates the output from the `debug atm t3e3 trace` command when it is run without the `debug atm t3e3 sar` command being activated:

```
Router# debug atm t3e3 trace
SAR CMD/ACK debugging is on
Router# configure terminal
Router(config)# interface atm 2/0
router(config-if)# pvc 2/2
Router(config-if-atm-vc)# exit
```

The table below describes the significant fields shown in the display.
Table 33: debug atm t3e3 trace Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date or timestamp for the command dialog.</td>
<td></td>
</tr>
<tr>
<td>word [0 - n]: 0xXXXXXXXX</td>
<td>Hexadecimal representation of the n-word command sent to the SAR (under Command Sent:) and the four-word acknowledgment returned by the SAR (under Command Indication):</td>
</tr>
<tr>
<td>ACK received</td>
<td>Time (in microseconds) between sending the command to the SAR and receiving the acknowledgment.</td>
</tr>
</tbody>
</table>

The second example illustrates the output from the `debugatmt3e3trace` command run in conjunction with the `debugatmt3e3sar` command.

In this example, each command sent to the SAR is displayed by the `debugatmt3e3sar` command. Then the hexadecimal representation of the command and its acknowledgement are displayed by the `debugatmt3e3trace` command.

```
Router# debug atm t3e3 trace
SAR CMD/ACK debugging is on
Router# debug atm t3e3 sar
SAR debugging is on
Router# configure terminal
Router(config)# interface atm 2/0
router(config-if)# pvc 2/2
Router(config-if-atm-vc)# exit
Router(config-if)#
*Jun 27 22:15:09.284: ATM2/0: Open_Channel(RSY): CH (1), VPI (2), VCI (2)
*Jun 27 22:15:09.284: Command Sent:
*Jun 27 22:15:09.284: word 0: 0x00000480
*Jun 27 22:15:09.284: word 1: 0x00012010
*Jun 27 22:15:09.284: word 2: 0x00000000
*Jun 27 22:15:09.284: word 3: 0x00000000
*Jun 27 22:15:09.284: word 4: 0x00000000
*Jun 27 22:15:09.284: word 5: 0x00000000
*Jun 27 22:15:09.284: word 6: 0x00000000
*Jun 27 22:15:09.284: word 7: 0x00000000
*Jun 27 22:15:09.284: word 8: 0x00000000
*Jun 27 22:15:09.284: word 9: 0x00000000
*Jun 27 22:15:09.284: Command Indication:
*Jun 27 22:15:09.284: word 0: 0x00000000
*Jun 27 22:15:09.284: word 1: 0x01042110
*Jun 27 22:15:09.284: word 2: 0x01050000
*Jun 27 22:15:09.284: word 3: 0x0000003B
*Jun 27 22:15:09.284: word 4: 0x00200020
*Jun 27 22:15:09.284: word 5: 0x64B30000
*Jun 27 22:15:09.284: word 6: 0x10C00000
*Jun 27 22:15:09.284: word 7: 0x86850000
*Jun 27 22:15:09.284: word 8: 0x00010040
*Jun 27 22:15:09.284: word 9: 0x00000000
```
*Jun 27 22:15:09.284: Command Indication:
*Jun 27 22:15:09.284: word 0: 0x00010000
*Jun 27 22:15:09.284: word 1: 0x00011110
*Jun 27 22:15:09.284: word 2: 0x02000000
*Jun 27 22:15:09.284: word 3: 0x0001003D
*Jun 27 22:15:09.284: ACK received - 200 usecs
debug audit

To display debug messages for the audit subsystem, use the **debug audit** command in privileged EXEC mode. To disable debugging for the audit subsystem, use the **no** form of this command.

**debug audit**

**no debug audit**

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(18)S</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.0(27)S</td>
<td>This feature was integrated into Cisco IOS Release 12.0(27)S.</td>
</tr>
<tr>
<td>12.2(27)SBC</td>
<td>This command was integrated into Cisco IOS Release 12.2(27)SBC.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Audit files allow you to track changes that have been made to your router. Each change is logged as a syslog message, and all syslog messages are kept in the audit file, which is kept in the audit subsystem.

**Examples**

The following example is sample output from the **debug audit** command:

```
Router# debug audit
*Sep 14 18:37:31.535:disk0:/forensics.log -> File not found
*Sep 14 18:37:31.535:%AUDIT-1-RUN_VERSION:Hash:24D96B13B870106E76A7E5D1B3E80AD User:
*Sep 14 18:37:31.583:%AUDIT-1-RUN_CONFIG:Hash:4AC2D776AA6FC86FD763C3EB869B699 User:
*Sep 14 18:37:31.597:Audit:Trying to hash nvram:startup-config
*Sep 14 18:37:31.587:Audit:Trying to hash nvram:private-config
*Sep 14 18:37:31.591:Audit:Trying to hash nvram:underlying-config
*Sep 14 18:37:31.591:Audit:Trying to hash nvram:underlying-config
*Sep 14 18:37:31.595:Audit:Trying to hash nvram:ifIndex-table
*Sep 14 18:37:31.595:Audit:Skipping nvram:ifIndex-table
*Sep 14 18:37:31.595:%AUDIT-1-STARTUP_CONFIG:Hash:95DD497B1BB61AB33A629124CBFEC0FC User:
*Sep 14 18:37:31.591:Audit:Trying to hash filesystem disk0:
*Sep 14 18:37:31.775:Audit:Trying to hash attributes of disk0:c7200-p-mz.120-23.S
*Sep 14 18:37:32.103:Audit:Try c7200-p-mz.120-23.S DONE
```
*Sep 14 18:37:32.103:Audit:disk0:DONE
*Sep 14 18:37:32.103:Audit:Trying to hash filesystem bootflash:
  bootflash:c7200-kboot-mz.121-8a.E
*Sep 14 18:37:32.107:Audit:Trying to hash attributes of
  bootflash:crashinfo_20030115-182547 DONE
*Sep 14 18:37:32.107:Audit:Trying to hash attributes of
  bootflash:crashinfo_20030115-212157 DONE
*Sep 14 18:37:32.107:Audit:Trying to hash attributes of
  bootflash:crashinfo_20030603-155534 DONE
*Sep 14 18:37:32.107:Audit:Trying to hash attributes of
  bootflash:crashinfo_20030603-155534 DONE
*Sep 14 18:37:32.107:Audit:Trying to hash attributes of
  bootflash:crashinfo_20030603-155534 DONE
*Sep 14 18:37:32.107:Audit:Hashing entitymib entry for 7206VXR chassis,
  Hw Serial#:28710795, Hw Revision:A
*Sep 14 18:37:32.107:Audit:Hashing entitymib entry for NPE 400 Card, Hw
  Serial#:28710795, Hw Revision:A
*Sep 14 18:37:32.107:Audit:Hashing entitymib entry for I/O Dual
  FastEthernet Controller
*Sep 14 18:37:32.107:Audit:Hashing entitymib entry for 182543
  (Livengood)
*Sep 14 18:37:32.107:Audit:Hashing entitymib entry for 182543
  (Livengood)
*Sep 14 18:37:32.107:Audit:Hashing entitymib entry for Chassis Slot
  (Livengood)
*Sep 14 18:37:32.107:Audit:Hashing entitymib entry for Chassis Slot
  (Livengood)
*Sep 14 18:37:32.107:Audit:Hashing entitymib entry for Chassis Slot
  (Livengood)
*Sep 14 18:37:32.107:Audit:Hashing entitymib entry for Chassis Slot
  (Livengood)
*Sep 14 18:37:32.107:Audit:Hashing entitymib entry for Chassis Slot
  (Livengood)
*Sep 14 18:37:32.107:Audit:Hashing entitymib entry for Chassis Slot
  (Livengood)
*Sep 14 18:37:32.107:Audit:Hashing entitymib entry for Chassis Slot
  (Livengood)
*Sep 14 18:37:32.107:Audit:Hashing entitymib entry for Chassis Slot
  (Livengood)
*Sep 14 18:37:32.107:%AUDIT-1-HARDWARE_CONFIG:Hash:
  User:

---

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>audit filesize</td>
<td>Changes the size of the audit file.</td>
</tr>
<tr>
<td>audit interval</td>
<td>Changes the time interval that is used for calculating hashes.</td>
</tr>
</tbody>
</table>
debug authentication

To display debugging information about the Authentication Manager, use the `debug authentication` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debuge authentication {[feature feature-name] | {all| detail| errors| events| sync}}
no debug authentication {[feature feature-name] | {all| detail| errors| events| sync}}
```

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>feature feature-name</td>
<td>Displays debugging information about specific features. To display the valid feature names, use the question mark (?) online help function.</td>
</tr>
<tr>
<td>all</td>
<td>Displays all debugging information about the Authentication Manager and all features.</td>
</tr>
<tr>
<td>detail</td>
<td>Displays detailed debugging information.</td>
</tr>
<tr>
<td>errors</td>
<td>Displays debugging information about Authentication Manager errors.</td>
</tr>
<tr>
<td>events</td>
<td>Displays debugging information about Authentication Manager events.</td>
</tr>
<tr>
<td>sync</td>
<td>Displays debugging information about Authentication Manager stateful switchovers (SSOs) or In Service Software Upgrades (ISSUs).</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SXI</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>15.2(2)T</td>
<td>This command was integrated into Cisco IOS Release 15.2(2)T.</td>
</tr>
<tr>
<td>Cisco IOS XE Release 3.2SE</td>
<td>This command was modified. The <code>detail</code> keyword was added.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use the `debug authentication` command to troubleshoot the Authentication Manager.
The following example shows sample output from the `debug authentication` command when the `feature` and `events` keywords are configured:

```
Device# debug authentication feature mda events
Auth Manager mda events debugging is on
```

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug access-session</code></td>
<td>Displays debugging information about Session Aware Networking sessions.</td>
</tr>
<tr>
<td><code>debug dot1x</code></td>
<td>Displays 802.1x debugging information.</td>
</tr>
</tbody>
</table>
debug auto-config

To enable debugging for autoconfiguration applications, use the `debug auto-config` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
default auto-config {all | errors | events | parser}
no default auto-config {all | errors | events | parser}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>all</code></td>
<td>Displays all autoconfiguration debug trace.</td>
</tr>
<tr>
<td><code>errors</code></td>
<td>Displays autoconfiguration errors.</td>
</tr>
<tr>
<td><code>events</code></td>
<td>Displays autoconfiguration events.</td>
</tr>
<tr>
<td><code>parser</code></td>
<td>Displays autoconfiguration parser.</td>
</tr>
</tbody>
</table>

**Command Default**

Disabled

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(8)XY</td>
<td>This command was introduced on the Communication Media Module.</td>
</tr>
<tr>
<td>12.3(14)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(14)T.</td>
</tr>
<tr>
<td>12.4(3)</td>
<td>This command was integrated into Cisco IOS Release 12.4(3).</td>
</tr>
</tbody>
</table>

**Examples**

The following example shows the `debug auto-config` command used to enable debugging for autoconfiguration applications and to display autoconfiguration events:

```
Router# debug auto-config events
.
Feb 8 02:17:31.119: dnld_app_check_state (0x628C8164) ...
Feb 8 02:17:31.123: dnld_chk_app_handle (0x628C8164)
Feb 8 02:17:31.123: dnld_app_check_state: appl = 0x628C8164, state = 0x11
.
.
```

The table below describes significant fields shown in the display.
Table 34: debug auto-config Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x628C8164</td>
<td>Identifies the application handle, an auto-generated number for debugging.</td>
</tr>
<tr>
<td>0x11</td>
<td>Displays the state of the application. State values are as follows:</td>
</tr>
<tr>
<td></td>
<td>0x11--Registered and enabled.</td>
</tr>
<tr>
<td></td>
<td>0x1--Download application enabled.</td>
</tr>
<tr>
<td></td>
<td>0x10--Download application registered.</td>
</tr>
</tbody>
</table>

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto-config</td>
<td>Enables autoconfiguration or enters auto-config application configuration mode for the SCCP application.</td>
</tr>
<tr>
<td>debug sccp config</td>
<td>Enables SCCP event debugging.</td>
</tr>
<tr>
<td>show auto-config</td>
<td>Displays the current status of autoconfiguration applications.</td>
</tr>
</tbody>
</table>
debug auto-ip-ring

To display debugging information about a device in an auto-IP ring, use the debug auto-ip-ring command in privileged EXEC mode. To disable the debugging output, use the no form of this command.

```
deq debug auto-ip-ring {ring-id {errors | events} | errors | events}
no debug auto-ip-ring {ring-id {errors | events} | errors | events}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ring-id</td>
<td>Auto-IP ring identification number.</td>
</tr>
<tr>
<td>errors</td>
<td>Enables debugging output for auto-IP ring errors.</td>
</tr>
<tr>
<td>events</td>
<td>Enables debugging output for auto-IP ring events.</td>
</tr>
</tbody>
</table>

**Command Default**

Auto-IP ring debugging is disabled.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS XE Release 3.10S</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>15.3(3)S</td>
<td>This command was integrated into Cisco IOS Release 15.3(3)S</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

To display debugging output for errors or events on all the auto-IP-enabled node interfaces for a device, use the debug auto-ip-ring command without the ring-id argument.

To display debugging output for errors or events for a specific auto-IP ring, use the debug auto-ip-ring command with the ring-id argument.

**Examples**

The following is sample output for the debug auto-ip-ring command:

```
Device> enable
Device# debug auto-ip-ring 1 errors

Auto IP Ring errors debugging is on for the ring id : 1
*Jul 26 11:30:40.541: (Ethernet0/0) priority (value:1) conflict detected, need admin intervention
```
A conflict is detected in this example because the auto-IP TLV priority that is sent from the device and the priority that is received from the neighbor device is the same.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto-ip-ring</td>
<td>Enables the auto-IP functionality on the interfaces of a device.</td>
</tr>
<tr>
<td>show auto-ip-ring</td>
<td>Displays auto-IP ring information.</td>
</tr>
</tbody>
</table>
debug autoupgrade

To display the debug output of the Cisco Auto-Upgrade Manager (AUM), use the debug autoupgrade command in privileged EXEC mode. To disable debug output, use the no form of this command.

debug autoupgrade
no debug autoupgrade

Syntax Description
This command has no arguments or keywords.

Command Default
The debug output is disabled.

Command Modes
Privileged EXEC (#)

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(15)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>Cisco IOS XE Release 3.9S</td>
<td>This command was integrated into Cisco IOS XE Release 3.9S.</td>
</tr>
</tbody>
</table>

Usage Guidelines
Use the debug autoupgrade command when you encounter a problem with AUM and provide the output to TAC. Run the debug autoupgrade command and then run AUM to view the debug messages.

Examples
The following example shows how to enable the debugging of Cisco Auto-Upgrade Manager:

Device# debug autoupgrade
Auto Upgrade Manager debugging ON
Device#
Device# upgrade automatic getversion tftp://10.1.0.1/username/aaa
Image not found.
Device#
Jun 14 14:23:08.251 IST: AUM: Currently running software:
flash:c3825-adventerprisek9-mz.CALVIN_AUM_EFT1
Jun 14 14:23:08.251 IST: AUM: Reload type:2 hour:0 min:0
Jun 14 14:23:08.251 IST: AUM: Disk management: 1
Jun 14 14:23:08.251 IST: AUM: Get image tftp://10.1.0.1/username/aaa from local server and upgrade:
Jun 14 14:23:08.251 IST: AUM: Extracted image name: aaa
Jun 14 14:23:08.339 IST: AUM: get image info: failed to open url
**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>upgrade automatic getversion</td>
<td>Downloads a Cisco software image directly from <a href="http://www.cisco.com">www.cisco.com</a> or from a non-Cisco server.</td>
</tr>
</tbody>
</table>
debug backhaul-session-manager session through debug channel packets

- debug backhaul-session-manager session through debug channel packets, page 165
debug backhaul-session-manager session

To debug all the available sessions or a specified session, use the `debug backhaul-session-manager session` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug backhaul-session-manager session {state|xport} {all|session-id}
no debug backhaul-session-manager session {state|xport} {all|session-id}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>state</code></td>
<td>Shows information about state transitions. Possible states are as follows:</td>
</tr>
<tr>
<td></td>
<td>SESS_SET_IDLE: A session-set has been created.</td>
</tr>
<tr>
<td></td>
<td>SESS_SET_OOS: A session(s) has been added to session-group(s). No ACTIVE notification has been received from Virtual Switch Controller (VSC).</td>
</tr>
<tr>
<td></td>
<td>SESS_SET_ACTIVE_IS: An ACTIVE notification has been received over one in-service session-group. STANDBY notification has not been received on any available session-group(s).</td>
</tr>
<tr>
<td></td>
<td>SESS_SET_STANDBY_IS: A STANDBY notification is received, but there is no in-service active session-group available.</td>
</tr>
<tr>
<td></td>
<td>SESS_SET_FULL_IS: A session-group in-service that has ACTIVE notification and at least one session-group in-service that has STANDBY notification.</td>
</tr>
<tr>
<td></td>
<td>SESS_SET_SWITCH_OVER: An ACTIVE notification is received on session-group in-service, which had received STANDBY notification.</td>
</tr>
<tr>
<td><code>xport</code></td>
<td>Provides traces for all packets (protocol data units (PDUs)), application PDUs, and also session-manager messages.</td>
</tr>
<tr>
<td><code>all</code></td>
<td>All available sessions.</td>
</tr>
<tr>
<td><code>session-id</code></td>
<td>A specified session.</td>
</tr>
</tbody>
</table>

**Command Default**

Debugging for backhaul-session-manager session is not enabled.

**Command Modes**

Privileged EXEC
Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(1)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)T</td>
<td>Support for this command was introduced on the Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(4)T</td>
<td>This command was implemented on the Cisco 2600 series, Cisco 3600 series, and Cisco MC3810.</td>
</tr>
<tr>
<td>12.2(2)XB</td>
<td>This command was implemented on the Cisco AS5350 and Cisco AS5400.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850 platform.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td>This command was implemented on Cisco IAD2420 series integrated access devices (IADs). This command is not supported on the access servers in this release.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was implemented on Cisco AS5350, Cisco AS5400, and Cisco AS5850 platforms.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

Examples

The following is output for the `debug backhaul-session-manager session all` command:

```
Router# debug backhaul-session-manager session all
23:49:14:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
23:49:19:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
23:49:19:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
23:49:24:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
23:49:24:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
23:49:29:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
23:49:29:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
23:49:34:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
23:49:34:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
23:49:34:SESSION:XPORT:sig rcvd. session = 33, connid = 0x80BA14EC, sig = 1 (CONN-FAILED)
```

The following example displays output for the `debug backhaul-session-manager session state all` command:

```
Router# debug backhaul-session-manager session state all
```
The following example displays output for the `debug backhaul-session-manager session xport all` command:

```
Router# debug backhaul-session-manager session xport all
23:51:39:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
23:51:42:SESSION:XPORT:sig rcvd. session = 33, connid = 0x80BA14EC, sig = 5 (CONN-RESET)
23:51:44:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
```

Use caution when enabling this debug command in a live system. It produces significant amounts of output, which could lead to a disruption of service.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug backhaul-session-manager set</code></td>
<td>Traces state changes and receives messages and events for all available session-sets or a specified session-set.</td>
</tr>
</tbody>
</table>
**debug backhaul-session-manager set**

To trace state changes and receive messages and events for all the available session sets or a specified session set, use the `debug backhaul-session-manager set` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug backhaul-session-manager set {all|name set-name}
no debug backhaul-session-manager set {all|name set-name}
```

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>All available session sets.</td>
</tr>
<tr>
<td>name set-name</td>
<td>A specified session set.</td>
</tr>
</tbody>
</table>

**Command Default**

Debugging for backhaul session sets is not enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(1)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)T</td>
<td>Support for this command was introduced on the Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(4)T</td>
<td>This command was implemented on the Cisco 2600 series, Cisco 3600 series, and Cisco MC3810.</td>
</tr>
<tr>
<td>12.2(2)XB</td>
<td>This command was implemented on the Cisco AS5350 and Cisco AS5400.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850 platform.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td>This command was implemented on Cisco IAD2420 series integrated access devices (IADs). This command is not supported on the access servers in this release.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was implemented on Cisco AS5350, Cisco AS5400, and Cisco AS5850 platforms.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>
Examples

The following is output for the `debug backhaul-session-manager set` command for all available session sets:

```
Router# debug backhaul-session-manager set all
Router# debug bsm_command:DEBUG_BSM_SET_ALL
    Function set_proc_event() is called
Session-Set : test-set
Old State  : BSM_SET_OOS
New State  : BSM_SET_OOS
    Active-Grp : NONE
    Session-Grp : g-11
Old State  : Group-None
New State  : Group-None
Event rcvd : EVT_GRP_INS
BSM:Event BSM_SET_UP is sent to user
Session-Set : test-set
Old State  : BSM_SET_OOS
New State  : BSM_SET_OOS
    Active-Grp : g-11
    Session-Grp : g-11
Old State  : Group-None
New State  : Group-Active
Event rcvd : BSM_ACTIVE_TYPE
```

The following is output for the `debug backhaul-session-manager set name set1` command:

```
Router# debug backhaul-session-manager set name set1
Router# debug bsm_command:DEBUG_BSM_SET_NAME
Router# Function set_proc_event() is called
Session-Set : test-set
Old State  : BSM_SET_OOS
New State  : BSM_SET_OOS
    Active-Grp : NONE
    Session-Grp : g-11
Old State  : Group-None
New State  : Group-None
Event rcvd : EVT_GRP_INS
Router#BSM:Event BSM_SET_UP is sent to user
Session-Set : test-set
Old State  : BSM_SET_OOS
New State  : BSM_SET_OOS
    Active-Grp : g-11
    Session-Grp : g-11
Old State  : Group-None
New State  : Group-Active
Event rcvd : BSM_ACTIVE_TYPE
```

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug backhaul-session-manager session</code></td>
<td>Debugs all available sessions or a specified session.</td>
</tr>
</tbody>
</table>
debug backup

To monitor the transitions of an interface going down then back up, use the `debug backup` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug backup
no debug backup
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The `debug backup` command is useful for monitoring dual X.25 interfaces configured as primary and backup in a Telco data communication network (DCN).

**Examples**

The following example shows how to start the `debug backup` command:

```
Router# debug backup
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backup active interface</td>
<td>Activates primary and backup lines on specific X.25 interfaces.</td>
</tr>
<tr>
<td>show backup</td>
<td>Displays interface backup status.</td>
</tr>
</tbody>
</table>
**debug bert**

To display information on the bit error rate testing (BERT) feature, use the `debug bert` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug bert
no debug bert
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(2)XD</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The `debug bert` command output is used primarily by Cisco technical support representatives. The `debug bert` command displays debugging messages for specific areas of executed code.

**Examples**

The following is output from the `debug bert` command:

```
Router# debug bert
Bit Error Rate Testing debugging is on
Router# no debug bert
Bit Error Rate Testing debugging is off
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bert abort</td>
<td>Aborts a bit error rate testing session.</td>
</tr>
<tr>
<td>bert controller</td>
<td>Starts a bit error rate test for a particular port on a Cisco AS5300 router.</td>
</tr>
<tr>
<td>bert profile</td>
<td>Sets up various bit error rate testing profiles.</td>
</tr>
</tbody>
</table>
**debug bfd**

To display debugging messages about Bidirectional Forwarding Detection (BFD), use the `debug bfd` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

**Cisco IOS Release 12.2(18)SXE, 12.4(4)T, and 12.2(33)SRA**

```
dbg bfd {event| packet [ip-address| ipv6-address]}
n dbg bfd {event| packet [ip-address| ipv6-address]}
```

**Cisco IOS Release 12.0(31)S**

```
dbg bfd {event| packet [ ip-address ]] ipc-error| ipc-event| oir-error| oir-event}
n dbg bfd {event| packet [ ip-address ]] ipc-error| ipc-event| oir-error| oir-event}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>event</code></td>
<td>Displays debugging information about BFD state transitions.</td>
</tr>
<tr>
<td><code>packet</code></td>
<td>Displays debugging information about BFD control packets.</td>
</tr>
<tr>
<td><code>ip-address</code></td>
<td>(Optional) Displays debugging information about BFD only for the specified IP address.</td>
</tr>
<tr>
<td><code>ipv6-address</code></td>
<td>(Optional) Displays debugging information about BFD only for the specified IPv6 address.</td>
</tr>
<tr>
<td><code>ipc-error</code></td>
<td>(Optional) Displays debugging information with interprocess communication (IPC) errors on the Route Processor (RP) and line card (LC).</td>
</tr>
<tr>
<td><code>ipc-event</code></td>
<td>(Optional) Displays debugging information with IPC events on the RP and LC.</td>
</tr>
<tr>
<td><code>oir-error</code></td>
<td>(Optional) Displays debugging information with online insertion and removal (OIR) errors on the RP and LC.</td>
</tr>
<tr>
<td><code>oir-event</code></td>
<td>(Optional) Displays debugging information with OIR events on the RP and LC.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC
**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(18)SXE</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.0(31)S</td>
<td>This command was integrated into Cisco IOS Release 12.0(31)S.</td>
</tr>
<tr>
<td>12.4(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(4)T.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2(33)SRE</td>
<td>This command was modified. Support for IPv6 was added.</td>
</tr>
<tr>
<td>15.1(2)T</td>
<td>This command was modified. Support for IPv6 was added to Cisco IOS Release 15.1(2)T.</td>
</tr>
<tr>
<td>15.1(1)SG</td>
<td>This command was integrated into Cisco IOS Release 15.1(1)SG.</td>
</tr>
<tr>
<td>15.1(1)SY</td>
<td>This command was modified. Support for IPv6 was added to Cisco IOS Release 15.1(1)SY.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The **debug bfd** command can be used to troubleshoot the BFD feature.

Because BFD is designed to send and receive packets at a very high rate of speed, consider the potential effect on system resources before enabling this command, especially if there are a large number of BFD peers. The **debug bfd packet** command should be enabled only on a live network at the direction of Cisco Technical Assistance Center personnel.

**Examples**

The following example shows output from the **debug bfd packet** command. The IP address has been specified in order to limit the packet information to one interface:

```
Router# debug bfd packet 172.16.10.5
BFD packet debugging is on
*Jan 26 14:47:37.645: Tx*IP: dst 172.16.10.1, plen 24. BFD: diag 2, St/D/P/F (1/0/0/0),
  mult 5, len 24, loc/rem discr 1 1, tx 1000000, rx 1000000 100000, timer 1000 ms, #103
*Jan 26 14:47:37.645: %OSPF-5-ADJCHG: Process 10, Nbr 172.16.10.12 on Ethernet1/4 from FULL
to DOWN, Neighbor Down: BFD node down
*Jan 26 14:47:50.685: %OSPF-5-ADJCHG: Process 10, Nbr 172.16.10.12 on Ethernet1/4 from LOADING
to FULL, Loading Done
*Jan 26 14:48:00.905: Rx IP: src 172.16.10.1, plen 24. BFD: diag 0, St/D/P/F (1/0/0/0),
  mult 4, len 24, loc/rem discr 2 1, tx 1000000, rx 1000000 100000, timer 4000 ms, #50
*Jan 26 14:48:00.905: %OSPF-5-ADJCHG: Process 10, Nbr 172.16.10.12 on Ethernet1/4 from LOADING
to FULL, Loading Done
*Jan 26 14:48:00.905: Rx IP: src 172.16.10.1, plen 24. BFD: diag 0, St/D/P/F (2/0/0/0),
  mult 5, len 24, loc/rem discr 1 2, tx 1000000, rx 1000000 100000, timer 1000 ms, #131
*Jan 26 14:48:00.905: Rx IP: src 172.16.10.1, plen 24. BFD: diag 0, St/D/P/F (3/0/0/0),
  mult 4, len 24, loc/rem discr 2 1, tx 1000000, rx 1000000 100000, timer 4000 ms, #51
*Jan 26 14:48:00.905: %OSPF-5-ADJCHG: Process 10, Nbr 172.16.10.12 on Ethernet1/4 from LOADING
to FULL, Loading Done
```
The following example shows output from the `debug bfd event` command when an interface between two BFD neighbor routers fails and then comes back online:

```
Router# debug bfd event
22:53:49: Session [172.16.10.1,172.16.10.2,Fa0/1,1], event DETECT TIMER EXPIRED, state UP -> FAILING
22:56:37: Session [172.16.10.1,172.16.10.2,Fa0/1,1], event RX IHY 0, state FAILING -> DOWN
22:56:37: Session [172.16.10.1,172.16.10.2,Fa0/1,1], event RX IHY 0, state DOWN -> INIT
22:56:37: Session [172.16.10.1,172.16.10.2,Fa0/1,1], event RX IHY 1, state INIT -> UP
```

The table below describes the significant fields shown in the display.

### Table 35: debug bfd event Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bfd_neighbor - action:DESTROY</td>
<td>The BFD neighbor will tear down the BFD session.</td>
</tr>
<tr>
<td>Session [172.16.10.1, 172.16.10.2, Fa0/1,1]</td>
<td>IP addresses of the BFD neighbors holding this session that is carried over FastEthernet interface 0/1.</td>
</tr>
<tr>
<td>event DETECT TIMER EXPIRED</td>
<td>The BFD neighbor has not received BFD control packets within the negotiated interval and the detect timer has expired.</td>
</tr>
<tr>
<td>state UP -&gt; FAILING</td>
<td>The BFD event state is changing from Up to Failing.</td>
</tr>
<tr>
<td>Session [172.16.10.1, 172.16.10.2, Fa0/1,1], event RX IHY 0</td>
<td>The BFD session between the neighbors indicated by the IP addresses that is carried over FastEthernet interface 0/1 is changing state from Failing to Down. The I Hear You (IHY) bit value is shown as 0 to indicate that the remote system is tearing down the BFD session.</td>
</tr>
<tr>
<td>event RX IHY 0, state DOWN -&gt; INIT</td>
<td>The BFD session is still considered down, and the IHY bit value still is shown as 0, and the session state changes from DOWN to INIT to indicate that the BFD session is again initializing, as the interface comes back up.</td>
</tr>
<tr>
<td>event RX IHY 1, state INIT -&gt; UP</td>
<td>The BFD session has been reestablished, and the IHY bit value changes to 1 to indicate that the session is live. The BFD session state changes from INIT to UP.</td>
</tr>
</tbody>
</table>
The following example shows output from the **debug bfd packet** command when an interface between two BFD neighbor routers fails and then comes back online. The diagnostic code changes from 0 (No Diagnostic) to 1 (Control Detection Time Expired) because no BFD control packets could be sent (and therefore detected by the BFD peer) after the interface fails. When the interface comes back online, the diagnostic code changes back to 0 to signify that BFD packets can be sent and received by the BFD peers.

```
Router# debug bfd packet
23:03:25: Rx IP: src 172.16.10.2, plen 24. BFD: diag 0, H/D/P/F (0/0/0/0), mult 3, len 24, loc/rem discr 5 1, tx 1000000, rx 100007
23:03:25: Tx IP: dst 172.16.10.2, plen 24. BFD: diag 1, H/D/P/F (0/0/0/0), mult 5, len 24, loc/rem discr 1 5, tx 1000000, rx 1000008
23:03:25: Tx IP: dst 172.16.10.2, plen 24. BFD: diag 1, H/D/P/F (1/0/0/0), mult 5, len 24, loc/rem discr 1 5, tx 1000000, rx 1000009
```

The table below describes the significant fields shown in the display.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rx IP: src 172.16.10.2</td>
<td>The router has received this BFD packet from the BFD router with source address 172.16.10.2.</td>
</tr>
<tr>
<td>plen 24</td>
<td>Length of the BFD control packet, in bytes.</td>
</tr>
<tr>
<td>diag 0</td>
<td>A diagnostic code specifying the local system’s reason for the last transition of the session from Up to some other state.</td>
</tr>
</tbody>
</table>

State values are as follows:

- 0—No Diagnostic
- 1—Control Detection Time Expired
- 2—Echo Function Failed
- 3—Neighbor Signaled Session Down
- 4—Forwarding Plane Reset
- 5—Path Down
- 6—Concentrated Path Down
- 7—Administratively Down
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H/D/P/F (0/0/0/0)</td>
<td>H bit--Hear You bit. This bit is set to 0 if the transmitting system either is not receiving BFD packets from the remote system or is tearing down the BFD session. During normal operation the H Hear You bit is set to 1. D bit--Demand Mode bit. If the Demand Mode bit set, the transmitting system wants to operate in demand mode. BFS has two modes--asynchronous and demand. The Cisco implementation of BFD supports only asynchronous mode. P bit--Poll bit. If the Poll bit is set, the transmitting system is requesting verification of connectivity or of a parameter change. F bit--Final bit. If the Final bit is set, the transmitting system is responding to a received BFC control packet that had a Poll (P) bit set.</td>
</tr>
<tr>
<td>mult 3</td>
<td>Detect time multiplier. The negotiated transmit interval, multiplied by the detect time multiplier, determines the detection time for the transmitting system in BFD asynchronous mode. The detect time multiplier is similar to the hello multiplier in IS-IS, which is used to determine the hold timer: (hellointerval) * (hellomultiplier) = hold timer. If a hello packet is not received within the hold-timer interval, a failure has occurred. Similarly, for BFD: (transmit interval) * (detect multiplier) = detect timer. If a BFD control packet is not received from the remote system within the detect-timer interval, a failure has occurred.</td>
</tr>
<tr>
<td>len 24</td>
<td>The BFD packet length.</td>
</tr>
</tbody>
</table>
| loc/rem discr 5 1 | The values for My Discriminator (local) and Your Discriminator (remote) BFD neighbors.  
  - My Discriminator--Unique, nonzero discriminator value generated by the transmitting system, used to demultiplex multiple BFD sessions between the same pair of systems.  
  - Your Discriminator--The discriminator received from the corresponding remote system. This field reflects the received value of My Discriminator, or is zero if that value is unknown. |
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tx 1000000</td>
<td>Desired minimum transmit interval.</td>
</tr>
<tr>
<td>rx 100007</td>
<td>Required minimum receive interval.</td>
</tr>
</tbody>
</table>
debug bgp ipv6 dampening

To display debugging messages for IPv6 Border Gateway Protocol (BGP) dampening, use the `debug bgp ipv6 dampening` command in privileged EXEC mode. To disable debugging messages for IPv6 BGP dampening, use the `no` form of this command.

`debug bgp ipv6 {unicast|multicast} dampening [prefix-list prefix-list-name]`

`no debug bgp ipv6 {unicast|multicast} dampening [prefix-list prefix-list-name]`

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>unicast</code></td>
<td>Specifies IPv6 unicast address prefixes.</td>
</tr>
<tr>
<td><code>multicast</code></td>
<td>Specifies IPv6 multicast address prefixes.</td>
</tr>
<tr>
<td><code>prefix-list</code></td>
<td>(Optional) Name of an IPv6 prefix list.</td>
</tr>
<tr>
<td><code>prefix-list-name</code></td>
<td>(Optional) Name of an IPv6 prefix list.</td>
</tr>
</tbody>
</table>

**Command Default**

Debugging for IPv6 BGP dampening packets is not enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(2)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.0(21)ST</td>
<td>This command was integrated into Cisco IOS Release 12.0(21)ST.</td>
</tr>
<tr>
<td>12.0(22)S</td>
<td>This command was integrated into Cisco IOS Release 12.0(22)S.</td>
</tr>
<tr>
<td>12.2(13)T</td>
<td>The <code>prefix-list</code> keyword was added.</td>
</tr>
<tr>
<td>12.0(24)S</td>
<td>The <code>prefix-list</code> keyword was added.</td>
</tr>
<tr>
<td>12.2(14)S</td>
<td>This command was integrated into Cisco IOS Release 12.2(14)S.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(28)SB.</td>
</tr>
<tr>
<td>12.2(25)SG</td>
<td>This command was integrated into Cisco IOS Release 12.2(25)SG.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2(33)SXH</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SXH.</td>
</tr>
<tr>
<td>Cisco IOS XE Release 2.1</td>
<td>This command was introduced on Cisco ASR 1000 Series Routers.</td>
</tr>
</tbody>
</table>
Usage Guidelines

The **debug bgp ipv6 dampening** command is similar to the **debug ip bgp dampening** command, except that it is IPv6-specific.

Use the **prefix-list** keyword and an argument to filter BGP IPv6 dampening debug information through an IPv6 prefix list.

By default, the network server sends the output from debug commands and system error messages to the console. To redirect debugging output, use the **logging** command options within global configuration mode. Destinations are the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server.

Examples

The following is sample output from the **debug bgp ipv6 dampening** command:

```
Router# debug bgp ipv6 dampening
00:13:28:BGP(1):flapped 1 times since 00:00:00. New penalty is 1000
00:13:28:BGP(1):flapped 1 times since 00:00:00. New penalty is 1000
00:13:28:BGP(1):flapped 1 times since 00:00:00. New penalty is 1000
00:16:03:BGP(1):charge penalty for 2000:0:0:1::/64 path 2 1 with halflife-time 15 reuse/suppress 750/2000
00:16:03:BGP(1):flapped 2 times since 00:02:35. New penalty is 1892
00:18:28:BGP(1):suppress 2000:0:0:1::/80 path 2 1 for 00:27:30 (penalty 2671)
00:18:28:halflife-time 15, reuse/suppress 750/2000
00:18:28:BGP(1):suppress 2000:0:0:1:1::/64 path 2 1 for 00:27:20 (penalty 2664)
00:18:28:halflife-time 15, reuse/suppress 750/2000
```

The following example shows output for the **debug bgp ipv6 dampening** command filtered through the prefix list named marketing:

```
Router# debug bgp ipv6 dampening prefix-list marketing
00:16:08:BGP(1):charge penalty for 2001:0DB8::/64 path 30 with halflife-time 15 reuse/suppress 750/2000
00:16:08:BGP(1):flapped 1 times since 00:00:00. New penalty is 10
```

The table below describes the fields shown in the display.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>penalty</td>
<td>Numerical value of 1000 assigned to a route by a router configured for route dampening in another autonomous system each time a route flaps. Penalties are cumulative. The penalty for the route is stored in the BGP routing table until the penalty exceeds the suppress limit. If the penalty exceeds the suppress limit, the route state changes from history to damp.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>flapped</td>
<td>Number of times a route is available, then unavailable, or vice versa.</td>
</tr>
<tr>
<td>halflife-time</td>
<td>Amount of time (in minutes) by which the penalty is decreased after the route is assigned a penalty. The halflife-time value is half of the half-life period (which is 15 minutes by default). Penalty reduction happens every 5 seconds.</td>
</tr>
<tr>
<td>reuse</td>
<td>The limit by which a route is unsuppressed. If the penalty for a flapping route decreases and falls below this reuse limit, the route is unsuppressed. That is, the route is added back to the BGP table and once again used for forwarding. The default reuse limit is 750. Routes are unsuppressed at 10-second increments. Every 10 seconds, the router determines which routes are now unsuppressed and advertises them to the world.</td>
</tr>
<tr>
<td>suppress</td>
<td>Limit by which a route is suppressed. If the penalty exceeds this limit, the route is suppressed. The default value is 2000.</td>
</tr>
<tr>
<td>maximum suppress limit (not shown in sample output)</td>
<td>Maximum amount of time (in minutes) a route is suppressed. The default value is four times the half-life period.</td>
</tr>
<tr>
<td>damp state (not shown in sample output)</td>
<td>State in which the route has flapped so often that the router will not advertise this route to BGP neighbors.</td>
</tr>
</tbody>
</table>

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug bgp ipv6 updates</td>
<td>Displays debugging messages for IPv6 BGP update packets.</td>
</tr>
</tbody>
</table>
debug bgp ipv6 updates

To display debugging messages for IPv6 Border Gateway Protocol (BGP) update packets, use the debug bgp ipv6 updates command in privileged EXEC mode. To disable debugging messages for IPv6 BGP update packets, use the no form of this command.

```
depug bgp ipv6 {unicast| multicast} updates [ipv6-address ] [prefix-list prefix-list-name] [in| out]
nob depug bgp ipv6 {unicast| multicast} updates [ipv6-address ] [prefix-list prefix-list-name] [in| out]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unicast</td>
<td>Specifies IPv6 unicast address prefixes.</td>
</tr>
<tr>
<td>multicast</td>
<td>Specifies IPv6 multicast address prefixes.</td>
</tr>
<tr>
<td>ipv6-address</td>
<td>(Optional) The IPv6 address of a BGP neighbor. This argument must be in the form documented in RFC 2373 where the address is specified in hexadecimal using 16-bit values between colons.</td>
</tr>
<tr>
<td>prefix-list prefix-list-name</td>
<td>(Optional) Name of an IPv6 prefix list.</td>
</tr>
<tr>
<td>in</td>
<td>(Optional) Indicates inbound updates.</td>
</tr>
<tr>
<td>out</td>
<td>(Optional) Indicates outbound updates.</td>
</tr>
</tbody>
</table>

**Command Default**

Debugging for IPv6 BGP update packets is not enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(2)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.0(21)ST</td>
<td>This command was integrated into Cisco IOS Release 12.0(21)ST.</td>
</tr>
<tr>
<td>12.0(22)S</td>
<td>This command was integrated into Cisco IOS Release 12.0(22)S.</td>
</tr>
<tr>
<td>12.2(13)T</td>
<td>The prefix-list keyword was added.</td>
</tr>
<tr>
<td>12.0(24)S</td>
<td>The prefix-list keyword was added.</td>
</tr>
<tr>
<td>12.2(14)S</td>
<td>This command was integrated into Cisco IOS Release 12.2(14)S.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(28)SB.</td>
</tr>
</tbody>
</table>
This command was integrated into Cisco IOS Release 12.2(25)SG. This command was integrated into Cisco IOS Release 12.2(33)SRA. This command was integrated into Cisco IOS Release 12.2(33)SXH.

This command was introduced on Cisco ASR 1000 Series Routers. Cisco IOS XE Release 2.1

### Usage Guidelines

The **debug bgp ipv6 updates** command is similar to the **debug ip bgp updates** command, except that it is IPv6-specific.

Use the **prefix-list** keyword to filter BGP IPv6 updates debugging information through an IPv6 prefix list.

By default, the network server sends the output from **debug** commands and system error messages to the console. To redirect debugging output, use the **logging** command options within global configuration mode. Destinations are the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server. For complete information on **debug** commands and redirecting debugging output, refer to the Release 12.2 Cisco IOS Debug Command Reference.

### Examples

The following is sample output from the **debug bgp ipv6 updates** command:

```
Router# debug bgp ipv6 updates
14:04:22:BGPF(1):2000:0:0:2::2 computing updates, afi 1, neighbor version 0, table version 1, starting at ::
14:04:22:BGPF(1):2000:0:0:2::2 update run completed, afi 1, ran for 0ms, neighbor version 0, start version 1, throttled to 1
14:04:19:BGPF(1):sourced route for 2000:0:0:2::1/64 path #0 changed (weight 32768)
14:04:19:BGPF(1):2000:0:0:2::1/64 route sourced locally
14:04:19:BGPF(1):2000:0:0:2:1::/80 route sourced locally
14:04:19:BGPF(1):2000:0:0:4:2/64 route sourced locally
14:04:22:BGPF(1):2000:0:0:2::2 computing updates, afi 1, neighbor version 1, table version 6, starting at ::
14:04:22:BGPF(1):2000:0:0:2::2 send UPDATE (format) 2000:0:0:2:1::1/64, next 2000:0:0:2:1::1, metric 0, path
14:04:22:BGPF(1):2000:0:0:2::2 send UPDATE (format) 2000:0:0:2:1/80, next 2000:0:0:2:1::1, metric 0, path
14:04:22:BGPF(1):2000:0:0:2::2 send UPDATE (prepend, chgflags:0x208) 2000:0:0:3:2/64, next 2000:0:0:2::1, metric 0, path
14:04:22:BGPF(1):2000:0:0:2::2 send UPDATE (prepend, chgflags:0x208) 2000:0:0:4:2/64, next 2000:0:0:2::1, metric 0, path
```

The following is sample output from the **debug bgp ipv6 updates prefix-list sales** command filtered through the prefix list named sales:

```
Router# debug bgp ipv6 updates prefix-list sales
00:18:26:BGPF(1):2000:8493:1::2 send UPDATE (prepend, chgflags:0x208) 7878:7878::/64, next 2001:0DB8::36C, metric 0, path
```

The table below describes the significant fields shown in the display.
### Table 38: `debug bgp ipv6 updates` Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGP(1):</td>
<td>BGP debugging for address family index (afi) 1.</td>
</tr>
<tr>
<td>afi</td>
<td>Address family index.</td>
</tr>
<tr>
<td>neighbor version</td>
<td>Version of the BGP table on the neighbor from which the update was received.</td>
</tr>
<tr>
<td>table version</td>
<td>Version of the BGP table on the router from which you entered the <code>debug bgp ipv6 updates</code> command.</td>
</tr>
<tr>
<td>starting at</td>
<td>Starting at the network layer reachability information (NLRI). BGP sends routing update messages containing NLRI to describe a route and how to get there. In this context, an NLRI is a prefix. A BGP update message carries one or more NLRI prefixes and the attributes of a route for the NLRI prefixes; the route attributes include a BGP next hop gateway address, community values, and other information.</td>
</tr>
<tr>
<td>route sourced locally</td>
<td>Indicates that a route is sourced locally and that updates are not sent for the route.</td>
</tr>
<tr>
<td>send UPDATE (format)</td>
<td>Indicates that an update message for a reachable network should be formatted. Addresses include prefix and next hop.</td>
</tr>
<tr>
<td>send UPDATE (prepend, chgflags:0x208)</td>
<td>Indicates that an update message about a path to a BGP peer should be written.</td>
</tr>
</tbody>
</table>

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug bgp ipv6 dampening</code></td>
<td>Displays debugging messages for IPv6 BGP dampening packets.</td>
</tr>
</tbody>
</table>
**debug bgp l2vpn evpn updates**

To display information related to the Border Gateway Protocol (BGP) update messages, use the `debug bgp l2vpn evpn updates` command in privileged EXEC mode. To disable the display of such debugging information, use the `no` form of this command.

```
debug bgp l2vpn evpn updates [ip-address] [{access-list|expanded-access-list} [in|out]] [events|in|out]
no debug bgp l2vpn evpn updates [ip-address] [{access-list|expanded-access-list} [in|out]|events|in|out]
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ip-address</code></td>
<td>(Optional) BGP neighbor address in the format A.B.C.D.</td>
</tr>
<tr>
<td><code>access-list</code></td>
<td>(Optional) Number of the access list used to filter debugging messages. The range is from 1 to 199.</td>
</tr>
<tr>
<td><code>expanded-access-list</code></td>
<td>(Optional) Number of the expanded access list used to filter debugging messages. The range is from 1300 to 2699.</td>
</tr>
<tr>
<td><code>events</code></td>
<td>(Optional) Specifies debugging messages for BGP update events.</td>
</tr>
<tr>
<td><code>in</code></td>
<td>Specifies debugging messages for inbound BGP update information.</td>
</tr>
<tr>
<td><code>out</code></td>
<td>Specifies debugging messages for outbound BGP update information.</td>
</tr>
</tbody>
</table>

### Command Modes

Privileged EXEC (#)

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS XE Release 3.11S</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>15.4(1)S</td>
<td>This command was integrated into Cisco IOS Release 15.4(1)S.</td>
</tr>
</tbody>
</table>

### Examples

The following example shows how to enable the `debug bgp l2vpn evpn updates` command:

```
Device> enable
Device# debug bgp l2vpn evpn updates
```
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug ip bgp updates</code></td>
<td>Displays information about the processing of BGP updates.</td>
</tr>
<tr>
<td><code>show bgp l2vpn evpn</code></td>
<td>Displays L2VPN Ethernet VPN address family information from the BGP table.</td>
</tr>
</tbody>
</table>
debug bgp l2vpn vpls updates

To enable debugging of the L2VPN VPLS address family updates from the BGP table, use the `debug bgp l2vpn vpls updates` command in privileged EXEC mode. To disable the display of the messages, use the `no` form of this command.

```
ddebug bgp l2vpn vpls updates [access-list | expanded-access-list | bgp-neighbor-address | events | {in | out }]
nodebug bgp l2vpn vpls updates [access-list | expanded-access-list | bgp-neighbor-address | events | {in | out }]
```

**Syntax Description**

- `access-list` (Optional) Number of an access list used to filter debugging messages. The range is from 1 to 199.
- `expanded-access-list` (Optional) Number of an expanded access list used to filter debugging messages. The range is from 1300 to 2699.
- `bgp-neighbor-address` (Optional) BGP neighbor address in the format A.B.C.D.
- `events` (Optional) Specifies debugging messages for BGP update events.
- `in` Specifies debugging messages for inbound BGP update information.
- `out` Specifies debugging messages for outbound BGP update information.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS XE Release 3.8S</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following shows how to enable the `debug bgp l2vpn vpls updates` command:

```
Device> enable
Device# debug bgp l2vpn vpls updates
BGP updates debugging is on for address family: L2VPN Vpl
```
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug ip bgp updates</code></td>
<td>Displays information about the processing of BGP updates.</td>
</tr>
<tr>
<td><code>show bgp l2vpn vpls</code></td>
<td>Displays L2VPN VPLS address family information from the BGP table.</td>
</tr>
</tbody>
</table>
debug bgp nsap

To enable the display of Border Gateway Protocol (BGP) debugging information specific to the network service access point (NSAP) address family, use the `debug bgp nsap` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

**debug bgp nsap**

**no debug bgp nsap**

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Debugging of BGP NSAP address-family code is not enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(8)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRB</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRB.</td>
</tr>
<tr>
<td>Cisco IOS XE 2.6</td>
<td>This command was integrated into Cisco IOS XE Release 2.6.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The `debug bgp nsap` command is similar to the `debug ip bgp` command, except that it is specific to the NSAP address family.

**Note**

By default, the network server sends the output from `debug` commands and system error messages to the console. To redirect debug output, use the `logging` command options within global configuration mode. Destinations include the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server.

**Examples**

The following example shows output for the `debug bgp nsap` command. The BGP(4) identifies that BGP version 4 is operational.

```
Router# debug bgp nsap
00:46:46: BGP(4): removing CLNS route to 49.0101
00:46:46: BGP(4): removing CLNS route to 49.0303
00:46:46: BGP(4): 10.1.2.1 removing CLNS route 49.0101.1111.1111.1111.1111.00 to eBGP-neighbor
00:46:46: BGP(4): 10.1.2.1 removing CLNS route 49.0101.1111.1111.1111.1111.00 to eBGP-neighbor
```
Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug bgp nsap dampening</td>
<td>Displays debug messages for BGP NSAP prefix dampening events.</td>
</tr>
<tr>
<td>debug bgp nsap updates</td>
<td>Displays debug messages for BGP NSAP prefix update packets.</td>
</tr>
</tbody>
</table>
**debug bgp nsap dampening**

To display debug messages for Border Gateway Protocol (BGP) network service access point (NSAP) prefix address dampening, use the `debug bgp nsap dampening` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug bgp nsap dampening [filter-list access-list-number]
no debug bgp nsap dampening [filter-list access-list-number]
```

**Syntax Description**

| filter-list access-list-number | (Optional) Displays debug messages for BGP NSAP dampening events that match the access list. The acceptable access list number range is from 1 to 199. |

**Command Default**

Debugging for BGP NSAP dampening events is not enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(8)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRB</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRB.</td>
</tr>
<tr>
<td>Cisco IOS XE 2.6</td>
<td>This command was integrated into Cisco IOS XE Release 2.6.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The `debug bgp nsap dampening` command is similar to the `debug ip bgp dampening` command, except that it is specific to the NSAP address family.

**Note**

By default, the network server sends the output from `debug` commands and system error messages to the console. To redirect debug output, use the `logging` command options within global configuration mode. Destinations include the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server.

**Examples**

The following example shows output for the `debug bgp nsap dampening` command:

```
Router# debug bgp nsap dampening
```
Only one line of output is displayed unless the `bgpdampening` command is configured with a route map in NSAP address family configuration mode. The following example shows output for the `debug bgp nsap dampening` command when a route map is configured:

```
20:07:19: BGP(4): charge penalty for 49.0404 path 65202 65404 with halflife-time 15
reuse/suppress 750/2000
20:07:19: BGP(4): flapped 1 times since 00:00:00. New penalty is 1000
20:08:59: BGP(4): charge penalty for 49.0404 path 65202 65404 with halflife-time 15
reuse/suppress 750/2000
20:08:59: BGP(4): flapped 2 times since 00:01:39. New penalty is 1928
20:10:04: BGP(4): charge penalty for 49.0404 path 65202 65404 with halflife-time 15
reuse/suppress 750/2000
20:10:04: BGP(4): flapped 3 times since 00:02:44. New penalty is 2839
20:10:48: BGP(4): suppress 49.0404 path 65202 65404 for 00:28:10 (penalty 2752)
```

The table below describes the significant fields shown in the display.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>penalty</td>
<td>Numerical value of 1000 assigned to a route by a router configured for route dampening in another autonomous system each time a route flaps. Penalties are cumulative. The penalty for the route is stored in the BGP routing table until the penalty exceeds the suppress limit. If the penalty exceeds the suppress limit, the route state changes from history to damp.</td>
</tr>
<tr>
<td>halflife-time</td>
<td>Amount by which the penalty is decreased after the route is assigned a penalty. The half-life-time value is half of the half-life period (which is 15 minutes by default). Penalty reduction occurs every 5 seconds.</td>
</tr>
<tr>
<td>flapped</td>
<td>Number of times a route is available, then unavailable, or vice versa.</td>
</tr>
<tr>
<td>reuse</td>
<td>The limit by which a route is unsuppressed. If the penalty for a flapping route decreases and falls below this reuse limit, the route is unsuppressed. That is, the route is added back to the BGP table and once again used for forwarding. The default reuse limit is 750. Unsuppressing of routes occurs at 10-second increments. Every 10 seconds, the router learns which routes are now unsuppressed and advertises them throughout the network.</td>
</tr>
<tr>
<td>suppress</td>
<td>Limit by which a route is suppressed. If the penalty exceeds this limit, the route is suppressed. The default value is 2000.</td>
</tr>
<tr>
<td>maximum suppress limit</td>
<td>Maximum amount of time a route is suppressed. The default value is four times the half-life period.</td>
</tr>
</tbody>
</table>
State in which the route has flapped so often that the router will not advertise this route to BGP neighbors.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug bgp nsap</code></td>
<td>Displays debug messages for BGP NSAP packets.</td>
</tr>
<tr>
<td><code>debug bgp nsap updates</code></td>
<td>Displays debug messages for BGP NSAP update events.</td>
</tr>
</tbody>
</table>
debug bgp nsap updates

To display debug messages for Border Gateway Protocol (BGP) network service access point (NSAP) prefix address update packets, use the `debug bgp nsap updates` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
depug bgp nsap updates [ ip-address ] [ in | out ] [ filter-set clns-filter-set-name ]
no debug bgp nsap updates [ ip-address ] [ in | out ] [ filter-set clns-filter-set-name ]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip-address</td>
<td>(Optional) The IP address of a BGP neighbor.</td>
</tr>
<tr>
<td>in</td>
<td>(Optional) Indicates inbound updates.</td>
</tr>
<tr>
<td>out</td>
<td>(Optional) Indicates outbound updates.</td>
</tr>
<tr>
<td>filter-set</td>
<td>(Optional) Name of a Connectionless Network Service (CLNS) filter set.</td>
</tr>
</tbody>
</table>

**Command Default**

Debugging for BGP NSAP prefix update packets is not enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(8)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRB</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRB.</td>
</tr>
<tr>
<td>Cisco IOS XE 2.6</td>
<td>This command was integrated into Cisco IOS XE Release 2.6.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The `debug bgp nsap updates` command is similar to the `debug bgp updates` command, except that it is specific to the NSAP address family.

Use the `ip-address` argument to display the BGP update debug messages for a specific BGP neighbor. Use the `clns-filter-set-name` argument to display the BGP update debug messages for a specific NSAP prefix.
By default, the network server sends the output from debug commands and system error messages to the console. To redirect debug output, use the logging command options within global configuration mode. Destinations include the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server.

**Examples**

The following example shows output for the `debug bgp nsap updates` command:

```
Router# debug bgp nsap updates
02:13:45: BGP(4): 10.0.3.4 send UPDATE (format) 49.0101, next 49.0303.3333.3333.3333.3333.00, metric 0, path 65202 65101
02:13:45: BGP(4): 10.0.3.4 send UPDATE (format) 49.0202, next 49.0303.3333.3333.3333.3333.00, metric 0, path 65202
02:13:45: BGP(4): 10.0.3.4 send UPDATE (format) 49.0303, next 49.0303.3333.3333.3333.3333.00, metric 0, path
02:13:45: BGP(4): 10.0.2.2 send UPDATE (format) 49.0404, next 49.0303.3333.3333.3333.3333.00, metric 0, path 65404
```

The table below describes the significant fields shown in the display.

**Table 40: debug bgp nsap updates Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGP(4):</td>
<td>BGP debug for address family index (afi) 4.</td>
</tr>
<tr>
<td>route sourced locally (not shown in display)</td>
<td>Indicates that a route is sourced locally and that updates are not sent for the route.</td>
</tr>
<tr>
<td>send UPDATE (format)</td>
<td>Indicates that an update message for a reachable network should be formatted. Addresses include NSAP prefix and next hop.</td>
</tr>
<tr>
<td>rcv UPDATE (not shown in display)</td>
<td>Indicates that an update message about a path to a BGP peer has been received. Addresses include NSAP prefix.</td>
</tr>
</tbody>
</table>

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug bgp nsap</td>
<td>Displays debug messages for BGP NSAP packets.</td>
</tr>
<tr>
<td>debug bgp nsap dampening</td>
<td>Displays debug messages for BGP NSAP prefix dampening events.</td>
</tr>
</tbody>
</table>
**debug bgp vpnv6 unicast**

To display Border Gateway Protocol (BGP) virtual private network (VPN) debugging output, use the `debug bgp vpnv6 unicast` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug bgp vpnv6 unicast
no debug bgp vpnv6
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SRB</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SB.</td>
</tr>
<tr>
<td>12.2(33)SXI</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SXI.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use the `debug bgp vpnv6 unicast` command to help troubleshoot the BGP VPN.

**Note**

By default, the network server sends the output from debug commands and system error messages to the console. To redirect debugging output, use the logging command options within global configuration mode. Destinations are the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server. For complete information on debug commands and redirecting debugging output, refer to the Cisco IOS Debug Command Reference, Release 12.4.

**Examples**

The following example enables BGP debugging output for IPv6 VPN instances:

```
Router# debug bgp vpnv6 unicast
```
debug bri-interface

To display debugging information on ISDN BRI routing activity, use the `debug bri-interface` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

**debug bri-interface**

**no debug bri-interface**

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

The `debug bri-interface` command indicates whether the ISDN code is enabling and disabling the B channels when attempting an outgoing call. This command is available for the low-end router products that have a multi-BRI network interface module installed.

⚠️ **Caution**

Because the `debug bri-interface` command generates a substantial amount of output, use it only when traffic on the IP network is low, so other activity on the system is not adversely affected.

**Examples**

The following is sample output from the `debug bri-interface` command:

```
Router# debug bri-interface
BRI: write_sid: wrote 1B for subunit 0, slot 1.
BRI: write_sid: wrote 15 for subunit 0, slot 1.
BRI: write_sid: wrote 17 for subunit 0, slot 1.
BRI: write_sid: wrote 6 for subunit 0, slot 1.
BRI: write_sid: wrote 11 for subunit 0, slot 1.
BRI: write_sid: wrote 13 for subunit 0, slot 1.
BRI: write_sid: wrote 29 for subunit 0, slot 1.
BRI: write_sid: wrote 1B for subunit 0, slot 1.
BRI: Starting Power Up timer for unit = 0.
BRI: write_sid: wrote 3 for subunit 0, slot 1.
BRI: Starting T3 timer after expiry of PUP timeout for unit = 0, current state is F4.
BRI: write_sid: wrote FF for subunit 0, slot 1.
BRI: Activation for unit = 0, current state is F7.
BRI: enable channel B1
BRI: write_sid: wrote 14 for subunit 0, slot 1.
%LINK-3-UPDOWN: Interface BRI0: B-Channel 1, changed state to up
%LINK-5-CHANGED: Interface BRI0: B-Channel 1, changed state to up.!!!
BRI: disable channel B1
BRI: write_sid: wrote 15 for subunit 0, slot 1.
%LINK-3-UPDOWN: Interface BRI0: B-Channel 1, changed state to down
%LINK-5-CHANGED: Interface BRI0: B-Channel 1, changed state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0: B-Channel 1, changed state to down
```
The following line indicates that an internal command was written to the interface controller. The subunit identifies the first interface in the slot.

BRI: write_sid: wrote 1B for subunit 0, slot 1.
The following line indicates that the power-up timer was started for the named unit:

BRI: Starting Power Up timer for unit = 0.
The following lines indicate that the channel or the protocol on the interface changed state:

%LINK-3-UPDOWN: Interface BRI0: B-Channel 1, changed state to up
%LINK-5-CHANGED: Interface BRI0: B-Channel 1, changed state to up.!!!
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0: B-Channel 1, changed state to down
The following line indicates that the channel was disabled:

BRI: disable channel B1
Lines of output not described are for use by support staff only.

## Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug isdn event</td>
<td>Displays ISDN events occurring on the user side (on the router) of the ISDN interface.</td>
</tr>
<tr>
<td>debug isdn q921</td>
<td>Displays data link-layer (Layer 2) access procedures that are taking place at the router on the D channel (LSPD).</td>
</tr>
<tr>
<td>debug isdn q931</td>
<td>Displays information about call setup and teardown of ISDN network connections (Layer 3) between the local router (user side) and the network.</td>
</tr>
</tbody>
</table>
debug bsc event

To display all events occurring in the Binary Synchronous Communications (Bisync) feature, use the `debug bsc event` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug bsc event [ number ]
no debug bsc event [ number ]
```

**Syntax Description**

<table>
<thead>
<tr>
<th><code>number</code></th>
<th>(Optional) Group number.</th>
</tr>
</thead>
</table>

**Command Modes**

Privileged EXEC

**Usage Guidelines**

This command traces all interfaces configured with a `bscprotocol-group number` command.

**Examples**

The following is sample output from the `debug bsc event` command:

```
Router# debug bsc event
BSC: Serial2 POLLEE-FSM inp:E_LineFail old_st:CU_Down new_st:TCU_EOFile
BSC: Serial2 POLLEE-FSM inp:E_LineFail old_st:CU_Down new_st:TCU_EOFile
BSC: Serial2 POLLEE-FSM inp:E_LineFail old_st:CU_Down new_st:TCU_EOFile
0:04:32: BSC: Serial2 :SDI-rx: 9 bytes
BSC: Serial2 POLLEE-FSM inp:E_RxEnq old_st:CU_Down new_st:TCU.EOFile
BSC: Serial2 POLLEE-FSM inp:E_Timeout old_st:CU_Down new_st:TCU_EOFile
BSC: Serial2 POLLEE-FSM inp:E_Timeout old_st:CU_Down new_st:TCU_EOFile
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2, changed state to up
%LINK-3-UPDOWN: Interface Serial2, changed state to up
0:04:35: BSC: Serial2 :SDI-rx: 9 bytes
BSC: Serial2 POLLEE-FSM inp:E_Timeout old_st:CU_Idle new_st:TCU_InFile
BSC: Serial2 POLLEE-FSM inp:E_Timeout old_st:CU_Idle new_st:TCU_InFile
0:04:35: BSC: Serial2 :SDI-rx: 5 bytes
BSC: Serial2 POLLEE-FSM inp:E_RxEnq old_st:CU_Idle new_st:TCU_InFile
0:04:35: BSC: Serial2 :SDI-rx: 3 bytes
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug bsc packet</td>
<td>Displays all frames traveling through the Bisync feature.</td>
</tr>
<tr>
<td>debug bstun events</td>
<td>Displays BSTUN connection events and status.</td>
</tr>
</tbody>
</table>
debug bsc packet

To display all frames traveling through the Binary Synchronous Communications (Bisync) feature, use the `debug bsc packet` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

`debug bsc packet [group number] [buffer-size bytes]`
`no debug bsc packet [group number] [buffer-size bytes]`

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>group number</code></td>
<td>(Optional) Group number.</td>
</tr>
<tr>
<td><code>buffer-size bytes</code></td>
<td>(Optional) Number of bytes displayed per packet (defaults to 20).</td>
</tr>
</tbody>
</table>

**Command Default**

The default number of bytes displayed is 20.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

This command traces all interfaces configured with a `bscprotocol-group number` command.

**Examples**

The following is sample output from the `debug bsc packet` command:

```
Router# debug bsc packet
0:23:33: BSC: Serial2 : NDI-rx : 27 bytes 401A400227F5C31140C11D60C8C5D3D51D4013
0:23:33: BSC: Serial2 : SDI-tx : 12 bytes 00323237FF3232606040402D
0:23:33: BSC: Serial2 : SDI-rx : 2 bytes 1070
0:23:33: BSC: Serial2 : SDI-tx : 27 bytes 401A400227F5C31140C11D60C8C5D3D51D4013
0:23:33: BSC: Serial2 : SDI-rx : 2 bytes 1061
0:23:33: BSC: Serial2 : SDI-tx : 5 bytes 00323237FF
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug bsc event</code></td>
<td>Displays all events occurring in the Bisync feature.</td>
</tr>
<tr>
<td><code>debug bstun events</code></td>
<td>Displays BSTUN connection events and status.</td>
</tr>
</tbody>
</table>
debug bstun events

To display BSTUN connection events and status, use the `debug bstun events` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug bstun events [ number ]
no debug bstun events [ number ]
```

**Syntax Description**

| number     | (Optional) Group number. |

**Command Modes**

Privileged EXEC

**Usage Guidelines**

When you enable the `debug bstun events` command, messages showing connection establishment and other overall status messages are displayed.

You can use the `debug bstun events` command to assist you in determining whether the BSTUN peers are configured correctly and are communicating. For example, if you enable the `debug bstun packet` command and you do not see any packets, you may want to enable event debugging.

**Note**

Also refer to the `debug bscp packet` and `debug bscevent` commands. Currently, these two commands support the only protocol working through the BSTUN tunnel. Sometimes frames do not go through the tunnel because they have been discarded at the Bisync protocol level.

**Examples**

The following is sample output from the `debug bstun events` command of keepalive messages working correctly. If the routers are configured correctly, at least one router will show reply messages.

```
Router# debug bstun events
BSTUN: Received Version Reply opcode from (all[2]) 172.16.12.2/1976 at 1360
BSTUN: Received Version Request opcode from (all[2]) 172.16.12.2/1976 at 1379
BSTUN: Received Version Reply opcode from (all[2]) 172.16.12.2/1976 at 1390
```

**Note**

In a scenario where there is constantly loaded bidirectional traffic, you might not see keepalive messages because they are sent only when the remote end has been silent for the keepalive period.

The following is sample output from the `debug bstun events` output of an event trace in which the wrong TCP address has been specified for the remote peer. These are non-keepalive related messages.

```
Router# debug bstun events
BSTUN: Change state for peer (C1[1]) 172.16.12.22/1976 (closed->opening)
BSTUN: Change state for peer (C1[1]) 172.16.12.22/1976 (opening->open wait)
%BSTUN-6-OPENING: CONN: opening peer (C1[1]) 172.16.12.22/1976, 3
BSTUN: tcpd sender in wrong state, dropping packet
BSTUN: tcpd sender in wrong state, dropping packet
```
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug bsc event</td>
<td>Displays all events occurring in the Bisync feature.</td>
</tr>
<tr>
<td>debug bsc packet</td>
<td>Displays all frames traveling through the Bisync feature.</td>
</tr>
<tr>
<td>debug bstun packet</td>
<td>Displays packet information on packets traveling through the BSTUN links.</td>
</tr>
</tbody>
</table>
**debug bstun packet**

To display packet information on packets traveling through the BSTUN links, use the `debug bstun packet` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug bstun packet [group number] [buffer-size bytes]
no debug bstun packet [group number] [buffer-size bytes]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>group number</code></td>
<td>(Optional) BSTUN group number.</td>
</tr>
<tr>
<td><code>buffer-size bytes</code></td>
<td>(Optional) Number of bytes displayed per packet (defaults to 20).</td>
</tr>
</tbody>
</table>

**Command Default**

The default number of bytes displayed is 20.

**Command Modes**

Privileged EXEC

**Examples**

The following is sample output from the `debug bstun packet` command:

```
Router# debug bstun packet
BSTUN bsc-local-ack: 0:00:00 Serial2 SDI: Addr: 40 Data: 02C1C1C1C1C1C1C1C1C1
BSTUN bsc-local-ack: 0:00:00 Serial2 SDI: Addr: 40 Data: 02C1C1C1C1C1C1C1C1C1
BSTUN bsc-local-ack: 0:00:06 Serial2 NDI: Addr: 40 Data: 0227F5C31140C11D60C8
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug bstun events</code></td>
<td>Displays BSTUN connection events and status.</td>
</tr>
</tbody>
</table>
debug bundle errors

To enable the display of information on bundle errors, use the `debug bundle errors` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug bundle errors
no debug bundle errors
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(3)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command to enable the display of error information for a bundle, such as reports of inconsistent mapping in the bundle.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bump</td>
<td>Configures the bumping rules for a VC class that can be assigned to a VC bundle.</td>
</tr>
<tr>
<td>bundle</td>
<td>Creates a bundle or modifies an existing bundle to enter bundle configuration mode.</td>
</tr>
<tr>
<td>debug bundle events</td>
<td>Enables display of bundle events when use occurs.</td>
</tr>
</tbody>
</table>
debug bundle events

To enable display of bundle events when use occurs, use the `debug bundle events` command in privileged EXEC mode. To disable the display, use the `no` form of this command.

```
debug bundle events
no debug bundle events
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(3)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command to enable the display of bundle events, such as occurrences of VC bumping, when bundles were brought up, when they were taken down, and so forth.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug bstun packet</td>
<td>Enables the display of information on bundle errors.</td>
</tr>
</tbody>
</table>
debug call-home diagnostic-signature

To enable the debugging of call-home diagnostic signature flags on a device, use the `debug call-home diagnostic-signature` command in privileged EXEC mode. To disable debugging, use the `no` form of this command.

```
debug call-home diagnostic-signature {action | all | api | cli | download | event-registration | parsing}
no debug call-home diagnostic-signature {action | all | api | cli | download | event-registration | parsing}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Displays debugging information associated with the execution of any call-home diagnostic signature action defined in the diagnostic signature file.</td>
</tr>
<tr>
<td>all</td>
<td>Displays debugging information about all flags associated with the call-home diagnostic signature.</td>
</tr>
<tr>
<td>api</td>
<td>Displays debugging information associated with call-home diagnostic signature internal operations or function calls.</td>
</tr>
<tr>
<td>cli</td>
<td>Displays debugging information associated with the call-home diagnostic signature to run the CLI commands as part of the diagnostic signature actions.</td>
</tr>
<tr>
<td>download</td>
<td>Displays debugging information associated with the downloading of call-home diagnostic signature files from the HTTP/HTTPS servers.</td>
</tr>
<tr>
<td>event-registration</td>
<td>Displays debugging information associated with the registration of call-home diagnostic signature events.</td>
</tr>
<tr>
<td>parsing</td>
<td>Displays debugging information associated with the parsing of call-home diagnostic-signature files.</td>
</tr>
</tbody>
</table>

### Command Modes

- Privileged EXEC (#)

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.3(2)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

### Examples

The following is sample output from the `debug call-home diagnostic-signature action` command:

```
Device# debug call-home diagnostic-signature action
Jan 29 10:42:22.698 CST: DS-ACT-TRACE: call_home_ds_eem_cmd_run[969], cli cmd "show version", expect string ",", max run time 20000
```
The following is sample output from the `debug call-home diagnostic-signature api` command:

```
Device# debug call-home diagnostic-signature api
```

```
Jan 29 10:41:24.902 CST: DS-API-TRACE: call_home_all_lock[101], lock callhome and ds mutex
Jan 29 10:41:24.902 CST: DS-API-TRACE: call_home_ds_lock[42], lock call home ds semaphore
Jan 29 10:41:24.902 CST: DS-API-TRACE: call_home_all_unlock[109], unlock callhome and ds mutex
Jan 29 10:41:24.902 CST: DS-API-TRACE: call_home_ds_unlock[52], unlock call home ds semaphore

The following is sample output from the `debug call-home diagnostic-signature cli` command:

```
Device# debug call-home diagnostic-signature cli
```

```
Jan 29 10:44:31.402 CST: DS-CLI-TRACE: call_home_ds_eem_cmd_run[981], the first 100 chars of output cmd: show version
Cisco IOS Software, C1861 Software (C1861-ADVENTERPRISEK9-M), Experimental Version 15.

Jan 29 10:44:31.442 CST: DS-CLI-TRACE: call_home_ds_eem_cmd_run[981], the first 100 chars of output cmd: show logging
Syslog logging: enabled (0 messages dropped, 3 messages rate-limited, 0 flushes, 0 over
```

The following is sample output from the `debug call-home diagnostic-signature download` command:

```
Device# debug call-home diagnostic-signature download
```

```
Jan 29 10:40:11.050 CST: DS-DNLD-TRACE: call_home_ds_update_thread_create[239], Creating a download process
Jan 29 10:40:11.054 CST: DS-DNLD-TRACE: ds_collect_content_prolog_values[370], Collecting XML content prolog values
Jan 29 10:40:11.054 CST: DS-DNLD-TRACE: ds_collect_content_prolog_values[489], System Name:C1861-1
Jan 29 10:40:11.054 CST: DS-DNLD-TRACE: ds_collect_content_prolog_values[494], Unable to get SNMP contact string
Jan 29 10:40:11.054 CST: DS-DNLD-TRACE: ds_collect_content_epilog_values[550], Collecting XML content epilog values
Jan 29 10:40:11.054 CST: DS-DNLD-TRACE: ds_collect_request_values[621], Collecting XML DS request values
```

The following is sample output from the `debug call-home diagnostic-signature event-registration` command:

```
Device# debug call-home diagnostic-signature event-registration
```

```
Jan 29 10:40:16.734 CST: DS-REG-TRACE: ds_content_event_reg[515], ds "6030"
Jan 29 10:40:16.734 CST: DS-REG-TRACE: ds_event_sig_reg[304], ds "6030", index 0, event number 1
Jan 29 10:40:16.734 CST: DS-REG-TRACE: call_home_ds_esid_reg[323], ds "6030", index 0, T = 41, S = 3FCH
```
The following is sample output from the **debug call-home diagnostic-signature parsing** command:

```
Device# debug call-home diagnostic-signature parsing
Jan 29 10:40:16.734 CST: DS-REG-TRACE: call_home_ds_content_reparse[3108], reparse ds "6030" content
Jan 29 10:40:16.734 CST: DS-PARSE-TRACE: ds_content_var_reparse[2915], reparse ds var in 6030
Jan 29 10:40:16.734 CST: DS-PARSE-TRACE: ds_sys_var_local_queue_init[2860], copy sys var ds_signature_id
Jan 29 10:40:16.734 CST: DS-PARSE-TRACE: ds_sys_var_local_queue_init[2860], copy sys var ds_hostname
```

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>call-home diagnostic-signature</strong></td>
<td>Downloads, installs, and uninstalls diagnostic signature files on a device.</td>
</tr>
<tr>
<td><strong>show call-home diagnostic-signature statistics</strong></td>
<td>Displays statistics and attributes of a diagnostic signature file on a device.</td>
</tr>
</tbody>
</table>
debug call-mgmt

To display debugging information for call accounting, including modem and time slot usage, for active and recent calls, use the `debug call-mgmt` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug call-mgmt
no debug call-mgmt
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output after the `debug call-mgmt` command has been enabled:

```
Router# debug call-mgmt
Call Management debugging is on
Router# Dec 26 13:57:27.710: msg_to_calls_mgmt: msg type CPM_NEW_CALL_CSM_CONNECT received
Router# Dec 26 13:57:27.714: In actv_c_proc_message,
    access type CPM_INSERT_NEW_CALL,
    call type CPM_ISDN_ANALOG:
        CSM completed connecting a new modem call
    .
    .
Router# Dec 26 13:57:45.906: msg_to_calls_mgmt: msg type CPM_NEW_CALL_ISDN_CONNECT received
Router# Dec 26 13:57:45.906: In actv_c_proc_message,
    access type CPM_INSERT_NEW_CALL,
    call type CPM_ISDN_ANALOG:
        Added a new ISDN analog call to the active-calls list
        CC-Slot#7, DSX1-Ctrlr#17, DS0-Timeslot#1
        Mdm-Slot#1, Mdm-Port#3, TTY#219
    .
    .
Router# Dec 26 13:58:25.682: Call mgmt per minute statistics:
    active list length: 1
    history list length: 3
Router# Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 1
Router# Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 2
Router# Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 3
Router# Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 4
```
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 5
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 6
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 7
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 8
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 9
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 10
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 11
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 12
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 13
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 14
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 15
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 16
Dec 26 13:58:25.682: 1 timeslots active at slot 7, ctrlr 17
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 18
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 19
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 20
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 21
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 22
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 23
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 24
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 25
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 26
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 27
Dec 26 13:58:25.682: 0 timeslots active at slot 7, ctrlr 28

Router# clear int as1/03
Dec 26 13:58:26.538: msg_to_calls_mgmt: msg type CPM_VOICE_CALL_REJ_NO_MOD_AVAIL received
Dec 26 13:58:26.538: In actv_c_proc_message,
   access type CPM_REMOVE_DISC_CALL,
   call type CPM_ISDN_ANALOG:
   Removed a disconnected ISDN analog call
   CC-Slot#7, DSX1-Ctrlr#17, DS0-Timeslot#1
   Mdm-Slot#1, Mdm-Port#3, TTY#219

The table below describes the significant fields shown in the display.

### Table 41: debug call-mgmt Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPM_NEW_CALL_CSM_CONNECT</td>
<td>Indicates the arrival of a new call.</td>
</tr>
<tr>
<td>access type CPM_INSERT_NEW_CALL, call type CPM_ISDN_ANALOG:</td>
<td>Indicates that the new call is an analog ISDN B channel call (either a voice call or a call over an analog modem), rather than a digital (V.110) call.</td>
</tr>
<tr>
<td>CC-Slot#7, DSX1-Ctrlr#17, DS0-Timeslot#1 Mdm-Slot#1, Mdm-Port#3, TTY#219</td>
<td>Indicates that the call is connected via the B channel on Serial7/17:1 to the asynchronous modem resource 1/03 (interface async1/03, also known as line tty219).</td>
</tr>
<tr>
<td>Dec 26 13:58:25.682: Call mgmt per minute statistics: active list length: 1 history list length: 3</td>
<td>Displays periodic statistics that give the allocation state of each DSX1 interface present in the system, as well as the number of current (active) and recent (history) calls.</td>
</tr>
<tr>
<td>Dec 26 13:58:26.538: msg_to_calls_mgmt: msg type CPM_VOICE_CALL_REJ_NO_MOD_AVAIL received</td>
<td>Indicates that the analog ISDN B channel call has been disassociated from a modem.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>access type CPM_REMOVE_DISC_CALL, call type CPM_ISDN_ANALOG: Removed a disconnected ISDN analog call</td>
<td>Indicates that the analog ISDN B channel call has been disconnected.</td>
</tr>
<tr>
<td>CC-Slot#7, DSX1-Ctrlr#17, DS0-Timeslot#1 Dec 26 13:58:26.538: Mdm-Slot#1, Mdm-Port#3, TTY#219</td>
<td>Indicates that the call has been disconnected via the B channel on Serial7/17:1 to the asynchronous modem resource 1/03 (interface async1/03, also known as line tty219).</td>
</tr>
</tbody>
</table>
debug call fallback detail

To display details of the call fallback, use the `debug call fallback detail` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug call fallback detail
no debug call fallback detail
```

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

**Command Default**
Disabled

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850 platform.</td>
</tr>
<tr>
<td>12.2(4)T</td>
<td>This command was implemented on the Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(4)T3</td>
<td>This command was implemented on the Cisco 7500 series routers routers.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(11)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
Every time a call request is received, the `debug call fallback detail` command displays in the command-line interface (CLI) cache lookup and call acceptance/rejection information. Use this command to monitor call requests as they enter the call fallback subsystem.

If you have a large amount of calls in your router, enabling this command can cause delays in your routing functions as the debug statistics are constantly compiled and sent to your terminal. Also, debug messages on your terminal may make for difficult CLI configuring.

**Examples**
The following example depicts a call coming in to 10.1.1.4 with codec g729r8. Because there is no cache entry for this destination, a probe is sent and values are inserted into the cache. A lookup is performed again, entry is found, and a fallback decision is made to admit the call.

```
Router# debug call fallback detail
When cache is empty:
debug call fallback detail:
2d19h:fb_lookup_cache:10.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:No entry found.
2d19h:fb_check:No entry exists, enqueueing probe info... 10.1.1.4, codec:g729r8
2d19h:fb_main:Got FB_APP_INQ event
```
2d19h:fb_main:Dequeued prob info: 10.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:10.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:No entry found.
2d19h:fb_cache_insert:insert:10.1.1.4, codec:g729r8
2d19h:fb_cache_insert:returning entry:10.1.1.4, codec:g729r8
2d19h:fb_initiate_probe:Creating probe... 10.1.1.4, codec:g729r8
2d19h:fb_initiate_probe:Created and started on probe #13, 10.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:10.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:Found entry.
2d19h:fb_check:returned FB_CHECK_TRUE, 10.1.1.4, codec:g729r8
2d19h:fb_main:calling callback function with:TRUE

The following example depicts a call coming in to 10.1.1.4 with codec g729r8. A lookup is performed, entry is found, and a fallback decision is made to admit the call.

Router# debug call fallback detail
When cache is full:
2d19h:fb_lookup_cache:10.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:Found entry.
2d19h:fb_check:Returned FB_CHECK_TRUE, 10.1.1.4, codec:g729r8
2d19h:fb_main:calling callBack function with:TRUE
debug call fallback probe

To display details of the call fallback probes, use the `debug call fallback probe` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug call fallback probe
no debug call fallback probe
```

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

**Command Default**
Disabled

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)XA</td>
<td>The <code>call fallback</code> and <code>call fallback reject-cause-code</code> commands were introduced.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850 platform.</td>
</tr>
<tr>
<td>12.2(4)T</td>
<td>This command was implemented on the Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(4)T3</td>
<td>This command was implemented on the Cisco 7500 series routers.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
Every time a probe is received, the `debug call fallback probe` command displays in the command-line interface (CLI) network traffic information collected by the probe. Use this command to monitor the network traffic information the probes carry as they enter the call fallback subsystem and log cache entries.

If you have frequent return of probes to your router, enabling this command can cause delays in your routing functions as the debug statistics are constantly compiled and sent to your terminal. Also, debug messages on your terminal may make for difficult CLI configuring.

**Examples**
The following example depicts a call coming in to 10.1.1.4 and codec type g729r8. Because there is no cache entry for this IP address, a g729r8 probe is initiated. The probe consists of 20 packet returns with an average delay of 43 milliseconds. The "jitter out" is jitter from source to destination router and "jitter in" is jitter from destination to source router. The delay, loss, and Calculated Planning Impairment Factor (ICPIF) values following `g113_calc_icpif` are the instantaneous values, whereas those values following "New smoothed values" are the values after applying the smoothing with weight 65.

```
Router# debug call fallback probe
```
debug backhaul-session-manager session through debug channel packets

depend call fallback probe

```plaintext
2d19h:fb_initiate_probe: Probe payload is 32
2d19h:fb_main: NumOfRTT=20, RTTSum=120, loss=0, delay=43, jitter in=0, jitter out=0 -> 10.1.1.4, codec:g729r8
2d19h:g113_calc_icpif (delay (w/codec delay)=43, loss=0, expect_factor=10) Icpif=0
2d19h:fb_main: Probe timer expired, 10.1.1.4, codec:g729r8
2d19h:fb_main: NumOfRTT=20, RTTSum=120, loss=0, delay=43, jitter in=0, jitter out=0 -> 10.1.1.4, codec:g729r8
2d19h:g113_calc_icpif (delay (w/codec delay)=43, loss=0, expect_factor=10) Icpif=0
2d19h:fb_main: New smoothed values: inst_weight=65, ICPIF=0, Delay=43, Loss=0 -> 10.1.1.4, codec:g729r8
```
debug call filter detail

To display details of the debug trace inside the generic call filter module (GCFM), use the debug call filter detail command in privileged EXEC mode. To disable debugging output, use the no form of this command.

```
debug call filter detail
no debug call filter detail
```

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

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(4)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**
The following sample output from the `debug call filter detail` command shows the detailed activity of the GCFM, which is the internal module that controls the debug filtering.

```
Router# debug call filter detail
5d18h: gcfs_call_get_hash_address: hashtable index = 345
5d18h: gcfs_call_search_hash: no found
5d18h: gcfs_init_call_record:
5d18h: gcfs_init_percall_matchlist:
5d18h: === list 1: service_state=2, callp's: 0
5d18h: gcfs_call_get_hash_address: hashtable index = 345
5d18h: gcfs_call_enlist: count before this enlist 0 on 624D6000
5d18h: gcfs_call_enlist: tail is empty guid=C2E4C789-214A-11D4-804C-000A8A389BA8
5d18h: gcfs_call_search_hash: hashtable index = 345
5d18h: gcfs_call_search_hash: search requested guid=C2E4C789-214A-11D4-804C-000A8A389BA8
vs the entry guid=C2E4C789-214A-11D4-804C-000A8A389BA8
5d18h: gcfs_call_search_hash: found
5d18h: gcfs_update_percall_condlist_context:
5d18h: gcfs_update_percall_condlist_context: check cond = 2
5d18h: gcfs_copy_match_cond:
5d18h: gcfs_update_cond_through_matchlist:
5d18h: gcfs_check_percond_with_matchlist: check match-list 1
5d18h: gcfs_matchList_perCond_Check:
5d18h: gcfs_matchList_perCond_check: check cond=2
5d18h: gcfs_matchList_perCond_check: compare 42300 to configured 42300
5d18h: gcfs_check_cond_tei_number:
5d18h: gcfs_check_cond_tei_number; matched
5d18h: gcfs_matchList_perCond_check: checked result is 1
5d18h: gcfs_is_bitfield_identical:
5d18h: gcfs_update_cond_through_matchlist: service=1, percall_matchlist tag=1, current_status
5d18h: gcfs_perall_notify_condition: not linked call record
```

The table below describes the significant fields shown in the display.
### Table 42: debug call filter detail Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5d18h: gcfm_init_percall_matchlist:</td>
<td>Shows that the filtering has been initiated.</td>
</tr>
<tr>
<td>guid=C2E4C789-214A-11D4-804C-000A8A389BA8</td>
<td>Shows the global unique identifier (GUID) for the call.</td>
</tr>
<tr>
<td>5d18h: gcfm_check_percond_with_matchlist: check match-list 1</td>
<td>Shows which match list is being checked.</td>
</tr>
<tr>
<td>5d18h: gcfm_matchlist_percond_check: checked result is 1</td>
<td>Shows that the call matched conditions in match list 1.</td>
</tr>
</tbody>
</table>

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug call filter inout</td>
<td>Displays the debug trace inside the GCFM.</td>
</tr>
<tr>
<td>debug condition match-list</td>
<td>Runs a filtered debug on a voice call.</td>
</tr>
<tr>
<td>show call filter components</td>
<td>Displays the components used for filtering calls.</td>
</tr>
</tbody>
</table>
debug call filter inout

To display the debug trace inside the generic call filter module (GCFM), use the debug call filter inout command in privileged EXEC mode. To disable debugging output, use the no form of this command.

debug call filter inout
no debug call filter inout

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

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(4)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**
The following sample output from the debug call filter inout command shows the incoming and outgoing activity of the GCFM, which is the internal module that controls the debug filtering.

```
Router# debug call filter inout
5d18h: gcfm_generate_guid:
    component ISDN gets guid
5d18h: gcfm_percall_register:
    component ISDN
5d18h: gcfm_percall_register: component ISDN return selected=0
5d18h: gcfm_percall_notify_condition:
    component ISDN for sync=1
5d18h: gcfm_percall_notify_condition: component ISDN successfully selected = 0
5d18h: gcfm_check_percall_status:
    component TGRM
5d18h: gcfm_check_percall_status: component TGRM return selected=0
5d18h: gcfm_check_percall_status: component TGRM return selected=0
5d18h: gcfm_percall_register:
    component VTSP
5d18h: gcfm_percall_register: component VTSP for return selected value 0
5d18h: gcfm_percall_notify_condition: component VTSP for sync=1
5d18h: gcfm_percall_notify_condition: component VTSP successfully selected = 0
5d18h: gcfm_percall_register: component CCAPI
5d18h: gcfm_percall_register: component CCAPI for return selected value 0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0
5d18h: gcfm_check_percall_status: component VOICE-IVR-V2
5d18h: gcfm_check_percall_register: component VOICE-IVR-V2 for return selected value 0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component DIAL-PEER return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component DIAL-PEER
```
Table 43: debug call filter inout Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcfm_generate_guid</td>
<td>Shows that a GUID has been generated.</td>
</tr>
<tr>
<td>gcfm_percall_register</td>
<td>Shows components that have been registered for the call.</td>
</tr>
<tr>
<td>gcfm_percall_notify_condition</td>
<td>Shows that a component has been notified of the call.</td>
</tr>
<tr>
<td>gcfm_check_percall_status</td>
<td>Shows the status of a component of the call.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>gcfm_per call_register:</td>
<td>Shows that a component has been registered.</td>
</tr>
<tr>
<td>gcfm_clear_condition:</td>
<td>Shows that a condition is cleared for a component.</td>
</tr>
<tr>
<td>gcfm_per call_deregister:</td>
<td>Shows that a component has been deregistered.</td>
</tr>
<tr>
<td>gcfm_terminate_track_guid:</td>
<td>Shows that the router is no longer tracking the GUID.</td>
</tr>
</tbody>
</table>

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug call filter detail</td>
<td>Displays the details of the debug trace inside the GCFM.</td>
</tr>
<tr>
<td>debug condition match-list</td>
<td>Runs a filtered debug on a voice call.</td>
</tr>
<tr>
<td>show call filter components</td>
<td>Displays the components used for filtering calls.</td>
</tr>
</tbody>
</table>
debug call rsvp-sync events

To display events that occur during Resource Reservation Protocol (RSVP) setup, use the `debug call rsvp-sync events` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

debug call rsvp-sync events
no debug call rsvp-sync events

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XI</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.1(5)T</td>
<td>This command was integrated into Cisco IOS Release 12.1(5)T.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>Support for the command was implemented in Cisco AS5850 images.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

It is highly recommended that you log the output from the `debug call rsvp-sync events` 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 a portion of sample output for a call initiating RSVP when using the `debug call rsvp-sync events` command:

```
00:03:25: QoS Primitive Event for Call id 0x1 : QoS Listen
00:03:25: Lookup to be done on hashkey 0x1 in hash table 0x61FC2498
00:03:25: Hashed entry 0x1 in call table 0x61FC2498
00:03:25: Entry Not found
00:03:25: Parameters: localip: 10.19.101.117
00:03:25: remoteip: 10.19.101.116
00:03:25: QoSpcb : 0x61FC34D8
00:03:25: Response Status : 0
Starting timer for call with CallId 0x1 for 10000 secs
00:03:25: Handling QoS Primitive QoS Listen
```
00:03:25: Establishing RSVP RESV state : rsvp_request_reservation()
00:03:25: RSVP Confirmation required
00:03:25: QoS Primitive Event for Call id 0x1 : QoS Resv
00:03:25: Lookup to be done on hashkey 0x1 in hash table 0x61FC2498
00:03:25: Hashed entry 0x1 in call table 0x61FC2498
00:03:25: Initiating RSVP PATH messages to be Sent : reg_invoke_rsvp_advertise_sender()
00:03:25: RESV notification event received is : 2
00:03:25: Received RESVCONFIRM
00:03:25: RESV CONFIRM message received from 10.19.101.116 for RESV setup from 10.19.101.117
00:03:25: RESV event received is : 0
00:03:25: RESERVATIONS ESTABLISHED : CallId: 1 Stop timer and notify Session Protocol
of Success (ie. if notification requested)
00:03:25: Invoking spQoSresvCallback with Success

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>call rsvp-sync</td>
<td>Enables synchronization between RSVP and the H.323 voice signaling protocol.</td>
</tr>
<tr>
<td>call rsvp-sync resv-timer</td>
<td>Sets the timer for RSVP reservation setup.</td>
</tr>
<tr>
<td>debug call rsvp-sync func-trace</td>
<td>Displays messages about the software functions called by RSVP synchronization.</td>
</tr>
<tr>
<td>show call rsvp-sync conf</td>
<td>Displays the RSVP synchronization configuration.</td>
</tr>
<tr>
<td>show call rsvp-sync stats</td>
<td>Displays statistics for calls that attempted RSVP reservation.</td>
</tr>
</tbody>
</table>
debug call rsvp-sync func-trace

To display messages about software functions called by Resource Reservation Protocol (RSVP), use the `debug call rsvp-sync func-trace` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug call rsvp-sync func-trace
no debug call rsvp-sync func-trace
```

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

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XI1</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.1(5)T</td>
<td>This command was integrated into Cisco IOS Release 12.1(5)T.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
It is highly recommended that you log the output from the `debug call rsvp-sync func-trace` 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 a portion of sample output for a call initiating RSVP when using the `debug call rsvp-sync func-trace` command in conjunction with the `debug call rsvp-syncevents` command:

```
00:03:41: Entering Function QoS_Listen
00:03:41: Parameters:localip:10.10.101.116:localport:17568
00:03:41:remoteip:10.10.101.117:remoteport:0
00:03:41: Entering Function qos_dequeue_event
00:03:41: Entering Function process_queue_event
00:03:41: QoS Primitive Event for Call id 0x2:QoS Listen
00:03:41: Entering Function get_pcb
00:03:41: Entering Function hash_tbl_lookup
00:03:41: Lookup to be done on hashkey 0x2 in hash table 0x61FAECD8
00:03:41: Entering Function hash_func
00:03:41: Hashed entry 0x2 in call table 0x61FAECD8
00:03:41: Entry Not found
00:03:41: Entering Function qos_dequeue_pcb
00:03:41: Entering Function qos_initialize_pcb
```
debug call rsvp-sync func-trace

00:03:41: Parameters: localip: 10.10.101.116
00:03:41: remoteip: 10.10.101.117
00:03:41: QoSpcb: 0x61FAFD18
00:03:41: Response Status: 0
00:03:41: Entering Function hash_tbl_insert_entry
00:03:41: Entering Function hash_func
00:03:41: Handling QoS Primitive QoS Listen
00:03:41: Entering Function qos_dequeue_hash_port_entry
00:03:41: Entering Function qos_port_tbl_insert_entry
00:03:41: Entering Function hash_func
00:03:41: Doing RSVP Listen: rsvp_add_ip_listen_api()

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>call rsvp-sync</td>
<td>Enables synchronization between RSVP and the H.323 voice signaling protocol.</td>
</tr>
<tr>
<td>call rsvp-sync resv-timer</td>
<td>Sets the timer for RSVP reservation setup.</td>
</tr>
<tr>
<td>debug call rsvp-sync events</td>
<td>Displays the events that occur during RSVP synchronization.</td>
</tr>
<tr>
<td>show call rsvp-sync conf</td>
<td>Displays the RSVP synchronization configuration.</td>
</tr>
<tr>
<td>show call rsvp-sync stats</td>
<td>Displays statistics for calls that attempted RSVP reservation.</td>
</tr>
</tbody>
</table>
**debug call threshold**

To see details of the trigger actions, use the `debug call threshold` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

**debug call threshold module**

`no debug call threshold`

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>The module argument can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td>module</td>
<td>• core --Traces the resource information.</td>
</tr>
<tr>
<td></td>
<td>• detail --Traces for detail information.</td>
</tr>
</tbody>
</table>

**Command Default**

Disabled

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(2)XA</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(4)T</td>
<td>The command was integrated into Cisco IOS Release 12.2(4)T. Support for the Cisco AS5300, Cisco AS5350, and Cisco AS5400 is not included in this release.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850 platform.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(8)T and implemented on Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>Support for this command was implemented on Cisco AS5850, Cisco AS5800, Cisco AS5300, Cisco AS5350, and Cisco AS5400 series images.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug call threshold core` command:

```
Router# debug call threshold core
RSCCAC Core info debugging is on
```

The following is sample output from the `debug call threshold detail` command:

```
Router# debug call threshold detail
All RSCCAC info debugging is on
```
debug call treatment action

To debug the call treatment actions, use the `debug call treatment action` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug call treatment action
no debug call treatment action
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Disabled

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(2)XA</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(4)T</td>
<td>The command was integrated into Cisco IOS Release 12.2(4)T.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850 platform.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(8)T and implemented on Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>Support for this command was implemented on Cisco AS5850, Cisco AS5800, Cisco AS5300, Cisco AS5350, and Cisco AS5400 series images.</td>
</tr>
</tbody>
</table>

**Examples**

Debug actions are performed on calls by call treatment. The following sample output shows that call treatment is turned on:

```
Router# debug call treatment action
Call treatment action debugging is on
```
debug callback

To display callback events when the router is using a modem and a chat script to call back on a terminal line, use the `debugcallback` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
deb ug callback
no debug callback
```

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

This command is useful for debugging chat scripts on PPP and AppleTalk Remote Access Protocol (ARAP) lines that use callback mechanisms. The output provided by the `debugcallback` command shows you how the call is progressing when used with the `debugppp` or `debugarap` commands.

Examples

The following is sample output from the `debugcallback` command:

```
Router# debug callback
TTY7 Callback process initiated, user: exec_test dialstring 123456
TTY7 Callback forced wait = 4 seconds
TTY7 Exec Callback Successful - await exec/autoselect pickup
TTY7: Callback in effect
```

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cable env</td>
<td>Displays ARAP events.</td>
</tr>
<tr>
<td>debug ppp</td>
<td>Displays information on traffic and exchanges in an internetwork implementing the PPP.</td>
</tr>
</tbody>
</table>
debug capf-server

To collect debug information about the CAPF server, use the `debug capf-server` command in privileged EXEC mode. To disable collection of debug information, use the `no` form of this command.

```
debug capf-server {all| error| events| messages}
no debug capf-server
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Collect all CAPF information available.</td>
</tr>
<tr>
<td>error</td>
<td>Collect only information about CAPF errors.</td>
</tr>
<tr>
<td>events</td>
<td>Collect only information about CAPF status events.</td>
</tr>
<tr>
<td>messages</td>
<td>Collect only CAPF system messages.</td>
</tr>
</tbody>
</table>

**Command Default**

Collection of CAPF debug information is disabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Cisco IOS Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(4)XC</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(9)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(9)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

This command is used with Cisco Unified CallManager Express phone authentication.

**Examples**

The following example shows debug messages for the CAPF server.

```
Router# debug capf-server all
001891: .Jul 21 18:17:07.014: %IPPHONE-6-UNREGISTER_NORMAL: ephone-1:SEP000E325C9A43
IP:10.10.10.194 Socket:3 DeviceType:Phone has unregistered normally.
001897: .Jul 21 18:17:22.555: ephone_capf_send_auth_req:
```

Cisco IOS Debug Command Reference - Commands A through D

228
debug backhaul-session-manager session through debug channel packets

debug capf-server

001902: Jul 21 18:17:22.711: ephone_capf_send_delete_cert_req_msg: SEP000E325C9A43
001903: Jul 21 18:17:22.711: ephone_capf_ssl_write: 8 bytes
001904: Jul 21 18:17:23.891: ephone_capf_ssl_read: Read 12 bytes
001905: Jul 21 18:17:23.891: ephone_capf_handle_phone_msg: msgtype 14
001906: Jul 21 18:17:23.891: certificate delete successful for SEP000E325C9A43
001907: Jul 21 18:17:24.695: ephone_capf_release_session: SEP000E325C9A43
001908: Jul 21 18:17:24.695: ephone_capf_send_end_session_msg: SEP000E325C9A43
001909: Jul 21 18:17:24.695: ephone_capf_ssl_write: 12 bytes
001910: Jul 21 18:17:25.095: %IPPHONE-6-REG_ALARM: 22: Name=SEP000E325C9A43 Load=7.2(2.0) Last=Reset
001911: Jul 21 18:18:25.099: %IPPHONE-6-REGISTER: ephone-1:SEP000E325C9A43 IP:10.10.10.194 Socket:2 DeviceType:Phone has registered.
001912: Jul 21 18:18:05.171: %IPPHONE-6-UNREGISTER_NORMAL: ephone-1:SEP000E325C9A43 IP:1.1.1.127 Socket:2 DeviceType:Phone has unregistered normally.
001913: Jul 21 18:18:18.288: New Connection from phone, socket 1
001914: Jul 21 18:18:18.288: Created New Handshake Process
001915: Jul 21 18:18:18.292: SSL Handshake Error -6983
001916: Jul 21 18:18:19.292: SSL Handshake Error -6983
001917: Jul 21 18:18:20.348: SSL Handshake Successful
001918: Jul 21 18:18:20.348: ephone_capf_send_auth_req:
001919: Jul 21 18:18:20.348: ephone_capf_ssl_write: 12 bytes^Z
001920: Jul 21 18:18:20.492: ephone_capf_ssl_read: Read 35 bytes
001923: Jul 21 18:18:20.492: ephone_capf_send_PhKeyGenReq_msg: SEP000E325C9A43 KeySize 1024
001924: Jul 21 18:18:20.492: ephone_capf_ssl_write: 13 bytes
001925: Jul 21 18:18:20.540: ephone_capf_ssl_read: Read 8 bytes
001926: Jul 21 18:18:20.540: ephone_capf_handle_phone_msg: msgtype 17
001927: Jul 21 18:18:20.540: ephone_capf_process_req_in_progress: SEP000E325C9A43 delay 0sh
001928: Jul 21 18:18:21.924: %SYS-5-CONFIG_I: Configured from console by user1 on console

Cisco IOS Debug Command Reference - Commands A through D 229
debug cas

To debug channel-associated signaling (CAS) messages and to debug the establishment of a time-division multiplexing (TDM) connection between a DS0 and a digital modem, use the `debug cas` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cas slot slot number port port number
no debug cas slot slot number port port number
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slot</td>
<td>slot number. Valid values are 0 and 1.</td>
</tr>
<tr>
<td>port</td>
<td>port number. Valid values are 0 and 1.</td>
</tr>
</tbody>
</table>

**Command Default**
Disabled

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(7)T</td>
<td>This command was introduced for the Cisco AS5200 and AS5300 platforms.</td>
</tr>
<tr>
<td>12.2(2)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(2)T and support was added for the Cisco 2600 series and Cisco 3600 series platforms.</td>
</tr>
<tr>
<td>12.3(1)</td>
<td>This command was integrated into Cisco IOS Release 12.3(1) and support was added for the Cisco 2600 XM series, Cisco 2691, and Cisco 3700 series platforms.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

When the NM-xCE1T1PRI network module is used with an NM-xDM and a DS0-group is configured under the controller, you can use the `debug cas` command to debug CAS signaling messages and the establishment of a TDM connection between a DS0 and a digital modem. Use the `debug cas` command to identify and troubleshoot call connection problems on a T1/E1 interface. With this command, you can trace the complete sequence of incoming and outgoing calls.
The following shows an example session to enable debugging CAS and generate troubleshooting output:

Router# `show debug`

Router# `debug cas slot 1 port 0`

CAS debugging is on

Router# `debug-cas is on at slot(1) dsx1(0)`

CAS debugging is on

The following example shows output for the first outgoing call:

Router# `p 1.1.1.2`

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 1.1.1.2, timeout is 2 seconds:

* Mar 2 00:17:45: dsx1_alloc_cas_channel: channel 0 dsx1_timeslot
  1(0/0): TX SEIZURE (ABCD=0001)(0/0): RX SEIZURE_ACK (ABCD=1101)(0/1):
  RX_IDLE (ABCD=1001)(0/2): RX_IDLE (ABCD=1001)(0/3): RX_IDLE
  (Type e000)(0/4): RX_IDLE (ABCD=1001)(0/5): RX_IDLE (ABCD=1001)(0/6):
  RX_IDLE (ABCD=1001)(0/7): RX_IDLE (ABCD=1001)(0/8): RX_IDLE
  (ABCD=1001)(0/9): RX_IDLE (ABCD=1001)(0/10): RX_IDLE (ABCD=1001)(0/11):
  RX_IDLE (ABCD=1001)(0/12): RX_IDLE (ABCD=1001)(0/13): RX_IDLE
  (ABCD=1001)(0/14): RX_IDLE (ABCD=1001)(0/16): RX_IDLE (ABCD=1001)(0/17):
  RX_IDLE (ABCD=1001)(0/18): RX_IDLE (ABCD=1001)(0/19): RX_IDLE
  (ABCD=1001)(0/20): RX_IDLE (ABCD=1001)(0/21): RX_IDLE
  (ABCD=1001)(0/22): RX_IDLE (ABCD=1001)(0/23): RX_IDLE
  (ABCD=1001)(0/24): RX_IDLE (ABCD=1001)(0/25): RX_IDLE (ABCD=1001)(0/26):
  RX_IDLE (ABCD=1001)(0/27): RX_IDLE (ABCD=1001)(0/28): RX_IDLE
  (ABCD=1001)(0/29): RX_IDLE (ABCD=1001)(0/30): RX_IDLE
  (ABCD=1001)...(0/0): RX_ANSWERED (ABCD=0101).

Success rate is 0 percent (0/5)

Router# `clear int dialer 1`

Router# `p 1.1.1.2`

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 1.1.1.2, timeout is 2 seconds:

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 160/180/236 ms

The following example shows that the call is cleared on the router:

Router# `clear int dialer 1`

Router# `p 1.1.1.2`

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 1.1.1.2, timeout is 2 seconds:

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 160/180/236 ms

The following example shows a subsequent outbound CAS call:

Router# `clear int dialer 1`

Router# `p 1.1.1.2`

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 1.1.1.2, timeout is 2 seconds:

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 160/180/236 ms

The following example shows a subsequent outbound CAS call:
Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>show debug</td>
<td>Displays information about the types of debugging that are enabled for your router.</td>
</tr>
</tbody>
</table>
debug ccaal2 session

To display the ccaal2 function calls during call setup and teardown, use the **debug ccaal2 session** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
default ccaal2 session
no default ccaal2 session
```

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

**Command Default**
Debugging for ATM Adaptation Layer type 2 (AAL2) sessions is not enabled.

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(1)XA</td>
<td>This command was introduced for the Cisco MC3810 series.</td>
</tr>
<tr>
<td>12.1(2)T</td>
<td>This command was integrated in Cisco IOS Release 12.1(2)T.</td>
</tr>
<tr>
<td>12.2(2)T</td>
<td>Support for this command was implemented on the Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
Use this command when troubleshooting an AAL2 trunk setup or teardown problem.

**Examples**
The following example shows sample output from the **debug ccaal2 session** command for a forced shutdown of a voice port:

```
Router# debug ccaal2 session
CCAAL2 Session debugging is on
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# voice-port 2/0:0
Router(config-voiceport)# shutdown
00:32:45:ccaal2_call_disconnect:peer tag 0
00:32:45:ccaal2_evhandle_call_disconnect:Entered
00:32:45:ccaal2_call_cleanup:freeccb i, call_disconnected 1
00:32:45:starting incoming timer:Setting accept_incoming to FALSE and
00:32:45:timer 2:(0x622F6270)starts - delay (70000)
00:32:45:ccaal2_call_cleanup:Generating Call record
00:32:45:cause=81 tcause=81 cause_text=unspecified
00:32:45:ccaal2_call_cleanup:ccb 0x63FF1700, vdbPtr 0x62DFF2E0
  freeccb_flag=1, call_disconnected_flag=1
00:32:45:%LINK-3-UPDOWN:Interface receive and transmit 2/0:0(1), changed state to Administrative Shutdown
```
The following example shows sample output from the `debug ccaal2 session` command for a trunk setup on a voice port:

Router# debug ccaal2 session
Router(config-voiceport)# no shutdown
Router(config-voiceport)#
00:35:28:%LINK-3-UPDOWN:Interface recEive and transMit2/0:0(1), changed state to up
00:35:35:ccaal2_call_setup_request:Entered
00:35:35:ccaal2_evhandle_call_setup_request:Entered
00:35:35:ccaal2_initialize_ccb:preferred_codec set(-1)(0)
00:35:35:ccaal2_evhandle_call_setup_request:preferred_codec set(5)(40). VAD is 1
00:35:35:ccaal2_call_setup_trunk:subchannel linking successful
ccaal2_receive:xmitFunc is NULL
00:35:35:ccaal2_caps_ind:PeerTag = 49
00:35:35: codec(preferred) = 1, fax_rate = 2, vad = 2
00:35:35: codec_bytes = 40, signal_type=8
00:35:36:%HTSP-5-UPDOWN:Trunk port(channel) [2/0:0(1)] is up
Router(config-voiceport)#

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>show debug</code></td>
<td>Shows which debug commands are enabled.</td>
</tr>
</tbody>
</table>
debug cce dp named-db urlfilter

To enable debug information of the Common Classification Engine Data-Plane (CCE DP) URL Filtering Classification module, use the `debug cce dp named-db urlfilter` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

dep cce dp named-db urlfilter
no debug cce dp named-db urlfilter

Syntax Description
This command has no keywords or arguments.

Command Default
No debugging information is generated for the the CCE DP URL Filtering Classification module.

Command Modes
Privileged EXEC (#)

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(15)XZ</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(20)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(20)T.</td>
</tr>
</tbody>
</table>

Examples
The following is sample output from the `debug cce dp named-db urlfilter` command at the time that a URL request to the untrusted domain www.example.com was made:

```
Router# debug cce dp named-db urlfilter
CCE DP Named DB URLF functionality debugging is on on
Router#
*Apr 4 10:38:08.043: CCE* FUNC: cce_dp_named_db_urlf_pkt_classify -- Didn't get token
*Apr 4 10:38:08.043: CCE* FUNC: cce_dp_urlf_truncate_url -- Truncating URL upto script before sending to the trend for classification
*Apr 4 10:38:08.043: CCE* FUNC: urlf_trend_find_cache_entry -- The host tree in bucket 1248 is empty
*Apr 4 10:38:08.043: CCE* FUNC: cce_dp_named_db_urlf_pkt_classify -- Didn't find in cache
*Apr 4 10:38:08.051: CCE FUNC: urlf_trend_store_response -- Host node with given domain name not found.
*Apr 4 10:38:08.051: CCE FUNC: urlf_trend_store_response -- Create domain type cache entry.
*Apr 4 10:38:08.051: CCE FUNC: cache_size_limit_check -- New cache size=73, existing cache size=0, cache size limit=131072000
*Apr 4 10:38:08.051: CCE FUNC: create_domain_cache_entry -- Domain cache entry 0x65EE0ED0 created.
*Apr 4 10:38:08.051: CCE FUNC: create_and_insert_domain_cache_entry --
*Apr 4 10:38:08.051: Domain cache entry 0x65EE0ED0 created and inserted into host tree with root=0x65EE0ED0, root left=0x0, root right=0x0; new node left=0x0, new node right=0x0
*Apr 4 10:38:08.051: CCE FUNC: cce_dp_named_db_urlf_gen_match_token -- pushing match-info token - class 0x000000E; filter 45; category 21
*Apr 4 10:38:08.051: CCE FUNC: cce_dp_named_db_urlf_non_pkt_classify -- Class 0x65C5D484 matched
*Apr 4 17:38:08.051: %URLF-4-URL_BLOCKED: Access denied URL 'http://www.example.com/'
```

Cisco IOS Debug Command Reference - Commands A through D
debug cce dp named-db urlfilter

*Apr 4 10:38:08.055: CCE* FUNC: cce_dp_named_db_urlf_pkt_classify -- Didn't get token
*Apr 4 10:38:08.055: CCE FUNC: cce_dp_named_db_urlf_pkt_classify -- Didn't get token
debug ccfrf11 session

To display the ccfrf11 function calls during call setup and teardown, use the **debug ccfrf11 session** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug ccfrf11 session**

**no debug ccfrf11 session**

**Syntax Description**

This command has no keywords or arguments.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(3)XG</td>
<td>This command was introduced for the Cisco 2600 and Cisco 3600 series routers.</td>
</tr>
<tr>
<td>12.0(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.0(4)T.</td>
</tr>
<tr>
<td>12.0(7)XK</td>
<td>This command was first supported on the Cisco MC3810 series.</td>
</tr>
<tr>
<td>12.1(2)T</td>
<td>Support for this command was implemented in Cisco MC3810 images.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command to display debug information about the various FRF.11 VoFR service provider interface (SPI) functions. Note that this debug command does not display any information regarding the proprietary Cisco switched-VoFR SPI.

This debug is useful only when the session protocol is "frf11-trunk."

**Examples**

The following is sample output from the **debug ccfrf11 session** command:

```
Router# debug ccfrf11 session
INCOMING CALL SETUP (port setup for answer-mode):
*Mar 6 18:04:07.693:ccfrf11_process_timers:scb (0x60EB6040) timer (0x60EB6098) expired
*Mar 6 18:04:07.693:Setting accept_incoming to TRUE
*Mar 6 18:04:11.213:ccfrf11_incoming_request:peer tag 800:callingNumber=+2602100,
calledNumber=+3622110
*Mar 6 18:04:11.217: codec(preferred) = 4, fax_rate = 2, vad = 2
*Mar 6 18:04:11.217: cid = 30, config_bitmask = 0, codec_bytes = 20, signal_type=2
*Mar 6 18:04:11.217: required_bandwidth 8192
*Mar 6 18:04:11.221:ccfrf11_evhandle_call_connect:Entered
```

Cisco IOS Debug Command Reference - Commands A through D
CALL SETUP (MASTER):
5d22h:ccfrf11_call_setup_request:Entered
5d22h:ccfrf11_evhandle_call_setup_request:Entered
5d22h:ccfrf11_initialize_ccb:preferred_codec set(-1)(0)
5d22h:ccfrf11_evhandle_call_setup_request:preferred_codec set(9)(24)
5d22h:ccfrf11_call_setup_trunk:subchannel linking successful
5d22h:ccfrf11_caps_ind:PeerTag = 810
5d22h: codec(preferred) = 512, fax_rate = 2, vad = 2
5d22h: cid = 30, config_bitmask = 1, codec_bytes = 24, signal_type=2
5d22h: required_bandwidth 6500
5d22h:ccfrf11_caps_ind:Bandwidth reservation of 6500 bytes succeeded.

CALL TEARDOWN:
*Mar 6 18:09:14.805:ccfrf11_call_cleanup:freeccb 1, call_connected 1
*Mar 6 18:09:14.809:timer 2:(0x60EB6098)starts - delay (70000)
*Mar 6 18:09:14.809:timer 1:(0x60F64104) stops
*Mar 6 18:09:14.809:ccfrf11_call_cleanup:ccb 0x60F6404C, vdbPtr 0x610DB7A4
freeccb_flag=1, call_disconnect_flag=1

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug call rsvp-sync events</td>
<td>Displays the ccsvoice function calls during call setup and teardown.</td>
</tr>
<tr>
<td>debug ccsvoice vofr-session</td>
<td>Displays the ccsvoice function calls during call setup and teardown.</td>
</tr>
<tr>
<td>debug vtsp session</td>
<td>Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.</td>
</tr>
</tbody>
</table>
debug cch323

To provide debugging output for various components within the H.323 subsystem, use the debug cch323 command in privileged EXEC mode. To disable debugging output, use the no form of this command.

```
debug cch323 { all| error| h225| h245| nxe| ras| rawmsg| session }
no debug cch323
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Enables all debug cch323 commands.</td>
</tr>
<tr>
<td>error</td>
<td>Traces errors encountered in the H.323 subsystem and can be used to help troubleshoot problems with H.323 calls.</td>
</tr>
<tr>
<td>h225</td>
<td>Traces the state transition of the H.225 state machine on the basis of the processed event.</td>
</tr>
<tr>
<td>h245</td>
<td>Traces the state transition of the H.245 state machine on the basis of the processed events.</td>
</tr>
<tr>
<td>nxe</td>
<td>Displays Annex E events that have been transmitted and received.</td>
</tr>
<tr>
<td>ras</td>
<td>Traces the state transition of the Registration, Admission, and Status (RAS) state machine on the basis of the processed events.</td>
</tr>
<tr>
<td>rawmsg</td>
<td>Troubleshoots raw message buffer problems.</td>
</tr>
<tr>
<td>session</td>
<td>Traces general H.323 events and can be used to troubleshoot H.323 problems.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(6)NA2</td>
<td>The debug cch323 command and the following keywords were introduced: h225, h245, and ras.</td>
</tr>
<tr>
<td>12.2(2)XA</td>
<td>The nxe keyword was added.</td>
</tr>
</tbody>
</table>
The following keywords were introduced: all, error, rawmsg, and session. The nxe keyword was integrated into Cisco IOS Release 12.2(4)T on all Cisco H.323 platforms. This command does not support the Cisco access server platforms in this release.

This command was implemented on the Cisco AS5850.

Usage Guidelines

The debug cch323 Command with the all Keyword

When used with the debug cch323 command, the all keyword provides debug output for various components within the H.323 subsystem.

The debug cch323 command used with the all keyword enables the following debug cch323 commands:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>Enables a CCH323 Service Provider Interface (SPI) trace.</td>
</tr>
<tr>
<td>h225</td>
<td>Enables an H225 state machine debugging trace.</td>
</tr>
<tr>
<td>h245</td>
<td>Enables an H245 state machine debugging trace.</td>
</tr>
<tr>
<td>nxe</td>
<td>Enables an Annex E debugging trace.</td>
</tr>
<tr>
<td>ras</td>
<td>Enables a RAS state machine debugging trace.</td>
</tr>
<tr>
<td>rawmsg</td>
<td>Enables a CCH323 RAWMSG debugging trace.</td>
</tr>
<tr>
<td>session</td>
<td>Enables a Session debugging trace.</td>
</tr>
</tbody>
</table>

Caution

Using the debug cch323 all command could slow your system and flood the TTY if there is significant call traffic.

The debug cch323 Command with the error Keyword

When used with the debug cch323 command, the error keyword allows you to trace errors encountered in the H.323 subsystem.

Note

There is little or no output from this command when there is a stable H.323 network.

The debug cch323 Command with the h225 Keyword

When used with the debug cch323 command, the h225 keyword allows you to trace the state transition of the H.225 state machine on the basis of the processed event.

The definitions of the different states of the H.225 state machine follow:
• H225_IDLE--This is the initial state of the H.225 state machine. The H.225 state machine is in this state before issuing a call setup request (for the outbound IP call case) or when ready to receive an incoming IP call.

• H225_SETUP--This is the call setup state. The state machine changes to this state after sending out a call setup request or after receiving an incoming call indication.

• H225_ALERT--This is the call alerting state. The state machine changes to this state after sending the alerting message or after receiving an alerting message from the peer.

• H225_CALLPROC--This is the call proceeding state.

• H225_ACTIVE--This is the call connected state. In this state, the call is active. The state machine changes to this state after sending the connect message to the peer or after receiving the connect message from the peer.

• H225_WAIT_FOR_ARQ--This is the state in which the H.225 state machine is waiting for the completion of the Admission Request (ARQ) process from the RAS state machine.

• H225_WAIT_FOR_DRQ--This is the state in which the H.225 state machine is waiting for the completion of the Disengage Request (DRQ) process from the RAS state machine.

• H225_WAIT_FOR_H245--This is the state in which the H.225 state machine is waiting for the success or failure from the H.245 state machine.

The definitions of the different events of the H.225 state machine follow:

• H225_EVENT_NONE--There is no event.

• H225_EVENT_ALERT--This event instructs the H.225 state machine to send an alert message to the peer.

• H225_EVENT_ALERT_IND--This event indicates to the H.225 state machine that an alert message arrived from the peer.

• H225_EVENT_CALLPROC--This event instructs the H.225 state machine to send a call proceeding message to the peer.

• H225_EVENT_CALLPROC_IND--This event indicates to the H.225 state machine that a call proceeding message has been received from the peer.

• H225_EVENT_REJECT--This event instructs the H.225 state machine to reject the call setup request from the peer.

• H225_EVENT_REJECT_IND--This event indicates to the H.225 state machine that a call setup request to the peer has been rejected.

• H225_EVENT_RELEASE--This event instructs the H.225 state machine to send a release complete message to the peer.

• H225_EVENT_RELEASE_IND--This event indicates to the H.225 state machine that a release complete message has been received from the peer.

• H225_EVENT_SETUP--This event instructs the H.225 state machine to send a setup message to the peer.

• H225_EVENT_SETUP_IND--This event indicates to the H.225 state machine that a setup message has been received from the peer.
• H225_EVENT_SETUP_CFM--This event instructs the H.225 state machine to send a connect message to the peer.
• H225_EVENT_SETUP_CFM_IND--This event indicates to the H.225 state machine that a connect message arrived from the peer.
• H225_EVENT_RAS_SUCCESS--This event indicates to the H.225 state machine that the pending RAS operation succeeded.
• H225_EVENT_RAS_FAILED--This event indicates to the H.225 state machine that the pending RAS operation failed.
• H225_EVENT_H245_SUCCESS--This event indicates to the H.225 state machine that the pending H.245 operation succeeded.
• H225_EVENT_H245_FAILED--This event indicates to the H.225 state machine that the pending H.245 operation failed.

The debug cch323 Command with the h245 Keyword

When used with the debug cch323 command, the h245 keyword allows you to trace the state transition of the H.245 state machine on the basis of the processed event.

The H.245 state machines include the following three state machines:

• Master slave determination (MSD) state machine
• Capability exchange (CAP) state machine
• Open logical channel (OLC) state machine

The state definitions follow:

• H245_MS_NONE--This is the initial state of the MSD state machine.
• H245_MS_WAIT--In this state, an MSD message is sent, and the device is waiting for the reply.
• H245_MS_DONE--The result is in.
• H245_CAP_NONE--This is the initial state of the CAP state machine.
• H245_CAP_WAIT--In this state, a CAP message is sent, and the device is waiting for the reply.
• H245_CAP_DONE--The result is in.
• H245_OLC_NONE--This is the initial state of the OLC state machine.
• H245_OLC_WAIT--In this state, an OLC message is sent, and the device is waiting for the reply.
• H245_OLC_DONE--The result is in.

The event definitions follow:

• H245_EVENT_MSD--Send MSD message.
• H245_EVENT_MS_CFM--Send MSD acknowledge message.
• H245_EVENT_MS_REJ--Send MSD reject message.
• H245_EVENT_MS_IND--Received MSD message.
• H245_EVENT_CAP--Send CAP message.
• H245_EVENT_CAP_CFM--Send CAP acknowledge message.
• H245_EVENT_CAP_REJ--Send CAP reject message.
• H245_EVENT_CAP_IND--Received CAP message.
• H245_EVENT_OLC--Send OLC message.
• H245_EVENT_OLC_CFM--Send OLC acknowledge message.
• H245_EVENT_OLC_REJ--Send OLC reject message.
• H245_EVENT_OLC_IND--Received OLC message.

The debug cch323 Command with the nxe Keyword
When used with the debug cch323 command, the nxe keyword allows you to display the Annex E events that have been transmitted and received.

The debug cch323 Command with the ras Keyword
When used with the debug cch323 command, the ras keyword allows you to trace the state transition of the RAS state machine based on the processed events.

RAS operates in two state machines. One global state machine controls the overall RAS operation of the gateway. The other state machine is a per-call state machine that controls the active calls.

The definitions of the different states of the RAS state machine follow:

• CCH323_RAS_STATE_NONE--This is the initial state of the RAS state machine.
• CCH323_RAS_STATE_GRQ--The state machine is in the Gatekeeper Request (GRQ) state. In this state, the gateway is discovering a gatekeeper.
• CCH323_RAS_STATE_RRQ--The state machine is in the Registration Request (RRQ) state. In this state, the gateway is registering with a gatekeeper.
• CCH323_RAS_STATE_IDLE--The global state machine is in the idle state.
• CCH323_RAS_STATE_URQ--The state machine is in the Unregistration Request (URQ) state. In this state, the gateway is in the process of unregistering with a gatekeeper.
• CCH323_RAS_STATE_ARQ--The per-call state machine is in the process of admitting a new call.
• CCH323_RAS_STATE_ACTIVE--The per-call state machine is in the call active state.
• CCH323_RAS_STATE_DRQ--The per-call state machine is in the process of disengaging an active call.

The definitions of the different events of the RAS state machine follow:

• CCH323_RAS_EVENT_NONE--Nothing.
• CCH323_RAS_EVENT_GWUP--Gateway is coming up.
• CCH323_RAS_EVENT_GWDWN--Gateway is going down.
• CCH323_RAS_EVENT_NEWCALL--New call.
• CCH323_RAS_EVENT_CALLDISC--Call disconnect.
• CCH323_RAS_EVENT_GCF--Received Gatekeeper Confirmation (GCF).
• CCH323_RAS_EVENT_GRJ--Received Gatekeeper Rejection (GRJ).
• CCH323_RAS_EVENT_ACF--Received Admission Confirmation (ACF).
• CCH323_RAS_EVENT_ARJ--Received Admission Reject (ARJ).
• CCH323_RAS_EVENT_SEND_RRQ--Send Registration Request (RRQ).
• CCH323_RAS_EVENT_RCF--Received Registration Confirmation (RCF).
• CCH323_RAS_EVENT_RRJ--Received Registration Rejection (RRJ).
• CCH323_RAS_EVENT_SEND_URQ--Send Unregistration Request (URQ).
• CCH323_RAS_EVENT_URQ--Received URQ.
• CCH323_RAS_EVENT_UCF--Received Unregister Confirmation (UCF).
• CCH323_RAS_EVENT_SEND_UCF--Send UCF.
• CCH323_RAS_EVENT_URJ--Received Unregister Reject (URJ).
• CCH323_RAS_EVENT_BCF--Received Bandwidth Confirm (BCF).
• CCH323_RAS_EVENT_BRJ--Received Bandwidth Rejection (BRJ).
• CCH323_RAS_EVENT_DRQ--Received Disengage Request (DRQ).
• CCH323_RAS_EVENT_DCF--Received Disengage Confirm (DCF).
• CCH323_RAS_EVENT_SEND_DCF--Send DCF.
• CCH323_RAS_EVENT_DRJ--Received Disengage Reject (DRJ).
• CCH323_RAS_EVENT_IRQ--Received Interrupt Request (IRQ).
• CCH323_RAS_EVENT_IRR--Send Information Request (IRR).
• CCH323_RAS_EVENT_TIMEOUT--Message timeout.

The debug cch323 Command with the rawmsg Keyword

When used with the debugcch323 command, the rawmsg keyword allows you to troubleshoot raw message buffer problems.

⚠️ Caution

Using the debugcch323 command with the rawmsg keyword could slow your system and flood the TTY if there is significant call traffic.

The debug cch323 Command with the session Keyword

Used with the debugcch323 command, the session keyword allows you to trace general H.323 events.

⚠️ Caution

Using the debugcch323session command could slow your system and flood the TTY if there is significant call traffic.

Examples

The debugcch323all command and keyword combination provides output for the following keywords: error, h225, h245, nxe, ras, rawmsg, and session. Examples of output for each keyword follow.
The following is sample output from a typical debug cch323 error request on a Cisco 3640 router:

```
Router# debug cch323 error
cch323_h225_receiver:received msg of unknown type 5
```

The following is sample output from a typical debug cch323 h225 request on a Cisco 3640 router:

```
Router# debug cch323 h225
20:59:17: Set new event H225_EVENT_SETUP
20:59:17: H225 FSM: received event H225_EVENT_SETUP while at state H225_IDLE
20:59:17: Changing from H225_IDLE state to H225_SETUP state
20:59:17: cch323_h225_receiver: received msg of type SETUPCFM_CHOSEN
20:59:17: H225 FSM: received event H225_EVENT_SETUP_CFM_IND while at state H225_SETUP
20:59:17: Changing from H225_SETUP state to H225_ACTIVE state
20:59:17: H225 FSM: received event H225_EVENT_H245_SUCCESS while at state H225_ACTIVE
20:59:20: Set new event H225_EVENT_RELEASE
20:59:20: H225 FSM: received event H225_EVENT_RELEASE while at state H225_ACTIVE
20:59:20: Changing from H225_ACTIVE state to H225_WAIT_FOR_DRQ state
20:59:20: H225 FSM: received event H225_EVENT_RAS_SUCCESS while at state H225_WAIT_FOR_DRQ
20:59:20: Changing from H225_WAIT_FOR_DRQ state to H225_IDLE state
```

The table below describes the significant fields shown in the display.

### Table 44: debug cch323 h225 Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H225_EVENT_SETUP</td>
<td>This event instructs the H.225 state machine to send a setup message to the peer.</td>
</tr>
<tr>
<td>H225_IDLE</td>
<td>The initial state of the H.225 state machine. The H.225 state machine is in this state before issuing a call setup request (for the outbound IP call case) or when ready to receive an incoming IP call.</td>
</tr>
<tr>
<td>H225_SETUP</td>
<td>The call setup state. The state machine changes to this state after sending out a call setup request or after receiving an incoming call indication.</td>
</tr>
<tr>
<td>SETUPCFM_CHOSEN</td>
<td>The H225 connect message that has been received from a remote H323 endpoint.</td>
</tr>
<tr>
<td>H225_EVENT_SETUP_CFM_IND</td>
<td>This event indicates to the H.225 state machine that a connect message arrived from the peer.</td>
</tr>
<tr>
<td>H225_ACTIVE</td>
<td>The call connected state. In this state, the call is active. The state machine changes to this state after sending the connect message to the peer or after receiving the connect message from the peer.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>H225_EVENT_H245_SUCCESS</td>
<td>This event indicates to the H.225 state machine that the pending H.245 operation succeeded.</td>
</tr>
<tr>
<td>H225_EVENT_RELEASE</td>
<td>This event instructs the H.225 state machine to send a release complete message to the peer.</td>
</tr>
<tr>
<td>H225_WAIT_FOR_DRQ</td>
<td>The state in which the H.225 state machine is waiting for the completion of the DRQ process from the RAS state machine.</td>
</tr>
<tr>
<td>H225_EVENT_RAS_SUCCESS</td>
<td>This event indicates to the H.225 state machine that the pending RAS operation succeeded.</td>
</tr>
<tr>
<td>H225 FSM</td>
<td>The finite state machine.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from a typical `debug cch323 h245` request on a Cisco 3640 router:

```
Router# debug cch323 h245
20:58:23:Changing to new event H245_EVENT_MSD
20:58:23:H245 MS FSM:received event H245_EVENT_MSD while at state H245_MS_NONE
20:58:23:Changing from H245_MS_NONE state to H245_MS_WAIT state
20:58:23:Changing to new event H245_EVENT_CAP
20:58:23:H245 CAP FSM:received event H245_EVENT_CAP while at state H245_CAP_NONE
20:58:23:changing from H245_CAP_NONE state to H245_CAP_WAIT state
20:58:23:cch323_h245_receiver:received msg of type M_H245_MS_DETERMINE_INDICATION
20:58:23:Changing to new event H245_EVENT_MS_IND
20:58:23:H245 MS FSM:received event H245_EVENT_MS_IND while at state H245_MS_WAIT
20:58:23:cch323_h245_receiver:received msg of type M_H245_CAP_TRANSFER_INDICATION
20:58:23:Changing to new event H245_EVENT_CAP_IND
20:58:23:H245 CAP FSM:received event H245_EVENT_CAP_IND while at state H245_CAP_WAIT
20:58:23:cch323_h245_receiver:received msg of type M_H245_MS_DETERMINE_CONFIRM
20:58:23:Changing to new event H245_EVENT_MS_CFM
20:58:23:H245 MS FSM:received event H245_EVENT_MS_CFM while at state H245_MS_WAIT
20:58:23:changing from H245_MS_WAIT state to H245_MS_DONE state
0:58:23:cch323_h245_receiver:received msg of type M_H245_CAP_TRANSFER_CONFIRM
20:58:23:Changing to new event H245_EVENT_CAP_CFM
20:58:23:H245 CAP FSM:received event H245_EVENT_CAP_CFM while at state H245_CAP_WAIT
20:58:23:changing from H245_CAP_WAIT state to H245_CAP_DONE state
20:58:23:Changing to new event H245_EVENT_OLC
20:58:23:H245 OLC FSM:received event H245_EVENT_OLC while at state H245_OLC_NONE
20:58:23:changing from H245_OLC_NONE state to H245_OLC_WAIT state
20:58:23:cch323_h245_receiver:received msg of type M_H245_UCHAN_ESTABLISH_INDICATION
20:58:23:Changing to new event H245_EVENT_OLC_IND
20:58:23:H245 OLC FSM:received event H245_EVENT_OLC_IND while at state H245_OLC_WAIT
20:58:23:cch323_h245_receiver:received msg of type M_H245_UCHAN_ESTAB_ACK
20:58:23:Changing to new event H245_EVENT_OLC_CFM
20:58:23:H245 OLC FSM:received event H245_EVENT_OLC_CFM while at state H245_OLC_DONE
```

20:58:23: changing from H245_OLC_WAIT state to H245_OLC_DONE state

The table below describes the significant fields shown in the display.

**Table 45: debug cch323 h245 Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H245_EVENT_MSD</td>
<td>Send MSD event message to the state machine.</td>
</tr>
<tr>
<td>H245 MS FSM</td>
<td>An H225 master slave determination finite state machine.</td>
</tr>
<tr>
<td>H245_MS_NONE</td>
<td>The initial state of the MSD state machine.</td>
</tr>
<tr>
<td>H245_MS_WAIT</td>
<td>In this state, a MSD message is sent, and the device is waiting for the reply.</td>
</tr>
<tr>
<td>H245_EVENT_CAP</td>
<td>Send CAP event message.</td>
</tr>
<tr>
<td>H245 CAP FSM</td>
<td>This is the H245 terminal CAP finite state machine.</td>
</tr>
<tr>
<td>H245_CAP_NONE</td>
<td>The initial state of the CAP state machine.</td>
</tr>
<tr>
<td>H245_CAP_WAIT</td>
<td>In this state, a CAP message is sent, and the device is waiting for the reply.</td>
</tr>
<tr>
<td>M_H245_MS_DETERMINE_INDICATION</td>
<td>The MSD message that has been received by an H245 terminal from a remote H323 endpoint.</td>
</tr>
<tr>
<td>H245_EVENT_MS_IND</td>
<td>Received MSD event message.</td>
</tr>
<tr>
<td>M_H245_CAP_TRANSFER_INDICATION</td>
<td>A CAP message that has been received by the H245 terminal from an H323 remote endpoint.</td>
</tr>
<tr>
<td>H245_EVENT_CAP_IND</td>
<td>Received CAP event message.</td>
</tr>
<tr>
<td>M_H245_MS_DETERMINE_CONFIRM</td>
<td>A confirmation message that the H245 master slave termination message was sent.</td>
</tr>
<tr>
<td>H245_EVENT_MS_CFM</td>
<td>Send MSD acknowledge event message.</td>
</tr>
<tr>
<td>H245_MS_DONE</td>
<td>The result is in.</td>
</tr>
<tr>
<td>M_H245_CAP_TRANSFER_CONFIRM</td>
<td>An indication to the H245 terminal that the CAP message was sent.</td>
</tr>
<tr>
<td>H245_EVENT_CAP_CFM</td>
<td>Send CAP acknowledge event message.</td>
</tr>
<tr>
<td>H245_CAP_DONE</td>
<td>The result is in.</td>
</tr>
<tr>
<td>H245_EVENT_OLC</td>
<td>Send OLC event message.</td>
</tr>
</tbody>
</table>
The initial state of the OLC state machine. **H245 OLC_NONE**

In this state, an OLC message is sent, and the device is waiting for the reply.

**H245 OLC_WAIT**

The OLC message received by an H245 terminal from a remote H323 endpoint.

**M_H245 UCHAN_ESTABLISH_INDICATION**

Received OLC event message. **H245 EVENT OLC_IND**

The OLC message acknowledgment received by an H245 terminal from a remote H323 endpoint.

**M_H245 UCHAN_ESTAB_ACK**

Send OLC acknowledge event message. **H245 EVENT OLC_CFM**

Send OLC acknowledge event message.

**H245 OLC FSM**

The result is in.

**H245 OLC_DONE**

The following is sample output from a `debugcch323nxe` request:

```
Router# debug cch323 nxe
00:15:54:nxe_handle_usrmgs_to_remote:User Message size is 227
00:15:54:nxe_msg_send_possible:Msg put in the active Q for CRV [3, direction flag 0]
00:15:54:nxe_send_msg:H323chan returns bytes sent=241, the actual len=241, to IPaddr [0xA4D4A02], Port [2517]
00:15:54:nxe_handle_usrmgs_to_remote:Usr Msg sent for IPaddr [0xA4D4A02], Port [2517], CRV [3, direction flag 0]
00:15:54:nxe_parse_msg_from_remote:Msg received from IP [0xA4D4A02], Port [2517]
00:15:54:nxe_parse_msg_from_remote:Value of PDU flags = 0x2
00:15:54:nxe_parse_payload:Transport msg type, Payload flag = 0x0
00:15:54:nxe_receive_ack:Ack received for 1 pdu
00:15:54:nxe_receiver:Ack received for seqnum=13 from IPAddr [0xA4D4A02], Port [2517]
00:15:54:nxe_parse_msg_from_remote:Msg received from IP [0xA4D4A02], Port [2517]
00:15:54:nxe_parse_msg_from_remote:Value of PDU flags = 0x3
00:15:54:nxe_parser:Static msg type, Payload flag = 0xA0
00:15:54:nxe_parser:Rx H225 msg from IPaddr [0xA4D4A02], Port [2517], CRV [3, direction flag 0]
00:15:54:nxe_make_ackmsg:NXE ACK Msg made to ack seqnum=14
00:15:54:nxe_send_msg:H323chan returns bytes sent=16, the actual len=16, to IPaddr [0xA4D4A02], Port [2517]
00:15:54:nxe_parser:Ack sent for Destination IPaddr [0xA4D4A02], Port [2517]
00:15:54:nxe_parser:Msg received from IP [0xA4D4A02], Port [2517]
00:15:54:nxe_parser:Msg received from IP [0xA4D4A02], Port [2517]
00:15:54:nxe_parser:Value of PDU flags = 0x3
00:15:54:nxe_parser:Static msg type, Payload flag = 0xA0
00:15:54:nxe_parser:Rx H225 msg from IPaddr [0xA4D4A02], Port [2517], CRV [3, direction flag 0]
```
Examples

The following is sample output from a typical `debug cch323 ras` request on a Cisco 3640 router:

```
Router# debug cch323 ras
20:58:49: Changing to new event CCH323_RAS_EVENT_SEND_RRQ
   cch323_run_ras_sm: received event CCH323_RAS_EVENT_SEND_RRQ while at CCH323_RAS_STATE_IDLE state
   cch323_ras_receiver: received msg of Type RCF_CHOSEN
   cch323_run_ras_sm: changing to CCH323_RAS_STATE_RRQ state
   cch323_ras_receiver: received msg of Type RCF_CHOSEN
   cch323_run_ras_sm: received event CCH323_RAS_EVENT_RCF while at CCH323_RAS_STATE_RRQ state
   cch323_run_ras_sm: changing to CCH323_RAS_STATE_IDLE state
20:58:59:cch323_percall_ras_sm: received event CCH323_RAS_EVENT_NEWCALL while at CCH323_RAS_STATE_IDLE state
20:58:59:cch323_percall_ras_sm: changing to new state CCH323_RAS_STATE_ARQ
   cch323_ras_receiver: received msg of Type ACF_CHOSEN
20:58:59:cch323_percall_ras_sm: received event CCH323_RAS_EVENT_ACF while at CCH323_RAS_STATE_ARQ state
20:58:59:cch323_percall_ras_sm: changing to new state CCH323_RAS_STATE_ACTIVE
20:59:02:cch323_percall_ras_sm: received event CCH323_RAS_EVENT_CALLDISC while at CCH323_RAS_STATE_ACTIVE state
20:59:02:cch323_percall_ras_sm: changing to new state CCH323_RAS_STATE_DRQ
   cch323_ras_receiver: received msg of Type DCF_CHOSEN
20:59:02:cch323_percall_ras_sm: received event CCH323_RAS_EVENT_DCF while at CCH323_RAS_STATE_DRQ state
20:59:02:cch323_percall_ras_sm: changing to new state CCH323_RAS_STATE_IDLE
20:59:04:cch323_percall_ras_sm: received event CCH323_RAS_EVENT_I RR while at CCH323_RAS_STATE_ACTIVE state
20:59:04:cch323_percall_ras_sm: changing to new state CCH323_RAS_STATE_ACTIVE
```

The table below describes the significant fields shown in the display.

### Table 46: `debug cch323 ras` Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCH323_RAS_EVENT_SEND_RRQ</td>
<td>Send RRQ event message.</td>
</tr>
<tr>
<td>CCH323_RAS_STATE_IDLE</td>
<td>The global state machine is in the idle state.</td>
</tr>
<tr>
<td>CCH323_RAS_STATE_RRQ</td>
<td>The state machine is in the RRQ state. In this state, the gateway is registering with a gatekeeper.</td>
</tr>
<tr>
<td>RCF_CHOSEN</td>
<td>A registration confirm message that has been received from a gatekeeper.</td>
</tr>
<tr>
<td>CCH323_RAS_EVENT_RCF</td>
<td>Received RCF event message.</td>
</tr>
<tr>
<td>CCH323_RAS_EVENT_NEWCALL</td>
<td>New call event.</td>
</tr>
<tr>
<td>CCH323_RAS_STATE_ARQ</td>
<td>The per-call state machine is in the process of admitting a new call.</td>
</tr>
<tr>
<td>ACF_CHOSEN</td>
<td>ACF message that has been received from a gatekeeper.</td>
</tr>
<tr>
<td>CCH323_RAS_EVENT_ACF</td>
<td>Received ACF event message.</td>
</tr>
</tbody>
</table>
The per-call state machine is in the call active state.

**CCH323_RAS_STATE_ACTIVE**
The per-call state machine is in the process of disengaging an active call.

**CCH323_RAS_STATE_DRQ**
The disengage confirm message that has been received from a gatekeeper.

**DCF_CHOSEN**
Received DCF event message.

**CCH323_RAS_EVENT_DCF**
Send IRR event message.

**CCH323_RAS_EVENT_IRR**

### Examples

The following is sample output from a typical `debug cch323 rawmsg` request on a Cisco 3640 router:

```plaintext
Router# debug cch323 rawmsg
00:32:04:cch323_h225_progress_ind: raw message is 4 bytes: 1E 02 81 88
00:32:22:cch323_h225_release_ind: raw message is 80 bytes: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00:32:22:cch323_h225_release_notify: raw message is 80 bytes: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00:32:22:cch323_h225_release_notify: raw message is 80 bytes: 00 00 00 00 00 00 00 00 00 00
```

### Examples

Following are two examples of output using the `debug cch323 session` command and keyword combination. The first example is for a call setup on an originating gateway. The second example is for a call setup on a terminating gateway.

The following is sample output from a typical `debug cch323 session` request for a call setup on an originating gateway:

```plaintext
Router# debug cch323 session
00:33:49:cch323_call_setup: gw_id=1, callID=16
00:33:49:cch323_get_new_ccb: ccb (0x81D12D2C) is in use
00:33:49:cch323_call_setup: inserted ccb
00:33:49:cch323_get_peer_info: preffered_codec set to G729IETF with Bytes = 20
00:33:49:cch323_get_peer_info: peer: 81FC0D14, peer->voice_peer_tag: 12D, ccb: 81D12D2C
00:33:49: Call_setup Playout Mode: 0, Init 60, Min 40, Max 200
00:33:49: No account/pin number available
00:33:49: timer (0x81D130D4) starts - delay (15000)
00:33:49: generic_open_logical_channel: codec is g729
00:33:49: Filling in qosCapability field to 0
00:33:49: timer (0x81D130D4) starts - delay (15000)
00:33:49: timer (0x81D130D4) stops
00:33:49: Near-end Pref Codecs = G.729 IETF
00:33:49: generic_open_logical_channel: codec is g729
00:33:49: timer (0x81D130D4) starts - delay (15000)
00:33:49: timer (0x81D130D4) stops
00:33:49: timer (0x81D130D4) starts - delay (15000)
00:33:49: timer (0x81D130D4) stops
```

Cisco IOS Debug Command Reference - Commands A through D
The following is sample output from a typical `debug cch323` session request for a call setup on a terminating gateway:

```
00:33:49:cch323_gw_process_read_socket:received msg for H.225
00:33:49:timer (0x81D130D4) starts - delay (180000)
00:33:49:Codec:loc(16), rem(16), Bytes:loc(20), Fwd(20), Rev(20)
00:33:49:cch323_rtp_open_notify:
00:33:50:cch323_ct_main:SOCK 1 Event 0x1
00:33:50:[1]towner_data=0x81D13C88, len=71, msgPtr=0x81F1F2E0
00:33:50:cch323_gw_process_read_socket:received msg for H.225
00:33:50:cch323_caps_ind:cap_modem_proto:0, cap_modem_codec:0, cap_modem_redundancy:0 payload 100
00:33:50:cch323_caps_ind:Load DSP with Negotiated codec(16) g729r8, Bytes=20
00:33:50:cch323_caps_ind:set DSP for dtmf-relay = CC_CAP_DTMF_RELAY_INBAND_VOICE
```

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear h323 gateway</td>
<td>Clears the H.323 gateway counters.</td>
</tr>
<tr>
<td>debug h323-annexg</td>
<td>Displays all pertinent AnnexG messages that have been transmitted and received.</td>
</tr>
<tr>
<td>debug voip rawmsg</td>
<td>Displays the raw message owner, length, and pointer.</td>
</tr>
<tr>
<td>show h323 gateway</td>
<td>Displays statistics for H.323 gateway messages that have been sent and received and displays the reasons for which H.323 calls have been disconnected.</td>
</tr>
</tbody>
</table>
**debug cch323 capacity**

To track the call capacity of the gatekeeper, use the `debug cch323 capacity` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```plaintext
debug cch323 capacity
no debug cch323 capacity
```

**Syntax Description**

This command has no keywords or arguments.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(11)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use the `debug cch323 capacity` command to track the maximum and current call capacity values in the Registration, Admission, and Status (RAS) Protocol messages and to debug capacity-related problems while sending RAS messages. This command is entered on the gateway to monitor the call capacity of the gatekeeper.

The command lists the values for current and maximum call capacity provided by the trunk group capacity resource manager if and when the H.323 Service Provider Interface (SPI) requests the information for all or specific groups of circuits.

**Examples**

The following is sample output from the `debug cch323 capacity` command:

```
Router# debug cch323 capacity
Call Capacity Information tracing is enabled
5d00h: cch323_process_carrier_update: Registered = 1, Event = 1, Reason = 1
5d00h: cch323_process_carrier_update: CarrierId = CARRIERA_NEWENGLAND
5d00h: cch323_fill_crm_CallCapacities: Reason = 1, GroupID = CARRIERA_NEWENGLAND
5d00h: Capacity Details: Maximum Channels in Group: 23
    Active Voice Calls(In): 5, Active Voice Calls(Out): 7
The gatekeeper displays this output when trunk groups are added, deleted, or modified or when circuits in a trunk group are deactivated or activated (similar to ISDN layer 2 down/up).

5d00h: cch323_process_carrier_update: Registered = 1, Event = 1, Reason = 1
5d00h: cch323_process_carrier_update: CarrierId = CARRIERA_NEWENGLAND
```

The table below describes the significant fields shown in the display.
Table 47: debug cch323 capacity Update Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registered</td>
<td>Gateway registration:</td>
</tr>
<tr>
<td></td>
<td>• 0=Gateway is not registered to the gatekeeper</td>
</tr>
<tr>
<td></td>
<td>• 1=Gateway is registered to the gatekeeper at the time of the change</td>
</tr>
<tr>
<td>Event</td>
<td>Carriers updated:</td>
</tr>
<tr>
<td></td>
<td>• 0=All carriers updated</td>
</tr>
<tr>
<td></td>
<td>• 1=Single carrier updated</td>
</tr>
<tr>
<td>Reason</td>
<td>Reason for the update notification:</td>
</tr>
<tr>
<td></td>
<td>• 0=CURRENT_CAPACITY_UPDATE</td>
</tr>
<tr>
<td></td>
<td>• 1=MAX_CAPACITY_UPDATE</td>
</tr>
<tr>
<td></td>
<td>• 2=BOTH_CAPACITY_UPDATE</td>
</tr>
<tr>
<td>CarrierID</td>
<td>ID of the trunk group or carrier to which the change applies.</td>
</tr>
</tbody>
</table>

The gatekeeper displays this output whenever call capacity information is sent to the gatekeeper.

5d00h: cch323_fillcrm_CallCapacities: Reason = 1, GroupID = CARRIERA_NEWENGLAND
5d00h: Capacity Details: Maximum Channels in Group: 23
Active Voice Calls(In): 5, Active Voice Calls(Out): 7

The table below describes the significant fields shown in the display.

Table 48: debug cch323 capacity Call Capacity Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GroupID</td>
<td>The circuit’s carrier identification (ID) or trunk group label.</td>
</tr>
<tr>
<td>Maximum Channels in Group</td>
<td>Maximum number of physical (or configured) circuits.</td>
</tr>
<tr>
<td>Max. Voice Calls(In)</td>
<td>Maximum number of allowed incoming voice and data calls.</td>
</tr>
<tr>
<td>Max. Voice Calls(Out)</td>
<td>Maximum number of allowed outgoing voice and data calls.</td>
</tr>
</tbody>
</table>
### Field | Description
--- | ---
Active Voice Calls(In) | Current number of active incoming voice and data calls.
Active Voice Calls(Out) | Current number of active outgoing voice and data calls.
Max. Voice Calls(to GK) | Maximum call capacity value to be sent to the gatekeeper in the RAS message.
Avail. Voice Calls(to GK) | Available call capacity value to be sent to the gatekeeper in the RAS message.

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>endpoint circuit-id h323id</td>
<td>Associates a carrier with a non-Cisco endpoint.</td>
</tr>
</tbody>
</table>
**debug cch323 h225**

To provide the trace of the state transition of the H.225 state machine based on the processed events, use the `debug cch323 h225` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cch323 h225
no debug cch323 h225
```

**Syntax Description**

This command has no keywords or arguments.

**Command Default**

Disabled

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(6)NA2</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(11)T.</td>
</tr>
</tbody>
</table>

**State Descriptions**

The state definitions of the different states of the H.225 state machine are as follows:

- **H225_IDLE**--This is the initial state of the H.225 state machine. The H.225 state machine is in this state before issuing a call setup request (for the outbound IP call case) or ready to receive an incoming IP call.
- **H225_SETUP**--This is the call setup state. The state machine transitions to this state after sending out a call setup request, or after the reception of an incoming call indication.
- **H225_ALERT**--This is the call alerting state. The state machine transitions to this state after sending the alerting message or after the reception of an alerting message from the peer.
- **H225_CALLPROC**--This is the call proceeding state.
- **H225_ACTIVE**--This is the Call connected state. In this state, the call is active. The state machine transitions to this state after sending the connect message to the peer or after the reception of the connect message from the peer.
- **H225_WAIT_FOR_ARQ**--This is the state where the H.225 state machine is waiting for the completion of the ARQ process from the Registration, Admission, and Status Protocol (RAS) state machine.
• H225_WAIT_FOR_DRQ--This is the state where the H.225 state machine is waiting for the completion of the DRQ process from the RAS state machine.

• H225_WAIT_FOR_H245--This is the state where the H.225 state machine is waiting for the success or failure from the H.245 state machine.

**Events Description**

The event definitions of the different events of the H.225 state machine are as follows:

• H225_EVENT_NONE--No event.

• H225_EVENT_ALERT--This event indicates the H.225 state machine to send an alerting message to the peer.

• H225_EVENT_ALERT_IND--This event indicates the H.225 state machine that an alerting message is received from the peer.

• H225_EVENT_CALLPROC--This event indicates the H.225 state machine to send a call proceeding message to the peer.

• H225_EVENT_CALLPROC_IND--This event indicates the H.225 state machine that a call proceeding message is received from the peer.

• H225_EVENT_REJECT--This event indicates the H.225 state machine to reject the call setup request from the peer.

• H225_EVENT_REJECT_IND--This event indicates the H.225 state machine that a call setup request to the peer is rejected.

• H225_EVENT_RELEASE--This event indicates the H.225 state machine to send a release complete message to the peer.

• H225_EVENT_RELEASE_IND--This event indicates the H.225 state machine that a release complete message is received from the peer.

• H225_EVENT_SETUP--This event indicates the H.225 state machine to send a setup message to the peer.

• H225_EVENT_SETUP_IND--This event indicates the H.225 state machine that a setup message is received from the peer.

• H225_EVENT_SETUP_CFM--This event indicates the H.225 state machine to send a connect message to the peer.

• H225_EVENT_SETUP_CFM_IND--This event indicates the H.225 state machine that a connect message from the peer.

• H225_EVENT_RAS_SUCCESS--This event indicates the H.225 state machine that the pending RAS operation is successful.

• H225_EVENT_RAS_FAILED--This event indicates the H.225 state machine that the pending RAS operation failed.

• H225_EVENT_H245_SUCCESS--This event indicates the H.225 state machine that the pending H.245 operation is successful.

• H225_EVENT_H245_FAILED--This event indicates the H.225 state machine that the pending H.245 operation failed.
The following is sample output from the `debug cch323 h225` command:

```
Router# debug cch323 h225
20:59:17: Set new event H225_EVENT_SETUP
20:59:17: H225 FSM: received event H225_EVENT_SETUP while at state H225_IDLE
20:59:17: Changing from H225_IDLE state to H225_SETUP state
20:59:17: cch323_h225_receiver: received msg of type SETUPCFM_CHOSEN
20:59:17: H225 FSM: received event H225_EVENT_SETUP_CFM_IND while at state H225_SETUP
20:59:17: Changing from H225_SETUP state to H225_ACTIVE state
20:59:17: Set new event H225_EVENT_H245_SUCCESS
20:59:17: H225 FSM: received event H225_EVENT_H245_SUCCESS while at state H225_ACTIVE
20:59:20: Set new event H225_EVENT_RELEASE
20:59:20: H225 FSM: received event H225_EVENT_RELEASE while at state H225_ACTIVE
20:59:20: Changing from H225_ACTIVE state to H225_WAIT_FOR_DRQ state
20:59:20: Set new event H225_EVENT_RAS_SUCCESS
20:59:20: H225 FSM: received event H225_EVENT_RAS_SUCCESS while at state H225_WAIT_FOR_DRQ
20:59:20: Changing from H225_WAIT_FOR_DRQ state to H225_IDLE state
```
debug cch323 h245

To provide the trace of the state transition of the H.245 state machine based on the processed events, use the debug cch323 h245 command in privileged EXEC mode. To disable debugging output, use the no form of this command.

debug cch323 h245
no debug cch323 h245

Syntax Description
This command has no arguments or keywords.

Command Default
Disabled

Command Modes
Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(6)NA2</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(11)T.</td>
</tr>
</tbody>
</table>

Usage Guidelines
The H.245 state machines include the following three state machines:

- Master Slave Determination (MSD) state machine
- Capability Exchange (CAP) state machine
- Open Logical Channel (OLC) state machine

State Definitions
The definitions are as follows:

- **H245_MS_NONE**-- This is the initial state of the master slave determination state machine.
- **H245_MS_WAIT**--In this state, a Master Slave Determination message is sent, waiting for the reply.
- **H245_MS_DONE**-- The result is in.
- **H245_CAP_NONE**--This is the initial state of the capabilities exchange state machine.
- **H245_CAP_WAIT**--In this state, a cap exchange message is sent, waiting for reply.
- **H245_CAP_DONE**--The result is in.
- **H245_OLC_NONE**--This is the initial state of the open logical channel state machine.
- H245_OLC_WAIT: OLC message sent, waiting for reply.
- H245_OLC_DONE: OLC done.

Event definitions
- H245_EVENT_MSD--Send MSD message
- H245_EVENT_MS_CFM--Send MSD acknowledge message
- H245_EVENT_MS_REJ--Send MSD reject message
- H245_EVENT_MS_IND--Received MSD message
- H245_EVENT_CAP--Send CAP message
- H245_EVENT_CAP_CFM--Send CAP acknowledge message
- H245_EVENT_CAP_REJ--Send CAP reject
- H245_EVENT_CAP_IND--Received CAP message
- H245_EVENT_OLC--Send OLC message
- H245_EVENT_OLC_CFM--Send OLC acknowledge message
- H245_EVENT_OLC_REJ--Send OLC reject message
- H245_EVENT_OLC_IND--Received OLC message

Examples
The following is sample output from the debugcch323h245 command:

Router# debug cch323 h245
20:58:23: Changing to new event H245_EVENT_MSD
20:58:23: H245 MS FSM: received event H245_EVENT_MSD while at state H245_MS_NONE
20:58:23: changing from H245_MS_NONE state to H245_MS_WAIT state
20:58:23: Changing to new event H245_EVENT_CAP
20:58:23: H245 CAP FSM: received event H245_EVENT_CAP while at state H245_CAP_NONE
20:58:23: changing from H245_CAP_NONE state to H245_CAP_WAIT state
20:58:23: cch323_h245_receiver: received msg of type M_H245_MS_DETERMINE_INDICATION
20:58:23: Changing to new event H245_EVENT_MS_IND
20:58:23: H245 MS FSM: received event H245_EVENT_MS_IND while at state H245_MS_WAIT
20:58:23: cch323_h245_receiver: received msg of type M_H245_CAP TRANSFER INDICATION
20:58:23: changing to new event H245_EVENT_CAP_IND
20:58:23: H245 CAP FSM: received event H245_EVENT_CAP_IND while at state H245_CAP_WAIT
20:58:23: cch323_h245_receiver: received msg of type M_H245_MS_DETERMINE_CONFIRM
20:58:23: Changing to new event H245_EVENT_MS_CFM
20:58:23: H245 MS FSM: received event H245_EVENT_MS_CFM while at state H245_MS_WAIT
20:58:23: changing from H245_MS_WAIT state to H245_MS_DONE state
20:58:23: cch323_h245_receiver: received msg of type M_H245_CAP TRANSFER_CONFIRM
20:58:23: Changing to new event H245_EVENT_CAP_CFM
20:58:23: H245 CAP FSM: received event H245_EVENT_CAP_CFM while at state H245_CAP_WAIT
20:58:23: changing from H245_CAP_WAIT state to H245_CAP_DONE state
20:58:23: H245 OLC FSM: received event H245_EVENT_OLC while at state H245 OLC NONE
20:58:23: changing from H245_OLC_NONE state to H245_OLC_WAIT state
20:58:23:cch323_h245_receiver: received msg of type M_H245_UCHAN_ESTABLISH_INDICATION
20:58:23:Changing to new event H245_EVENT_OLC_IND
20:58:23:H245 OLC FSM: received event H245_EVENT_OLC_IND while at state H245_OLC_WAIT
20:58:23:cch323_h245_receiver: received msg of type M_H245_UCHAN_ESTAB_ACK
20:58:23:Changing to new event H245_EVENT_OLC_CFM
20:58:23:H245 OLC FSM: received event H245_EVENT_OLC_CFM while at state H245_OLC_WAIT
20:58:23:changing from H245_OLC_WAIT state to H245_OLC_DONE state
debug cch323 preauth

To enable diagnostic reporting of authentication, authorization, and accounting (AAA) call preauthentication for H.323 calls, use the debug cch323 preauth command in privileged EXEC mode. To disable debugging output, use the no form of this command.

depth cch323 preauth
no debug cch323 preauth

Syntax Description
This command has no arguments or keywords.

Command Default
No default behavior or values

Command Modes
Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(11)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

Examples
The following is debugging output for a single H.323 call:

```
Router# debug cch323 preauth
CCH323 preauth tracing is enabled
Jan 23 18:39:56.393: In cch323_send_preauth_req for preauth_id = -1
Jan 23 18:39:56.393: Entering rpms_proc_print_preauth_req
Jan 23 18:39:56.393: Request = 0
Jan 23 18:39:56.393: Preauth_id = 86514
Jan 23 18:39:56.393: EndPt Type = 1
Jan 23 18:39:56.393: EndPt = 192.168.81.102
Jan 23 18:39:56.393: Resource Service = 1
Jan 23 18:39:56.393: Call_origin = answer
Jan 23 18:39:56.393: Call_type = voip
Jan 23 18:39:56.393: Calling_num = 2230001
Jan 23 18:39:56.393: Called_num = 1#1130001
Jan 23 18:39:56.393: Protocol = 0
Jan 23 18:39:56.393: cch323_insert_preauth_tree:Created node with preauth_id = 86514 , ccb 6852D5BC , node 651F87FC
Jan 23 18:39:56.393: rpms_proc_create_node:Created node with preauth_id = 86514
Jan 23 18:39:56.393: rpms_proc_send_aaa_req:uid got is 466725
Jan 23 18:39:56.397: rpms_proc_preauth_response:Context is for preauth_id 86514, aaa_uid 466725
Jan 23 18:39:56.397: cch323_rpms_proc_callback_func:PREAUTH_SUCCESS for preauth id 86514, aaa_uid 466725
Jan 23 18:39:56.397: Deleting Tree node for preauth id 86514 uid 466725
Jan 23 18:39:56.397: cch323_get_ccb_and_delete_from_preauth_tree:Preauth id 86514 ccb 6852D5BC
```

The table below describes the significant fields shown in the display.
Table 49: debug cch323 preauth Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Request Type--0 for preauthentication, 1 for disconnect.</td>
</tr>
<tr>
<td>Preauth id</td>
<td>Identifier for the preauthentication request.</td>
</tr>
<tr>
<td>EndPt Type</td>
<td>Call Origin End Point Type--1 for IP address, 2 for IZCT value.</td>
</tr>
<tr>
<td>EndPt</td>
<td>Call Origin End Point Value--An IP address or IZCT value.</td>
</tr>
<tr>
<td>Resource Service</td>
<td>Resource Service Type--1 for Reservation, 2 for Query.</td>
</tr>
<tr>
<td>Call_origin</td>
<td>Answer.</td>
</tr>
<tr>
<td>Call_type</td>
<td>Voice over IP (VoIP).</td>
</tr>
<tr>
<td>Calling_num</td>
<td>Calling Party Number (CLID).</td>
</tr>
<tr>
<td>Called_num</td>
<td>Called Party Number (DNIS).</td>
</tr>
<tr>
<td>Protocol</td>
<td>0 for H.323, 1 for SIP.</td>
</tr>
<tr>
<td>function reports</td>
<td>Various identifiers and status reports for executed functions.</td>
</tr>
</tbody>
</table>
**debug cch323 ras**

To provide the trace of the state transition of the Registration, Admission, and Status (RAS) Protocol state machine based on the processed events, use the `debug cch323 ras` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cch323 ras
no debug cch323 ras
```

**Syntax Description**

This command has no keywords or arguments.

**Command Default**

Disabled

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(6)NA2</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(11)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

RAS operates in two state machines. One global state machine controls the overall RAS operation of the Gateway. The other state machine is a per call state machine that controls the active calls.

**State definitions**

The state definitions of the different states of the RAS state machine follow:

- **CCH323_RAS_STATE_NONE**--This is the initial state of the RAS state machine.
- **CCH323_RAS_STATE_GRQ**--The state machine is in the Gatekeeper Request (GRQ) state. In this state, the gateway is in the process of discovering a gatekeeper.
- **CCH323_RAS_STATE_RRQ**--The state machine is in the Registration Request (RRQ) state. In this state, the gateway is in the process of registering with a gatekeeper.
- **CCH323_RAS_STATE_IDLE**--The global state machine is in the idle state.
- **CCH323_RAS_STATE_URQ**--The state machine is in the Unregistration Request (URQ) state. In this state, the gateway is in the process of unregistering with a gatekeeper.
- **CCH323_RAS_STATE_ARQ**--The per call state machine is in the process of admitting a new call.
- **CCH323_RAS_STATE_ACTIVE**--The per call state machine is in the call active state.
• CCH323_RAS_STATE_DRQ--The per call state machine is in the process of disengaging an active call.

**Event Definitions**

These are the event definitions of the different states of the RAS state machine:

• CCH323_RAS_EVENT_NONE--Nothing.
• CCH323_RAS_EVENT_GWUP--Gateway is coming up.
• CCH323_RAS_EVENT_GWDWN--Gateway is going down.
• CCH323_RAS_EVENT_NEWCALL--New call.
• CCH323_RAS_EVENT_CALLDISC--Call disconnect.
• CCH323_RAS_EVENT_GCF--Received Gatekeeper Confirmation (GCF).
• CCH323_RAS_EVENT_GRJ--Received Gatekeeper Rejection (GRJ).
• CCH323_RAS_EVENT_ACF--Received Admission Confirmation (ACF).
• CCH323_RAS_EVENT_ARJ--Received Admission Rejection (ARJ).
• CCH323_RAS_EVENT_SEND_RRQ--Send Registration Request (RRQ).
• CCH323_RAS_EVENT_RCF--Received Registration Confirmation (RCF).
• CCH323_RAS_EVENT_RRJ--Received Registration Rejection (RRJ).
• CCH323_RAS_EVENT_SEND_URQ--Send URQ.
• CCH323_RAS_EVENT_UUCF--Received Unregister Confirmation (UCF).
• CCH323_RAS_EVENT_SEND_UCF--Send Unregister Confirmation (UCF).
• CCH323_RAS_EVENT_URJ--Received Unregister Reject (URJ).
• CCH323_RAS_EVENT_BCF--Received Bandwidth Confirm (BCF).
• CCH323_RAS_EVENT_BRJ--Received Bandwidth Rejection (BRJ).
• CCH323_RAS_EVENT_DRQ--Received Disengage Request (DRQ).
• CCH323_RAS_EVENT_DCF--Received Disengage Confirm (DCF).
• CCH323_RAS_EVENT_SEND_DCF--Send Disengage Confirm (DCF).
• CCH323_RAS_EVENT_DRJ--Received Disengage Reject (DRJ).
• CCH323_RAS_EVENT_IRQ--Received Interrupt Request (IRQ).
• CCH323_RAS_EVENT_IRR--Send Information Request (IRR).
• CCH323_RAS_EVENT_TIMEOUT--Message timeout.

**Examples**

The following is sample output from the **debug cch323 preauth** command:

```
Router# debug cch323 preauth
20:58:49:Changing to new event CCH323_RAS_EVENT_SEND_RRQ
```
cch323_run_ras_sm: received event CCH323_RAS_EVENT_SEND_RRQ while at CCH323_RAS_STATE_IDLE state
cch323_run_ras_sm: changing to CCH323_RAS_STATE_RRQ state
cch323_ras_receiver: received msg of Type RCF_CHOSEN
20:58:59: cch323_perccall_ras_sm: received event CCH323_RAS_EVENT_RCF while at CCH323_RAS_STATE_RRQ state
cch323_ras_receiver: received msg of Type RCF_CHOSEN
cch323_run_ras_sm: changing to CCH323_RAS_STATE_IDLE state
20:58:59: cch323_perccall_ras_sm: received event CCH323_RAS_EVENT_NEWCALL while at CCH323_RAS_STATE_IDLE state
cch323_ras_receiver: received msg of type ACF_CHOSEN
20:58:59: cch323_perccall_ras_sm: changing to new state CCH323_RAS_STATE_ARQ
20:58:59: cch323_perccall_ras_sm: received event CCH323_RAS_EVENT_ACF while at CCH323_RAS_STATE_ARQ state
20:58:59: cch323_perccall_ras_sm: changing to new state CCH323_RAS_STATE_ACTIVE
20:59:02: cch323_perccall_ras_sm: received event CCH323_RAS_EVENT_CALLDISC while at CCH323_RAS_STATE_ACTIVE state
20:59:02: cch323_perccall_ras_sm: changing to new state CCH323_RAS_STATE_DRQ
20:59:02: cch323_perccall_ras_sm: received msg of type DCF_CHOSEN
20:59:02: cch323_perccall_ras_sm: changing to new state CCH323_RAS_STATE_DRQ state
20:59:02: cch323_perccall_ras_sm: changing to new state CCH323_RAS_STATE_IDdle
20:59:04: cch323_perccall_ras_sm: received event CCH323_RAS_EVENT_IRR while at CCH323_RAS_STATE_ACTIVE state
debug cch323 video

To provide debugging output for video components within the H.323 subsystem, use the `debug cch323 video` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cch323 video
no debug cch323 video
```

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

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Cisco IOS Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(4)XC</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(9)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(9)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
Use this command to enable a debugging trace for the video component in an H.323 network.

**Examples**
The following is sample output of the debugging log for an originating Cisco Unified CallManager Express (Cisco Unified CME) gateway after the `debug cch323 video` command was enabled:

```
Router# show log
Syslog logging: enabled (11 messages dropped, 487 messages rate-limited, 0 flushes, 0 overruns, xml disabled, filtering disabled)
  Console logging: disabled
  Monitor logging: level debugging, 0 messages logged, xml disabled, filtering disabled
  Buffer logging: level debugging, 1144 messages logged, xml disabled, filtering disabled
  Logging Exception size (4096 bytes)
  Count and timestamp logging messages: disabled
  Trap logging: level informational, 1084 message lines logged
Log Buffer (6000000 bytes):
  Jun 13 09:19:42.006: //103030/C7838B198002/H323/cch323_get_peer_info: Entry
  Jun 13 09:19:42.006: //103030/C7838B198002/H323/cch323_get_peer_info: Have peer
  Jun 13 09:19:42.006: //103030/C7838B198002/H323/cch323_set_pref_codec_list: First preferred codec(bytes)=16(20)
  Jun 13 09:19:42.006: //103030/C7838B198002/H323/cch323_set_h323_control_options_outgoing: h245 sm mode = 8463
```
Examples

The following is sample output of the debugging log for a terminating Cisco Unified Survivable Remote Site Telephony (Cisco Unified SRST) gateway after the `debug cch323 video` command was enabled:

```
Router# show log
Syslog logging: enabled (11 messages dropped, 466 messages rate-limited, 0 flushes, 0 overruns, xml disabled, filtering disabled)  
Console logging: disabled  
Monitor logging: level debugging, 0 messages logged, xml disabled, filtering disabled  
Buffer logging: level debugging, 829 messages logged, xml disabled, filtering disabled  
Logging Exception size (4096 bytes)  
Count and timestamp logging messages: disabled  
Trap logging: level informational, 771 message lines logged  
Log Buffer (200000 bytes):  
Jun 13 09:19:42.011: //103034/C7838B198002/H323/setup_ind: Receive bearer cap info XRate 24, rateMult 12  
Jun 13 09:19:42.011: //103034/C7838B198002/H323/cch323_set_h245_state_mc_mode_incoming: h245 state m/c mode=0x10F, h323_ctl=0x2F  
Jun 13 09:19:42.015: //1-1/xxxxxxxxxxxx/H323/cch245_event_handler: callID=103034  
Jun 13 09:19:42.019: //1-1/xxxxxxxxxxxx/H323/cch323_set_mode: callID=103034, flow Mode=1, spi_mode=0x6  
Jun 13 09:19:42.019: //103034/C7838B198002/H323/cch323_do_call_proceeding: set_mode NOT called yet...saved deferred CALL_PROC  
Jun 13 09:19:42.019: //103034/C7838B198002/H323/cch323_h245_connection_sm: state=0, event=0, ccb=4461B518, listen state=0  
Jun 13 09:19:42.019: //103034/C7838B198002/H323/cch323_process_set_mode: Setting inbound leg mode flags to flush=0, 0-mode to FLOW THROUGH  
Jun 13 09:19:42.019: //103034/C7838B198002/H323/cch323_process_set_mode: Sending deferred CALL_PROC  
Jun 13 09:19:42.019: //103034/C7838B198002/H323/cch323_process_set_mode: Setting mode called so we can proceed with CALLPROC  
Jun 13 09:19:42.027: //103034/C7838B198002/H323/cch323_send_cap_request: Setting mode to VIDEO MODE  
Jun 13 09:19:42.031: //103034/C7838B198002/H323/cch323_h245_cap_ind: Masks au=0xC data=0x2 uinp=0x32
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug ephone video</code></td>
<td>Sets video debugging for the Cisco Unified IP phone.</td>
</tr>
<tr>
<td><code>show call active video</code></td>
<td>Displays call information for SCCP video calls in progress.</td>
</tr>
<tr>
<td><code>show call history video</code></td>
<td>Displays call history information for SCCP video calls.</td>
</tr>
<tr>
<td><code>show debugging</code></td>
<td>Displays information about the types of debugging that are enabled for your router.</td>
</tr>
</tbody>
</table>
**debug ccm-manager**

To display debugging information about Cisco CallManager, use the `debug ccm-manager` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```plaintext
debug ccm-manager {backhaul {errors|events|packets} | config-download {all|errors|events|packets|tone|xml|errors|events|music-on-hold {errors|events|packets}} | packets}
no debug ccm-manager
```

### Syntax Description

<table>
<thead>
<tr>
<th>backhaul</th>
<th>Enables debugging of the Cisco CallManager backhaul. The keywords are as follows:</th>
</tr>
</thead>
<tbody>
<tr>
<td>errors</td>
<td>--Displays Cisco CallManager backhaul errors.</td>
</tr>
<tr>
<td>events</td>
<td>--Displays Cisco CallManager backhaul events.</td>
</tr>
<tr>
<td>packets</td>
<td>--Displays Cisco CallManager packets.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>config-download</th>
<th>Enables debugging of the Cisco CallManager configuration download. The keywords are as follows:</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>--Displays all Cisco CallManager configuration parameters.</td>
</tr>
<tr>
<td>errors</td>
<td>--Displays Cisco CallManager configuration errors.</td>
</tr>
<tr>
<td>events</td>
<td>--Displays Cisco CallManager configuration events.</td>
</tr>
<tr>
<td>packets</td>
<td>--Displays Cisco CallManager configuration packets.</td>
</tr>
<tr>
<td>tone</td>
<td>--Displays Cisco CallManager downloaded custom tones.</td>
</tr>
<tr>
<td>xml</td>
<td>--Displays the Cisco CallManager configuration XML parser.</td>
</tr>
</tbody>
</table>

| errors | Displays errors related to Cisco CallManager. |
| events | Displays Cisco CallManager events, such as when the primary Cisco CallManager server fails and control is switched to the backup Cisco CallManager server. |
## music-on-hold

<table>
<thead>
<tr>
<th>music-on-hold</th>
<th>Displays music on hold (MOH). The keywords are as follows:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>errors</strong></td>
<td>-- Displays MOH errors.</td>
</tr>
<tr>
<td>- <strong>events</strong></td>
<td>-- Displays MOH events.</td>
</tr>
<tr>
<td>- <strong>packets</strong></td>
<td>-- Displays MOH packets.</td>
</tr>
</tbody>
</table>

| packets       | Displays Cisco CallManager packets.                        |

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)T</td>
<td>This command was introduced for Cisco CallManager Version 3.0 and the Cisco VG200.</td>
</tr>
</tbody>
</table>

| 12.2(2)XA | This command was implemented on Cisco 2600 series and Cisco 3600 series routers. |

| 12.2(2)XN | Support for enhanced MGCP voice gateway interoperability was added to Cisco CallManager Version 3.1 for the Cisco 2600 series, Cisco 3600 series, and Cisco VG200. |

| 12.2(11)T | This command was integrated into Cisco IOS Release 12.2(11)T and implemented on the Cisco IAD2420 series. |

| 12.2(15)XJ | The **tone** keyword was added for the following platforms: Cisco 2610XM, Cisco 611XM, Cisco 2620XM, Cisco 2621XM, Cisco 2650XM, Cisco 2651XM, Cisco 2691, Cisco 3640A, Cisco 3660, Cisco 3725, and Cisco 3745. |

| 12.3(4)T | The **tone** keyword was added. |

| 12.3(14)T | New output was added relating to the SCCP protocol. |

| 12.2SX | This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware. |

### Examples

The following is sample output from the `debug ccm-manager events` command:

```
Router# debug ccm-manager events
*Feb 28 22:56:05.873: cmapp_mgcapp_go_down: Setting mgc status to NO_RESPONSE
*Feb 28 22:56:05.873: cmapp_host_fsm: New state DOWN for host 0 (172.20.71.38)
*Feb 28 22:56:05.873: cmapp_mgr_process_ev_active_host_failed: Active host 0 (172.20.71.38) failed
*Feb 28 22:56:05.873: cmapp_mgr_check_hostlist: Active host is 0 (172.20.71.38)
```
You can use the `debug ccm-manager config-download tone` command to verify the parameters assigned to each locale. The following sample output shows the locale name United Kingdom and lists all the dual-tone parameters for that region:

```
Router# debug ccm-manager config-download tone
00:09:07: cmapp_prefix_process_tag_tones:
00:09:07: cmaapp_process_tag_trkLocaleName: region = United Kingdom
00:09:07: cmaapp_process_tag_pulse_ratio: pulse ratio = 40
00:09:07: cmaapp_process_tag_dtmf_llevel: low frequency level = 65438
00:09:07: cmaapp_process_tag_dtmf_hlevel: high frequency level = 65463
00:09:07: cmaapp_process_tag_special_oper: operation = uLaw
00:09:07: cmaapp_prefix_process_tag_lpig:
00:09:07: cmaapp_process_tag_fxs: ignore LPIG for fxs
00:09:07: cmaapp_process_tag_fxo: ignore LPIG for fxo
00:09:07: cmaapp_process_tag_digital: ignore LPIG for digital
00:09:07: cmaapp_prefix_process_tag_dxat:
00:09:07: cmaapp_process_tag_fxs: ignore PDOG for fxs
00:09:07: cmaapp_process_tag_fxo: ignore PDOG for fxo
00:09:07: cmaapp_process_tag_digital: ignore PDOG for digital
00:09:07: cmaapp_prefix_process_tag_tonetable_info:
00:09:07: cmaapp_prefix_process_tag_dualtone: TID=[0:CPTONE_BUSY]
00:09:07: cmaapp_process_tag_nf: number of frequencies = 1
00:09:07: cmaapp_process_tag_dr: direction = 0
00:09:07: cmaapp_process_tag_fof: frequency 1 = 400
00:09:07: cmaapp_process_tag_fos: frequency 2 = 0
00:09:07: cmaapp_process_tag_fot: frequency 3 = 0
00:09:07: cmaapp_process_tag_fo4: frequency 4 = 0
00:09:07: cmaapp_prefix_process_tag_asd_level:
00:09:07: cmaapp_process_tag_fxs: amplitude of 1st = -200
00:09:07: cmaapp_process_tag_fxo: amplitude of 1st = -200
00:09:07: cmaapp_process_tag_digital: amplitude of 1st = -240
00:09:07: cmaapp_prefix_process_tag_acs_level:
00:09:07: cmaapp_process_tag_fxs: amplitude of 2nd = 0
00:09:07: cmaapp_process_tag_fxo: amplitude of 2nd = 0
00:09:07: cmaapp_process_tag_digital: amplitude of 2nd = 0
00:09:07: cmaapp_prefix_process_tag_acg_level:
00:09:07: cmaapp_process_tag_fxs: amplitude of 3rd = 0
00:09:07: cmaapp_process_tag_fxo: amplitude of 3rd = 0
00:09:07: cmaapp_process_tag_digital: amplitude of 3rd = 0
00:09:07: cmaapp_prefix_process_tag_aos_level:
00:09:07: cmaapp_process_tag_fxs: amplitude of 4th = 0
00:09:07: cmaapp_process_tag_fxo: amplitude of 4th = 0
00:09:07: cmaapp_process_tag_digital: amplitude of 4th = 0
00:09:07: cmaapp_prefix_process_tag_ontf:
00:09:07: cmaapp_process_tag_fxs: frequency 1 on time = 375
00:09:07: cmaapp_process_tag_fxo: frequency 1 off time = 375
00:09:07: cmaapp_process_tag_digital: frequency 1 on time = 375
00:09:07: cmaapp_process_tag_ofst:
00:09:07: cmaapp_process_tag_fxs: frequency 2 on time = 0
00:09:07: cmaapp_process_tag_fxo: frequency 2 off time = 0
00:09:07: cmaapp_process_tag_digital: frequency 2 on time = 0
00:09:07: cmaapp_process_tag_ontt:
00:09:07: cmaapp_process_tag_fxs: frequency 3 on time = 0
00:09:07: cmaapp_process_tag_fxo: frequency 3 off time = 0
00:09:07: cmaapp_process_tag_digital: frequency 3 on time = 0
00:09:07: cmaapp_process_tag_oftt:
00:09:07: cmaapp_process_tag_fxs: frequency 4 on time = 0
00:09:07: cmaapp_process_tag_fxo: frequency 4 off time = 0
00:09:07: cmaapp_process_tag_digital: frequency 4 on time = 0
```
00:09:07: cmapp_process_tag_fos3: frequency 2 cadence 3 = 0
00:09:07: cmapp_process_tag_fos4: frequency 2 cadence 4 = 0
00:09:07: cmapp_process_tag_rct1: cadence 1 repeat count = 0
00:09:07: cmapp_process_tag_rct2: cadence 2 repeat count = 0
00:09:07: cmapp_process_tag_rct3: cadence 3 repeat count = 0
00:09:07: cmapp_process_tag_rct4: cadence 4 repeat count = 0
00:09:07: cmapp_prefix_process_tag_dualtone: TID=[1:CPTONE_RING_BACK]
00:09:07: cmapp_process_tag_nf: number of frequencies = 2
00:09:07: cmapp_process_tag_dr: direction = 0
00:09:07: cmapp_process_tag_fo1: frequency 1 = 400
00:09:07: cmapp_process_tag_fo2: frequency 2 = 450
00:09:07: cmapp_process_tag_fo4: frequency 4 = 0
00:09:07: cmapp_prefix_process_tag_aof_level:
00:09:07: cmapp_process_tag_fxs: amplitude of 1st = -190
00:09:07: cmapp_process_tag_fxo: amplitude of 1st = -190
00:09:07: cmapp_process_tag_digital: amplitude of 1st = -190
00:09:07: cmapp_prefix_process_tag_aos_level:
00:09:07: cmapp_process_tag_fxs: amplitude of 2nd = -190
00:09:07: cmapp_process_tag_fxo: amplitude of 2nd = -190
00:09:07: cmapp_process_tag_digital: amplitude of 2nd = -190
00:09:07: cmapp_prefix_process_tag_aot_level:
00:09:07: cmapp_process_tag_fxs: amplitude of 3rd = 0
00:09:07: cmapp_process_tag_fxo: amplitude of 3rd = 0
00:09:07: cmapp_process_tag_digital: amplitude of 3rd = 0
00:09:07: cmapp_prefix_process_tag_ao4_level:
00:09:07: cmapp_process_tag_fxs: amplitude of 4th = 0
00:09:07: cmapp_process_tag_fxo: amplitude of 4th = 0
00:09:07: cmapp_process_tag_digital: amplitude of 4th = 0
00:09:07: cmapp_process_tag_ontf: frequency 1 on time = 400
00:09:07: cmapp_process_tag_oftf: frequency 1 off time = 200
00:09:07: cmapp_process_tag_onts: frequency 2 on time = 400
00:09:07: cmapp_process_tag_ofts: frequency 2 off time = 2000
00:09:07: cmapp_process_tag_ontt: frequency 3 on time = 0
00:09:07: cmapp_process_tag_oftt: frequency 3 off time = 0
00:09:07: cmapp_process_tag_ont4: frequency 4 on time = 0
00:09:07: cmapp_process_tag_oft4: frequency 4 off time = 0
00:09:07: cmapp_process_tag_ontf2: frequency 1 cadence 2 = 0
00:09:07: cmapp_process_tag_onts2: frequency 2 cadence 2 = 0
00:09:07: cmapp_process_tag_ontt2: frequency 3 cadence 2 = 0
00:09:07: cmapp_process_tag_ont42: frequency 4 cadence 2 = 0
00:09:07: cmapp_prefix_process_tag_dualtone: TID=[2:CPTONE_CONGESTION]
00:09:07: cmapp_process_tag_nf: number of frequencies = 1
00:09:07: cmapp_process_tag_dr: direction = 0
00:09:07: cmapp_process_tag_fo1: frequency 1 = 400
00:09:07: cmapp_process_tag_fo2: frequency 2 = 0
00:09:07: cmapp_process_tag_fo4: frequency 4 = 0
00:09:07: cmapp_prefix_process_tag_aof_level:
00:09:07: cmapp_process_tag_fxs: amplitude of 1st = -200
00:09:07: cmapp_process_tag_fxo: amplitude of 1st = -200
00:09:07: cmapp_process_tag_digital: amplitude of 1st = -200
00:09:07: cmapp_prefix_process_tag_aos_level:
00:09:07: cmapp_process_tag_fxs: amplitude of 2nd = 0
00:09:07: cmapp_process_tag_fxo: amplitude of 2nd = 0
00:09:07: cmapp_process_tag_digital: amplitude of 2nd = 0
00:09:07: cmapp_prefix_process_tag_aot_level:
00:09:07: cmapp_process_tag_fxs: amplitude of 3rd = 0
00:09:07: cmapp_process_tag_fxo: amplitude of 3rd = 0
00:09:07: cmapp_process_tag_digital: amplitude of 3rd = 0
00:09:07: cmapp_prefix_process_tag_ao4_level:
00:09:07: cmapp_process_tag_fxs: amplitude of 4th = 0
00:09:07: cmapp_process_tag_fxo: amplitude of 4th = 0
00:09:07: cmapp_process_tag_digital: amplitude of 4th = 0
debug ccm-manager

The following is sample output from the `debugccm-managerconfig-downloadall` command for an error case in which the configuration file cannot be accessed for a Skinny Client Control Protocol (SCCP) download:

*Jan  9 07:28:33.499: cmapp_xml_process_timer:
*Jan  9 07:28:33.499: cmapp_xml_find_ep_by_name: Checking for ep_name [*]
*Jan  9 07:28:33.499: cmapp_xml_exec_fsm: Endpoint is [*]

The following is sample output from the `debugccm-managerconfig-downloadall` command for a successful SCCP download:

*Jan  9 09:44:45.543: cmapp_sccp_config:
*Jan  9 09:44:45.543: cmapp_sccp_reset_curcfg:
*Jan  9 09:44:45.543: cmapp_sccp_init_curcfg:
*Jan  9 09:44:45.543: cmapp_sccp_download_gw_config_file:
*Jan  9 09:44:45.543: cmapp_sccp_get_gw_name:
*Jan  9 09:44:45.543: cmapp_sccp_get_gw_name: XML file name generated->SKIGW0C85226910.cnf.xml
*Jan  9 09:44:45.543: cmapp_sccp_get_xml_file_via_tftp:
*Jan  9 09:44:45.543: cmapp_sccp_tftp_download_file:
*Jan  9 09:44:45.543: cmapp_sccp_tftp_get_file_size:
*Jan  9 09:44:45.563: cmapp_sccp_get_buffer:
*Jan  9 09:44:45.575: cmapp_sccp_tftp_download_file: File tftp://10.2.6.101/SKIGW0C85226910.cnf.xml read 8162 bytes
*Jan  9 09:44:45.575: cmapp_sccp_get_xml_file via tftp: Read file tftp://10.2.6.101/SKIGW0C85226910.cnf.xml, len = 8162
*Jan  9 09:44:45.575: cmapp_parse_gw_xml_file:
*Jan  9 09:44:45.579: cmapp_sccp_gw_chardata_handler: ccm found, priority=0

The following lines show the conversion of XML data into router configuration information for the endpoint:

*Jan  9 09:44:45.579: cmapp_sccp_gw_start_element_handler: Unit has been set to 1
*Jan  9 09:44:45.579: cmapp_sccp_gw_start_element_handler: Subunit has been set to 0
*Jan  9 09:44:45.579: cmapp_sccp_gw_start_element_handler: Endpoint has been set to 0
*Jan  9 09:44:45.579: cmapp_sccp_gw_start_element_handler: Endpoint has been set to 1
*Jan  9 09:44:45.579: cmapp_sccp_gw_start_element_handler: Endpoint has been set to 2
*Jan  9 09:44:45.579: cmapp_sccp_gw_start_element_handler: Endpoint has been set to 3
Table 50: debug ccm-manager Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$nn:nn:nn$</td>
<td>Timestamp time in hours (military format), minutes, and seconds that indicates when the Cisco CallManager event occurred.</td>
</tr>
<tr>
<td>cmapp_error message</td>
<td>The Cisco CallManager routine in which the error event occurred.</td>
</tr>
<tr>
<td>LocaleName</td>
<td>Region name, such as United Kingdom.</td>
</tr>
<tr>
<td>low frequency level</td>
<td>DTMF low frequency.</td>
</tr>
<tr>
<td>high frequency level</td>
<td>DTMF high frequency.</td>
</tr>
<tr>
<td>operation</td>
<td>Special operations, such as uLaw.</td>
</tr>
</tbody>
</table>

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>show ccm-manager</td>
<td>Displays a list of Cisco CallManager servers, their current status, and their availability.</td>
</tr>
</tbody>
</table>
debug ccsip all

To enable all Session Initiation Protocol (SIP)-related debugging, use the `debug ccsip all` command in privileged EXEC mode. To disable debugging output, use the `no debug ccsip all` command.

```plaintext
debug ccsip all
no debug ccsip all
```

### Syntax Description

This command has no arguments or keywords.

### Command Default

No default behavior or values

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(1)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.1(3)T</td>
<td>The output of this command was changed.</td>
</tr>
<tr>
<td>12.2(2)XA</td>
<td>Support was added for the Cisco AS5350 and Cisco AS5400 universal gateways.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850 universal gateway.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(8)T and implemented on Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(11)T. Support for the Cisco AS5300 universal access server, Cisco AS5350, Cisco AS5400, and Cisco AS5850 universal gateway is not included in this release.</td>
</tr>
</tbody>
</table>

### Usage Guidelines

The `debug ccsip all` command enables the following SIP debug commands:

- `debug ccsip events`
- `debug ccsip error`
- `debug ccsip states`
- `debug ccsip messages`
- `debug ccsip calls`

### Examples

The following example displays debug output from one side of the call:

```plaintext
Router# debug ccsip all
```
All SIP call tracing enabled

Router1# * Mar 6 14:10:42: 0x624CFEF8 : State change from (STATE_NONE, SUBSTATE_NONE) to (STATE_IDLE, SUBSTATE_NONE)
* Mar 6 14:10:42: Queued event from SIP SPI : SIPSPI_EV_CC_CALL_SETUP
* Mar 6 14:10:42: Queued event from SIP SPI : SIPSPI_EV_CREATE_CONNECTION
* Mar 6 14:10:42: 0x624CFEF8 : State change from (STATE_IDLE, SUBSTATE_NONE) to (STATE_IDLE, SUBSTATE_CONNECTING)
* Mar 6 14:10:42: REQUEST CONNECTION TO IP:166.34.245.231 PORT:5060
* Mar 6 14:10:42: 0x624CFEF8 : State change from (STATE_IDLE, SUBSTATE_CONNECTING) to (STATE_IDLE, SUBSTATE_NONE)
* Mar 6 14:10:42: act_idle_connection_created: Connid(1) created to 166.34.245.231:5060, local_port 54113
* Mar 6 14:10:42: Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
* Mar 6 14:10:42: 0x624CFEF8 : State change from (STATE_IDLE, SUBSTATE_CONNECTING) to (STATE_SENT_INVITE, SUBSTATE_NONE)
* Mar 6 14:10:42: Sent:
  INVITE sip:3660210@166.34.245.231;user=phone;phone-context=unknown SIP/2.0
  Via: SIP/2.0/UDP 166.34.245.230:54113
  From: "3660110" <sip:3660110@166.34.245.230>
  To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
  Date: Sat, 06 Mar 1993 19:10:42 GMT
  Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
  Cisco-Guid: 2881152943-2184249548-0-483039712
  User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
  CSeq: 101 INVITE
  Max-Forwards: 6
  Timestamp: 731427042
  Contact: <sip:3660110@166.34.245.230:5060;user=phone>
  Expires: 180
  Content-Type: application/sdp
  Content-Length: 137
v=0
o=CiscoSystemsSIP-GW-UserAgent 1212 283 IN IP4 166.34.245.230
s=SIP Call
t=0 0
m=audio 20208 RTP/AVP 0
* Mar 6 14:10:42: Received:
  SIP/2.0 100 Trying
  Via: SIP/2.0/UDP 166.34.245.230:54113
  From: "3660110" <sip:3660110@166.34.245.230>
  To: <sip:3660110@166.34.245.230;user=phone;phone-context=unknown>
  Date: Sat, 06 Mar 1993 19:10:42 GMT
  Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
  Cisco-Guid: 2881152943-2184249548-0-483039712
  User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
  Content-Length: 0
  Content-Type: application/sdp
  CSeq: 101 INVITE
* Mar 6 14:10:42: HandleUdpSocketReads : Msg enqueued for SPI with IPaddr: 166.34.245.231
* Mar 6 14:10:42: Received:
  SIP/2.0 180 Ringing
  Via: SIP/2.0/UDP 166.34.245.230:54113
  From: "3660110" <sip:3660110@166.34.245.230>
  To: <sip:3660110@166.34.245.230;user=phone;phone-context=unknown>
  Date: Mon, 08 Mar 1993 22:36:40 GMT
  Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
  Timestamp: 731427042
  Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
  CSeq: 101 INVITE
  Content-Size: 0
  Content-Type: application/sdp
Content-Length: 137
v=0
o=CiscoSystemsSIP-GW-UserAgent 969 7889 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20038 RTP/AVP 0

*Mar 6 14:10:42: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr: 166.34.245.231:5060
*Mar 6 14:10:42: Roundtrip delay 8 milliseconds for method INVITE
*Mar 6 14:10:42: HandleSIP1xxRinging: SDP MediaTypes negotiation successful!
Negotiated Codec  : g711ulaw , bytes :160
Inband Alerting   : 0
*Mar 6 14:10:42: 0x624CFEF8 : State change from (STATE_RECV_PROCEEDING, SUBSTATE_PROCESSING_PROCEEDING) to (STATE_RECV_PROCEEDING, SUBSTATE_PROCESSING_ALERTING)
*Mar 6 14:10:46: Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
Date: Mon, 08 Mar 1993 22:36:40 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/SIP enabled
Contact: <sip:3660210@166.34.245.231:5060;user=phone>
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 137
v=0
o=CiscoSystemsSIP-GW-UserAgent 969 7889 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20038 RTP/AVP 0

*Mar 6 14:10:46: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr: 166.34.245.231:5060
*Mar 6 14:10:46: Roundtrip delay 3536 milliseconds for method INVITE
Negotiated Codec  : g711ulaw , bytes :160
*Mar 6 14:10:46: CCSIP-SPICONTROL: sipSPIReconnectConnection
*Mar 6 14:10:46: Queued event from SIP SPI : SIPSPI_EV_RECONNECT_CONNECTION
*Mar 6 14:10:46: Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar 6 14:10:46: 0x624CFEF8 : State change from (STATE_RECV_PROCEEDING, SUBSTATE_PROCESSING_ALERTING) to (STATE_ACTIVE, SUBSTATE_NONE)
*Mar 6 14:10:46: The Call Setup Information is :
Call Control Block (CCB) : 0x624CFEF8
State of The Call : STATE_ACTIVE
TCP Sockets Used : NO
Calling Number : 3660110
Called Number : 3660210
Negotiated Codec  : g711ulaw
Source IP Address (Media): 166.34.245.230
Source IP Port (Media): 20208
Destn IP Address (Media) : 166.34.245.231
Destn IP Port (Media): 20038
Destn SIP Addr (Control) : 166.34.245.231
Destn SIP Port (Control) : 5060
Destination Name : 166.34.245.231
*Mar 6 14:10:46: HandleUdpReconnection: Udp socket connected for fd: 1 with 166.34.245.231:5060
*Mar 6 14:10:46: Sent:
ACK sip:3660210@166.34.245.231:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP 166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
Date: Sat, 06 Mar 1993 19:10:42 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Max-Forwards: 6
Content-Type: application/sdp
Content-Length: 137
CSeq: 101 ACK

v=0
o=CiscoSystemsSIP-GW-UserAgent 1212 283 IN IP4 166.34.245.230
t=0 0
m=audio 20208 RTP/AVP 0

*Mar 6 14:10:46: ccsip_caps_ind: Load DSP with codec (5) g711ulaw, Bytes=160
*Mar 6 14:10:50: Received:
BYE sip:3660110@166.34.245.230:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP 166.34.245.231:54835
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
To: "3660110" <sip:3660110@166.34.245.230>
Date: Sat, 06 Mar 1993 19:10:50 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Max-Forwards: 6
Timestamp: 731612207
CSeq: 101 BYE
Content-Length: 0

*Mar 6 14:10:50: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr: 166.34.245.231:54835
*Mar 6 14:10:50: CCSIP-SPI-CONTROL: act_active_new_message
*Mar 6 14:10:50: CCSIP-SPI-CONTROL: acf_active_new_message_request
*Mar 6 14:10:50: CCSIP-SPI-CONTROL: sip_stats_method
*Mar 6 14:10:50: Queued event from SIP SPI : SIPSI_EV_SEND_MESSAGE
*Mar 6 14:10:50: CCSIP-SPI-CONTROL: sip_stats_status_code
*Mar 6 14:10:50: CCSIP-SPI-CONTROL: sipSPInitiateCallDisconnect : Initiate call disconnect(16) for outgoing call
*Mar 6 14:10:50: 0x624CFEF8 : State change from (STATE_ACTIVE, SUBSTATE_NONE) to (STATE_DISCONNECTING, SUBSTATE_NONE)
*Mar 6 14:10:50: Sent:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 166.34.245.231:54835
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
To: "3660110" <sip:3660110@166.34.245.230>
Date: Sat, 06 Mar 1993 19:10:50 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Timestamp: 731612207
Content-Length: 0
CSeq: 101 BYE

*Mar 6 14:10:50: Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_DISCONNECT
*Mar 6 14:10:50: CCSIP-SPI-CONTROL: act_disconnecting_disconnect
*Mar 6 14:10:50: CCSIP-SPI-CONTROL: sipSPICallCleanup
*Mar 6 14:10:50: Queued event from SIP SPI : SIPSI_EV_CLOSE_CONNECTION
*Mar 6 14:10:50: CLOSE CONNECTION TO CONNID:1
*Mar 6 14:10:50: 0x624CFEF8 : State change from (STATE_DISCONNECTING, SUBSTATE_NONE) to (STATE_DEAD, SUBSTATE_NONE)
*Mar 6 14:10:50: The Call Setup Information is :
Call Control Block (CCB) : 0x624CFEF8
State of The Call : STATE_DEAD
TCP Sockets Used : NO
Calling Number : 3660110
Called Number : 3660210
Negotiated Codec : g711ulaw
Source IP Address (Media): 166.34.245.230
Source IP Port (Media): 20208
Destn IP Address (Media): 166.34.245.231
Destn IP Port (Media): 20208
Destn SIP Addr (Control) : 166.34.245.231
Destn SIP Port (Control) : 5060
Destination Name : 166.34.245.231

Cisco IOS Debug Command Reference - Commands A through D
The following example displays debut output from the other side of the call:

Router# debug ccsip all
All SIP call tracing enabled
3660-2#
*Mar 8 17:36:40: Received:
INVITE sip:3660110@166.34.245.230;user=phone;phone-context=unknown SIP/2.0
Via: SIP/2.0/UDP 166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Sat, 06 Mar 1993 14:10:42 GMT
Call-ID: ABBAE7AF-823100CR-0-1CCAA69CQ172.18.192.194
Cisco-Guid: 2881152943-218429548-0-483039712
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Max-Forwards: 6
Timestamp: 731427042
Contact: <sip:3660110@166.34.245.230:5060;user=phone>
Expires: 180
Content-Type: application/sdp
Content-Length: 137
v=0
o=CiscoSystemsSIP-GW-UserAgent 1212 283 IN IP4 166.34.245.230
s=SIP Call
t=0 0
c=IN IP4 166.34.245.230
m=audio 20208 RTP/AVP 0
*Mar 8 17:36:40: act_recd_invite: Media Negotiation successful for an incoming call
*Mar 8 17:36:40: act_recd_invite: Negotiated Codec : g711ulaw, bytes :160
Preferred Codec : g711ulaw, bytes :160
*Mar 8 17:36:40: Queued event From SIP SPI : SIP/2.0/UDP 166.34.245.230:54113
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:36:40 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69CQ172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Length: 0
*Mar 8 17:36:40: Queued event From SIP SPI : SIP/2.0/UDP 166.34.245.230:54113
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:36:40 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69CQ172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Length: 0
*Mar 8 17:36:40: 180 Ringing with SDP - not likely
*Mar 8 17:36:40: Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar 8 17:36:40: CCSIP-SPI-CONTROL: aip_stats_status_code
*Mar 8 17:36:40: 0x624D8CCC : State change from (STATE_RECD_INVITE, SUBSTATE_RECD_INVITE_CALL_SETUP) to (STATE_SENT_ALERTING, SUBSTATE_NONE)
*Mar 8 17:36:40: Sent:
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:36:40 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 137
v=0
o=CiscoSystemsSIP-GW-UserAgent 969 7889 IN IP4 166.34.245.231
a=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20038 RTP/AVP 0
*Mar 8 17:36:44: Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_CONNECT
*Mar 8 17:36:44: CCSIP-SPI-CONTROL: act_sentalert_connect
*Mar 8 17:36:44: sipSPIAddLocalContact
*Mar 8 17:36:44: Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar 8 17:36:44: CCSIP-SPI-CONTROL: aip_stats_status_code
*Mar 8 17:36:44: 0x624D8CCC : State change from (STATE_SENT_ALERTING, SUBSTATE_NONE) to (STATE_SENT_SUCCESS, SUBSTATE_NONE)
*Mar 8 17:36:44: Sent:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
Date: Sat, 06 Mar 1993 19:10:42 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Contact: <sip:3660210@166.34.245.231:5060;user=phone>
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 137
v=0
c=IN IP4 166.34.245.231
m=audio 20038 RTP/AVP 0
*Mar 8 17:36:44: Received:
ACK sip:3660210@166.34.245.231:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP 166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
Date: Sat, 06 Mar 1993 19:10:42 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Max-Forwards: 6
Content-Type: application/sdp
Content-Length: 137
CSeq: 101 ACK
v=0
c=IN IP4 166.34.245.231
m=audio 20038 RTP/AVP 0
State of The Call : STATE_ACTIVE
TCP Sockets Used : NO
Calling Number : 3660110
Called Number : 3660210
Negotiated Codec : g711ulaw
Source IP Address (Media): 166.34.245.231
Source IP Port (Media): 20038
Destn IP Address (Media): 166.34.245.230
Destn IP Port (Media): 20208
Destn SIP Addr (Control) : 166.34.245.230
Destn SIP Port (Control) : 5060
Destination Name : 166.34.245.230

Mar 8 17:36:47: Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_DISCONNECT
Mar 8 17:36:47: CCSIP-SPI-CONTROL: act_active_disconnect
Mar 8 17:36:47: Queued event from SIP SPI : SIPSPI_EV_CREATE_CONNECTION
Mar 8 17:36:47: 0x624D8CCC : State change from (STATE_ACTIVE, SUBSTATE_NONE) to (STATE_ACTIVE, SUBSTATE_CONNECTING)
Mar 8 17:36:47: REQUEST CONNECTION TO IP:166.34.245.230 PORT:5060
Mar 8 17:36:47: 0x624D8CCC : State change from (STATE_ACTIVE, SUBSTATE_CONNECTING) to (STATE_ACTIVE, SUBSTATE_CONNECTING)
Mar 8 17:36:47: CCSIP-SPI-CONTROL: act_active_connection_created
Mar 8 17:36:47: CCSIP-SPI-CONTROL: sipSPICheckSocketConnection
Mar 8 17:36:47: CCSIP-SPI-CONTROL: sipSPICheckSocketConnection: Connid(1) created to 166.34.245.230:5060, local_port 54835
Mar 8 17:36:47: Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
Mar 8 17:36:47: CCSIP-SPI-CONTROL: sip_stats_method
Mar 8 17:36:47: 0x624D8CCC : State change from (STATE_ACTIVE, SUBSTATE_CONNECTING) to (STATE_DISCONNECTING, SUBSTATE_NONE)
Mar 8 17:36:47: Close:
BYE sip:3660110@166.34.245.230:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP 166.34.245.231:54835
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
To: "3660110" <sip:3660110@166.34.245.230>
Date: Mon, 08 Mar 1993 22:36:44 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C0172.18.192.194
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Max-Forwards: 6
Timestamp: 731612207
CSeq: 101 BYE
Content-Length: 0

Mar 8 17:36:47: Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 166.34.245.231:54835
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
To: "3660110" <sip:3660110@166.34.245.230>
Date: Sat, 06 Mar 1993 19:10:50 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C0172.18.192.194
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Max-Forwards: 6
Timestamp: 731612207
CSeq: 101 BYE
Content-Length: 0

Mar 8 17:36:47: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr: 166.34.245.230:54113
Mar 8 17:36:47: CCSIP-SPI-CONTROL: act_disconnecting_new_message
Mar 8 17:36:47: CCSIP-SPI-CONTROL: sact_disconnecting_new_message_response
Mar 8 17:36:47: CCSIP-SPI-CONTROL: sipSPICheckResponse
Mar 8 17:36:47: CCSIP-SPI-CONTROL: sip_stats_status_code
Mar 8 17:36:47: Roundtrip delay 4 milliseconds for method BYE
Mar 8 17:36:47: CCSIP-SPI-CONTROL: sipSPICallCleanup
Mar 8 17:36:47: Queued event from SIP SPI : SIPSPI_EV_CLOSE_CONNECTION
Mar 8 17:36:47: CLOSE CONNECTION TO CONNID:1
Mar 8 17:36:47: 0x624D8CCC : State change from (STATE_DISCONNECTING, SUBSTATE_NONE) to (STATE_DEAD, SUBSTATE_NONE)
Mar 8 17:36:47: The Call Setup Information is:
\nCall Control Block (CCB) : 0x624D8CCC
State of The Call : STATE_DEAD
TCP Sockets Used : NO
Calling Number : 3660110
Called Number : 3660210
Negotiated Codec : g711ulaw
Source IP Address (Media): 166.34.245.231
Source IP Port (Media): 20038
Destn IP Address (Media): 166.34.245.230
Destn IP Port   (Media): 20208
Destn SIP Addr (Control) : 166.34.245.230
Destn SIP Port (Control) : 5060
Destination Name     : 166.34.245.230

*Mar  8 17:36:47: Disconnect Cause (CC) : 16
      Disconnect Cause (SIP) : 200

*Mar  8 17:36:47: udpsock_close_connect: Socket fd: 1 closed for connid 1 with remote port: 5060

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip calls</td>
<td>Shows all SIP SPI call tracing.</td>
</tr>
<tr>
<td>debug ccsip error</td>
<td>Shows SIP SPI errors.</td>
</tr>
<tr>
<td>debug ccsip events</td>
<td>Shows all SIP SPI events tracing.</td>
</tr>
<tr>
<td>debug ccsip info</td>
<td>Shows all SIP SPI message tracing.</td>
</tr>
<tr>
<td>debug ccsip states</td>
<td>Shows all SIP SPI state tracing.</td>
</tr>
</tbody>
</table>
debug ccsip calls

To show all Session Initiation Protocol (SIP) Service Provider Interface (SPI) call tracing, use the `debug ccsip calls` command in privileged EXEC mode. To disable debugging output, use the no form of this command.

```
debug ccsip calls
no debug ccsip calls
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

- 12.1(1)T This command was introduced.
- 12.1(3)T The output of this command was changed.
- 12.2(2)XA Support was added for the Cisco AS5350 and Cisco AS5400 universal gateways.
- 12.2(2)XB1 This command was introduced on the Cisco AS5850 universal gateway.
- 12.2(8)T This command was integrated into Cisco IOS Release 12.2(8)T and implemented on Cisco 7200 series routers.
- 12.2(11)T This command was integrated into Cisco IOS Release 12.2(11)T. Support for the Cisco AS5300 universal access server, Cisco AS5350, Cisco AS5400, and Cisco AS5850 universal gateway is not included in this release.

**Usage Guidelines**

This command traces the SIP call details as they are updated in the SIP call control block.

**Examples**

The following example displays debug output from one side of the call:

```
Router1# debug ccsip calls
SIP Call statistics tracing is enabled
Router1#
*Mar  6 14:12:33: The Call Setup Information is :
   Call Control Block (CCB) : 0x624D078C
   State of The Call    : STATE_ACTIVE
   Calling Number       : 3660110
   Called Number        : 3660210
   Negotiated Codec     : g711ulaw
   Source IP Address (Media): 166.34.245.230
```
Source IP Port (Media): 20644
Destn IP Address (Media): 166.34.245.231
Destn IP Port (Media): 20500
Destn SIP Addr (Control): 166.34.245.231
Destn SIP Port (Control): 5060
Destination Name : 166.34.245.231

*Mar 6 14:12:40: The Call Setup Information is :
Call Control Block (CCB) : 0x624D078C
State of The Call : STATE_DEAD
TCP Sockets Used : NO
Calling Number : 3660110
Called Number : 3660210
Negotiated Codec : g711ulaw
Source IP Address (Media): 166.34.245.230
Source IP Port (Media): 20644
Destn IP Address (Media): 166.34.245.231
Destn IP Port (Media): 20500
Destn SIP Addr (Control): 166.34.245.231
Destn SIP Port (Control): 5060
Destination Name : 166.34.245.231

*Mar 6 14:12:40:
Disconnect Cause (CC) : 16
Disconnect Cause (SIP) : 200

The following example displays debug output from the other side of the call:

Router2# debug ccsip calls
SIP Call statistics tracing is enabled

*Mar 8 17:38:31: The Call Setup Information is :
Call Control Block (CCB) : 0x624D9560
State of The Call : STATE_ACTIVE
TCP Sockets Used : NO
Calling Number : 3660110
Called Number : 3660210
Negotiated Codec : g711ulaw
Source IP Address (Media): 166.34.245.231
Source IP Port (Media): 20500
Destn IP Address (Media): 166.34.245.230
Destn IP Port (Media): 20644
Destn SIP Addr (Control): 166.34.245.230
Destn SIP Port (Control): 5060
Destination Name : 166.34.245.230

*Mar 8 17:38:38: The Call Setup Information is:
Call Control Block (CCB) : 0x624D9560
State of The Call : STATE_DEAD
TCP Sockets Used : NO
Calling Number : 3660110
Called Number : 3660210
Negotiated Codec : g711ulaw
Source IP Address (Media): 166.34.245.231
Source IP Port (Media): 20500
Destn IP Address (Media): 166.34.245.230
Destn IP Port (Media): 20644
Destn SIP Addr (Control): 166.34.245.230
Destn SIP Port (Control): 5060
Destination Name : 166.34.245.230

*Mar 8 17:38:38:
Disconnect Cause (CC) : 16
Disconnect Cause (SIP) : 200

<table>
<thead>
<tr>
<th>Related Commands</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip all</td>
<td>Enables all SIP-related debugging.</td>
<td></td>
</tr>
<tr>
<td>debug ccsip error</td>
<td>Shows SIP SPI errors.</td>
<td></td>
</tr>
<tr>
<td>debug ccsip events</td>
<td>Shows all SIP SPI events tracing.</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>debug ccsip info</td>
<td>Shows all SIP SPI message tracing.</td>
<td></td>
</tr>
<tr>
<td>debug ccsip states</td>
<td>Shows all SIP SPI state tracing.</td>
<td></td>
</tr>
</tbody>
</table>
debug ccsip dhcp

To display debugging related information on Session Initiation Protocol (SIP) and Dynamic Host Configuration Protocol (DHCP) interaction, when SIP parameters are provisioned by DHCP, use the `debug ccsip dhcp` command in privileged EXEC mode. To disable debugging output, use the `no debug ccsip dhcp` command.

**debug ccsip dhcp**

**no debug ccsip dhcp**

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

**Command Modes**
Privileged EXEC (#)

**Command History**
- 12.4(22)YB This command was introduced.
- 15.0(1)M This command was integrated in Cisco IOS Release 15.0(1)M.

**Usage Guidelines**
The debug ccsip dhcp command can be enabled by executing the command itself or by issuing the debug ccsip all command.

**Examples**
The following example displays debug output from the debug ccsip dhcp command:

```
Router# debug ccsip dhcp
Nov 18 17:20:48.881: //1/xxxxxxxxxxxx/SIP/Info/ccsip_register_configured_dest_patterns:
No destination patterns to Register
Nov 18 17:20:48.881: //1/xxxxxxxxxxxx/SIP/Info/ccsip_register_reset_dns_cache:
CCSIP_REGISTER:: Primary registrar DNS resolved addr Reset
Nov 18 17:21:00.965: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/config_credential_trigger_reg: Query DHCP for provisioned info upon credential dhcp config
Nov 18 17:21:00.965: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/sipua_query_dhcp_reg_info: DHCP provisioned option 125 available
Nov 18 17:21:00.965: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: parsing data in option 125 of length 73
Nov 18 17:21:00.965: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: enterprise ID 210
Nov 18 17:21:00.965: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: total option data length 80
Nov 18 17:21:00.965: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: sub-option type 201 of length 6
Nov 18 17:21:00.965: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: sub-option type 202 of length 6
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: sub-option type 203 of length 6
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: secondary # 222222 (index 0)
```
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: sub-option
type 203 of length 6
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_subopt_addn_num:
secondary # 333333 (index 1)
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: sub-option
type 203 of length 6
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_subopt_addn_num:
secondary # 444444 (index 2)
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: sub-option
type 203 of length 6
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_subopt_addn_num:
secondary # 555555 (index 3)
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: sub-option
type 203 of length 6
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_subopt_addn_num:
secondary # 666666 (index 4)
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: parsing of
DHCP option 125 succeeded
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_subopt_domain:
domain sublen 5
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_subopt_domain:
domain sublen 3
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_subopt_domain:
domain dns:cisco.com
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: parsing of
DHCP option 125 succeeded
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: parsing of
DHCP option 125 succeeded
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: parsing of
DHCP option 125 succeeded
Nov 18 17:21:00.977: //1/xxxxxxxxxxxx/SIP/Info/sip_dns_type_a_aaaa_query: 9.13.2.36
Nov 18 17:21:00.977: //1/xxxxxxxxxxxx/SIP/Info/ccsip_process_sipspi_queue_event: ccsip_spi_get_msg_type returned: 2 for event 43
Nov 18 17:21:00.977: //1/xxxxxxxxxxxx/SIP/Info/ccsip_register_set_dns_resolved_address: CCSIP_REGISTER:: Primary registrar DNS resolved addr set to 0.0.0.0:15184760
Nov 18 17:21:00.977: //1/xxxxxxxxxxxx/SIP/Info/ccsipRegisterStartExpiresTimer: Starting timer for pattern for 3600 seconds
Nov 18 17:21:00.977: //1034/000000000000/SIP/State/sipSPIChangeState: 0x476FD758 : State change from (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_SENT_DNS) to (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_NONE)
Nov 18 17:21:00.977: //1/xxxxxxxxxxxx/SIP/Info/ccsipRegisterStartExpiresTimer: Starting timer for pattern for 3600 seconds
Nov 18 17:21:00.977: //1034/000000000000/SIP/State/sipSPIChangeState: 0x476FD758 : State change from (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_NONE) to (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_NONE)

Nov 18 17:21:00.981: //1034/000000000000/SIP/Info/sipSPISetDateHeader: Clock Time Zone is UTC, same as GMT: Using GMT
Nov 18 17:21:00.981: //1034/000000000000/SIP/Transport/sipSPISendRegister: Associated container=0x46794ACC to Register
Nov 18 17:21:00.981: //1034/000000000000/SIP/Transport/sipSPIGetSwitchTransportFlag: Return the Dial peer configuration, Switch Transport is FALSE
Nov 18 17:21:00.981: //1034/000000000000/SIP/Transport/sipSPITransportSendMessage: msg=0x4707F998, addr=9.13.2.36, port=5060, sentBy_port=0, is_req=1, transport=1, switch=0, callback=0x415A53B0
Nov 18 17:21:00.981: //1034/000000000000/SIP/Transport/sipSPITransportSendMessage: Proceedable for sending msg immediately
Nov 18 17:21:00.981: //1034/000000000000/SIP/State/sipSPIChangeState: 0x476FD758 : State change from (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_NONE) to (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_NONE)

Nov 18 17:21:00.981: //1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg: Sent:
REGISTER sip:cisco.com:5060 SIP/2.0
Date: Tue, 18 Nov 2008 17:21:00 GMT
From: <sip:777777@cisco.com>;tag=34FBAED8-131
Supported: path
Timestamp: 1227028860
User-Agent: Cisco-SIPGateway/IOS-12.x
To: <sip:777777@cisco.com>
Contact: <sip:777777@9.13.8.183:5060>
Expires: 3600
Call-ID: 1AF6E28A-B4CC11DD-81078B9C-6E99E02B
Via: SIP/2.0/UDP 9.13.8.183:5060;branch=z9hG4bK3F522D9
CSeq: 2 REGISTER
Max-Forwards: 70
Nov 18 17:21:00.981: //1/xxxxxxxxxxxx/SIP/Info/HandleUdpIPv4SocketReads: Msg enqueued for SPI with IP addr: [9.13.2.36]:56305
Nov 18 17:21:00.981: //1/xxxxxxxxxxxx/SIP/Info/ccsip_process_sipspi_queue_event: ccsip_spi_get_msg_type returned: 2 for event 1
Nov 18 17:21:00.981: //1/xxxxxxxxxxxx/SIP/Info/ccsip_register_set_dns_resolved_address: CCSIP_REGISTER:: Primary registrar DNS resolved addr set to 0.0.0.0:15184760
Nov 18 17:21:00.981: //1034/000000000000/SIP/State/sipSPIChangeState: 0x476FD758 : State change from (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_SENT_DNS) to (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_NONE)
Nov 18 17:21:00.981: //1/xxxxxxxxxxxx/SIP/Info/HandleUdpIPv4SocketReads: Msg enqueued for SPI with IP addr: [9.13.2.36]:56305
Nov 18 17:21:00.981: //1/xxxxxxxxxxxx/SIP/Info/ccsip_process_sipspi_queue_event: ccsip_spi_get_msg_type returned: 2 for event 1
Nov 18 17:21:00.981: //1/xxxxxxxxxxxx/SIP/State/sipSPIChangeState: 0x476FD758 : State change from (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_NONE) to (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_NONE)
Nov 18 17:21:00.981: //1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg: Received:
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 9.13.8.183:5060;received=9.13.8.183;branch=z9hG4k3F522D9
Call-ID: 1AF6E28A-B4CC11DD-81078B9C-6E99E02B
From: <sip:777777@cisco.com>;tag=34FBAED8-131
To: <sip:777777@cisco.com>
CSeq: 2 REGISTER
Content-Length: 0
Nov 18 17:21:01.077: //1/xxxxxxxxxxxx/SIP/Info/ccsip_process_sipspi_queue_event: ccsip_spi_get_msg_type returned: 2 for event 1
Nov 18 17:21:01.077: //1/xxxxxxxxxxxx/SIP/State/sipSPIChangeState: 0x476FD758 : State change from (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_NONE) to (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_NONE)
Nov 18 17:21:01.077: //1/xxxxxxxxxxxx/SIP/Info/ccsip_process_sipspi_queue_event: ccsip_spi_get_msg_type returned: 2 for event 1
Nov 18 17:21:01.077: //1/xxxxxxxxxxxx/SIP/State/sipSPIChangeState: 0x476FD758 : State change from (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_NONE) to (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_NONE)
Nov 18 17:21:01.077: //1/yyyyyyyyyy/SIP/Msg/ccsipDisplayMsg: Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 9.13.8.183:5060;received=9.13.8.183;branch=z9hG4bK3F522D9
Call-ID: 1AF6E28A-B4CC11DD-81078B9C-6E99E02B
From: <sip:777777@cisco.com>;tag=34FBAED8-131
To: <sip:777777@cisco.com>
CSeq: 2 REGISTER
Contact: <sip:777777@9.13.8.183:5060>;expires=3600
Content-Length: 0

Nov 18 17:21:01.077: //1/yyyyyyyyyy/SIP/Info/ccsipRegisterStartExpiresTimer: Starting timer for pattern 77777 for 2880 seconds
Nov 18 17:21:01.077: //1/yyyyyyyyyy/SIP/Info/sipSPIUdeleteCcbFromTable: Deleting from table. ccb=0x476FD758 key=1AF6E28A-B4CC11DD-81078B9C-6E99E02B
Nov 18 17:21:01.077: //1/yyyyyyyyyy/SIP/Info/sipSPIFlushEventBufferQueue: There are 0 events on the internal queue that are going to be free'd
Nov 18 17:21:01.077: //1/yyyyyyyyyy/SIP/Info/ccsip_qos_cleanup: Entry
Nov 18 17:21:01.077: //1/yyyyyyyyyy/SIP/Info/ccsip_qos_free_codec_profile: Codec Profiles Freed
Nov 18 17:21:02.761: %SYS-5-CONFIG_I: Configured from console by console CUBE-DHCP-CLIENT1#
debug ccsip error

To show Session Initiation Protocol (SIP) Service Provider Interface (SPI) errors, use the debug ccsip error command in privileged EXEC mode. To disable debugging output, use the no form of this command.

**debug ccsip error**

**no debug ccsip error**

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Version</th>
<th>Release</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(1)T</td>
<td></td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)XA</td>
<td></td>
<td>Support was added for the Cisco AS5350 and Cisco AS5400 universal gateways.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td></td>
<td>This command was implemented on the Cisco AS5850 universal gateway.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td></td>
<td>This command was integrated into Cisco IOS Release 12.2(8)T and implemented on Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td></td>
<td>This command was integrated into Cisco IOS Release 12.2(11)T. Support for the Cisco AS5300 universal access server, Cisco AS5350, Cisco AS5400, and Cisco AS850 universal gateway is not included in this release.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

This command traces all error messages generated from errors encountered by the SIP subsystem.

**Examples**

The following example displays debug output from one side of the call:

```
Router1# debug ccsip error
SIP Call error tracing is enabled
Router1# *Mar 6 14:16:41: CCSIP-SPI-CONTROL: act_idle_call_setup
*Mar 6 14:16:41: act_idle_call_setup: Not using Voice Class Codec
*Mar 6 14:16:41: REQUEST_CONNECTION TO IP:166.34.245.231 PORT:5060
*Mar 6 14:16:41: CCSIP-SPI-CONTROL: act_idle_connection_created
*Mar 6 14:16:41: CCSIP-SPI-CONTROL: act_idle_connection_created: Connid(1) created to 166.34.245.231:5060, local_port 55674
*Mar 6 14:16:41: sipSPIAddLocalContact
*Mar 6 14:16:41: sipSPIAddLocalContact
*Mar 6 14:16:41: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr: 166.34.245.231:5060
```
The following example displays debug output from the other side of the call:

Router2# debug ccsip error
SIP Call error tracing is enabled
Router2# debug backhaul-session-manager session through debug channel packets
**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip all</td>
<td>Enables all SIP-related debugging.</td>
</tr>
<tr>
<td>debug ccsip calls</td>
<td>Shows all SIP SPI call tracing.</td>
</tr>
<tr>
<td>debug ccsip events</td>
<td>Shows all SIP SPI events tracing.</td>
</tr>
<tr>
<td>debug ccsip info</td>
<td>Shows all SIP SPI message tracing.</td>
</tr>
<tr>
<td>debug ccsip states</td>
<td>Shows all SIP SPI state tracing.</td>
</tr>
</tbody>
</table>
debug ccsip events

To enable tracing of events that are specific to service provider interface (SPI), use the `debug ccsip events` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug ccsip events
no debug ccsip events
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(1)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)XA</td>
<td>Support was added for the Cisco AS5350 and Cisco AS5400 universal gateways.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was introduced on the Cisco AS5850 universal gateway.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(11)T.</td>
</tr>
<tr>
<td>12.2(15)T</td>
<td>Much of the information formerly found in the output of the <code>debug ccsip events</code> command is now reported in the output of the <code>debug ccsipinfo</code> and <code>debug ccsipmedia</code> commands. The <code>debug ccsip events</code> command now displays only the debugging information specifically related to SIP events.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

This command previously traced all events posted to Session Initiation Protocol (SIP) SPI from all interfaces and also provided general SIP SPI information. Beginning with Cisco IOS Release 12.2(15)T, the `debug ccsip events` command displays only debugging information specifically related to SIP SPI events. Media stream and SIP SPI information is now reported in the `debug ccsipmedia` and `debug ccsipinfo` command output.

**Note**

This command is intended for use by Cisco technicians only.

**Examples**

The following is sample output from the `debug ccsip events` command for a Cisco 3660:

```
Router# debug ccsip events
SIP Call events tracing is enabled
```
Router#
Nov 15 18:20:25.779: Queued event from SIP SPI : SIPSPI_EV_CC_CALL_SETUP
Nov 15 18:20:25.779: Queued event from SIP SPI : SIPSPI_EV_CREATE_CONNECTION
Nov 15 18:20:25.783: Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
Nov 15 18:20:25.815: Queued event from SIP SPI : SIPSPI_EV_CREATE_CONNECTION
Nov 15 18:20:25.819: Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
Nov 15 18:20:50.844: Queued event from SIP SPI : SIPSPI_EV_CLOSE_CONNECTION
Nov 15 18:20:50.844: Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
Nov 15 18:20:50.848: Queued event from SIP SPI : SIPSPI_EV_CC_CALL_DISCONNECT

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip all</td>
<td>Enables all SIP-related debugging.</td>
</tr>
<tr>
<td>debug ccsip info</td>
<td>Enables tracing of general SIP SPI information.</td>
</tr>
<tr>
<td>debug ccsip media</td>
<td>Enables tracing of SIP call media streams.</td>
</tr>
</tbody>
</table>
debug ccsip feature

To filter the DEBUG CCSIP INFO debugs based on various features, use the debug ccsip feature command in privileged EXEC mode. To disable filtering of the INFO debugs, use the no form of this command.

`debug ccsip feature feature-name1 [feature-name2] [feature-name3] [feature-name...]`

`no debug ccsip`

**Syntax Description**

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>audio</td>
<td>Enables audio debugging.</td>
</tr>
<tr>
<td>cac</td>
<td>Enables call admission control (CAC) debugging.</td>
</tr>
<tr>
<td>config</td>
<td>Enables configuration debugging.</td>
</tr>
<tr>
<td>control</td>
<td>Enables control debugging.</td>
</tr>
<tr>
<td>dtmf</td>
<td>Enables Dual Tone Multi-Frequency (DTMF) debugging.</td>
</tr>
<tr>
<td>fax</td>
<td>Enables fax debugging.</td>
</tr>
<tr>
<td>line</td>
<td>Enables SIP Cisco Call Manager Express line debugging.</td>
</tr>
<tr>
<td>misc-features</td>
<td>Enables miscellaneous feature debugging.</td>
</tr>
<tr>
<td>miscellaneous</td>
<td>Enables misc (catch all) debugging.</td>
</tr>
<tr>
<td>Feature Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>parse</td>
<td>Enables parse debugging.</td>
</tr>
<tr>
<td>registration</td>
<td>Enables SIP registration debugging.</td>
</tr>
<tr>
<td>sdp-negotiation</td>
<td>Enables Session Description Protocol (SDP) negotiation debugging.</td>
</tr>
<tr>
<td>sdp-passthrough</td>
<td>Enables SDP-passthrough debugging.</td>
</tr>
<tr>
<td>sip-profiles</td>
<td>Enables SIP-profiles debugging.</td>
</tr>
<tr>
<td>sip-transport</td>
<td>Enables SIP transport debugging.</td>
</tr>
<tr>
<td>srtp</td>
<td>Enables Secure Real-Time Transport Protocol (SRTP) debugging.</td>
</tr>
<tr>
<td>supplementary-services</td>
<td>Enables supplementary services debugging.</td>
</tr>
<tr>
<td>transcoder</td>
<td>Enables transcoder debugging.</td>
</tr>
<tr>
<td>video</td>
<td>Enables video debugging.</td>
</tr>
</tbody>
</table>

**Note**

To activate feature debugging you must first enable CCSIP debugging using either the `debug ccsip info` or `debug ccsip all` command.

**Examples**

The following example shows how to enable CCSIP feature debugging for audio, fax, DTMF, CAC and SIP registration:

```
Device# debug ccsip info
SIP Call info tracing is enabled
device debug ccsip feature audio cac dtmf fax registration
audio debugging for ccsip info is enabled (active)
fax debugging for ccsip info is enabled (active)
dtmf debugging for ccsip info is enabled (active)
cac debugging for ccsip info is enabled (active)
registration debugging for ccsip info is enabled (active)
```
debug ccsip info

To enable tracing of general service provider interface (SPI) information, use the `debug ccsip info` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug ccsip info
no debug ccsip info
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(15)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Beginning in Cisco IOS Release 12.2(15)T, the `debug ccsip info` command is a separate option that displays general SIP SPI information for debug purposes. In past releases, this output was part of the `debug ccsip events` command.

**Note**

This command is intended for use by Cisco technicians only.

**Examples**

The following is sample output from the `debug ccsip info` command for a Cisco 3660:

```
Router# debug ccsip info
SIP Call info tracing is enabled
Router# Nov 15 18:19:22.670: ****Adding to UAC table
Nov 15 18:19:22.670: adding call id E to table
Nov 15 18:19:22.670: sipSPICopyPeerDataToCCB: From CLI: Modem NSE payload = 100, Passthrough = 0, Modem relay = 0, Gw-Xid = 1
SPRT latency 200, SPRT Retries = 12, Dict Size = 1024
String Len = 32, Compress dir = 3
Nov 15 18:19:22.670: ****Deleting from UAC table
Nov 15 18:19:22.670: ****Adding to UAC table
Nov 15 18:19:22.670: sipSPIUsetBillingProfile: sipCallId for billing records = 20A40C3B-D92C11D5-8015E1CC-C91F3F10@12.18.195.49
Nov 15 18:19:22.674: CCSIP-SPI-CONTROL: sipSPIOutgoingCallSDP
Nov 15 18:19:22.674: convert_codec_bytes_to_ptime: Values :Codec: g729r8 codecbytes :20, ptime: 10
```
debug backhaul-session-manager session through debug channel packets

debuge ccsp info
Nov 15 18:19:24.706: sip_stats_status_code
Nov 15 18:19:24.706: Roundtrip delay 2032 milliseconds for method PRACK
Nov 15 18:19:24.710: CCSIP-SPI-CONTROL: act_recproc_continue_200_processing: *** This ccb
is the parent
Nov 15 18:19:24.710: sipSPICompareRespMediaInfo
Nov 15 18:19:24.710: sipSPIDoMediaNegotiation: number of m lines is 1
Nov 15 18:19:24.710: sipSPIDoAudioNegotiation: Codec (g729r8) Negotiation Successful on
Static Payload
Nov 15 18:19:24.710: sipSPIDoPtimeNegotiation: One ptime attribute found = value:10
Nov 15 18:19:24.710: convert_ptime_to_codec_bytes: Values :Codec: g729r8 ptime :10,
codecbytes: 20
Nov 15 18:19:24.710: convert_codec_bytes_to_ptime: Values :Codec: g729r8 codecbytes :20,
ptime: 10
Nov 15 18:19:24.710: sipSPIDoDTMFRelayNegotiation: m-line index 1
Nov 15 18:19:24.710: sip_sdp_get_modem_relay_cap_params:
Nov 15 18:19:24.710: sip_sdp_get_modem_relay_cap_params: NSE payload from X-cap = 0
Nov 15 18:19:24.710: sip_do_nse_negotiation: NSE Payload 100 found in SDP
Nov 15 18:19:24.710: sip_do_nse_negotiation: Remote NSE payload = local one = 100, Use it
Nov 15 18:19:24.710: sip_select_modem_relay_params: X-tmr not present in SDP. Disable modem
relay
Nov 15 18:19:24.710: sipSPIProcessMediaChanges
Nov 15 18:19:24.710: ccsip_process_response_contact_record_route
Nov 15 18:19:24.710: sip_stats_method
Nov 15 18:19:24.710: udpsock_close_connect: Socket fd: 1 closed for connid 1 with remote
port: 5060
Nov 15 18:19:37.479: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
172.18.193.190:52180
Nov 15 18:19:37.483: ****Found CCB in UAC table
Nov 15 18:19:37.483: ****Found CCB in UAC table
Nov 15 18:19:37.483: CCSIP-SPI-CONTROL: act_active_new_message
Nov 15 18:19:37.483: CCSIP-SPI-CONTROL: act_active_new_message_request
Nov 15 18:19:37.483: sip_stats_method
Nov 15 18:19:37.483: sip_stats_status_code
Nov 15 18:19:37.483: CCSIP-SPI-CONTROL: sipSPIInitiateCallDisconnect : Initiate call
disconnect(16) for outgoing call
Nov 15 18:19:37.483: udpsock_close_connect: Socket fd: 2 closed for connid 2 with remote
port: 5060
Nov 15 18:19:37.483: debug ccsip info

---

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip all</td>
<td>Enables all SIP-related debugging.</td>
</tr>
<tr>
<td>debug ccsip events</td>
<td>Enables tracing of events that are specific to SIP SPI.</td>
</tr>
<tr>
<td>debug ccsip media</td>
<td>Enables tracing of SIP call media streams.</td>
</tr>
</tbody>
</table>
**debug ccsip level**

To enable filtering of the DEBUG CCSIP INFO messages based on importance, use the `debug ccsip level` command in privileged EXEC mode. To disable filtering of the INFO debugs, use the `no` form of this command.

`debug ccsip level {critical| info| notify| verbose}`

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>critical</td>
<td>Enables critical level debugging (Severity 1).</td>
</tr>
<tr>
<td>info</td>
<td>Enables information level debugging (Severity 3).</td>
</tr>
<tr>
<td>notify</td>
<td>Enables notification level debugging (Severity 2).</td>
</tr>
<tr>
<td>verbose</td>
<td>Enables verbose level debugging (Severity 4).</td>
</tr>
</tbody>
</table>

**Command Default**

Verbose-level debugging is enabled.

**Command Modes**

Privileged EXEC#

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.3(3)M</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

This command sets the level of debugging messages. Only one level of debug messages can be set.

When critical-level debugging is selected (Severity 1), messages are generated for specific errors such as resource failures. This information is used by the engineer to troubleshoot the system. When notify-level debugging is selected (Severity 2), messages include information on important milestones that were reached and important processing steps that were met. When information-level debugging is selected, (Severity 3), messages contain all details needed to understand workflow. When verbose-level debugging is selected, (Severity 4), messages contain a high level of detail, which might not be useful when troubleshooting issues.

**Examples**

The following example shows how to enable debugging messages for critical level events only:

```
Device# debug ccsip level critical
Critical mode tracing for ccsip info is enabled (active)
```
debug ccsip media

To enable tracing of Session Initiation Protocol (SIP) call media streams, use the debug ccsip media command in privileged EXEC mode. To disable debugging output, use the no form of this command.

debug ccsip media
no debug ccsip media

to enable tracing of Session Initiation Protocol (SIP) call media streams, use the debug ccsip media command in privileged EXEC mode. To disable debugging output, use the no form of this command.

debug ccsip media
no debug ccsip media

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(15)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

Usage Guidelines

Beginning in Cisco IOS Release 12.2(15)T, the debug ccsip media command is a separate option that displays debugging information specific to SIP media stream processing. In past releases, this output was part of the debug ccsip events command.

Note

This command is intended for use by Cisco technicians only.

Examples

The following is sample output from the debug ccsip media command for a Cisco 3660:

Router# debug ccsip media

SIP Call media tracing is enabled
Router#
Nov 15 18:19:53.835: sipSPIReserveRtpPort: reserved port 16500 for stream 1
Nov 15 18:19:53.867: sipSPIReplaceSDP
Nov 15 18:19:53.871: sipSPICopySdpInfo
Nov 15 18:19:53.871: sipSPIUpdateCallWithSdpInfo:
  Preferred Codec: g729r8, bytes: 20
  Preferred DTMF relay: inband-voice
  Preferred NTE payload: 101
  Early Media: No
  Delayed Media: No
  Bridge Done: No
  New Media: No
  DSP DNLD Req: No
Nov 15 18:19:53.871: sipSPIUpdateCallWithSdpInfo:
  M-line Index: 1
  State: STREAMADDING (3)
  Callid: -1
  Negotiated Codec: g729r8, bytes: 20
Negotiated DTMF relay: inband-voice
Negotiated NTE payload: 0
Media Src addr/Port: 172.18.195.49:16500
Media Dest addr/Port: 172.18.193.190:19148

Nov 15 18:19:53.871: sipSPIProcessRtpSessions
Nov 15 18:19:53.871: sipSPIAddStream: Adding stream 1 (callid 16) to the VOIP RTP library
Nov 15 18:19:53.871: sipSPIUpdateRtcpSession: for m-line 1
Nov 15 18:19:53.871: sipSPIUpdateRtcpSession: rtcp_session_info
laddr = 172.18.195.49, lport = 16500, raddr = 172.18.193.190, rport=19148
Nov 15 18:19:53.871: sipSPIUpdateRtcpSession: No rtcp session, creating a new one
Nov 15 18:19:53.871: sipSPIUpdateRtcpSession: rtcp_session_info
laddr = 172.18.195.49, lport = 16500, raddr = 172.18.193.190, rport=19148
Nov 15 18:19:53.871: sipSPIUpdateRtcpSession: No rtcp session, creating a new one
Nov 15 18:19:53.871: sipSPIUpdateRtcpSession: rtcp_session_info
laddr = 172.18.195.49, lport = 16500, raddr = 172.18.193.190, rport=19148
Nov 15 18:19:53.871: sipSPIUpdateRtcpSession: rtcp_session_info
laddr = 172.18.195.49, lport = 16500, raddr = 172.18.193.190, rport=19148
Nov 15 18:19:53.871: sipSPIUpdateRtcpSession: No rtcp session, creating a new one
Nov 15 18:19:53.871: sipSPIUpdateRtcpSession: rtcp_session_info
laddr = 172.18.195.49, lport = 16500, raddr = 172.18.193.190, rport=19148
Nov 15 18:19:53.871: sipSPIUpdateRtcpSession: No rtcp session, creating a new one
Nov 15 18:19:53.871: sipSPIUpdateRtcpSession: rtcp_session_info
laddr = 172.18.195.49, lport = 16500, raddr = 172.18.193.190, rport=19148
Nov 15 18:19:53.871: sipSPIUpdateRtcpSession: No rtcp session, creating a new one
Nov 15 18:19:53.871: sipSPIUpdateRtcpSession: rtcp_session_info
laddr = 172.18.195.49, lport = 16500, raddr = 172.18.193.190, rport=19148

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip all</td>
<td>Enables all SIP-related debugging.</td>
</tr>
<tr>
<td>debug ccsip events</td>
<td>Enables tracing of events that are specific to SIP SPI.</td>
</tr>
<tr>
<td>debug ccsip info</td>
<td>Enables tracing of general SIP SPI events.</td>
</tr>
</tbody>
</table>
**debug ccsip messages**

To show all Session Initiation Protocol (SIP) Service Provider Interface (SPI) message tracing, use the `debug ccsip messages` command in privileged EXEC mode. To disable debugging output, use the `no debug ccsip messages` command.

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(1)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.1(3)T</td>
<td>The output of this command was changed.</td>
</tr>
<tr>
<td>12.2(2)XA</td>
<td>Support was added for the Cisco AS5350 and Cisco AS5400 universal gateways.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850 universal gateway.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td>This command was implemented on Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(11)T. Support for the Cisco AS5300 universal access server, Cisco AS5350, Cisco AS5400, and Cisco AS5850 universal gateway is not included in this release.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
<tr>
<td>IOS Release XE 2.5</td>
<td>This command was integrated into Cisco IOS XE Release 2.5.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

This command traces the Session Initiation Protocol (SIP) messages exchanged between the SIP UA client (UAC) and the access server.
The following example shows debug output from one side of the call:

Router1# debug ccsip messages
SIP Call messages tracing is enabled

* Mar 6 14:19:14: Sent:
  INVITE sip:3660210@166.34.245.231;user=phone;phone-context=unknown SIP/2.0
  Via: SIP/2.0/UDP 166.34.245.230:55820
  From: "3660110" <sip:3660110@166.34.245.230>
  To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
  Date: Sat, 06 Mar 1993 19:19:14 GMT
  Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
  Cisco-Guid: 2881152943-2184249568-0-483551624
  User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
  CSeq: 101 INVITE
  Max-Forwards: 6
  Timestamp: 731427554
  Contact: <sip:3660110@166.34.245.230;user=phone>
  Expires: 180
  Content-Type: application/sdp
  Content-Length: 138
  v=0
  o=CiscoSystemsSIP-GW-UserAgent 5596 7982 IN IP4 166.34.245.230
  s=SIP Call
  t=0 0
  c=IN IP4 166.34.245.230
  m=audio 20762 RTP/AVP 0

* Mar 6 14:19:14: Received:
  SIP/2.0 100 Trying
  Via: SIP/2.0/UDP 166.34.245.230:55820
  From: "3660110" <sip:3660110@166.34.245.230>
  To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
  Date: Mon, 08 Mar 1993 22:45:12 GMT
  Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
  Timestamp: 731427554
  Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
  CSeq: 101 INVITE
  Content-Length: 0

* Mar 6 14:19:14: Received:
  SIP/2.0 180 Ringing
  Via: SIP/2.0/UDP 166.34.245.230:55820
  From: "3660110" <sip:3660110@166.34.245.230>
  To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
  Date: Mon, 08 Mar 1993 22:45:12 GMT
  Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
  Timestamp: 731427554
  Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
  CSeq: 101 INVITE
  Content-Type: application/sdp
  Content-Length: 138
  v=0
  o=CiscoSystemsSIP-GW-UserAgent 1193 7927 IN IP4 166.34.245.231
  s=SIP Call
  t=0 0
  c=IN IP4 166.34.245.231
  m=audio 20224 RTP/AVP 0

* Mar 6 14:19:16: Received:
  SIP/2.0 200 OK
  Via: SIP/2.0/UDP 166.34.245.230:55820
  From: "3660110" <sip:3660110@166.34.245.230>
  To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
  Date: Mon, 08 Mar 1993 22:45:12 GMT
  Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
  Timestamp: 731427554
  Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
  CSeq: 101 INVITE
  Content-Type: application/sdp
Content-Length: 138
v=0
o=CiscoSystemsSIP-GW-UserAgent 1193 7927 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20224 RTP/AVP 0

*Mar 6 14:19:16: Sent: ACK sip:3660210@166.34.245.231:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP 166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.231>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
Call-ID: ABBAE7AF-823100E2-0-1CD274BC0172.18.192.194
Max-Forwards: 6
Content-Type: application/sdp
Content-Length: 138
CSeq: 101 ACK
v=0
o=CiscoSystemsSIP-GW-UserAgent 5596 7982 IN IP4 166.34.245.230
s=SIP Call
t=0 0
c=IN IP4 166.34.245.230
m=audio 20762 RTP/AVP 0

*Mar 6 14:19:19: Received: BYE sip:3660110@166.34.245.230:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP 166.34.245.231:53600
From: "3660210" <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
To: "3660110" <sip:3660110@166.34.245.230>
Date: Mon, 08 Mar 1993 22:45:14 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC0172.18.192.194
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Max-Forwards: 6
Timestamp: 731612717
CSeq: 101 BYE
Content-Length: 0

*Mar 6 14:19:19: Sent:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 166.34.245.231:53600
From: "3660210" <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
To: "3660110" <sip:3660110@166.34.245.230>
Date: Sat, 06 Mar 1993 19:19:19 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC0172.18.192.194
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Timestamp: 731612717
Content-Length: 0
CSeq: 101 BYE

The following example show debug output from the other side of the call:

Router2# debug ccsip messages
SIP Call messages tracing is enabled
Router2#

*Mar 8 17:45:12: Received:
INVITE sip:3660210@166.34.245.231;user=phone;phone-context=unknown SIP/2.0
Via: SIP/2.0/UDP 166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.231>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Sat, 06 Mar 1993 19:19:19 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC0172.18.192.194
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Max-Forwards: 6
Timestamp: 731427554
Contact: <sip:3660110@166.34.245.230:5060;user=phone>
Expires: 180
Content-Type: application/sdp
Content-Length: 138
v=0
o=CiscoSystemsSIP-GW-UserAgent 5596 7982 IN IP4 166.34.245.230
s=SIP Call
t=0 0
Cisco IOS Debug Command Reference - Commands A through D
Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip all</td>
<td>Enables all SIP-related debugging.</td>
</tr>
<tr>
<td>debug ccsip calls</td>
<td>Shows all SIP SPI call tracing.</td>
</tr>
<tr>
<td>debug ccsip error</td>
<td>Shows SIP SPI errors.</td>
</tr>
<tr>
<td>debug ccsip events</td>
<td>Shows all SIP SPI events tracing.</td>
</tr>
<tr>
<td>debug ccsip states</td>
<td>Shows all SIP SPI state tracing.</td>
</tr>
</tbody>
</table>
debug ccsip non-call

To enable non-call-context tracing, use the `debug ccsip non-call` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command or the `undebug ccsip non-call` command.

```
debug ccsip non-call
no debug ccsip non-call
```

Syntax Description
This command has no arguments or keywords.

Command Default
No default behavior or values

Command Modes
Privileged EXEC (#)

Command History
15.4(2)T  This command was introduced.
Cisco IOS XE Release 3.12S  This command was integrated into Cisco IOS XE Release 3.12S.

Usage Guidelines
Use the `debug ccsip non-call` command to enable debugging of non-call-context trace messages (OPTIONS, REGISTER, SUBSCRIBE, and NOTIFY). Usually, SIP debugs contain unrelated debugs that are not useful in debugging. This command will give you the control to enable or disable debugs without any call context; if a message is not part of any call, the debugs are not printed.

Examples
```
Device# debug ccsip non-call
SIP Out-of-Dialog tracing is enabled
```

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip calls</td>
<td>Shows all SIP SPI call tracing.</td>
</tr>
<tr>
<td>debug ccsip error</td>
<td>Shows SIP SPI errors.</td>
</tr>
<tr>
<td>debug ccsip events</td>
<td>Shows all SIP SPI events tracing.</td>
</tr>
<tr>
<td>debug ccsip info</td>
<td>Shows all SIP SPI message tracing.</td>
</tr>
<tr>
<td>debug ccsip states</td>
<td>Shows all SIP SPI state tracing.</td>
</tr>
</tbody>
</table>
To enable diagnostic reporting of authentication, authorization, and accounting (AAA) preauthentication for Session Initiation Protocol (SIP) calls, use the `debug ccsip preauth` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

**debug ccsip preauth**

**no debug ccsip preauth**

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(11)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following example shows debug output for a single SIP call:

```
Router# debug ccsip preauth
SIP Call preauth tracing is enabled
Jan 23 18:43:17.898: Preauth Required
Jan 23 18:43:17.898: In sipSPISendPreauthReq for preauth_id = 86515, ccb = 67AF4E10
Jan 23 18:43:17.898: Entering rpms_proc_print_preauth_req
Jan 23 18:43:17.898: Request = 0
Jan 23 18:43:17.898: Preauth id = 86515
Jan 23 18:43:17.898: EndPt Type = 1
Jan 23 18:43:17.898: EndPt = 192.168.80.70
Jan 23 18:43:17.898: Resource Service = 1
Jan 23 18:43:17.898: Call_origin = answer
Jan 23 18:43:17.898: Call_type = voip
Jan 23 18:43:17.898: Calling_num = 2270001
Jan 23 18:43:17.898: Called_num = 1170001
Jan 23 18:43:17.898: Protocol = 1
Jan 23 18:43:17.898: sipSPISendPreauthReq:Created node with preauth_id = 86515, ccb 67AF4E10, node 6709C280
Jan 23 18:43:17.898: rpms_proc_create_node: Created node with preauth_id = 86515
Jan 23 18:43:17.898: rpms_proc_send_aaa_req:uid got is 466728
Jan 23 18:43:17.902: rpms_proc_preauth_response:Context is for preauth_id 86515, aaa_uid 466728
Jan 23 18:43:17.902: Deleting Tree node for preauth id 86515 uid 466728
Jan 23 18:43:17.902: Deleting Tree node for preauth id 86515 uid 466728
```

The table below describes the significant fields shown in the display.
Table 51: debug ccsip preauth Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Request Type--0 for preauthentication, 1 for disconnect.</td>
</tr>
<tr>
<td>Preauth id</td>
<td>Identifier for the preauthentication request.</td>
</tr>
<tr>
<td>EndPt Type</td>
<td>Call Origin End Point Type--1 for IP address, 2 for Interzone ClearToken (IZCT) value.</td>
</tr>
<tr>
<td>EndPt</td>
<td>Call Origin End Point Value--An IP address or IZCT value.</td>
</tr>
<tr>
<td>Resource Service</td>
<td>Resource Service Type--1 for Reservation, 2 for Query.</td>
</tr>
<tr>
<td>Call_origin</td>
<td>Answer.</td>
</tr>
<tr>
<td>Call_type</td>
<td>Voice over IP (VoIP).</td>
</tr>
<tr>
<td>Calling_num</td>
<td>Calling Party Number (CLID).</td>
</tr>
<tr>
<td>Called_num</td>
<td>Called Party Number (DNIS).</td>
</tr>
<tr>
<td>Protocol</td>
<td>0 for H.323, 1 for SIP.</td>
</tr>
<tr>
<td>function reports</td>
<td>Various identifiers and status reports for executed functions.</td>
</tr>
</tbody>
</table>
**debug ccsip states**

To show all Session Initiation Protocol (SIP) Service Provider Interface (SPI) state tracing, use the `debug ccsip states` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug ccsip states
no debug ccsip states
```

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

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(1)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)XA</td>
<td>Support was added for the Cisco AS5350 and Cisco AS5400 universal gateways.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850 universal gateway.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td>This command was implemented on Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(11)T. Support for the Cisco AS5300 universal access server, Cisco AS5350, Cisco AS5400, and Cisco AS5850 universal gateway is not included in this release.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
This command traces the state machine changes of SIP SPI and displays the state transitions.

**Examples**
The following example shows all SIP SPI state tracing:

```
Router1# debug ccsip states
SIP Call states tracing is enabled
Router1#
*Jan 2 18:34:37.793:0x6220C634:State change from (STATE_NONE, SUBSTATE_NONE) to (STATE_IDLE, SUBSTATE_NONE)
*Jan 2 18:34:37.797:0x6220C634:State change from (STATE_IDLE, SUBSTATE_NONE) to (STATE_IDLE, SUBSTATE_CONNECTING)
```
debug ccsip states

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip all</td>
<td>Enables all SIP-related debugging.</td>
</tr>
<tr>
<td>debug ccsip calls</td>
<td>Shows all SIP SPI call tracing.</td>
</tr>
<tr>
<td>debug ccsip error</td>
<td>Shows SIP SPI errors.</td>
</tr>
<tr>
<td>debug ccsip events</td>
<td>Shows all SIP SPI events tracing.</td>
</tr>
<tr>
<td>debug ccsip info</td>
<td>Shows all SIP SPI message tracing.</td>
</tr>
</tbody>
</table>
debug ccsip transport

To enable tracing of the Session Initiation Protocol (SIP) transport handler and the TCP or User Datagram Protocol (UDP) process, use the `debug ccsip transport` command in privileged EXEC mode. To disable debugging output, use the `no debug ccsip transport` command.

**debug ccsip transport**

**no debug ccsip transport**

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(8)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

12.2 SX

This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.

**Usage Guidelines**

Use the `debug ccsip transport` command to debug issues related to connection and transport usage and to see the flow of the messages being sent or received.

**Examples**

The following is sample output from the `debug ccsip transport` command for a Cisco 3660:

```
Router# debug ccsip transport
.
.
1w1d: //18/8E16980D800A/SIP/Transport/sipSPISendInvite: Sending Invite to the transport layer
1w1d: //18/8E16980D800A/SIP/Transport/sipSPIGetSwitchTransportFlag: Return the Global configuration, Switch Transport is TRUE
1w1d: //18/8E16980D800A/SIP/Transport/sipSPITransportSendMessage: msg=0x64082D50, addr=172.18.194.183, port=5060, sentBy_port=0, is_req=1, transport=1, switch=1, callBack=0x614FAB58
1w1d: //18/8E16980D800A/SIP/Transport/sipSPITransportSendMessage: Proceedable for sending msg immediately
1w1d: //18/8E16980D800A/SIP/Transport/sipTransportLogicSendMsg: switch transport is 1
1w1d: //18/8E16980D800A/SIP/Transport/sipTransportGetInterfaceMtuSize: MTU size for remote address 172.18.194.183 is 500
1w1d: //18/8E16980D800A/SIP/Transport/sipTransportVerifyMsgForMTUThreshold: Interface MTU Size 500, Msg Size 1096
1w1d: //18/8E16980D800A/SIP/Transport/sipTransportLogicSendMsg: Switching msg=0x64082D50 transport UDP->TCP
1w1d: //18/8E16980D800A/SIP/Transport/sipTransportSetAgeingTimer: Aging timer initiated for holder=0x64084058,addr=172.18.194.183
1w1d: //18/8E16980D800A/SIP/Transport/sipCreateConnHolder: Created new holder=0x64084058, addr=172.18.194.183
```
The table below describes the significant fields shown in the display.

**Table 52: debug ccsip transport Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sending Invite to the transport layer</td>
<td>Indicates that the SIP signaling state machine has invoked transport layer operations such as transport arbitration logic and the connection management interface.</td>
</tr>
<tr>
<td>switch transport is 1</td>
<td>Indicates that the gateway has been provisioned to enable the transport switching functionality based on the message size. 1 is true and 0 is false.</td>
</tr>
<tr>
<td>MTU size for remote address</td>
<td>Indicates that the bound outgoing Ethernet interface that sends the message to the given remote address is configured for an MTU size of the indicated value.</td>
</tr>
<tr>
<td>Interface MTU Size 500, Msg Size 1096</td>
<td>Indicates that the size of the message is larger than the size of the MTU; thus transport switching (from UDP to TCP) should be enabled.</td>
</tr>
<tr>
<td>Switching msg=... transport UDP-&gt;TCP</td>
<td>Indicates that transport switching from UDP to TCP is occurring for the handled message because of the large size of the message.</td>
</tr>
<tr>
<td>Aging timer initiated for holder</td>
<td>Indicates that the connection algorithm is started; that is, the counter begins to age out the TCP or UDP connection if inactivity occurs.</td>
</tr>
<tr>
<td>Posting TCP conn create request</td>
<td>Indicates a request for a TCP connection from a lower TCP process.</td>
</tr>
</tbody>
</table>

1w1d: //-1/xxxxxxxxxxxx/SIP/Transport/sipTransportPostRequestConnection: Posting TCP conn create request for addr=172.18.194.183, port=5060, context=0x64128D5C
1w1d: //-1/xxxxxxxxxxxx/SIP/Transport/sipTransportSetConnWaitTimer: Wait timer set for connection=0x64129BF4, addr=172.18.194.183, port=5060
1w1d: //-1/xxxxxxxxxxxx/SIP/Transport/sipCreateConnInstance: Created new initiated conn=0x64129BF4, connid=-1, addr=172.18.194.183, port=5060, transport=tcp
1w1d: //-1/xxxxxxxxxxxx/SIP/Transport/sipConnectionManagerProcessConnCreated: gConnTab=0x64128D5C, addr=172.18.194.183, port=5060, connid=1, transport=tcp
1w1d: //-1/xxxxxxxxxxxx/SIP/Transport/sipInstanceHandleConnectionCreated: Moving connection=0x64129BF4, connid=1 state to pending
1w1d: //-1/xxxxxxxxxxxx/SIP/Transport/sipTransportProcessNWConnectionCreated: context=0x64128D5C
1w1d: //-1/xxxxxxxxxxxx/SIP/Transport/sipConnectionManagerProcessConnCreated: gConnTab=0x64128D5C, addr=172.18.194.183, port=5060, connid=1, transport=tcp
1w1d: //-1/xxxxxxxxxxxx/SIP/Transport/sipTransportPostSendMessage: Posting send for msg=0x64082D50, addr=172.18.194.183, port=5060, connId=1 for TCP
Indicates all the transport related attributes that the SIP signaling state machine originally gives to the transport layer to send out the message. The attributes are:

- transport: 1 for UDP; 2 for TCP.
- switch (switching transport enabled or disabled for large messages): 1 for enabled; 0 for disabled.

Posting send for msg=0x64082D50, addr=...for TCP

Indicates that all transport and connection related operations are complete. The message is sent out on the network targeted to the given address, port, and transport.

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip all</td>
<td>Enables all SIP-related debugging.</td>
</tr>
<tr>
<td>debug ccsip info</td>
<td>Enables tracing of general SIP SPI information.</td>
</tr>
<tr>
<td>transport switch</td>
<td>Enables switching between UDP and TCP transport mechanisms globally for large SIP messages.</td>
</tr>
<tr>
<td>voice-class sip transport switch</td>
<td>Enables switching between UDP and TCP transport mechanisms for large SIP messages for a specific dial peer.</td>
</tr>
</tbody>
</table>
debug ccswvoice vo-debug

To display detailed debugging information related to ccswvoice function calls during call setup and teardown, use the `debug ccswvoice vo-debug` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
display ccswvoice vo-debug
no display ccswvoice vo-debug
```

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

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(1)MA</td>
<td>This command was introduced on the Cisco MC3810 networking device.</td>
</tr>
<tr>
<td>12.0(7)XK</td>
<td>This command was implemented on the Cisco 3600 series router.</td>
</tr>
<tr>
<td>12.1(2)T</td>
<td>This command was integrated into Cisco IOS Release 12.1(2)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
Use this command when attempting to troubleshoot a Vo call that uses the "cisco-switched" session protocol. This command provides the same information as the `debug ccswvoice vo-session` command, but includes additional debugging information relating to the calls.

**Examples**
The following shows sample output from the `debug ccswvoice vo-debug` command:

```
Router# debug ccswvoice vo-debug
2w2d: ccswvoice: callID 529927 pvcid -1 cid -1 state NULL event O/G SETUP
2w2d: ccswvoice_out_callinit_setup: callID 529927 using pvcid 1 cid 15
2w2d: ccswvoice: callID 529927 pvcid 1 cid 15 state O/G INIT event I/C PROC
2w2d: ccswvoice: callID 529927 pvcid 1 cid 15 state O/G PROC event I/C ALERTccfrf11_caps_ind: codec(preferred) = 1
2w2d: ccswvoice: callID 529927 pvcid 1 cid 15 state O/G ALERT event I/C CONN
2w2d: ccswvoice: callID 529927 pvcid 1 cid 15 state O/G ALERT event I/C RELCOMP
2w2d: ccswvoice_bridge_drop: dropping bridge calls src 529927 dst 529926 pvcid 1 cid 15 state ACTIVE
2w2d: ccswvoice: callID 529927 pvcid 1 cid 15 state ACTIVE event O/G REL
2w2d: ccswvoice: callID 529927 pvcid 1 cid 15 state RELEASE event I/C RELCOMP
2w2d: ccswvoice_store_call_history_entry: cause=10 tcause=10 cause_text=normal call clearing.
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccswvoice vo-session</td>
<td>Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>debug backhaul-session-manager session through debug channel packets</code></td>
<td>Triggers debug information for backhaul session manager session through debug channel packets.</td>
</tr>
<tr>
<td><code>debug ccswvoice vo-debug</code></td>
<td>Triggers debug information for Cisco CallManager with Video session manager (CCSW) voice-related operations.</td>
</tr>
</tbody>
</table>
**debug ccswvoice vofr-debug**

To display the ccswvoice function calls during call setup and teardown, use the `debug ccswvoice vofr-debug` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug ccswvoice vofr-debug
no debug ccswvoice vofr-debug
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(3)XG</td>
<td>This command was introduced on the Cisco 2600 and Cisco 3600 series routers.</td>
</tr>
<tr>
<td>12.0(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.0(4)T.</td>
</tr>
<tr>
<td>12.0(7)XK</td>
<td>This command was implemented on the Cisco MC3810 networking device.</td>
</tr>
<tr>
<td>12.1(2)T</td>
<td>This command was integrated into Cisco IOS Release 12.1(2)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command when troubleshooting a VoFR call that uses the "cisco-switched" session protocol. This command provides the same information as the `debug ccswvoice vofr-session` command, but includes additional debugging information relating to the calls.

**Examples**

The following shows sample output from the `debug ccswvoice vofr-debug` command:

```
Router# debug ccswvoice vofr-debug
CALL TEARDOWN:
3640_vofr(config-voiceport)#
*Mar 1 03:02:08.719:ccswvofr_bridge_drop:dropping bridge calls src 17 dst 16 dlci 100 cid 9 state ACTIVE
*Mar 1 03:02:08.727:ccswvofr:callID 17 dlci 100 cid 9 state RELEASE event I/C RELCOMP
*Mar 1 03:02:08.735:ccswvofr_store_call_history_entry:cause=22 tcause=22 cause_text=no circuit.
3640_vofr(config-voiceport)#

CALL SETUP (outgoing):
*Mar 1 03:02:08.735:ccswvofr:callID 23 dlci -1 cid -1 state NULL event O/G SETUP
*Mar 1 03:02:08.735:ccswvofr_out_callinit_setup:callID 23 using dlci 100 cid 10
*Mar 1 03:02:08.735:ccswvofr:callID 23 dlci 100 cid 10 state O/G INIT event I/C PROC
*Mar 1 03:02:08.735:ccswvofr_store_call_history_entry:cause=22 tcause=22
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cch323</td>
<td>Displays the ccf11 function calls during call setup and teardown.</td>
</tr>
<tr>
<td>debug ccsw voice vo-debug</td>
<td>Displays the ccswvoice function calls during call setup and teardown.</td>
</tr>
<tr>
<td>debug vtsp session</td>
<td>Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.</td>
</tr>
</tbody>
</table>
debug ccsvoice vofr-session

To display the ccsvoice function calls during call setup and teardown, use the `debug ccsvoice vofr-session` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

`debug ccsvoice vofr-session`

`no debug ccsvoice vofr-session`

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.03XG</td>
<td>This command was introduced on the Cisco 2600 and Cisco 3600 series routers.</td>
</tr>
<tr>
<td>12.04T</td>
<td>This command was integrated into Cisco IOS Release 12.04T.</td>
</tr>
<tr>
<td>12.07XK</td>
<td>This command was implemented on the Cisco MC3810 networking device.</td>
</tr>
<tr>
<td>12.12T</td>
<td>This command was integrated into Cisco IOS Release 12.12T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command to show the state transitions of the cisco-switched-vofr state machine as a call is processed, and when attempting to troubleshoot a VoFR call that uses the "cisco-switched" session protocol.

**Examples**

The following shows sample output from the `debug ccsvoice vofr-session` command:

```
Router# debug ccsvoice vofr-session
CALL TEARDOWN:
3640_vofr(config-voiceport)#
*Mar 1 02:58:13.203:ccswvofr:callID 14 dlci 100 cid 8 state ACTIVE event O/G REL
*Mar 1 02:58:13.215:ccswvofr:callID 14 dlci 100 cid 8 state RELEASE event I/C RELCOMP
3640_vofr(config-voiceport)#
CALL SETUP (outgoing):
*Mar 1 02:59:46.551:ccswvofr:callID 17 dlci -1 dlci -1 state NULL event O/G SETUP
*Mar 1 02:59:46.559:ccswvofr:callID 17 dlci 100 cid 9 state O/G INIT event I/C PROC
*Mar 1 02:59:46.567:ccswvofr:callID 17 dlci 100 cid 9 state O/G PROC event I/C CONN
3640_vofr(config-voiceport)#
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug cch323</code></td>
<td>Displays the ccfrf11 function calls during call setup and teardown.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>debug call rsvp-sync events</td>
<td>Displays events that occur during RSVP setup.</td>
</tr>
<tr>
<td>debug vtsp session</td>
<td>Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.</td>
</tr>
</tbody>
</table>
**debug ccswvoice vo-session**

To display the first 10 bytes (including header) of selected VoFR subframes for the interface, use the `debug ccswvoice vo-session` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
default ccswvoice vo-session
no default ccswvoice vo-session
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(MA)</td>
<td>This command was introduced on the Cisco MC3810 networking device.</td>
</tr>
<tr>
<td>12.0(XK)</td>
<td>This command was implemented on the Cisco 3600 series router.</td>
</tr>
<tr>
<td>12.1(T)</td>
<td>This command was integrated into Cisco IOS Release 12.1(T).</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command to show the state transitions of the cisco-switched-vo state machine as a call is processed. This command should be used when attempting to troubleshoot a Vo call that uses the "cisco-switched" session protocol.

**Examples**

The following shows sample output from the `debug ccswvoice vo-session` command:

```
Router# debug ccswvoice vo-session
2w2d: ccswvoice: callID 529919 pvcid -1 cid -1 state NULL event O/G SETUP
2w2d: ccswvoice: callID 529919 pvcid 1 cid 11 state O/G INIT event I/C PROC
2w2d: ccswvoice: callID 529919 pvcid 1 cid 11 state O/G PROC event I/C ALERT
2w2d: ccswvoice: callID 529919 pvcid 1 cid 11 state O/G ALERT event I/C CONN
2w2d: ccswvoice: callID 529919 pvcid 1 cid 11 state ACTIVE event O/G REL
2w2d: ccswvoice: callID 529919 pvcid 1 cid 11 state RELEASE event I/C RELCOMP
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug ccswvoice vo-debug</code></td>
<td>Displays detailed debugging information related to ccswvoice function calls during call setup and teardown.</td>
</tr>
</tbody>
</table>
debug cdapi

To display information about the Call Distributor Application Programming Interface (CDAPI), use the `debug cdapi` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
default cdapi {detail|events}
no default cdapi {detail|events}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>detail</code></td>
<td>Displays when applications register or become unregistered with CDAPI, when calls are added or deleted from the CDAPI routing table, and when CDAPI messages are created and freed.</td>
</tr>
<tr>
<td><code>events</code></td>
<td>Displays the events passing between CDAPI and an application or signalling stack.</td>
</tr>
</tbody>
</table>

### Command Default

Debugging output is disabled.

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(6)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.1(5)XM2</td>
<td>This command was implemented on the Cisco AS5350 and Cisco AS5400.</td>
</tr>
<tr>
<td>12.3(2)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(2)T. This command was enhanced to show V.110 call types.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was enhanced to show V.120 call types.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

### Usage Guidelines

The `detail` keyword is useful for determining if messages are being lost (or not freed). It is also useful for determining the size of the raw messages passed between CDAPI and other applications to ensure that the correct number of bytes is being passed.

The `events` keyword is useful for determining if certain ISDN messages are not being received by an application and if calls are not being directed to an application.

The following bandwidths are supported:
56 kbps
64 kbps

Examples

The following Media Gateway Control Protocol (MGCP) packet received example shows V.110 call debugging output for the **debug cdapi** command. In this example, the modem is not yet in STEADY_STATE.

```
Router# debug cdapi detail
Sep 26 19:12:25.327:MGCP Packet received from 10.0.44.109:2427-
CRCX 6318 s7/ds1-0/24 MGCP 1.0
C:111
M:nas/data
L:lib:64, nas/bt:v.110, nas/cdn:234567
R:nas/au, nas/ax,nas/of, nas/crq
X:101
Sep 26 19:12:25.327:CDAPI:cdapi_create_msg():CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327:CDAPI Se7/1:23:cdapi_add_entry_callRoutingTbl() -
Sep 26 19:12:25.327: Added entry for call 0x7017 for application CSM
Sep 26 19:12:25.331:CDAPI:cdapi_free_msg():CDAPI Pool Count:958,
router# Raw Length = 0
Sep 26 19:12:25.331:CDAPI:cdapi_free_msg():CDAPI Pool Count:959
Sep 26 19:12:25.331:CDAPI:cdapi_free_msg():CDAPI Pool Count:959
Sep 26 19:12:25.331:CDAPI:cdapi_free_msg():CDAPI Pool Count:959
Sep 26 19:12:25.331:CDAPI:cdapi_free_msg():CDAPI Pool Count:959
Sep 26 19:12:25.331:CDAPI:cdapi_free_msg():CDAPI Pool Count:959
Sep 26 19:12:25.331:send mgcp msg, MGCP Packet sent to 10.0.44.109:2427 ---->
I:64524608
Sep 26 19:12:25.339:send mgcp msg, MGCP Packet sent to 10.0.44.109:2427 ---->
```

The following partial example shows V.120 call debugging output for the **debug cdapi** command:

```
Router# debug cdapi detail
May 14 19:12:25.327:MGCP Packet received from 10.0.44.109:2427-
CRCX 6318 s7/ds1-0/24 MGCP 1.0
C:111
M:nas/data
L:lib:64, nas/bt:v.120, nas/cdn:234567
R:nas/au, nas/ax,nas/of, nas/crq
X:101
May 14 19:12:25.327:CDAPI:cdapi_create_msg():CDAPI Pool Count:959, Raw Length = 0
May 14 19:12:25.327:CDAPI Se7/1:23:cdapi_add_entry_callRoutingTbl() -
May 14 19:12:25.327: Added entry for call 0x7017 for application CSM
May 14 19:12:25.331:CDAPI:cdapi_create_msg():CDAPI Pool Count:958,
router# Raw Length = 0
```

Cisco IOS Debug Command Reference - Commands A through D
The following MGCP packet received example shows V.120 call debugging output for the `debug cdapi events` command:

```
Router# debug cdapi events
Sep 26 19:14:39.027: MGCP Packet received from 10.0.44.109:2427-
CRCX 6322 s7/ds1-0/24 MGCP 1.0
C:111
Mnas/data
I:bt:120, nas/cdn:234567
R:nas/au, nas/ax,nas/of, nas/crq
X:101
Sep 26 19:14:39.027: Se7/0:23 CDAPI:TX -> CDAPI_MSG_CONNECT_IND to CSM call = 0x7017
Sep 26 19:14:39.027: From Appl/Stack = XCSP
Sep 26 19:14:39.027: Call Type = V.120
Sep 26 19:14:39.027: B Channel = 23
Sep 26 19:14:39.027: dsId = 0
Sep 26 19:14:39.027: IDb = 0
Sep 26 19:14:39.027: BChanIdb = 64519A14
Sep 26 19:14:39.027: Handle = 63CB8DF4
Sep 26 19:14:39.027: RPA = 638B506C
Sep 26 19:14:39.027: Cause = 0
Sep 26 19:14:39.027: ApplSpecData = 0
Sep 26 19:14:39.027: Overlap = 0
Sep 26 19:14:39.027: Calling Party Number =
Sep 26 19:14:39.027: Called Party Number = 234567
Sep 26 19:14:39.027: Overlap = 0
Sep 26 19:14:39.027: Se7/0:23 CDAPI:TX -> CDAPI_MSG_CONNECT_RESP to XCSP call = 0x7017
Sep 26 19:14:39.027: From Appl
router#26 19:14:39.027: Stack = CSM
Sep 26 19:14:39.027: Call Type = MODEM
Sep 26 19:14:39.027: B Channel = 23
Sep 26 19:14:39.027: dsId = 0
Sep 26 19:14:39.027: IDb = 0
Sep 26 19:14:39.027: BChanIdb = 64519A14
Sep 26 19:14:39.027: Handle = 63CB8DF4
Sep 26 19:14:39.027: RPA = 0
Sep 26 19:14:39.027: Cause = 0
Sep 26 19:14:39.027: ApplSpecData = 0
Sep 26 19:14:39.027: Overlap = 0
Sep 26 19:14:39.031: Se7/0:23 CDAPI:TX -> CDAPI_MSG_CONN_ACT_REQ to XCSP call = 0x7017
Sep 26 19:14:39.031: From Appl/Stack = CSM
Sep 26 19:14:39.031: Call Type = MODEM
Sep 26 19:14:39.031: B Channel = 23
Sep 26 19:14:39.031: dsId = 0
Sep 26 19:14:39.031: IDb = 0
Sep 26 19:14:39.031: BChanIdb = 64519A14
Sep 26 19:14:39.031: Handle = 63CB8DF4
Sep 26 19:14:39.031: RPA = 0
Sep 26 19:14:39.031: Cause = 0
Sep 26 19:14:39.031: ApplSpecData = 0
Sep 26 19:14:39.031: Overlap = 0
```

`debug cdapi events` command outputs detailed information about MGCP packets received and sent, including call types, channel information, and other relevant details. This can be useful for troubleshooting and monitoring call flows in a network environment.
The table below describes the significant fields shown in the displays.

Table 53: debug cdapi Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L:b:64, nas/bt</td>
<td>The bearer type parameter includes v.110 and v.120 for V.110 and V.120 calls.</td>
</tr>
<tr>
<td>Call Type</td>
<td>Call types are V.110, V.120, and modem.</td>
</tr>
</tbody>
</table>

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug mgcp packet</td>
<td>Displays the MGCP signaling message received and sent to the called agent.</td>
</tr>
<tr>
<td>debug voip rawmsg</td>
<td>Displays the raw message owner, length, and pointer.</td>
</tr>
</tbody>
</table>
**debug cdma pdsn a10 ahdlc**

To display debug messages for Asynchronous High-Level Data Link Control (AHDLC), use the `debug cdma pdsn a10 ahdlc` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cdma pdsn a10 ahdlc [errors| events]
no debug cdma pdsn a10 ahdlc [errors| events]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>errors</td>
<td>(Optional) Displays details of AHDLC packets in error.</td>
</tr>
<tr>
<td>events</td>
<td>(Optional) Displays AHDLC events.</td>
</tr>
</tbody>
</table>

**Command Default**

If the command is entered without any optional keywords, all of the types of debug information are enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(2)XC</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(8)BY</td>
<td>Keywords were made optional.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug cdma pdsn a10 ahdlc errors` command:

```
Router# debug cdma pdsn a10 ahdlc errors
ahdlc error packet display debugging is on
Router# debug cdma pdsn a10 ahdlc events
ahdlc events display debugging is on
Router#
*Jan 1 00:18:30:*LINK-3-UPDOWN:Interface Virtual-Access1, changed state to up
*Jan 1 00:18:30:*OPEN AHDLC*****
*Jan 1 00:18:30: ahdlc_mgr_channel_create
*Jan 1 00:18:30: ahdlc_mgr_allocate_available_channel:
*Jan 1 00:18:30:ahdlc:tell h/w open channel 9 from engine 0
```


**debug cdma pdsn a10 gre**

To display debug messages for A10 Generic Routing Encapsulation (GRE) interface errors, events, and packets, use the `debug cdma pdsn a10 gre` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```plaintext
debug cdma pdsn a10 gre [errors|events|packets] [tunnel-key key]
no debug cdma pdsn a10 gre [errors|events|packets]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>errors</td>
<td>(Optional) Displays A10 GRE errors.</td>
</tr>
<tr>
<td>events</td>
<td>(Optional) Displays A10 GRE events.</td>
</tr>
<tr>
<td>packets</td>
<td>(Optional) Displays transmitted or received A10 GRE packets.</td>
</tr>
<tr>
<td>tunnel-key key</td>
<td>(Optional) Specifies the GRE key.</td>
</tr>
</tbody>
</table>

**Command Default**

If the command is entered without any optional keywords, all of the types of debug information are enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(8)BY</td>
<td>The <code>tunnel-key</code> keyword was added and the existing keywords were made optional.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug cdma pdsn a10 gre events tunnel-key 1` command:

```plaintext
Router# debug cdma pdsn a10 gre events tunnel-key 1
Router# show debug
CDMA: CDMA PDSN A10 GRE events debugging is on for tunnel key 1
PDSN#
*Mar 1 04:00:57.847:CDMA-GRE:CDMA-Ix1 (GRE/CDMA) created with src 5.0.0.2 dst 0.0.0.0
*Mar 1 04:00:57.847:CDMA-GRE:(in) found session 5.0.0.2-4.0.0.1-1
*Mar 1 04:00:59.863:CDMA-GRE:(in) found session 5.0.0.2-4.0.0.1-1
*Mar 1 04:00:59.863:CDMA-GRE:(in) found session 5.0.0.2-4.0.0.1-1
*Mar 1 04:01:01.879:CDMA-GRE:(in) found session 5.0.0.2-4.0.0.1-1
```
debug cdma pdsn a10 gre

*Mar 1 04:01:01.879:CDMA-GRE:(in) found session 5.0.0.2-4.0.0.1-1
*Mar 1 04:01:03.899:CDMA-GRE:(in) found session 5.0.0.2-4.0.0.1-1
*Mar 1 04:01:03.899:CDMA-GRE:(in) found session 5.0.0.2-4.0.0.1-1
debug cdma pdsn a10 ppp

To display debug messages for A10 Point-to-Point protocol (PPP) interface errors, events, and packets, use the `debug cdma pdsn a10 ppp` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
default cdma pdsn a10 ppp [errors| events| packets]
no debug cdma pdsn a10 ppp [errors| events| packets]
```

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>errors</td>
<td>(Optional) Displays A10 PPP errors.</td>
</tr>
<tr>
<td>events</td>
<td>(Optional) Displays A10 PPP events.</td>
</tr>
<tr>
<td>packets</td>
<td>(Optional) Displays transmitted or received A10 PPP packets.</td>
</tr>
</tbody>
</table>

**Command Default**

If the command is entered without any optional keywords, all of the types of debug information are enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(8)BY</td>
<td>Keywords were made optional.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug cdma pdsn a10 ppp` command:

```
Router# debug cdma pdsn a10 ppp errors
CDMA PDSN A10 errors debugging is on
Router# debug cdma pdsn a10 ppp events
CDMA PDSN A10 events debugging is on
Router# debug cdma pdsn a10 ppp packets
CDMA PDSN A10 packet debugging is on
Router# show debug
*Jan 1 00:13:09:CDMA-PPP:create va tunnel=CDMA-Ix1 virtual-template template=Virtual-Template2 ip_enabled=1
*Jan 1 00:13:09:CDMA-PPP:create va va=Virtual-Access1
*Jan 1 00:13:09:CDMA-PPP:clone va=Virtual-Access1 subif_state=1 hwidb->state=0
*Jan 1 00:13:09: linestate=1 ppp_lineup=0
```
debug backhaul-session-manager session through debug channel packets

*Jan 1 00:13:09:%LINK-3-UPDOWN:Interface Virtual-Access1, changed state to up
*Jan 1 00:13:09:CDMA-PPP:clone va=Virtual-Access1 subif_state=1 hwidb->state=4
*Jan 1 00:13:09:linestate=0 ppp_lineup=0
*Jan 1 00:13:09:*****OPEN AHDLC*****
debug cdma pdsn a11

To display debug messages for A11 interface errors, events, and packets, use the `debug cdma pdsn a11` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cdma pdsn a11 [errors|events|packets] [mnid]
no debug cdma pdsn a11 [errors|events|packets]
```

### Syntax Description

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>errors</td>
<td>(Optional) Displays A11 protocol errors.</td>
</tr>
<tr>
<td>events</td>
<td>(Optional) Displays A11 events.</td>
</tr>
<tr>
<td>packets</td>
<td>(Optional) Displays transmitted or received packets.</td>
</tr>
<tr>
<td>mnid</td>
<td>(Optional) Specifies the ID of the mobile station.</td>
</tr>
</tbody>
</table>

### Command Default

If the command is entered without any optional keywords, all of the types of debug information are enabled.

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(8)BY</td>
<td>The <code>mnid</code> argument was added and the existing keywords were made optional.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

### Examples

The following is sample output from the `debug cdma pdsn a11` commands:

```
Router# debug cdma pdsn all errors
CDMA PDSN All errors debugging is on
Router# show debug
1d21h:CDMA-RP:(in) rpmsgs, code=1, status=0
1d21h:CDMA-RP:(enqueue req) type=1 homeagent=5.0.0.2 coaddr=4.0.0.1
1d21h: id=0xBEF750F0-0xBA53E0F lifetime=65535
1d21h:CDMA-RP:len=8, 00-00-00-00-00-00-00-F1 convert to 00000000000001 (14 digits), type=IMSI
1d21h:CDMA-RP:(req) process_rp_req, homeagent=5.0.0.2 coaddr=4.0.0.1
1d21h: lifetime=65535 id=BEF750F0-BA53E0F
imsi=0000000000001
1d21h:CDMA-RP:(req) rp_req_create, 5.0.0.2-4.0.0.1-1 imsi=00000000000001
1d21h:CDMA-RP:(out) rp_reply session=5.0.0.2-4.0.0.1-1, lifetime=65535
1d21h:CDMA-RP:(out) setup_rp_out_msg, ha=5.0.0.2 coa=4.0.0.1 key=1
```
Router# debug cdma pdsn all packets events
Router# show debug
CDMA:
   CDMA PDSN A11 packet debugging is on for m nid 000000000000001
   CDMA PDSN A11 all events debugging is on for m nid 000000000000001
Router# *Mar 1 03:15:32.507:CDMA-RP:len=8, 01-00-00-00-00-00-00-10 convert to 000000000000001 (15 digits), type=IMSI
*Mar 1 03:15:32.511:CDMA-RP:extension type=38, len=0
*Mar 1 03:15:32.511:CDMA-RP:extension type=38, len=0
*Mar 1 03:15:32.511:CDMA-RP:extension type=38, len=0
*Mar 1 03:15:32.511:CDMA-RP:extension type=32, len=20
*Mar 1 03:15:32.511: 00 00 01 00 EE 1F 43 0A 7D F9 36 29 C2 BA 28
*Mar 1 03:15:32.511: 5A 64 D5 9C
*Mar 1 03:15:32.511:CDMA-RP:(req) process_rp_req, homeagent=5.0.0.2 coaddr=4.0.0.1
*Mar 1 03:15:32.511: lifetime=1800 id=AF3BFE55-69A109D IMSI=000000000000001
*Mar 1 03:15:32.511:CDMA-RP:(req) rp_req_create, ha=5.0.0.2, coa=4.0.0.1, key=1
IMSI=000000000000001
*Mar 1 03:15:32.511:CDMA-RP:(req) rp_req_lifetime_zero 5.0.0.2-4.0.0.1-1, lifetime=1800
*Mar 1 03:15:32.511:CDMA-RP:(req) setRp Out message, ha=5.0.0.2 coa=4.0.0.1 key=1
*Mar 1 03:15:38.555:CDMA-RP:(out) rp_reply session=5.0.0.2-4.0.0.1-1, lifetime=1800
*Mar 1 03:15:38.555:CDMA-RP:(out) Setup RP out message, ha=5.0.0.2 coa=4.0.0.1 key=1
Router# debug cdma pdsn a11 packet mnid 000000000000001
Router# show debug
CDMA:
   CDMA PDSN A11 packet debugging is on for m nid 000000000000001
   CDMA PDSN A11 all events debugging is on for m nid 000000000000001
Router# *Mar 1 03:15:54.755:CDMA-RP:len=8, 01-00-00-00-00-00-00-10 convert to 000000000000001 (15 digits), type=IMSI
*Mar 1 03:15:54.755:CDMA-RP:extension type=38, len=0
*Mar 1 03:15:54.755: 00 00 01 00 EA 9C C6 4C BA B9 F9 B6 DD C4 19 76
*Mar 1 03:15:54.755:CDMA-RP:(req) process_rp_req, homeagent=5.0.0.2 coaddr=4.0.0.1
*Mar 1 03:15:54.755: lifetime=0 id=AF3BFE6B-4616E475 IMSI=000000000000001
*Mar 1 03:15:54.755:CDMA-RP:(req) rp_req_lifetime_zero 5.0.0.2-4.0.0.1-1, lifetime=0
*Mar 1 03:15:54.755:CDMA-RP:(req) setRp Out message, ha=5.0.0.2 coa=4.0.0.1 key=1
Router# close the session
Router# debug cdma pdsn all packet mnid 000000000000001
Router# show debug
CDMA:
   CDMA PDSN A11 packet debugging is on for m nid 000000000000001
Router# *Mar 1 03:09:34.339:CDMA-RP:len=8, 01-00-00-00-00-00-00-10 convert to 000000000000001 (15 digits), type=IMSI
*Mar 1 03:09:34.339:CDMA-RP:extension type=38, len=0
*Mar 1 03:09:34.339:CDMA-RP:extension type=38, len=0
*Mar 1 03:09:34.339:CDMA-RP:extension type=38, len=0
*Mar 1 03:09:34.339:CDMA-RP:extension type=32, len=20
*Mar 1 03:09:34.339: 51 5A 56 45
*Mar 1 03:09:34.339:CDMA-RP:(req) process_rp_req, homeagent=5.0.0.2 coaddr=4.0.0.1
*Mar 1 03:09:34.339: lifetime=1800 id=AF3BFE6B-4616E475 IMSI=000000000000001
*Mar 1 03:09:34.339:CDMA-RP:(req) rp_req_create, ha=5.0.0.2, coa=4.0.0.1, key=1
IMSI=000000000000001
*Mar 1 03:09:34.339:CDMA-RP:(req) rp_req_lifetime_zero 5.0.0.2-4.0.0.1-1, lifetime=0
*Mar 1 03:09:34.339:CDMA-RP:(req) setRp Out message, ha=5.0.0.2 coa=4.0.0.1 key=1
Router# close the session
Router# debug cdma pdsn all packet mnid 000000000000001
Router# show debug
CDMA:
   CDMA PDSN A11 packet debugging is on for m nid 000000000000001
Router#
debug backhaul-session-manager session through debug channel packets

debug cdma pdsn a11
debug cdma pdsn accounting

To display debug messages for accounting events, use the `debug cdma pdsn accounting` command in privileged EXEC mode. To disable debugging output, use the `no form` of this command.

```
Router# debug cdma pdsn accounting
CDMA PDSN accounting debugging is on
```

```
Router# debug cdma pdsn accounting
no cdma pdsn accounting
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug cdma pdsn accounting` command:

```
Router# debug cdma pdsn accounting
*Jan 1 00:15:32:CDMA/ACCT:NULL vaccess in session_start
*Jan 1 00:15:32:CDMA/ACCT: Current Attribute type:0x[1A] len:[9]
*Jan 1 00:15:32:CDMA/ACCT: VSA Vid:5535 type:[44] len:[3] 01 Processing Y1
*Jan 1 00:15:32:CDMA/ACCT: Setup airlink record received
*Jan 1 00:15:32:CDMA/ACCT: Current Attribute type:0x[1A] len:[12]
*Jan 1 00:15:32:CDMA/ACCT: VSA Vid:5535 type:[41] len:[6] 00 00 00 02 CDMA/ACCT: Processing Y2
*Jan 1 00:15:32:CDMA/ACCT: Current Attribute type:0x[1A] len:[9]
*Jan 1 00:15:32:CDMA/ACCT: Current Attribute type:0x[1A] len:[12]
*Jan 1 00:15:32:CDMA/ACCT: Current Attribute type:0x[1A] len:[14]
*Jan 1 00:15:32:CDMA/ACCT: VSA Vid:5535 type:[10] len:[8] 00 00 04 04 04 05 Processing D4
*Jan 1 00:15:32:CDMA/ACCT: Current Attribute type:0x[1A] len:[9]
*Jan 1 00:15:32:CDMA/ACCT: VSA Vid:5535 type:[44] len:[3] 02 Processing Y1
*Jan 1 00:15:32:CDMA/ACCT: Start airlink record received
*Jan 1 00:15:32:CDMA/ACCT: Current Attribute type:0x[1A] len:[12]
*Jan 1 00:15:32:CDMA/ACCT: VSA Vid:5535 type:[41] len:[6] 00 00 00 02 CDMA/ACCT: Processing Y2
*Jan 1 00:15:32:CDMA/ACCT: Current Attribute type:0x[1A] len:[9]
*Jan 1 00:15:32:CDMA/ACCT: VSA Vid:5535 type:[44] len:[3] 02 Processing Y1
*Jan 1 00:15:32:CDMA/ACCT: Start airlink record received
*Jan 1 00:15:32:CDMA/ACCT: Current Attribute type:0x[1A] len:[10]
*Jan 1 00:15:32:CDMA/ACCT: VSA Vid:5535 type:[11] len:[4] 00 02 Processing E1
```
debug backhaul-session-manager session through debug channel packets
debag cdma pdsn accounting

*Jan 1 00:15:32:CDMA/ACCT:  Current Attribute type:0x[1A] len:[10]
*Jan 1 00:15:32:CDMA/ACCT:  VSA Vid:5535 type:[12] len:[4] 00 F1   Processing F1
debug cdma pdsn accounting flow

To display debug messages for accounting flow, use the **debug cdma pdsn accounting flow** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug cdma pdsn accounting flow
no debug cdma pdsn accounting flow
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(2)XC</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the **debug cdma pdsn accounting flow** command:

```
Router# debug cdma pdsn accounting flow
CDMA PDSN flow based accounting debugging is on
pdsn-6500#
01:59:40:CDMA-SM:cdma_pdsn_flow_acct_upstream sess id 1 flow type 0 bytes 100 addr 20.20.20.1
01:59:40:CDMA-SM:cdma_pdsn_flow_acct_downstream sess id 1 flow type 0 bytes 100 addr 20.20.20.1
```
**debug cdma pdsn accounting time-of-day**

To display the timer value, use the `debug cdma pdsn accounting time-of-day` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

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

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug cdma pdsn accounting time-of-day` command:

```
Router# debug cdma pdsn accounting time-of-day
CDMA PDSN accounting time-of-day debugging is on
Feb 15 19:13:24.194:%SYS-5-CONFIG_I:Configured from console by console
Router#
Feb 15 19:13:45.635:CDMA-TOD:Timer expired...Rearrming timer
Feb 15 19:13:45.635:CDMA-TOD:Gathering session info
Feb 15 19:13:45.635:CDMA-TOD:Found 0 sessions
```
**debug cdma pdsn cluster**

To display the error messages, event messages, and packets received, use the `debug cdma pdsn cluster` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cdma pdsn cluster message [error|events|packets] redundancy [error|events|packets]
no debug cdma pdsn cluster message [error|events|packets] redundancy [error|events|packets]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>Displays cluster messages for errors, events and packets received.</td>
</tr>
<tr>
<td>redundancy</td>
<td>Displays redundancy information for errors, events, and sent or received packets.</td>
</tr>
<tr>
<td>error</td>
<td>Displays either cluster or redundancy error messages.</td>
</tr>
<tr>
<td>events</td>
<td>Displays either all cluster or all redundancy events.</td>
</tr>
<tr>
<td>packets</td>
<td>Displays all transmitted or received cluster or redundancy packets.</td>
</tr>
</tbody>
</table>

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

This debug is *only* allowed on PDSN c6-mz images, and helps to monitor prepaid information.

**Examples**

The following is sample output from the `debug cdma pdsn cluster` command:

```
Router# debug cdma pdsn cluster ?
message  Debug PDSN cluster controller messages
redundancy  Debug PDSN cluster controller redundancy
```
**debug cdma pdsn ipv6**

To display IPV6 error or event messages, use the `debug cdma pdsn IPV6` command in privileged EXEC mode. To disable debug messages, use the `no` form of this command.

```
ddebug cdma pdsn ipv6
do debug cdma pdsn ipv6
```

**Syntax Description**

There are no arguments or keywords for this command.

**Command Default**

No default behavior or values.

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(14)YX</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(11)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The following example illustrates the `debug cdmapdsnipv6` command:

```
Router# debug cdma pdsn ipv6
```
**debug cdma pdsn prepaid**

To display debug messages about prepaid flow, use the `debug cdma pdsn prepaid` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cdma pdsn prepaid
no debug cdma pdsn prepaid
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(8)BY</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

This debug is *only* allowed on PDSN c6-mz images, and helps to monitor prepaid information.

**Examples**

The following is sample output from the `debug cdma pdsn prepaid` command:

```
Router# debug cdma pdsn prepaid
*Mar 1 00:09:38.391: CDMA-PREPAID: Initialized the authorization request
*Mar 1 00:09:38.391: CDMA-PREPAID: Added username into A-V list
*Mar 1 00:09:38.391: CDMA-PREPAID: Added CLID into A-V list
*Mar 1 00:09:38.391: CDMA-PREPAID: Added session id for prepaid
*Mar 1 00:09:38.391: CDMA-PREPAID: Added correlation id into A-V list
*Mar 1 00:09:38.391: CDMA-PREPAID: Added auth reason for prepaid into A-V list
*Mar 1 00:09:38.391: CDMA-PREPAID: Added USER_ID for prepaid
*Mar 1 00:09:38.391: CDMA-PREPAID: Added service id for prepaid
*Mar 1 00:09:38.391: CDMA-PREPAID: Built prepaid VSAs
*Mar 1 00:09:38.391: CDMA-PREPAID: Sent the request to AAA
*Mar 1 00:09:38.391: CDMA-PREPAID: Auth_reason: CRB_RSP_PEND_INITIAL_QUOTA
*Mar 1 00:09:38.395: CDMA-PREPAID: Received prepaid response: status 2
*Mar 1 00:09:38.395: CDMA-PREPAID: AAA authorised parms being processed
*Mar 1 00:09:38.395: CDMA-PREPAID: Attr in Grp Prof: cb-duration
*Mar 1 00:09:38.395: (0x4B000000) CDMA/PREPAID: AAA_AT_CRB_ENTITY_TYPE
*Mar 1 00:09:38.395: (0x4B000000) CDMA/PREPAID: entity type returns 1
*Mar 1 00:09:38.395: CDMA-PREPAID: Attr in Grp Prof: cb-duration
*Mar 1 00:09:38.395: (0x4B000000) CDMA/PREPAID: AAA_AT_CRB_DURATON
*Mar 1 00:09:38.395: CDMA-PREPAID: duration returns 120
*Mar 1 00:09:38.395: CDMA-PREPAID: Retrieved attributes successfully
*Mar 1 00:09:38.395: CDMA-PREPAID: Reset duration to 120, mn 9.3.0.1
*Mar 1 00:09:38.395: CDMA-PREPAID: : Started duration timer for 120 sec
```
**debug cdma pdsn qos**

To display debug messages about quality of service features, use the `debug cdma pdsn qos` command in privileged EXEC mode. To disable debug messages, use the `no` form of this command.

```
debug cdma pdsn qos
no debug cdma pdsn qos
```

**Syntax Description**

There are no arguments or keywords for this command.

**Command Default**

There are no default values for this command.

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(8)XW</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(11)T.</td>
</tr>
</tbody>
</table>

**Examples**

There are currently no sample outputs for this command.
**debug cdma pdsn resource-manager**

To display debug messages that help you monitor the resource-manager information, use the `debug cdma pdsn resource-manager` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

`debug cdma pdsn resource-manager [error| events]`

`no debug cdma pdsn resource-manager [error| events]`

**Syntax Description**

| errors | Displays Packet Data Service node (PDSN) resource manager errors. |
| events | Displays PDSN resource manager events. |

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(8)BY</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug cdma pdsn resource-manager` command:

```
Router# debug cdma pdsn resource-manager
errors CDMA PDSN resource manager errors
events CDMA PDSN resource manager events
```
debug cdma pdsn selection

To display debug messages for the intelligent Packet Data Serving Node (PDSN) selection feature, use the `debug cdma pdsn selection` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cdma pdsn selection {errors| events| packets}
no debug cdma pdsn selection {errors| events| packets}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>errors</td>
<td>Displays PDSN selection errors.</td>
</tr>
<tr>
<td>events</td>
<td>Displays PDSN selection events.</td>
</tr>
<tr>
<td>packets</td>
<td>Displays transmitted or received packets.</td>
</tr>
</tbody>
</table>

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug cdma pdsn selection` command with the keyword `events` specified:

```
Router# debug cdma pdsn selection events
CDMA PDSN selection events debugging is on
00:27:46: CDMA-PSL: Message(IN) pdsn 51.4.2.40 interface 70.4.2.40
00:27:46:   Keepalive 10
00:27:46:   Count 0
00:27:46:   Capacity 16000
00:27:46:   Weight 0
00:27:46:   Hostname 11 7206-PDSN-2
00:27:46: CDMA-PSL: Reset keepalive, pdsn 51.4.2.40 current 10 new 10
00:27:46: CDMA-PSL: Message processed, pdsn 51.4.2.40 tsize 0 pendings 0
00:27:47: CDMA-PSL: Send KEEPALIVE, len 32
00:27:47: CDMA-PSL: Message(OUT) dest 224.0.0.11
00:27:47:   Keepalive 10
00:27:47:   Count 1
00:27:47:   Capacity 16000
00:27:47:   Weight 0
```
00:27:47: Hostname 11 7206-PDSN-1
00:27:47: CDMA-PSL: RRQ sent, s=70.4.1.40 (FastEthernet0/1), d=224.0.0.11
debug cdma pdsn service-selection

To display debug messages for service selection, use the `debug cdma pdsn service-selection` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cdma pdsn service-selection
no debug cdma pdsn service-selection
```

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

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug cdma pdsn service-selection` command:

```
Router# debug cdma pdsn service-selection
CDMA PDSN service provisioning debugging is on
1d02h:%LINK-3-UPDOWN:Interface Virtual-Access3, changed state to up
1d02h:Vi3 CDMA-SP: user_class=1, ms_ipaddr_req=1, apply_acl=0
1d02h:Vi3 CDMA-SP: Adding simple ip flow, user=bsip, mm=6.0.0.2,
1d02h:%LINEPROTO-5-UPDOWN:Line protocol on Interface Virtual-Access3, changed state to up
```
debug cdma pdsn session

To display debug messages for Session Manager errors, events, and packets, use the **debug cdma pdsn session** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug cdma pdsn session [errors| events]**

**no debug cdma pdsn session [errors| events]**

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>errors</strong></td>
<td>(Optional) Displays session protocol errors.</td>
</tr>
<tr>
<td><strong>events</strong></td>
<td>(Optional) Displays session events.</td>
</tr>
</tbody>
</table>

**Command Default**

If the command is entered without any optional keywords, all of the types of debug information are enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(8)BY</td>
<td>Keywords were made optional.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the **debug cdma pdsn session** command:

```
Router# debug cdma pdsn session events
CDMA: PDSN session events debugging is on
Router# debug cdma pdsn session errors
CDMA: PDSN session errors debugging is on
Router# show debug
CDMA:
  CDMA PDSN session events debugging is on
  CDMA PDSN session errors debugging is on
Router# *Jan 1 00:22:27:CDMA-SM:create_session 5.5.5.5-4.4.4.5-2
*Jan 1 00:22:27:CDMA-SM:create_tunnel 5.5.5.5-4.4.4.5
*Jan 1 00:22:27:%LINK-3-UPDOWN:Interface Virtual-Access1, changed state to up
*Jan 1 00:22:29:CDMA-SM:create_flow mn=0.0.0.0, ha=8.8.8.8 nai=l2tp2@cisco.com
*Jan 1 00:22:30:%LINEPROTO-5-UPDOWN:Line protocol on Interface Virtual-Access1, changed state to up
```
debug cdp

To enable debugging of the Cisco Discovery Protocol (CDP), use the `debug cdp` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cdp {packets| adjacency| events}
no debug cdp {packets| adjacency| events}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packets</td>
<td>Enables packet-related debugging output.</td>
</tr>
<tr>
<td>adjacency</td>
<td>Enables adjacency-related debugging output.</td>
</tr>
<tr>
<td>events</td>
<td>Enables output related to error messages, such as detecting a bad checksum.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(2)T</td>
<td>This command was introduced in a release earlier than Cisco IOS Release 12.4(2)T.</td>
</tr>
<tr>
<td>12.2(55)SE</td>
<td>This command was modified. The debug output was enhanced to display location Type-Length-Values (TLVs), location-server TLVs, and application TLV-related debugs.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use `debug cdp` commands to display information about CDP packet activity, activity between CDP neighbors, and various CDP events.

**Examples**

The following is sample output from the `debug cdp packets`, `debug cdp adjacency`, and `debug cdp events` commands:

```
Router# debug cdp packets
CDP packet info debugging is on
Router# debug cdp adjacency
CDP neighbor info debugging is on
Router# debug cdp events
CDP events debugging is on
CDP-FA: Packet sent out on Ethernet0
CDP-FA: Packet received from gray.cisco.com on interface Ethernet0
CDP-AD: Deleted table entry for violet.cisco.com, interface Ethernet0
CDP-AD: Interface Ethernet2 coming up
CDP-EV: Encapsulation on interface Serial2 failed
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdp tlv</td>
<td>Configures location support in CDP.</td>
</tr>
<tr>
<td>show cdp tlv</td>
<td>Displays information about CDP TLVs.</td>
</tr>
</tbody>
</table>
**debug cdp ip**

To enable debug output for the IP routing information that is carried and processed by the Cisco Discovery Protocol (CDP), use the `debug cdp ip` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cdp ip
no debug cdp ip
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

CDP is a media- and protocol-independent device-discovery protocol that runs on all Cisco routers. You can use the `debug cdp ip` command to determine the IP network prefixes CDP is advertising and whether CDP is correctly receiving this information from neighboring routers.

Use the `debug cdp ip` command with the `debug ip routing` command to debug problems that occur when on-demand routing (ODR) routes are not installed in the routing table at a hub router. You can also use the `debug cdp ip` command along with the `debug cdp packet` and `debug cdp adjacency` commands along with encapsulation-specific debug commands to debug problems that occur in the receipt of CDP IP information.

**Examples**

The following is sample output from the `debug cdp ip` command. This example shows the transmission of IP-specific information in a CDP update. In this case, three network prefixes are being sent, each with a different network mask.

```
Router# debug cdp ip
CDP-IP: Writing prefix 172.1.69.232.112/28
CDP-IP: Writing prefix 172.19.89.0/24
CDP-IP: Writing prefix 11.0.0.0/8
```

In addition to these messages, you might see the following messages:

- This message indicates that CDP is attempting to install the prefix 172.16.1.0/24 into the IP routing table:

```
CDP-IP: Updating prefix 172.16.1.0/24 in routing table
```

- This message indicates a protocol error occurred during an attempt to decode an incoming CDP packet:

```
CDP-IP: IP TLV length (3) invalid
```

- This message indicates the receipt of the IP prefix 172.16.1.0/24 from a CDP neighbor connected via Ethernet interface 0/0. The neighbor IP address is 10.0.0.1.

```
CDP-IP: Reading prefix 172.16.1.0/24 source 10.0.0.1 via Ethernet0/0
```
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ip routing</td>
<td>Displays information on RIP routing table updates and route cache updates.</td>
</tr>
</tbody>
</table>
debug cef

To enable the display of information about Cisco Express Forwarding events, use the `debug cef` command in privileged EXEC mode. To disable the display of Cisco Express Forwarding events, use the `no` form of this command.

```
default cef {all| assert| background| broker| consistency-check| elog| epoch| fib [attached export| subblock]| hardware [notification| queries]| hash| high-availability| interest| interface| iprm| issu| loadinfo| memory| non-ip| path [extension| list| scope]| subtree context| switching background| table| xdr}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>all</code></td>
<td>Displays debug messages for all Cisco Express Forwarding events.</td>
</tr>
<tr>
<td><code>assert</code></td>
<td>Displays debug messages for Cisco Express Forwarding assert events.</td>
</tr>
<tr>
<td><code>background</code></td>
<td>Displays debug messages for Cisco Express Forwarding background events.</td>
</tr>
<tr>
<td><code>broker</code></td>
<td>Displays debug messages for Cisco Express Forwarding broker events.</td>
</tr>
<tr>
<td><code>consistency-check</code></td>
<td>Displays debug messages for Cisco Express Forwarding consistency checker events.</td>
</tr>
<tr>
<td><code>elog</code></td>
<td>Displays debug messages for Cisco Express Forwarding elog events.</td>
</tr>
<tr>
<td><code>epoch</code></td>
<td>Displays debug messages for Cisco Express Forwarding epoch events.</td>
</tr>
<tr>
<td>`fib [attached export</td>
<td>subblock]]`</td>
</tr>
<tr>
<td>`hardware [notification</td>
<td>queries]</td>
</tr>
<tr>
<td><code>hash</code></td>
<td>Displays debug messages for Cisco Express Forwarding load-balancing hash algorithms.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>high-availability</td>
<td>Displays debug messages for Cisco Express Forwarding high availability events.</td>
</tr>
<tr>
<td>interest</td>
<td>Displays debug messages for Cisco Express Forwarding interest list events.</td>
</tr>
<tr>
<td>interface</td>
<td>Displays debug messages for Cisco Express Forwarding interface events.</td>
</tr>
<tr>
<td>iprm</td>
<td>Displays debug messages for Cisco Express Forwarding IP rewrite manager events. (This keyword is not available in Cisco IOS Release 12.2(33)SRA.)</td>
</tr>
<tr>
<td>issu</td>
<td>Displays debug messages for Cisco Express Forwarding In Service Software Upgrade (ISSU) events.</td>
</tr>
<tr>
<td>loadinfo</td>
<td>Displays debug messages for Cisco Express Forwarding loadinfo events.</td>
</tr>
<tr>
<td>memory</td>
<td>Displays debug messages for Cisco Express Forwarding memory events.</td>
</tr>
<tr>
<td>non-ip</td>
<td>Displays debug messages for Cisco Express Forwarding non-IP entry events.</td>
</tr>
<tr>
<td>path [extension</td>
<td>list</td>
</tr>
<tr>
<td>subtree context</td>
<td>Displays debug messages for Cisco Express Forwarding subtree context events.</td>
</tr>
<tr>
<td>switching background</td>
<td>Displays debug messages for Cisco Express Forwarding switching background events.</td>
</tr>
<tr>
<td>table</td>
<td>Displays debug messages for Cisco Express Forwarding table events.</td>
</tr>
<tr>
<td>xdr</td>
<td>Displays debug messages for Cisco Express Forwarding External Data Representation (XDR) events.</td>
</tr>
</tbody>
</table>

**Command Default**

Debugging information about Cisco Express Forwarding events is not displayed.

**Command Modes**

Privileged EXEC (#)
Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(25)S</td>
<td>This command was introduced. The <code>debug ceffibattachedexport</code> command replaces the <code>debugipcefadjfib</code> command.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(28)SB and implemented on the Cisco 10000 series routers.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2(33)SXH</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SXH.</td>
</tr>
<tr>
<td>12.4(20)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(20)T.</td>
</tr>
</tbody>
</table>

Usage Guidelines

Because debugging output is assigned high priority in the CPU process, you should use debug commands only to troubleshoot specific problems or during troubleshooting sessions with Cisco technical support staff. Moreover, you should use debug commands during periods of lower network traffic and fewer users. Debugging during these periods decreases the likelihood that increased debug command processing overhead will affect system use.

Examples

The following is sample output from the `debug cefall` command:

```
Router# debug cef all
06:23:38: HW-API: Counter poll: Label[label=implicit-null]
06:23:38: HW-API: Counter poll: Label[label=implicit-null]
06:23:43: FIBbg: Timer 'FIB checkers: IPv4 scan-rib-ios scanner' expired, calling 0x40FA03FC, context 0x00010003
06:23:43: FIBbg: Restarting timer 'FIB checkers: IPv4 scan-rib-ios scanner' with delay 60000
06:23:43: FIBbg: Timer 'FIB checkers: IPv4 scan-rios-rib scanner' expired, calling 0x40FA03FC, context 0x00010004
06:23:43: FIBbg: Restarting timer 'FIB checkers: IPv4 scan-rios-rib scanner' with delay 60000
06:23:43: FIBbg: Timer 'FIB checkers: IPv4 scan-rp-lc scanner' expired, calling 0x40FA03FC, context 0x00010002
06:23:43: FIBbg: Restarting timer 'FIB checkers: IPv4 scan-rp-lc scanner' with delay 60000
06:23:43: FIBbg: Timer 'FIB checkers: IPv6 scan-rios-rib scanner' expired, calling 0x40FA03FC, context 0x00020002
06:23:43: FIBbg: Restarting timer 'FIB checkers: IPv6 scan-rios-rib scanner' with delay 60000
06:23:48: HW-API: Counter poll: Label[label=implicit-null]
06:23:48: HW-API: Counter poll: Label[label=implicit-null]
06:23:48: HW-API: Counter poll: Label[label=implicit-null]
06:24:06: FIBtable: IPv4: Event modified, 0.0.0.0/0, vrf Default, 1 path, flags 00420005
06:24:06: FIBpath: Configuring IPv4 path 444B2AB0 from rib (idb=NULL, gw=9.1.41.1, gw_table=0, rr=1) and host prefix 0.0.0.0
```
06:24:06: FIBpath: Configured recursive-nexthop 9.1.41.1[0] 444B2AB0 path
06:24:06: FIBfib: [v4-0.0.0.0/0 (44AAC750)] Mod type - null
06:24:06: FIBtable: IPv4: Event up, default, 0.0.0.0/0, vrf Default, 1 path, flags 00420005
06:24:06: FIBtable: IPv4: Adding route for 0.0.0.0/0 but route already exists. Trying modify.
06:24:06: FIBpath: Configuring IPv4 path 444B2AA0 from rib (idb=NULL, gw=9.1.41.1, gw_table=0, rr=1) and host prefix 0.0.0.0/0
06:24:06: FIBpath: Configured recursive-nexthop 9.1.41.1[0] 444B2AA0 path
06:24:06: FIBfib: [v4-0.0.0.0/0 (44AAC750)] Mod type - null vrf
06:24:07: FIBbg: Timer 'FIB checkers: IPv4 scan-hw-sw scanner' expired, calling 0x40FA03FC, context 0x00010005
06:24:07: FIBbg: Restarting timer 'FIB checkers: IPv4 scan-hw-sw scanner' with delay 60000
06:24:07: FIBbg: Timer 'FIB checkers: IPv4 scan-sw-hw scanner' expired, calling 0x40FA03FC, context 0x00010006
06:24:07: FIBbg: Restarting timer 'FIB checkers: IPv4 scan-sw-hw scanner' with delay 60000

<table>
<thead>
<tr>
<th>Name</th>
<th>Default RD</th>
<th>Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>1:1</td>
<td>Ethernet4/0/5</td>
</tr>
</tbody>
</table>

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cef table consistency-check</td>
<td>Enables Cisco Express Forwarding consistency checker table values by type and parameter.</td>
</tr>
<tr>
<td>clear cef table</td>
<td>Clears the Cisco Express Forwarding tables.</td>
</tr>
<tr>
<td>clear ip cef inconsistency</td>
<td>Clears Cisco Express Forwarding inconsistency statistics and records found by the Cisco Express Forwarding consistency checkers.</td>
</tr>
<tr>
<td>debug ip cef table</td>
<td>Enables the collection of events that affect entries in the Cisco Express Forwarding tables.</td>
</tr>
<tr>
<td>show cef table consistency-check</td>
<td>Displays Cisco Express Forwarding consistency checker table values.</td>
</tr>
<tr>
<td>show ip cef inconsistency</td>
<td>Displays Cisco Express Forwarding IP prefix inconsistencies.</td>
</tr>
</tbody>
</table>
debug cell-hwic driver

To debug the Cisco IOS driver for the cellular interface, use the `debug cell-hwic driver` command in EXEC mode.

```
debug cell-hwic slotwic_slotport driver {crcdump|errdump|errors}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>slot/wic_slot/port</code></td>
<td>Numeric values that indicate the router slot, WAN interface card (WIC) slot, and port.</td>
</tr>
<tr>
<td><code>crcdump</code></td>
<td>CRC error details.</td>
</tr>
<tr>
<td><code>errdump</code></td>
<td>Other error details.</td>
</tr>
<tr>
<td><code>errors</code></td>
<td>Errors debugging.</td>
</tr>
</tbody>
</table>

**Command Default**

None

**Command Modes**

EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(11)XV</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command for debugging purposes only.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cellular messages async</td>
<td>Debugs cellular async.</td>
</tr>
<tr>
<td>debug cellular messages data</td>
<td>Prints Cisco IOS data path debug messages.</td>
</tr>
<tr>
<td>debug cellular firmware</td>
<td>Displays Cisco IOS firmware information.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>debug cellular messages management</td>
<td>Prints management path messages, such as CnS.</td>
</tr>
<tr>
<td>debug cellular messages dm</td>
<td>Prints diagnostics monitor (DM) messages from the Qualcomm CDMA chipset.</td>
</tr>
<tr>
<td>debug cell-hwic virt-con</td>
<td>Redirects the Nios II console driver messages to display them in the Cisco IOS router console environment.</td>
</tr>
</tbody>
</table>
debug cell-hwic firmware

To see the Cisco IOS firmware information, use the debug cell-hwic firmware command in EXEC mode.

d debug cellular slotwic_slotport firmware

Syntax Description

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slot/wic_slot/port</td>
<td>Numeric values that indicate the router slot, WAN interface card (WIC) slot, and port.</td>
</tr>
</tbody>
</table>

Command Default

None

Command Modes

EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(11)XV</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
<tr>
<td>12.4(22)YB1</td>
<td>This command was integrated into Cisco IOS Release 12.4(22)YB1.</td>
</tr>
</tbody>
</table>

Usage Guidelines

Use this command for debugging purposes only.

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cellular messages async</td>
<td>Debugs cellular async.</td>
</tr>
<tr>
<td>debug cellular messages data</td>
<td>Prints Cisco IOS data path debug messages.</td>
</tr>
<tr>
<td>debug cellular firmware</td>
<td>Debugs the Cisco IOS driver.</td>
</tr>
<tr>
<td>debug cellular messages dm</td>
<td>Prints diagnostics monitor (DM) messages from the Qualcomm CDMA chipset.</td>
</tr>
<tr>
<td>debug cell-hwic virt-con</td>
<td>Redirects the Nios II console driver messages to display them in the Cisco IOS router console environment.</td>
</tr>
</tbody>
</table>
debug cellular messages all

To print all Cisco IOS driver debug messages, use the `debugcellularmessagesall` command in EXEC mode.

```
ddebug cellular slotwic_slotport messages all
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slot/wic_slot/port</td>
<td>Numeric values that indicate the router slot, WAN interface card (WIC) slot, and port.</td>
</tr>
</tbody>
</table>

### Command Default

None

### Command Modes

EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(11)XV</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
</tbody>
</table>

### Usage Guidelines

Use this command for debugging purposes only.

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cellular messages async</td>
<td>Debugs cellular async.</td>
</tr>
<tr>
<td>debug cellular messages data</td>
<td>Prints Cisco IOS data path debug messages.</td>
</tr>
<tr>
<td>debug cell-hwic driver</td>
<td>Debugs the Cisco IOS driver.</td>
</tr>
<tr>
<td>debug cell-hwic firmware</td>
<td>Displays Cisco IOS firmware information.</td>
</tr>
<tr>
<td>debug cellular messages management</td>
<td>Prints management path messages, such as CnS.</td>
</tr>
<tr>
<td>debug cellular messages dm</td>
<td>Prints diagnostics monitor (DM) messages from the Qualcomm CDMA chipset.</td>
</tr>
<tr>
<td>debug cellular messages virt-con</td>
<td>Redirects the Nios II console driver messages to display them in the Cisco IOS router console environment.</td>
</tr>
</tbody>
</table>
debug backhaul-session-manager session through debug channel packets

ddebug cellular messages all
debug cellular messages async

To debug cellular async, use the `debugcellularmessagesasync` command in EXEC mode.

```
ddebug cellular slot/wic_slot/port messages async
```

**Syntax Description**

| slot/wic_slot/port | Numeric values that indicate the router slot, WAN interface card (WIC) slot, and port. |

**Command Default**

None

**Command Modes**

EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(11)XV</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command for debugging purposes only.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug cellular messages all</code></td>
<td>Prints all Cisco IOS driver debug messages.</td>
</tr>
<tr>
<td><code>debug cellular messages data</code></td>
<td>Prints Cisco IOS data path debug messages.</td>
</tr>
<tr>
<td><code>debug cellular driver</code></td>
<td>Debugs the Cisco IOS driver.</td>
</tr>
<tr>
<td><code>debug cellular firmware</code></td>
<td>Displays Cisco IOS firmware information.</td>
</tr>
<tr>
<td><code>debug cellular messages management</code></td>
<td>Prints management path messages, such as CnS.</td>
</tr>
<tr>
<td><code>debug cellular messages dm</code></td>
<td>Prints diagnostics monitor (DM) messages from the Qualcomm CDMA chipset.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>debug cellular messages virt-con</td>
<td>Redirects the Nios II console driver messages to display them in the Cisco IOS router console environment.</td>
</tr>
</tbody>
</table>
debug cellular messages data

To print Cisco IOS data path debug messages, use the `debugcellularmessagesdata` command in EXEC mode.

```
show cellular slot wic_slot port messages data
```

**Syntax Description**

| slot/wic_slot/port | Numeric values that indicate the router slot, WAN interface card (WIC) slot, and port. |

**Command Default**

None

**Command Modes**

EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(11)XV</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command for debugging purposes only.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cellular messages all</td>
<td>Prints all Cisco IOS driver debug messages.</td>
</tr>
<tr>
<td>debug cellular messages async</td>
<td>Debugs cellular async.</td>
</tr>
<tr>
<td>debug cell-hwic driver</td>
<td>Debugs the Cisco IOS driver.</td>
</tr>
<tr>
<td>debug cell-hwic firmware</td>
<td>Displays Cisco IOS firmware information.</td>
</tr>
<tr>
<td>debug cellular messages mgmt</td>
<td>Prints management path messages, such as CnS.</td>
</tr>
<tr>
<td>debug cellular messages dm</td>
<td>Prints diagnostics monitor (DM) messages from the Qualcomm CDMA chipset.</td>
</tr>
<tr>
<td>debug cellular messages virt-con</td>
<td>Redirects the Nios II console driver messages to display them in the Cisco IOS router console environment.</td>
</tr>
</tbody>
</table>
debug backhaul-session-manager session through debug channel packets

debug cellular messages data
**debug cellular messages dm**

To print Diagnostics Monitor (DM) messages from the Qualcomm CDMA chipset, use the `debug cellular messages dm` command in EXEC mode.

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>slot/wic_slot/port</code></td>
<td>Numeric values that indicate the router slot, WAN interface card (WIC) slot, and port.</td>
</tr>
</tbody>
</table>

**Command Default**

There is no default for this command.

**Command Modes**

EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(11)XV</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command for debugging purposes only.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug cellular messages all</code></td>
<td>Prints all Cisco IOS driver debug messages.</td>
</tr>
<tr>
<td><code>debug cellular messages async</code></td>
<td>Debugs cellular async.</td>
</tr>
<tr>
<td><code>debug cellular messages data</code></td>
<td>Prints Cisco IOS data path debug messages.</td>
</tr>
<tr>
<td><code>debug cell-hwic driver</code></td>
<td>Debugs the Cisco IOS driver.</td>
</tr>
<tr>
<td><code>debug cell-hwic firmware</code></td>
<td>Displays Cisco IOS firmware information.</td>
</tr>
<tr>
<td><code>debug cellular messages management</code></td>
<td>Prints management path messages, such as CnS.</td>
</tr>
<tr>
<td><code>debug cellular messages virt-con</code></td>
<td>Redirects the Nios II console driver messages to display them in the Cisco IOS router console environment.</td>
</tr>
</tbody>
</table>
debug backhaul-session-manager session through debug channel packets

debug cellular messages dm
debug cellular messages management

To print management path messages, such as CnS, use the `debugcellularmessagesmanagement` command in EXEC mode.

**debug cellular slot/wic_slot/port messages management**

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slot/wic_slot/port</td>
<td>Numeric values that indicate the router slot, WAN interface card (WIC) slot, and port.</td>
</tr>
</tbody>
</table>

**Command Default**

None

**Command Modes**

EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(11)XV</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command for debugging purposes only.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cellular messages all</td>
<td>Prints all Cisco IOS driver debug messages.</td>
</tr>
<tr>
<td>debug cellular messages async</td>
<td>Debugs cellular async.</td>
</tr>
<tr>
<td>debug cellular messages data</td>
<td>Prints Cisco IOS data path debug messages.</td>
</tr>
<tr>
<td>debug cell-hwic driver</td>
<td>Debugs the Cisco IOS driver.</td>
</tr>
<tr>
<td>debug cell-hwic firmware</td>
<td>Displays Cisco IOS firmware information.</td>
</tr>
<tr>
<td>debug cellular messages virt-con</td>
<td>Redirects the Nios II console driver messages to display them in the Cisco IOS router console environment.</td>
</tr>
</tbody>
</table>
debug cellular messages

To debug GPS or NMEA management messages, use the **debug cellular messages** command in privileged EXEC mode.

**debug cellular unit messages {gps|nmea}**

**Syntax Description**

- **unit**
  - EHWIC Router slot, WIC slot, and port separated by slashes (for example, 0/1/0).
  - For a fixed platform, the number is 0.
- **gps**
  - Debugs GPS management messages.
- **nmea**
  - Debugs NMEA management messages.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS Release 15.3(3)M</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command to collect debug GPS or NMEA data for evaluation.

**Note**

When the **debug cellular messages nmea** command is entered to troubleshoot NMEA streaming, a large amount of output data is generated, especially once a GPS fix is obtained.

**Examples**

The following example shows how the **debug cellular messages nmea** command is used to gather NMEA troubleshooting output data from an EHWIC:

```
Device# debug cellular 0/0/0 messages nmea

NMEA debugging is on

Device#
```

*Aug 15 17:03:42.631 PDT: Ignoring NMEA ID:5
*Aug 15 17:03:42.635 PDT: big nmea token width 26
*Aug 15 17:03:42.635 PDT: eccell_nmea_parse_gpgsv: num_satellites 10, num_of_sat in pak 2
*Aug 15 17:03:42.635 PDT: eccell_nmea_parse_gpgsv: raw_satellite_info[i].Sat_number 31, raw_satellite_info[i].elevation 11, raw_satellite_info[i].azimuth 314, raw_satellite_info[i].sig_noise_ratio 35
*Aug 15 17:03:42.635 PDT: eccell_nmea_parse_gpgsv: raw_satellite_info[i].Sat_number 21, raw_satellite_info[i].elevation 0, raw_satellite_info[i].azimuth 232, raw_satellite_info[i].sig_noise_ratio 0
The following example shows how the `debug cellular messages gps` command is used to gather GPS troubleshooting output data from an EHWIC:

```
Device# debug cellular 0/0/0 messages gps

Device# debug cellular 0/0/0 messages gps

GPS debugging is on

Device#
```

```
*Aug 15 17:15:52.975 PDT: ecell_set_gpio Assertions
*Aug 15 17:15:52.975 PDT: ecell_set_gpio Assertions
*Aug 15 17:15:52.975 PDT: no change in GPS state 3
*Aug 15 17:16:02.975 PDT: ecell_set_gpio Assertions
*Aug 15 17:16:02.975 PDT: no change in GPS state 3
```
**debug cellular messages sms**

To debug SMS data path errors received on the modem, use the `debug cellular messages sms` command in privileged EXEC mode.

`debug cellular unit messages sms`

**Syntax Description**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>unit</code></td>
<td>EHWIC Router slot, WIC slot, and port separated by slashes (for example, 0/1/0. For a fixed platform, the number is 0.)</td>
</tr>
</tbody>
</table>

**Command Default**

Privileged EXEC (#)

**Command Modes**

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS Release 15.3(3)M</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command to collect debug SMS message information for evaluation.

**Examples**

The following example shows the debugging of SMS messaging data on the modem:

```
Device# debug cellular 0/0/0 messages sms
```
**debug cell-hwic virt-con**

To redirect the Nios II console driver messages to display them in the Cisco IOS router console environment, use the `debug cell-hwic virt-con` command in EXEC mode.

```
debug cell-hwic slot/wic_slot/port virt-con {clear| disable| dump-data-structs| log| monitor| wrapper-on| wrapper-off}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slot/wic_slot/port</td>
<td>Numeric values that indicate the router slot, WAN interface card (WIC) slot, and port.</td>
</tr>
<tr>
<td>clear</td>
<td>Optional) Clears all virtual console debug log messages.</td>
</tr>
<tr>
<td>disable</td>
<td>Optional) Disables virtual console real-time debug monitoring.</td>
</tr>
<tr>
<td>dump-data-structs</td>
<td>Optional) Dumps virtual console data structures.</td>
</tr>
<tr>
<td>log</td>
<td>Optional) Displays virtual console messages from the debug log.</td>
</tr>
<tr>
<td>monitor</td>
<td>Optional) Enables monitoring of real-time virtual console debug messages.</td>
</tr>
<tr>
<td>wrapper-on</td>
<td>Optional) Disables wraparound for virtual console log messages.</td>
</tr>
<tr>
<td>wrapper-off</td>
<td>Optional) Enables wraparound for virtual console log messages.</td>
</tr>
</tbody>
</table>

**Command Default**

There is no default for this command.

**Command Modes**

EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(11)XV</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>
### Usage Guidelines

Use this command for debugging purposes only.

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cellular messages all</td>
<td>Prints all Cisco IOS driver debug messages.</td>
</tr>
<tr>
<td>debug cellular messages async</td>
<td>Debugs cellular async.</td>
</tr>
<tr>
<td>debug cellular messages data</td>
<td>Prints Cisco IOS data path debug messages.</td>
</tr>
<tr>
<td>debug cell-hwic driver</td>
<td>Debugs the Cisco IOS driver.</td>
</tr>
<tr>
<td>debug cell-hwic firmware</td>
<td>Displays Cisco IOS firmware information.</td>
</tr>
<tr>
<td>debug cellular messages management</td>
<td>Prints management path messages, such as CnS.</td>
</tr>
<tr>
<td>debug cellular messages dm</td>
<td>Prints diagnostics monitor (DM) messages from the Qualcomm CDMA chipset.</td>
</tr>
</tbody>
</table>
### debug cem ls errors

To debug connection errors or null data structures, use the `debug cem ls errors` command in privileged EXEC mode. To disable this form of debugging, use the `no` form of this command.

```
ddebug cem ls errors

no debug cem ls errors
```

#### Syntax Description

This command has no arguments or keywords.

#### Command Modes

Privileged EXEC (#)

#### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SRC</td>
<td>This command was introduced on the Cisco 7600 series routers.</td>
</tr>
</tbody>
</table>

#### Usage Guidelines

Use the `show debug` command to see debug information.

#### Examples

The following command turns on CEM local switching error debugging:

```
Router# debug cem ls errors
```

#### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cem ls events</td>
<td>Enables debugging of events relating to CEM local switching.</td>
</tr>
</tbody>
</table>
**debug cem ls events**

To debug CEM local switching events, use the debug cem ls events command in privileged EXEC mode. To disable this form of debugging, use the no form of this command.

```
display cem ls events
no display cem ls events
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SRC</td>
<td>This command was introduced on the Cisco 7600 series routers.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use the show debug command to see debug information.

**Examples**

The following command turns on debugging for CEM local switching events.

```
Router# debug cem ls events
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cem ls errors</td>
<td>Enables debugging of connection errors or null data structures.</td>
</tr>
</tbody>
</table>
**debug ces-conn**

To display information from circuit emulation service (CES) clients, use the `debug ces-conn` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug ces-conn [all| errors| events]
no debug ces-conn
```

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>all</code></td>
<td>(Optional) Displays all error and event information.</td>
</tr>
<tr>
<td><code>errors</code></td>
<td>(Optional) Displays only error information.</td>
</tr>
<tr>
<td><code>events</code></td>
<td>(Optional) Displays only event information.</td>
</tr>
</tbody>
</table>

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(5)XM</td>
<td>This command is supported on Cisco 3600 series routers.</td>
</tr>
<tr>
<td>12.2(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(4)T.</td>
</tr>
</tbody>
</table>

**Examples**

The following example shows debug output for a CES connection:

```
Router# debug ces-conn all
CES all debugging is on
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# connect conn1 t1 3/0 1 atm1/0 1/100
Router(config-ces-conn)# exit
Router(config)#
*Mar 6 18:32:27:CES_CLIENT:vc QoS parameters are PCR = 590, CDV = 5000, CAS_ENABLED = 1, partial fill = 0, multiplier = 8, cbr rate = 64, clock recovery = 0, service_type = 3, error method = 0, sdt_size = 196, billing count = 0
*Mar 6 18:32:27:CES_CLIENT:attempt 1 to activate segment>
```
debug cfm

To enable debugging of the data path of Ethernet connectivity fault management (CFM) on Cisco Catalyst 6500 series switches, use the `debug cfm` command in privileged EXEC mode. To disable the debugging function, use the `no` form of this command.

```
d debug cfm {all| api| cfmpal| common| db| isr}   
no debug cfm {all| api| cfmpal| common| db| isr} 
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Specifies all Catalyst 6500 switch-specific route processor and switch processor (RP/SP) events.</td>
</tr>
<tr>
<td>api</td>
<td>Specifies Catalyst 6500 switch-specific application program interface (API) events.</td>
</tr>
<tr>
<td>cfmpal</td>
<td>Specifies general Catalyst 6500 switch debugging.</td>
</tr>
<tr>
<td>common</td>
<td>Specifies common Catalyst 6500 switch RP/SP components.</td>
</tr>
<tr>
<td>db</td>
<td>Specifies Catalyst 6500 switch CFM database debugging.</td>
</tr>
<tr>
<td>isr</td>
<td>Specifies Catalyst 6500 switch-specific ingress CFM packet debugging.</td>
</tr>
</tbody>
</table>

**Command Default**

Debugging is disabled.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SXI2</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRE</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRE.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The output from this command is a log of activity.

Use this command to troubleshoot Ethernet CFM on Cisco Catalyst 6500 series switches.
The following example shows output of the `debug cfm all` command:

```
Device# debug cfm all
CFM DB events debugging is on
CFM Ingress ISR events debugging is on
CFMPAL events debugging is on
CFM API events debugging is on
CFM RP/SP COMMON events debugging is on
CFM packets debugging is on
```
debug channel events

To display processing events on Cisco 7000 series routers that occur on the channel adapter interfaces of all installed adapters, use the `debug channel events` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

`debug channel events`

`no debug channel events`

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(3)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

This command displays CMCC adapter events that occur on the Channel Interface Processor (CIP) or Channel Port Adapter (CPA) and is useful for diagnosing problems in an IBM channel attach network. It provides an overall picture of the stability of the network. In a stable network, the `debug channel events` command does not return any information. If the command generates numerous messages, the messages can indicate the possible source of the problems. To observe the statistic message (cip_love_letter) sent every 10 seconds, use the `debug channel love` command.

When configuring or making changes to a router or interface that supports IBM channel attach, enable the `debug channel events` command. Doing so alerts you to the progress of the changes or to any errors that might result. Also use this command periodically when you suspect network problems.

**Examples**

The following sample output is from the `debug channel events` command:

```
Router# debug channel events
Channel3/0: cip_reset(), state administratively down
Channel3/0: cip_reset(), state up
Channel3/0: sending nodeid
Channel3/0: sending command for vc 0, CLAW path C700, device C0
```

The following line indicates that the CIP is being reset to an administrative down state:

```
Channel3/0: cip_reset(), state administratively down
```

The following line indicates that the CIP is being reset to an administrative up state:

```
Channel3/0: cip_reset(), state up
```
The following line indicates that the node ID is being sent to the CIP. This information is the same as the "Local Node" information under the `show extendedchannelslot/port/subchannels` command. The CIP needs to send this information to the host mainframe.

Channel3/0: sending nodeid
The following line indicates that a Common Link Access for Workstations (CLAW) subchannel command is being sent from the Route Processor (RP) to the CIP. The value vc 0 indicates that the CIP will use virtual circuit number 0 with this device. The virtual circuit number also shows up when you use the `debugchannelpackets` command.

Channel3/0: sending command for vc 0, CLAW path C700, device C0
The following is a sample output that is generated by the `debugchannelevents` command when a CMPC+ IP TG connection is activated with the host:

1d05h:Channel4/2:Received route UP for tg (768)
1d05h:Adding STATIC ROUTE for vc:768
The following is a sample output from the `debugchannelevents` command when a CMPC+ IP TG connection is deactivated:

1d05h:Channel4/2:Received route DOWN for tg (768)
1d05h:Deleting STATIC ROUTE for vc:768

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug channel ilan</code></td>
<td>Displays CIP love letter events.</td>
</tr>
<tr>
<td><code>debug channel packets</code></td>
<td>Displays per-packet debugging output.</td>
</tr>
</tbody>
</table>
**debug channel ilan**

To display messages relating to configuration and bridging using Cisco Mainframe Channel Connection (CMCC) internal LANs and to help debug source-route bridging (SRB) problems related to CMCC internal LANs, use the `debug channel ilan` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug channel ilan
no debug channel ilan
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.0(3)</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The `debug channel ilan` command displays events related to CMCC internal LANs. This command is useful for debugging problems associated with CMCC internal LAN configuration. It is also useful for debugging problems related to SRB packet flows through internal LANs.

**Examples**

The following is sample output from the `debug channel ilan` command:

```
Router# debug channel ilan
Channel internal LANs debugging is on
The following line indicates that a packet destined for the CMCC via a configured internal MAC adapter configured on an internal LAN was dropped because the Logical Link Control (LLC) end station in Cisco IOS software did not exist:

CIP ILAN(Channel3/2-Token): Packet dropped - NULL LLC
The following line indicates that a packet destined for the CMCC via a configured internal MAC adapter configured on an internal LAN was dropped because the CMCC had not yet acknowledged the internal MAC adapter configuration command:

Channel3/2: ILAN Token-Ring 3 - CIP internal MAC adapter not acknowledged DMAC(4000.7000.0001) SMAC(0c00.8123.0023)
```
**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug channel events</td>
<td>Displays processing that occurs on the channel adapter interfaces of all installed adapters.</td>
</tr>
<tr>
<td>debug source bridge</td>
<td>Displays information about packets and frames transferred across a source-route bridge.</td>
</tr>
</tbody>
</table>
**debug channel love**

To display Channel Interface Processor (CIP) love letter events, use the **debugchannellove** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug channel love**

**no debug channel love**

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

This command displays CIP love letter events (an operating status or configuration message) that occur on the CIP interface processor and is useful for diagnosing problems in an IBM channel attach network. It provides an overall picture of the stability of the network. In a stable network, the **debugchannellove** command returns a statistic message (**cip_love_letter**) that is sent every 10 seconds. This command is valid for the Cisco 7000 series routers only.

**Examples**

The following is sample output from the **debugchannellove** command:

```
Router# debug channel love
Channel3/1: love letter received, bytes 3308
Channel3/0: love letter received, bytes 3336

cip_love_letter: received ll, but no cip_info
```

The following line indicates that data was received on the CIP:

```
Channel3/1: love letter received, bytes 3308
```

The following line indicates that the interface is enabled, but there is no configuration for it. It does not normally indicate a problem, just that the Route Processor (RP) got statistics from the CIP but has no place to store them.

```
cip_love_letter: received ll, but no cip_info
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>debug channel events</strong></td>
<td>Displays processing that occurs on the channel adapter interfaces of all installed adapters.</td>
</tr>
<tr>
<td><strong>debug channel packets</strong></td>
<td>Displays per-packet debugging output.</td>
</tr>
</tbody>
</table>
debug channel packets

To display per-packet debugging output, use the `debugchannelpackets` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
default channel packets
no debug channel packets
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

The `debugchannelpackets` command displays all process-level Channel Interface Processor (CIP) packets for both outbound and inbound packets. The output reports information when a packet is received or a transmission is attempted. You will need to disable fast switching and autonomous switching to obtain debugging output. This command is useful for determining whether packets are received or sent correctly.

This command is valid for the Cisco 7000 series routers only.

**Examples**

The following is sample output from the `debugchannelpackets` command:

```
Router# debug channel packets
(Channel3/0)-out size = 104, vc = 0000, type = 0800, src 172.24.0.11, dst 172.24.1.58
(Channel3/0)-in size = 48, vc = 0000, type = 0800, src 172.24.1.58, dst 172.24.15.197
(Channel3/0)-in size = 48, vc = 0000, type = 0800, src 172.24.1.58, dst 172.24.15.197
(Channel3/0)-out size = 71, vc = 0000, type = 0800, src 172.24.15.197, dst 172.24.1.58
(Channel3/0)-in size = 44, vc = 0000, type = 0800, src 172.24.1.58, dst 172.24.15.197
```

The table below describes the significant fields shown in the display.

**Table 54: debug channel packets Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Channel3/0)</td>
<td>Interface slot and port.</td>
</tr>
<tr>
<td>in/out</td>
<td>&quot;In&quot; is a packet from the mainframe to the router. &quot;Out&quot; is a packet from the router to the mainframe.</td>
</tr>
<tr>
<td>size</td>
<td>Number of bytes in the packet, including internal overhead.</td>
</tr>
<tr>
<td>vc</td>
<td>Value from 0 to 511 that maps to the claw interface configuration command. This information is from the MAC layer.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>type=</td>
<td>Encapsulation type in the MAC layer. The value 0800 indicates an IP datagram.</td>
</tr>
<tr>
<td>src</td>
<td>Origin, or source, of the packet, as opposed to the previous hop address.</td>
</tr>
<tr>
<td>dst</td>
<td>Destination of the packet, as opposed to the next-hop address.</td>
</tr>
</tbody>
</table>
debug backhaul-session-manager session through debug channel packets

ddebug channel packets
debug clns esis events through debug dbconn tcp

- debug clns esis events through debug dbconn tcp, page 385

ddebug clns esis events through debug dbconn tcp
debug clns esis events

To display uncommon End System-to-Intermediate System (ES-IS) events, including previously unknown neighbors, neighbors that have aged out, and neighbors that have changed roles (ES-IS, for example), use the `debug clns esis events` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
  debug clns esis events
  no debug clns esis events
```

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

**Command Modes**
Privileged EXEC

**Examples**
The following is sample output from the `debug clns esis events` command:

```
Router# debug clns esis events
ES-IS: ISH from aa00.0400.2c05 (Ethernet1), HT 30
ES-IS: ESH from aa00.0400.9105 (Ethernet1), HT 150
ES-IS: ISH sent to All ESs (Ethernet1): NET 49.0001.AA00.0400.6904.00, HT 299, HLEN 20
The following line indicates that the router received a hello packet (ISH) from the IS at MAC address aa00.0400.2c05 on Ethernet interface 1. The hold time (or number of seconds to consider this packet valid before deleting it) for this packet is 30 seconds.

ES-IS: ISH from aa00.0400.2c05 (Ethernet1), HT 30
The following line indicates that the router received a hello packet (ESH) from the ES at MAC address aa00.0400.9105 on the Ethernet interface 1. The hold time is 150 seconds.

ES-IS: ESH from aa00.0400.9105 (Ethernet1), HT 150
The following line indicates that the router sent an IS hello packet on the Ethernet interface 0 to all ESs on the network. The network entity title (NET) address of the router is 49.0001.0400.AA00.6904.00; the hold time for this packet is 299 seconds; and the header length of this packet is 20 bytes.

ES-IS: ISH sent to All ESs (Ethernet1): NET 49.0001.AA00.0400.6904.00, HT 299, HLEN 20
```
**debug clns esis packets**

To enable display information on End System-to-Intermediate System (ES-IS) packets that the router has received and sent, use the `debug clns esis packets` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```plaintext
debug clns esis packets
no debug clns esis packets
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Examples**

The following is sample output from the `debug clns esis packets` command:

```
Router# debug clns esis packets
ES-IS: ISH sent to All ESs (Ethernet0): NET 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 33
ES-IS: ISH sent to All ESs (Etherent1): NET 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 34
ES-IS: ISH from aa00.0400.6408 Ethernet0, HT 299
ES-IS: ESH from 0000.0c00.bda8 Ethernet0, HT 300
The following line indicates that the router has sent an IS hello packet on Ethernet interface 0 to all ESs on the network. This hello packet indicates that the NET of the router is 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00. The hold time for this packet is 299 seconds. The packet header is 33 bytes in length.

ES-IS: ISH sent to All ESs (Tunnel0): NET 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 33
The following line indicates that the router has sent an IS hello packet on Tunnel interface 0 to all ESs on the network. This hello packet indicates that the NET of the router is 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00. The hold time for this packet is 299 seconds. The packet header is 33 bytes in length.

ES-IS: ISH sent to All ESs (Tunnel1): NET 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 34
The following line indicates that the router received a hello packet on Tunnel interface 0 from an intermediate system, aa00.0400.6408. The hold time for this packet is 299 seconds.

ES-IS: ISH from aa00.0400.6408 Ethernet0, HT 299
The following line indicates that the router has sent an IS hello packet on Tunnel interface 0 to all ESs on the network. This hello packet indicates that the NET of the router is 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00. The hold time for this packet is 299 seconds. The packet header is 33 bytes in length.

ES-IS: ISH sent to All ESs (Tunnel0): NET 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 34
```
The following line indicates that on Ethernet interface 0, the router received a hello packet from an end system with an SNPA of 0000.0c00.bda8. The hold time for this packet is 300 seconds.

IS-IS: ESH from 0000.0c00.bda8 (Ethernet0), HT 300
**debug clns events**

To display Connectionless Network Service (CLNS) events that are occurring at the router, use the `debug clns events` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

**debug clns events**

**no debug clns events**

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Examples**

The following is sample output from the `debug clns events` command:

```
Router# debug clns events
CLNS: Echo PDU received on Ethernet3 from 39.0001.2222.2222.2222.00!
CLNS: Sending from 39.0001.3333.3333.3333.00 to 39.0001.2222.2222.2222.00 via 2222.2222.2222 (Ethernet3 0000.0c00.3a18)
CLNS: Forwarding packet size 117 from 39.0001.2222.2222.2222.00 to 49.0002.0001.AAAA.AAAA.AAAA.00 via 49.0002 (Ethernet3 0000.0c00.b5a3)
CLNS: RD Sent on Ethernet3 to 39.0001.2222.2222.2222.00 @ 0000.0c00.3a18, redirecting 49.0002.0001.AAAA.AAAA.AAAA.00 to 0000.0c00.b5a3
```

The following line indicates that the router received an echo protocol data unit (PDU) on Ethernet interface 3 from source network service access point (NSAP) 39.0001.2222.2222.2222.00. The exclamation point at the end of the line has no significance.

```
CLNS: Echo PDU received on Ethernet3 from 39.0001.2222.2222.2222.00!
The following lines indicate that the router at source NSAP 39.0001.3333.3333.3333.00 is sending a CLNS echo packet to destination NSAP 39.0001.2222.2222.2222.00 via an IS with system ID 2222.2222.2222. The packet is being sent on Ethernet interface 3, with a MAC address of 0000.0c00.3a18.
```

```
CLNS: Sending from 39.0001.3333.3333.3333.00 to 39.0001.2222.2222.2222.00 via 2222.2222.2222 (Ethernet3 0000.0c00.3a18)
The following lines indicate that a CLNS echo packet 117 bytes in size is being sent from source NSAP 39.0001.2222.2222.2222.00 to destination NSAP 49.0002.0001.AAAA.AAAA.AAAA.00 via the router at NSAP 49.0002. The packet is being forwarded on the Ethernet interface 3, with a MAC address of 0000.0c00.b5a3.
```

```
CLNS: Forwarding packet size 117 from 39.0001.2222.2222.2222.00 to 49.0002.0001.AAAA.AAAA.AAAA.00 via 49.0002 (Ethernet3 0000.0c00.b5a3)
The following lines indicate that the router sent a redirect packet on the Ethernet interface 3 to the NSAP 39.0001.2222.2222.2222.00 at MAC address 0000.0c00.3a18 to indicate that NSAP 49.0002.0001.AAAA.AAAA.AAAA.00 can be reached at MAC address 0000.0c00.b5a3.
```

```
CLNS: RD Sent on Ethernet3 to 39.0001.2222.2222.2222.00 @ 0000.0c00.3a18, redirecting 49.0002.0001.AAAA.AAAA.AAAA.00 to 0000.0c00.b5a3
```
**debug clns igrp packets**

To display debugging information on all ISO-IGRP routing activity, use the `debug clns igrp packets` privileged EXEC command. The `no` form of this command disables debugging output.

```
debug clns igrp packets
no debug clns igrp packets
```

**Syntax Description**

This command has no arguments or keywords.

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(13)T</td>
<td>This command is no longer supported in Cisco IOS Mainline or Technology-based (T) releases. It may continue to appear in Cisco IOS 12.2S-family releases.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug clns igrp packets` command:

```
Router# debug clns igrp packets
ISO-IGRP: Hello sent on Ethernet3 for DOMAIN_green1
ISO-IGRP: Received hello from 39.0001.3333.3333.3333.00, (Ethernet3), ht 51
ISO-IGRP: Originating level 1 periodic update
ISO-IGRP: Advertise dest: 2222.2222.2222
ISO-IGRP: Sending update on interface: Ethernet3
ISO-IGRP: Originating level 2 periodic update
ISO-IGRP: Advertise dest: 0001
ISO-IGRP: Sending update on interface: Ethernet3
ISO-IGRP: Received update from 3333.3333.3333 (Ethernet3)
ISO-IGRP: Opcode: area
ISO-IGRP: Received level 2 adv for 0001 metric 1100
ISO-IGRP: Opcode: station
ISO-IGRP: Received level 1 adv for 3333.3333.3333 metric 1100
The following line indicates that the router is sending a hello packet to advertise its existence in the DOMAIN_green1 domain:

ISO-IGRP: Hello sent on Ethernet3 for DOMAIN_green1
The following line indicates that the router received a hello packet from a certain network service access point (NSAP) on Ethernet interface 3. The hold time for this information is 51 seconds.

ISO-IGRP: Received hello from 39.0001.3333.3333.3333.00, (Ethernet3), ht 51
The following lines indicate that the router is generating a Level 1 update to advertise reachability to destination NSAP 2222.2222.2222 and that it is sending that update to all systems that can be reached through Ethernet interface 3:

ISO-IGRP: Originating level 1 periodic update
ISO-IGRP: Advertise dest: 2222.2222.2222
ISO-IGRP: Sending update on interface: Ethernet3
```
The following lines indicate that the router is generating a Level 2 update to advertise reachability to destination area 1 and that it is sending that update to all systems that can be reached through Ethernet interface 3:

ISO-IGRP: Originating level 2 periodic update  
ISO-IGRP: Advertise dest: 0001  
ISO-IGRP: Sending update on interface: Ethernet3

The following lines indicate that the router received an update from NSAP 3333.3333.3333 on Ethernet interface 3. This update indicated the area that the router at this NSAP could reach.

ISO-IGRP: Received update from 3333.3333.3333 (Ethernet3)  
ISO-IGRP: Opcode: area

The following lines indicate that the router received an update advertising that the source of that update can reach area 1 with a metric of 1100. A station opcode indicates that the update included system addresses.

ISO-IGRP: Received level 2 adv for 0001 metric 1100  
ISO-IGRP: Opcode: station
debug clns packet

To display information about packet receipt and forwarding to the next interface, use the `debug clns packet` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

**debug clns packet**

**no debug clns packet**

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Examples**

The following is sample output from the `debug clns packet` command:

```
Router# debug clns packet
CLNS: Forwarding packet size 157
      from 47.0023.0001.0000.0000.0000.0001.1920.3614.3002.00 STUPI-RBS
      to 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00
      via 1600.8906.4017 (Ethernet0 0000.0c00.bda8)
CLNS: Echo PDU received on Ethernet0 from
      47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00!
CLNS: Sending from 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00 to
      47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00
      via 1600.8906.4017 (Ethernet0 0000.0c00.bda8)
```

In the following lines, the first line indicates that a Connectionless Network Service (CLNS) packet of size 157 bytes is being forwarded. The second line indicates the network service access point (NSAP) and system name of the source of the packet. The third line indicates the destination NSAP for this packet. The fourth line indicates the next hop system ID, interface, and subnetwork point of attachment (SNPA) of the router interface used to forward this packet.

```
CLNS: Forwarding packet size 157
      from 47.0023.0001.0000.0000.0000.0001.1920.3614.3002.00 STUPI-RBS
      to 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00
      via 1600.8906.4017 (Ethernet0 0000.0c00.bda8)
```

In the following lines, the first line indicates that the router received an echo protocol data unit (PDU) on the specified interface from the source NSAP. The second line indicates which source NSAP is used to send a CLNS packet to the destination NSAP, as shown on the third line. The fourth line indicates the next hop system ID, interface, and SNPA of the router interface used to forward this packet.

```
CLNS: Echo PDU received on Ethernet0 from
      47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00!
CLNS: Sending from 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00 to
      47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00
      via 1600.8906.4017 (Ethernet0 0000.0c00.bda8)
```
**debug clns routing**

To display debugging information for all Connectionless Network Service (CLNS) routing cache updates and activities involving the CLNS routing table, use the `debug clns routing` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug clns routing
no debug clns routing
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Examples**

The following is sample output from the `debug clns routing` command:

```
Router# debug clns routing
CLNS-RT: cache increment:17
CLNS-RT: Add 47.0023.0001.0000.0000.0003.0001 to prefix table, next hop 1920.3614.3002
CLNS-RT: Aging cache entry for: 47.0023.0001.0000.0000.0003.0001.1920.3614.3002.06
CLNS-RT: Deleting cache entry for: 47.0023.0001.0000.0000.0003.0001.1920.3614.3002.06

The following line indicates that a change to the routing table has resulted in an addition to the fast-switching cache:

```
CLNS-RT: cache increment:17
```

The following line indicates that a specific prefix route was added to the routing table, and indicates the next hop system ID to that prefix route. In other words, when the router receives a packet with the prefix 47.0023.0001.0000.0000.0003.0001 in the destination address of that packet, it forwards that packet to the router with the MAC address 1920.3614.3002.

```
CLNS-RT: Add 47.0023.0001.0000.0000.0003.0001 to prefix table, next hop 1920.3614.3002
```

The following lines indicate that the fast-switching cache entry for a certain network service access point (NSAP) has been invalidated and then deleted:

```
CLNS-RT: Aging cache entry for: 47.0023.0001.0000.0000.0003.0001.1920.3614.3002.06
CLNS-RT: Deleting cache entry for: 47.0023.0001.0000.0000.0003.0001.1920.3614.3002.06
```
**debug cls message**

To display information about Cisco Link Services (CLS) messages, use the `debug cls message` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cls message
no debug cls message
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

The `debug cls message` command displays the primitives (state), selector, header length, and data size.

**Examples**

The following is sample output from the `debug cls message` command. For example, CLS-->DLU indicates the direction of the flow that is described by the status. From CLS to dependent logical unit (DLU), a request was established to the connection endpoint. The header length is 48 bytes, and the data size is 104 bytes.

```
Router# debug cls message
(FRAS Daemon:CLS-->DLU):
   ID_STN.Ind to uSAP: 0x607044C4 sel: LLC hlen: 40, dlen: 54
(FRAS Daemon:CLS-->DLU):
   ID_STN.Ind to uSAP: 0x6071B054 sel: LLC hlen: 40, dlen: 46
(FRAS Daemon:DLU-->SAP):
   REQ_OPNSTN.Req to pSAP: 0x608021F4 sel: LLC hlen: 48, dlen: 104
(FRAS Daemon:CLS-->DLU):
   REQ_OPNSTN.Cfm(NO_REMOTE_STN) to uCEP: 0x607FFE84 sel: LLC hlen: 48, dlen: 104
```

The status possibilities include the following: enabled, disabled, request open station, open station, close station, activate SA, deactivate service access point (SAP), XID, exchange identification (XID) station, connect station, signal station, connect, disconnect, connected, data, flow, unnumbered data, modify SAP, test, activate ring, deactivate ring, test station, and unnumbered data station.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug fras error</code></td>
<td>Displays information about FRAS protocol errors.</td>
</tr>
<tr>
<td><code>debug fras message</code></td>
<td>Displays general information about FRAS messages.</td>
</tr>
<tr>
<td><code>debug fras state</code></td>
<td>Displays information about FRAS data-link control state changes.</td>
</tr>
</tbody>
</table>

Cisco IOS Debug Command Reference - Commands A through D

394
debug cls vdlc

To display information about Cisco Link Services (CLS) Virtual Data Link Control (VDLC), use the debug cls vdlc command in privileged EXEC mode. To disable debugging output, use the no form of this command.

debug cls vdlc
no debug cls vdlc

Syntax Description
This command has no arguments or keywords.

Command Modes
Privileged EXEC

Usage Guidelines
The debug cls message command displays primitive state transitions, selector, and source and destination MAC and service access points (SAPs).
Also use the show cls command to display additional information on CLS VDLC.

Caution
Use the debug cls vdlc command with caution because it can generate a substantial amount of output.

Examples
The following messages are sample output from the debug cls vdlc command. In the following scenario, the systems network architecture (SNA) service point—also called nativeservicepoint (NSP)—is setting up two connections through VDLC and data-link switching (DLSw): one from NSP to VDLC and one from DLSw to VDLC. VDLC joins the two.
The NSP initiates a connection from 4000.05d2.0001 as follows:

VDLC: Req Open Stn Req PSap 0x7ACE00, port 0x79DF98
4000.05d2.0001(0C)-->4000.1060.1000(04)
In the next message, VDLC sends a test station request to DLSw for destination address 4000.1060.1000.

VDLC: Send UFrame E3: 4000.05d2.0001(0C)-->4000.1060.1000(00)
In the next two messages, DLSw replies with test station response, and NSP goes to a half-open state. NSP is waiting for the DLSw connection to VDLC.

VDLC: Sap to Sap TEST_STN_RSP VSap 0x7B68C0 4000.1060.1000(00)-->4000.05d2.0001(0C)
VDLC: 4000.05d2.0001(0C)-->4000.1060.1000(04): VDLC_OPENING-->VDLC_HALF_OPEN
The NSP sends an exchange identification (XID) and changes state as follows:

VDLC: 4000.05d2.0001(0C)-->4000.1060.1000(04): VDLC_HALF_OPEN-->VDLC_XID_RSP_PENDING
VDLC: CEP to SAP ID_REQ 4000.05d2.0001(0C)-->4000.1060.1000(04) via bridging SAP (DLSw)
In the next several messages, DLSw initiates its connection, which matches the half-open connection with NSP:

VDLC: Req Open Stn Req PSap 0x7B68C0, port 0x7992A0
4000.1060.1000(04)-->4000.05d2.0001(0C)
VDLC: two-way connection established
VDLC: 4000.1060.1000(04)->4000.05d2.0001(0C): VDLC_IDLE->VDLC_OPEN

In the following messages, DLSw sends an XID response, and the NSP connection goes from the state XID Response Pending to Open. The XID exchange follows:

VDLC: CEP to CEP ID_RSP 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_OPEN->VDLC_XID_RSP_PENDING
VDLC: CEP to CEP ID_REQ 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: CEP to CEP ID_RSP 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_OPEN->VDLC_XID_RSP_PENDING
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: CEP to CEP ID_REQ 4000.05d2.0001(0C)->4000.1060.1000(04)
VDLC: CEP to CEP ID_RSP 4000.05d2.0001(0C)->4000.1060.1000(04)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_OPEN->VDLC_XID_RSP_PENDING
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: CEP to CEP ID_REQ 4000.05d2.0001(0C)->4000.1060.1000(04)
VDLC: CEP to CEP ID_RSP 4000.05d2.0001(0C)->4000.1060.1000(04)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_OPEN->VDLC_XID_RSP_PENDING
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: CEP to CEP ID_REQ 4000.05d2.0001(0C)->4000.1060.1000(04)
VDLC: CEP to CEP ID_RSP 4000.05d2.0001(0C)->4000.1060.1000(04)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_OPEN->VDLC_XID_RSP_PENDING
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: CEP to CEP ID_REQ 4000.05d2.0001(0C)->4000.1060.1000(04)
VDLC: CEP to CEP ID_RSP 4000.05d2.0001(0C)->4000.1060.1000(04)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_OPEN->VDLC_XID_RSP_PENDING
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN

When DLSw is ready to connect, the front-end processor (FEP) sends a set asynchronous balanced mode extended (SABME) command as follows:

VDLC: CEP to CEP CONNECT_REQ 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN

In the following messages, NSP accepts the connection and sends an unnumbered acknowledgment (UA) to the FEP:

VDLC: CEP to CEP CONNECT_RSP 4000.05d2.0001(0C)->4000.1060.1000(04)
VDLC: FlowReq QUENCH OFF 4000.1060.1000(04)->4000.05d2.0001(0C)

The following messages show the data flow:

VDLC: DATA 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: DATA 4000.05d2.0001(0C)->4000.1060.1000(04)

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cls message</td>
<td>Displays information about CLS messages.</td>
</tr>
</tbody>
</table>
debug cme-xml

To generate debug messages for the Cisco Unified CallManager Express XML application, use the `debug cme-xml` command in privileged EXEC mode. To disable debugging, use the `no` form of the command.

```
debug cme-xml
no debug cme-xml
```

**Syntax Description**

This command has no keywords or arguments.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Cisco IOS Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(4)XC</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(9)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(9)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The `show fb-its-log` command displays the contents of the XML event table.

**Examples**

The following example shows the progress of an XML request that has been sent to Cisco Unified CallManager Express:

```
Router# debug cme-xml
*Aug 5 06:27:25.727: doc 0x63DB85E8, doc->doc_type 3, req 0x655FDCD0
*Aug 5 06:27:25.727: CME extracted a XML document
*Aug 5 06:27:25.727: Response buffer 0x63DCFD58, len = 4096
*Aug 5 06:27:25.727: First Tag ID SOAP_HEADER_TAG_ID 58720257
*Aug 5 06:27:25.727: First Attribute ID SOAP_ENV_ATTR 50331649
*Aug 5 06:27:25.727: cme_xml_process_soap_header
*Aug 5 06:27:25.727: cme_xml_process_soap_body
*Aug 5 06:27:25.731: cme_xml_process_axl
*Aug 5 06:27:25.731: cme_xml_process_request
*Aug 5 06:27:25.731: cme_xml_process_ISgetGlobal
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>show fb-its-log</td>
<td>Displays Cisco Unified CallManager Express XML API information.</td>
</tr>
</tbody>
</table>
debug cns config

To turn on debugging messages related to the Cisco Networking Services (CNS) configuration agent, use the `debug cns config` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
dump cns config {agent| all| connection| notify}
no dump cns config {agent| all| connection| notify}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agent</td>
<td>Displays debugging messages related to the CNS configuration agent.</td>
</tr>
<tr>
<td>all</td>
<td>Displays all debugging messages.</td>
</tr>
<tr>
<td>connection</td>
<td>Displays debugging messages related to configuration connections.</td>
</tr>
<tr>
<td>notify</td>
<td>Displays debugging messages related to CNS configurations.</td>
</tr>
</tbody>
</table>

### Command Default

No default behavior or values

### Command Modes

Privileged EXEC (#)

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(2)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.0(18)ST</td>
<td>This command was integrated into Cisco IOS Release 12.0(18)ST.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td>This command was implemented on the Cisco 2600 and Cisco 3600 series.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2(33)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SB.</td>
</tr>
<tr>
<td>12.2(33)SXI</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SXI.</td>
</tr>
</tbody>
</table>

### Usage Guidelines

Use this command to turn on or turn off debugging messages related to the CNS Configuration Agent.
In the following example, debugging messages are enabled for CNS configuration processes:

```plaintext
Router# debug cns config all
00:04:09: config_id_get: entered
00:04:09: config_id_get: Invoking cns_id_mode_get()
00:04:09: config_id_get: cns_id_mode_get() returned INTERNAL
00:04:09: config_id_get: successful exit cns_config_id=minna1,cns_config_id_len=6
00:04:09: cns_establish_connect_intf(): The device is already connected with the config server
00:04:09: cns_initial_config_agent(): connecting with port 80
00:04:09: pull_config() entered
00:04:09: cns_config_id(): returning config_id=minna1
00:04:09: Message finished 150 readend
00:04:09: %CNS-4-NOTE: SUCCESSFUL COMPLETION -Process= "CNS Initial Configuration Agent", ipl= 0, pid= 82
00:04:10: %SYS-5-CONFIG_I: Configured from console by console
```

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cns config cancel</td>
<td>Cancels a CNS configuration.</td>
</tr>
<tr>
<td>cns config initial</td>
<td>Starts the initial CNS Configuration Agent.</td>
</tr>
<tr>
<td>cns config partial</td>
<td>Starts the partial CNS Configuration Agent.</td>
</tr>
<tr>
<td>cns config retrieve</td>
<td>Gets the configuration of a routing device using CNS.</td>
</tr>
<tr>
<td>debug cns event</td>
<td>Displays information on CNS events.</td>
</tr>
<tr>
<td>debug cns exec</td>
<td>Displays information on CNS management.</td>
</tr>
<tr>
<td>debug cns xml-parser</td>
<td>Displays information on the CNS XML parser.</td>
</tr>
<tr>
<td>show cns config</td>
<td>Displays information about the CNS Configuration Agent.</td>
</tr>
</tbody>
</table>
debug cns events

To turn on debugging messages related to the Cisco Networking Services (CNS) Event Gateway, use the `debug cns event` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
depth cns event {agent| all| connection| subscriber}
no debug cns event {agent| all| connection| subscriber}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>agent</code></td>
<td>Displays debugging messages related to the event agent.</td>
</tr>
<tr>
<td><code>all</code></td>
<td>Displays all debugging messages.</td>
</tr>
<tr>
<td><code>connection</code></td>
<td>Displays debugging messages related to event connections.</td>
</tr>
<tr>
<td><code>subscriber</code></td>
<td>Displays debugging messages related to subscribers.</td>
</tr>
</tbody>
</table>

### Command Default

No default behavior or values

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(2)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.0(18)ST</td>
<td>This command was integrated into the Cisco IOS Release 12.0(18)ST.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td>This command was implemented on Cisco 2600 series and Cisco 3600 series routers.</td>
</tr>
<tr>
<td>12.2(31)SB2</td>
<td>This command was integrated into Cisco IOS Release 12.2(31)SB2.</td>
</tr>
</tbody>
</table>

### Usage Guidelines

Use this command to turn on or turn off debugging messages related to the CNS Event Gateway.

### Examples

In the following example, debugging messages about all CNS Events are enabled:

```
Router# debug cns event all
```

```
00:09:14: %CNS-4-NOTE: SUCCESSFUL_COMPLETION
  -Process= "CNS Initial Configuration Agent", ipl= 0, pid= 82
00:09:14: event_agent():event_agent starting ..
```
00:09:14: event_agent_open_connection(): attempting socket connect to Primary Gateway
00:09:14: event_agent_open_connection(): cns_socket_connect() succeeded: return_code=0
00:09:14: event_agent_open_connection(): timeout_len=1: ka_total_timeout =0:
               total_timeout=0
00:09:14: event_id_get: entered
00:09:14: event_id_get: Invoking cns_id_mode_get()
00:09:14: event_id_get: cns_id_mode_get() returned INTERNAL
00:09:14: event_id_get: successful exit cns_event_id=test1, cns_event_id_len=5
00:09:14: ea_devid_send(): devid sent DUMP OF DEVID MSG
82C920A0: 00120000 00010774 .......t
82C920B0: 65737431 00000402 020000 est1.......
00:09:14: event_agent_get_input(): cli timeout=0: socket:0x0
00:09:14: process_all_event_agent_event_items(): process_get_wakeup(&major, &minor)=TRUE:
       major=0
.
.
00:09:14: add_subjectANDhandle_to_subject_table(): p_subject_entry=0x82E3EEDC:
p_subject_entry_list=0x82619CD8
00:09:14: add_subjectANDhandle_to_subject_table(): add 'user_entry' entry succeeded:
user entry =0x82C92AF4: queue_handle=0x82C913FC
00:09:14: %SYS-5-CONFIG_I: Configured from console by console

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cns event</td>
<td>Configures the CNS Event Gateway.</td>
</tr>
<tr>
<td>show cns event</td>
<td>Displays information about the CNS Event Agent.</td>
</tr>
</tbody>
</table>
**debug cns exec**

To display debugging messages about Cisco Networking Services (CNS) exec agent services, use the `debug cns exec` command in privileged EXEC mode. To disable debugging output, use the `no` or `undebug` form of this command.

```
debug cns exec {agent| all| decode| messages}
no debug cns exec {agent| all| decode| messages}
undebug cns exec {agent| all| decode| messages}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>agent</code></td>
<td>Displays debugging messages related to the exec agent.</td>
</tr>
<tr>
<td><code>all</code></td>
<td>Displays all debugging messages.</td>
</tr>
<tr>
<td><code>decode</code></td>
<td>Displays debugging messages related to image agent connections.</td>
</tr>
<tr>
<td><code>messages</code></td>
<td>Displays debugging output related to messages generated by exec agent services.</td>
</tr>
</tbody>
</table>

### Command Default

Debugging output is disabled.

### Command Modes

Privileged EXEC (♯)

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(1)</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(25)S</td>
<td>This command was integrated into Cisco IOS Release 12.2(25)S.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2(33)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SB.</td>
</tr>
<tr>
<td>12.2(33)SXI</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SXI.</td>
</tr>
</tbody>
</table>

### Usage Guidelines

Use the `debug cns exec` command to troubleshoot CNS exec agent services.
Examples

The following example shows a debugging message for the CNS exec agent when a response has been posted to HTTP:

```
Router# debug cns exec agent
4d20h: CNS exec agent: response posted
```

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cns exec</td>
<td>Configures CNS Exec Agent services.</td>
</tr>
</tbody>
</table>
**debug cns image**

To display debugging messages about Cisco Networking Services (CNS) image agent services, use the `debug cns image` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cns image {agent| all| connection| error}
no debug cns image {agent| all| connection| error}
```

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Agent Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>agent</code></td>
<td>Displays debugging messages related to the image agent.</td>
</tr>
<tr>
<td><code>all</code></td>
<td>Displays all debugging messages.</td>
</tr>
<tr>
<td><code>connection</code></td>
<td>Displays debugging messages related to image agent connections.</td>
</tr>
<tr>
<td><code>error</code></td>
<td>Displays debugging messages related to errors generated by image agent services.</td>
</tr>
</tbody>
</table>

**Command Default**

If no keyword is specified, all debugging messages are displayed.

**Command Modes**

Privileged EXEC (`#`)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(1)</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(31)SB2</td>
<td>This command was integrated into Cisco IOS Release 12.2(31)SB2.</td>
</tr>
<tr>
<td>12.2(33)SRB</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRB.</td>
</tr>
<tr>
<td>12.2(33)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SB.</td>
</tr>
<tr>
<td>12.2(33)SXI</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SXI.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use the `debug cns image` command to troubleshoot CNS image agent services.
debug cns management

To display information about Cisco Networking Services (CNS) management, use the debug cns management command in privileged EXEC mode. To disable debugging output, use the no form of this command.

debug cns management {snmp| xml}
no debug cns management {snmp| xml}

Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>snmp</td>
<td>Displays debugging messages related to nongranular Simple Network Management Protocol (SNMP) encapsulated CNS-management events.</td>
</tr>
<tr>
<td>xml</td>
<td>Displays debugging messages related to granular eXtensible Markup Language (XML) encapsulated CNS-management events.</td>
</tr>
</tbody>
</table>

Command Modes Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(8)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

Examples

In the following example, debugging messages about SNMP- and XML-encapsulated CNS-management events are enabled:

```
Router# debug cns management snmp
Router# debug cns management xml
Router# show debugging
CNS Management (SNMP Encapsulation) debugging is on
CNS Management (Encap XML) debugging is on
Router# show running-config | include cns
cns mib-access encapsulation snmp
cns mib-access encapsulation xml
cns notifications encapsulation snmp
cns notifications encapsulation xml
cns event 10.1.1.1 11011
Router#
00:12:50: Enqueued a notification in notif_q
00:12:50: ea_produce succeeded Subject:cisco.cns.mibaccess:notification Message Length:385
00:12:50: Trap sent via CNS Transport Mapping.
Router#
00:13:31: Response sent via CNS Transport Mapping.
Router#
00:14:38: Received a request
00:14:38: ea_produce succeeded Subject:cisco.cns.mibaccess:response Message Length:241
```
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cns event</code></td>
<td>Configures the CNS event gateway, which provides CNS event services to Cisco IOS clients.</td>
</tr>
<tr>
<td><code>debug cns config</code></td>
<td>Displays information on CNS configurations.</td>
</tr>
<tr>
<td><code>debug cns xml-parser</code></td>
<td>Displays information on the CNS XML parser.</td>
</tr>
<tr>
<td><code>show debugging</code></td>
<td>Displays information about the types of debugging that are enabled for your router.</td>
</tr>
<tr>
<td><code>show running-config</code></td>
<td>Displays the current running configuration.</td>
</tr>
</tbody>
</table>
**debug cns xml**

To turn on debugging messages related to the Cisco Networking Services (CNS) eXtensible Markup Language (XML) parser, use the `debug cns xml` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cns xml {all| decode| dom| parser}
no debug cns xml {all| decode| dom| parser}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Displays all CNS debugging.</td>
</tr>
<tr>
<td>decode</td>
<td>Reports usage of common XML decoding library functions by applications and reports the decoded contents.</td>
</tr>
<tr>
<td>dom</td>
<td>Displays failures in the Document Object Model (DOM) infrastructure messages tree built by the XML parser.</td>
</tr>
<tr>
<td>parser</td>
<td>Displays failures and progress of the parsing of an XML message by the XML parser.</td>
</tr>
</tbody>
</table>

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(2)T</td>
<td>This command was introduced. This command replaces the <code>debug cns xml-parser</code> command.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
<tr>
<td>12.2(33)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SB.</td>
</tr>
<tr>
<td>12.2(33)SXI</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SXI.</td>
</tr>
</tbody>
</table>

### Examples

In the following example, debugging messages for the CNS XML parser are enabled:

```
Router# debug cns xml parser
```
00:12:05: Registering tag <config-server>
00:12:05: Registering tag <server-info>
00:12:05: Registering tag <ip-address>
00:12:05: Registering tag <web-page>
00:12:05: Registering tag <config-event>
00:12:05: Registering tag <identifier>
00:12:05: Registering tag <config-data>
00:12:05: Registering tag <cli>
00:12:05: Registering tag <error-info>
00:12:05: Registering tag <error-message>
00:12:05: Registering tag <line-number>
00:12:05: Registering tag <exec-cmd-event>
00:12:05: Registering tag <identifier-exec>
00:12:05: Registering tag <event-response>
00:12:05: Registering tag <reply-subject>
00:12:05: Registering tag <server-response>
00:12:05: Registering tag <ip-address-exec>
00:12:05: Registering tag <port-number>
00:12:05: Registering tag <url>
00:12:05: Registering tag <cli-exec>
00:12:05: Registering tag <config-pwd>
00:12:06: Pushing tag <config-data> on to stack
00:12:06: open tag is <config-data>
00:12:06: Pushing tag <config-id> on to stack
00:12:06: open tag is <config-id>
00:12:06: Popping tag <config-id> off stack
00:12:06: close tag is </config-id>
00:12:06: Pushing tag <cli> on to stack
00:12:06: open tag is <cli>
00:12:06: Popping tag <cli> off stack
00:12:06: close tag is </cli>
00:12:06: Popping tag <config-data> off stack
00:12:06: close tag is </config-data>
00:12:06: %CNS-4-NOTE: SUCCESSFUL_COMPLETION
-Process= "CNS Initial Configuration Agent", ipl= 0, pid= 96

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cns event</td>
<td>Configures the CNS Event Gateway.</td>
</tr>
<tr>
<td>show cns event</td>
<td>Displays information about the CNS Event Agent.</td>
</tr>
</tbody>
</table>
**debug cns xml-parser**

**Note**
Effective with Cisco IOS Release 12.3(2)T, the `debug cns xml-parser` command is replaced by the `debug cns xml` command. See the `debug cns xml` command for more information.

To turn on debugging messages related to the Cisco Networking Services (CNS) eXtensible Markup Language (XML) parser, use the `debug cns xml-parser` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

`debug cns xml-parser`
`no debug cns xml-parser`

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

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(2)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.0(18)ST</td>
<td>This command was integrated into Cisco IOS Release 12.0(18)ST.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td>This command was implemented on the Cisco 2600 and Cisco 3600 series.</td>
</tr>
<tr>
<td>12.3(2)T</td>
<td>This command was replaced by the <code>debug cns xml</code> command.</td>
</tr>
</tbody>
</table>

**Examples**
In the following example, debugging messages for the CNS XML parser are enabled:

```
Router# debug cns xml-parser
00:12:05: Registering tag <config-server>
00:12:05: Registering tag <server-info>
00:12:05: Registering tag <ip-address>
00:12:05: Registering tag <web-page>
00:12:05: Registering tag <config-event>
00:12:05: Registering tag <identifier>
00:12:05: Registering tag <config-id>
00:12:05: Registering tag <config-data>
00:12:05: Registering tag <cli>
00:12:05: Registering tag <error-info>
00:12:05: Registering tag <error-message>
00:12:05: Registering tag <line-number>
00:12:05: Registering tag <exec-cmd-event>
00:12:05: Registering tag <identifier-exec>
00:12:05: Registering tag <event-response>
```
Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cns event</td>
<td>Configures the CNS Event Gateway.</td>
</tr>
<tr>
<td>show cns event</td>
<td>Displays information about the CNS Event Agent.</td>
</tr>
</tbody>
</table>
**debug compress**

To debug compression, enter the `debug compress` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug compress
no debug compress
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Disabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command to display output from the compression and decompression configuration you made. Live traffic must be configured through the Cisco 2600 access router with a data compression Advanced Interface Module (AIM) installed for this command to work.

**Examples**

The following example is output from the `debug compress` command, which shows that compression is taking place on a Cisco 2600 access router using data compression AIM hardware compression is configured correctly:

```
Router# debug compress
COMPRESS debugging is on
compr-in: pak:0x810C6B10 npart:0 size:103
pak:0x810C6B10 start:0x02406BD4 size:103 npart:0
compr-out: pak:0x8118C8B8 stat:0x00000000 npart:1 size:71 lcb:0xED
pak:0x8118C8B8 start:0x0259CD3E size:71 npart:1
mp:0x8118A990 start:0x0259CD3E size:71
decmp-in: pak:0x81128B78 start:0x0255AF44 size:42 npart:1 hdr:0xC035
pak:0x81128B78 start:0x0255AF44 size:42 npart:1
mp:0x81174480 start:0x0255AF44 size:42
decmp-out: pak:0x8118C8B8 start:0x025B2C42 size:55 npart:1 stat:0
pak:0x8118C8B8 start:0x025B2C42 size:55 npart:1
mp:0x8118B700 start:0x025B2C42 size:55
```

The table below describes the significant fields shown in the display.
### Table 55: debug compress Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>compr-in</td>
<td>Indicates that a packet needs to be compressed.</td>
</tr>
<tr>
<td>compr-out</td>
<td>Indicates completion of compression of packet.</td>
</tr>
<tr>
<td>decmp-in</td>
<td>Indicates receipt of a compressed packet that needs to be decompressed.</td>
</tr>
<tr>
<td>decmp-out</td>
<td>Indicates completion of decompression of a packet.</td>
</tr>
<tr>
<td>pak:0x810C6B10</td>
<td>Provides the address in memory of a software structure that describes the compressed packet.</td>
</tr>
<tr>
<td>start:0x02406BD4 size:103 npart:0</td>
<td>The &quot;npart:0&quot; indicates that the packet is contained in a single, contiguous area of memory. The start address of the packet is 0x02406bd4 and the size of the packet is 103.</td>
</tr>
<tr>
<td>start:0x0259CD3E size:71 npart:1</td>
<td>The &quot;npart:1&quot; indicates that the packet is contained in 1 or more regions of memory. The start address of the packet is 0x0259CD3E and the size of the packet is 71.</td>
</tr>
<tr>
<td>mp:0x8118A980 start:0x0259CD3e size:71</td>
<td>Describes one of these regions of memory.</td>
</tr>
<tr>
<td>mp:0x8118A980</td>
<td>Provides the address of a structure describing this region.</td>
</tr>
<tr>
<td>start 0x0259CD3E</td>
<td>Provides the address of the start of this region.</td>
</tr>
</tbody>
</table>

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug frame-relay</td>
<td>Displays debugging information about the packets that are received on a Frame Relay interface.</td>
</tr>
<tr>
<td>debug ppp</td>
<td>Displays information on traffic and exchanges in an internetwork implementing the PPP.</td>
</tr>
<tr>
<td>show compress</td>
<td>Displays compression statistics.</td>
</tr>
<tr>
<td>show diag</td>
<td>Displays hardware information including DRAM, SRAM, and the revision-level information on the line card.</td>
</tr>
</tbody>
</table>
debug condition

To filter debugging output for certain debug commands on the basis of specified conditions, use the debug condition command in privileged EXEC mode. To remove the specified condition, use the no form of this command.

```
debug condition {called dial-string| caller dial-string| calling tidisn string| domain domain-name| fex fabric-extender-number module module-number| interface interface-id| ip ip-address| mac-address hexadecimal-MAC-address| module module-number| portbundle ip ip-address bundle bundle-number| session-id session-number| switch switch-number module module-number| username username| vcid ve-id| vlan vlan-id}

no debug condition {condition-id| all}
```

**Syntax Description**

- **called dial-string**
  Filters output on the basis of the called party number.

- **caller dial-string**
  Filters output on the basis of the calling party number.

- **calling tidisn string**
  Filters debug messages for general packet radio service (GPRS) tunneling protocol (GTP) processing on the gateway GPRS support node (GGSN) based on the tunnel identifier (TID) or international mobile system identifier (IMSI) in a Packet Data Protocol (PDP) Context Create Request message.

- **domain domain-name**
  Filters output on the basis of the specified domain.

- **fex fabric-extender-number module module-number**
  Filters output on the basis of the specified fabric extender and specified module.

- **interface interface-id**
  Filters output on the basis of the specified interface ID.

- **ip ip-address**
  Filters output on the basis of the specified IP address.

- **mac-address hexadecimal-MAC-address**
  Filters messages on the specified MAC address.

- **module module-number**
  Filters output on the basis of the specified module number.

- **portbundle ip ip-address**
  Filters output on the basis of the port-bundle host key (PBHK) that uniquely identifies the session.

- **bundle bundle-number**
  Specifies the port bundle.

- **session-id session-number**
  Filters output on the specified Intelligent Service Architecture (ISA) session identifier.
**debug condition**

<table>
<thead>
<tr>
<th>Command Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All debugging messages for enabled protocol-specific <code>debug</code> commands are generated.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command Modes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privileged EXEC (#)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command History</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Release</strong></td>
<td><strong>Modification</strong></td>
</tr>
<tr>
<td>11.3(2)AA</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.0(23)S</td>
<td>This command was modified. The <code>vcid</code> and <code>ip</code> keywords were added to support the debugging of Any Transport over MPLS (AToM) messages.</td>
</tr>
<tr>
<td>12.2(14)S</td>
<td>This command was integrated into Cisco IOS Release 12.2(14)S.</td>
</tr>
<tr>
<td>12.2(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(15)T.</td>
</tr>
<tr>
<td>12.3(2)XB</td>
<td>This command was modified. Support was added on the GGSN.</td>
</tr>
<tr>
<td>12.3(8)T</td>
<td>This command was modified. The calling keyword and <code>tid/imsi-string</code> argument were added.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was modified. The ability to filter output on the following conditions was added: domain, MAC address, PBHK, and ISA session ID.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>15.2(2)T</td>
<td>This command was modified. The <code>vlan vlan-id</code> keyword and argument and the <code>interface interface-id</code> keyword and argument were added.</td>
</tr>
</tbody>
</table>

### Syntax

- `switch switch-number module module-number`
  - Filters output on the basis of the specified switch and specified module.
- `username username`
  - Filters output on the basis of the specified username.
- `vcid vc-id`
  - Filters output on the basis of the specified VC ID.
- `vlan vlan-id`
  - Filters output on the basis of the specified VLAN ID.
- `condition-id`
  - ID number of the condition to be removed.
- `all`
  - Removes all debugging conditions, and conditions specified by the `debug condition interface` command. Use this keyword to disable conditional debugging and reenable debugging for all interfaces.

### Description

- `switch switch-number module module-number`: Filters output on the basis of the specified switch and specified module.
- `username username`: Filters output on the basis of the specified username.
- `vcid vc-id`: Filters output on the basis of the specified VC ID.
- `vlan vlan-id`: Filters output on the basis of the specified VLAN ID.
- `condition-id`: ID number of the condition to be removed.
- `all`: Removes all debugging conditions, and conditions specified by the `debug condition interface` command. Use this keyword to disable conditional debugging and reenable debugging for all interfaces.

### Notes

- This command was introduced.
- This command was modified. The `vcid` and `ip` keywords were added to support the debugging of Any Transport over MPLS (AToM) messages.
- This command was integrated into Cisco IOS Release 12.2(14)S.
- This command was integrated into Cisco IOS Release 12.2(15)T.
- This command was modified. Support was added on the GGSN.
- This command was modified. The calling keyword and `tid/imsi-string` argument were added.
- This command was modified. The ability to filter output on the following conditions was added: domain, MAC address, PBHK, and ISA session ID.
- This command was integrated into Cisco IOS Release 12.2(33)SRA.
- This command was modified. The `vlan vlan-id` keyword and argument and the `interface interface-id` keyword and argument were added.
Use the `debug condition` command to restrict the debug output for some commands. If any `debug condition` commands are enabled, output is generated only for interfaces associated with the specified keyword. In addition, this command enables debugging output for conditional debugging events. Messages are displayed as different interfaces meet specific conditions.

If multiple `debug condition` commands are enabled, output is displayed if at least one condition matches. All the conditions do not need to match.

The `no` form of this command removes the debug condition specified by the condition identifier. The condition identifier is displayed after you use a `debug condition` command or in the output of the `show debug condition` command. If the last condition is removed, debugging output resumes for all interfaces. You will be asked for confirmation before removing the last condition or all conditions.

Not all debugging output is affected by the `debug condition` command. Some commands generate output whenever they are enabled, regardless of whether they meet any conditions.

The following components are supported for Intelligent Service Architecture (ISA) distributed conditional debugging:

- Authentication, authorization, and accounting (AAA) and RADIUS
- ATM components
- Feature Manager
- Policy Manager
- PPP
- PPP over Ethernet (PPPoE)
- Session Manager
- Virtual Private Dialup Network (VPDN)

Ensure that you enable TID/IMSI-based conditional debugging by entering `debug condition calling` before configuring `debug gprs gtp` and `debug gprs charging`. In addition, ensure that you disable the `debug gprs gtp` and `debug gprs charging` commands using the `no debug all` command before disabling conditional debugging using the `no debug condition` command. This will prevent a flood of debugging messages when you disable conditional debugging.

In the following example, the router displays debugging messages only for interfaces that use a username of `user1`. The condition identifier displayed after the command is entered identifies this particular condition.

```
Router# debug condition username user1
Condition 1 set
```

The following example specifies that the router should display debugging messages only for VC 1000:

```
Router# debug condition vcid 1000
Condition 1 set
01:12:32: 1000 Debug: Condition 1, vcid 100 triggered, count 1
01:12:32: 1000 Debug: Condition 1, vcid 100 triggered, count 1
```
The following example enables other debugging commands. These debugging commands will only display information for VC 1000.

Router# debug mpls l2transport vc event
AToM vc event debugging is on
Router# debug mpls l2transport vc fsm
AToM vc fsm debugging is on

The following commands shut down the interface on which VC 1000 is established:

Router(config)# interface serial3/1/0
Router(config-if)# shut

The debugging output shows the change to the interface where VC 1000 is established:

01:15:59: AToM MGR [13.13.13.13, 1000]: Event local down, state changed from established to remote ready
01:15:59: AToM MGR [13.13.13.13, 1000]: Local end down, vc is down
01:15:59: AToM SMGR [13.13.13.13, 1000]: Processing imposition update, vc_handle 6227BCF0, update_action 0, remote_vc_label 18
01:15:59: AToM SMGR [13.13.13.13, 1000]: Processing disposition update, vc_handle 6227BCF0, update_action 0, local_vc_label 755
01:16:01:%LINK-5-CHANGED: Interface Serial3/1/0, changed state to administratively down
01:16:02:%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial3/1/0, changed state to down

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug condition interface</td>
<td>Limits output for some debugging commands based on the interfaces.</td>
</tr>
</tbody>
</table>
debug condition application voice

To display debugging messages for only the specified VoiceXML application, use the `debug condition application voice` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

**Syntax Description**

| `application-name` | Name of the VoiceXML application for which you want to display all enabled debugging messages. |

**Command Default**

If this command is not configured, debugging messages are enabled for all VoiceXML applications.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(11)T</td>
<td>This command was introduced for the Cisco 3640, Cisco 3660, Cisco AS5300, Cisco AS5350, and Cisco AS5400.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

- This command filters debugging output only for the `debugvxml` and `debughttpclient` commands, except that it does not filter output for the `debugxmlerror`, `debugxmlbackground`, `debughttpclienterror`, or `debughttpclientbackground` commands. It does not filter messages for any other debug commands such as the `debugvoipivr` command or the `debugvoiceivr` command.

- This command filters debugging output for all VoiceXML applications except the application named in the command. When this command is configured, the gateway displays debugging messages only for the specified VoiceXML application.

- To filter debugging output with this command, the `<cisco-debug>` element must be enabled in the VoiceXML document. For more information about the `<cisco-debug>` element, refer to the Cisco VoiceXML Programmer’s Guide.

- To see debugging output for VoiceXML applications, you must first configure global `debug` commands such as the `debugvxml` command or the `debughttpclient` command. If no global `debug` commands are turned on, you do not see debugging messages even if the `debug condition application voice` command is configured and the `<cisco-debug>` element is enabled in the VoiceXML document.

- This command can be configured multiple times to display output for more than one application.

- To see which debug conditions have been set, use the `show debug condition` command.
Examples

The following example disables debugging output for all applications except the myapp1 application, if the <cisco-debug> element is enabled in the VoiceXML documents that are executed by myapp1:

```plaintext
Router# debug condition application voice myapp1
```

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug http client</td>
<td>Displays debugging messages for the HTTP client.</td>
</tr>
<tr>
<td>debug vxml</td>
<td>Displays debugging messages for VoiceXML features.</td>
</tr>
<tr>
<td>show debug condition</td>
<td>Displays the debugging conditions that have been enabled for VoiceXML application.</td>
</tr>
</tbody>
</table>
debug condition glbp

To display debugging messages about Gateway Load Balancing Protocol (GLBP) conditions, use the `debug condition glbp` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
deb ug condition glbp interface-type interface-number group [forwarder]
no debug condition glbp interface-type interface-number group [forwarder]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>interface-type</code></td>
<td>Interface type for which output is displayed.</td>
</tr>
<tr>
<td><code>interface-number</code></td>
<td>Interface number for which output is displayed.</td>
</tr>
<tr>
<td><code>group</code></td>
<td>GLBP group number in the range from 0 to 1023.</td>
</tr>
<tr>
<td><code>forwarder</code></td>
<td>(Optional) Number in the range from 1 to 255 used to identify a virtual MAC address.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(14)S</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(15)T.</td>
</tr>
<tr>
<td>12.2(17b)SXA</td>
<td>This command was integrated into Cisco IOS Release 12.2(17b)SXA.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug condition glbp` command:

```
Router# debug condition glbp fastethernet 0/0 10 1
Condition 1 set
5d23h: Fa0/0 GLBP10.1 Debug: Condition 1, glbp Fa0/0 GLBP10.1 triggered, count 1
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug glbp errors</code></td>
<td>Displays debugging messages about GLBP errors.</td>
</tr>
<tr>
<td><code>debug glbp events</code></td>
<td>Displays debugging messages about GLBP events.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>debug glbp packets</td>
<td>Displays debugging messages about GLBP packets.</td>
</tr>
<tr>
<td>debug glbp terse</td>
<td>Displays a limited range of debugging messages about GLBP errors, events, and packets.</td>
</tr>
</tbody>
</table>
debug condition interface

To limit output for some debug commands on the basis of the interface, virtual circuit (VC), or VLAN, use the debug condition interface command in privileged EXEC mode. To remove the interface condition and reset the interface so that it must be triggered by a condition, use the no form of this command.

Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interface-type</td>
<td>Interface type and number. For more information, use the question mark (?) online help function.</td>
</tr>
<tr>
<td>interface-number</td>
<td></td>
</tr>
<tr>
<td>dlici dlci</td>
<td>(Optional) Specifies the data-link connection identifier (DLCI), if the interface to be debugged is a Frame Relay-encapsulated interface.</td>
</tr>
<tr>
<td>vc {vci</td>
<td>vpi/vci}</td>
</tr>
<tr>
<td>vlan-id vlan-id</td>
<td>(Optional) Specifies the VLAN ID, if the interface to be debugged is ATM, Ethernet, Fast Ethernet, or Gigabit Ethernet interface.</td>
</tr>
</tbody>
</table>

Command Default

All debugging messages for enabled debug commands are displayed.

Command Modes

Privileged EXEC (#)

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(11)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.0(28)S</td>
<td>This command was modified. The dlici and vc keywords were added for additional Frame Relay and ATM functionality.</td>
</tr>
<tr>
<td>12.2(25)S</td>
<td>This command was modified. It was integrated into Cisco IOS Release 12.2(25)S.</td>
</tr>
<tr>
<td>12.2(27)SBC</td>
<td>This command was modified. It was integrated into Cisco IOS Release 12.2(27)SBC.</td>
</tr>
</tbody>
</table>
### Usage Guidelines

Use this command to restrict the debugging output for some commands on the basis of an interface or virtual circuit. When you enter this command, debugging output is disabled for all interfaces except the specified interface or virtual circuit. In addition, this command enables conditional debugging to limit output for specific debugging events. Messages are displayed as different interfaces meet specific conditions.

The `no` form of this command performs the following functions:

- Disables the `debug condition interface` command for the specified interface. Output is no longer generated for the interface, assuming that the interface meets no other applicable conditions. If the interface meets other conditions that have been set by another `debug condition` command, debugging output will still be generated for the interface.

- If some other `debug condition` command has been enabled, output is stopped for that interface until the condition is met on the interface. You will be asked for confirmation before the last condition or all conditions are removed.

Not all debugging output is affected by the `debug condition` command. Some commands generate output whenever they are enabled, regardless of whether they meet any conditions. The commands that are affected by the `debug condition` commands are generally related to dial access functions, where a large amount of output is expected. Output from the following commands is controlled by the `debug condition` command:

- `debug aaa`
- `debug atm`
- `debug dialer events`
- `debug frame-relay`
- `debug isdn`
- `debug modem`
- `debug ppp`

One or more ATM-encapsulated interfaces must be enabled, and one or more of the following `debug` commands must be enabled to use conditional debugging with ATM:

- `debug atm arp`
One or more of the following **debug** commands must be enabled to use conditional debugging with Frame Relay:

- **debug frame-relay adjacency**
- **debug frame-relay ipc**
- **debug frame-relay lmi**
- **debug frame-relay packet**
- **debug frame-relay pseudowire**

### Examples

In the following example, only **debug** command output related to serial interface 1 is displayed. The condition identifier for this command is 1.

```
Router# debug condition interface serial 1
Condition 1 set
```

The following example shows how to enable an ATM interface, specifies an IP address for the interface, turns on conditional debugging for that interface with a VPI/VCI pair of 255/62610, and verifies that debugging has been enabled:

```
Router> enable
Password:
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# interface atm 2/0
Router(config-if)# ip address 209.165.201.2 255.255.255.0
Router(config-if)# pvc 255/62610
Router(config-if-atm-vc)# no shutdown
Router(config-if)# exit
Router(config)# exit
2w3d: %SYS-5-CONFIG_I: Configured from console by console
```

```
Router# debug condition interface atm 2/0 vc 255/62610
Condition 1 set
2w3d: ATM VC Debug: Condition 1, atm-vc 255/62610 AT2/0 triggered, count 1
Condition 1: atm-vc 255/62610 AT2/0 (1 flags triggered)
Flags: ATM VC
```

The following example shows how to enable Frame Relay conditional debugging on Frame Relay DLCI 105:

```
Router# debug condition interface serial 4/3 dlci 105
Router# debug frame-relay packet
```

The following example shows how to disable the conditional debugging on VC. A warning message is displayed when the last condition is removed.

```
Router> enable
Router# no debug condition interface atm 1/0 vc 4335
```
This condition is the last interface condition set. Removing all conditions may cause a flood of debugging messages to result, unless specific debugging flags are first removed.
Proceed with removal? [yes/no]: y
Condition 1 has been removed

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug condition</td>
<td>Limits output for some debug commands on the basis of specific conditions.</td>
</tr>
<tr>
<td>debug frame-relay</td>
<td>Displays debugging information about the packets received on a Frame Relay interface.</td>
</tr>
<tr>
<td>show debug condition</td>
<td>Displays the debugging filters that have been enabled for VoiceXML applications, ATM-enabled interfaces, or Frame Relay interfaces</td>
</tr>
</tbody>
</table>
debug condition match-list

To run a filtered debug on a voice call, use the `debug condition match-list` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug condition match-list number {exact-match|partial-match}
no debug condition match-list number {exact-match|partial-match}
```

**Syntax Description**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>number</strong></td>
<td>Numeric label that uniquely identifies the match list. Range is 1 to 16. The number for the match list is set using the <code>callfiltermatch-list</code> command.</td>
</tr>
<tr>
<td><strong>exact-match</strong></td>
<td>All related debug output is filtered until all conditions in the match list are explicitly met. This is the best choice for most situations because the output is the most concise.</td>
</tr>
<tr>
<td><strong>partial-match</strong></td>
<td>No related debug output is filtered until there is a single explicit match failure. As long as zero or more conditions are met, debug output will not be filtered. This choice is useful in debugging call startup problems like digit collection, but is not ideal for many situations because there is much debug output until matches explicitly fail.</td>
</tr>
</tbody>
</table>

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(4)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

In this output example, the following configuration was used:

```
call filter match-list 1 voice
incoming calling-number 8288807
incoming called-number 6560729
incoming port 7/0:D
```
The following is sample output for the `debug condition match-list 1` command. The next several lines match the above conditions.

```
Router# debug condition match-list 1
07:22:19://-1/3C0B9468-15C8-11D4-8013-000A8A389BA8/VTSP:(7/0:D):0:0:0/vtsp_gcfm_incoming_cond_notify:
  add incoming port cond success: 7/0:D
07:22:19://-1/3C0B9468-15C8-11D4-8013-000A8A389BA8/VTSP:(7/0:D):0:0:0/vtsp_gcfm_incoming_cond_notify:
  add incoming dialpeer tag success:1
07:22:19://-1/3C0B9468-15C8-11D4-8013-000A8A389BA8/VTSP:(7/0:D):0:0:0/vtsp_update_dsm_stream_mgr_filter_flag:
  cannot find dsp_stream_mgr_t
07:22:19://-1/3C0B9468-15C8-11D4-8013-000A8A389BA8/VTSP:(7/0:D):0:0:0/vtsp_update_dsm_stream_mgr_filter_flag:
  update dsp_stream_mgr_t debug flag
07:22:19://49/3C0B9468-15C8-11D4-8013-000A8A389BA8/VTSP:(7/0:D):0:0:0/vtsp_insert_cdb:
  cdb 0x6482C518, CallID=49
07:22:19://49/3C0B9468-15C8-11D4-8013-000A8A389BA8/VTSP:(7/0:D):0:0:0/vtsp_do_call_setup_ind:
  Call ID=98357, guid=3C0B9468-15C8-11D4-8013-000A8A389BA8
```

The table below describes the significant fields shown in the display.

**Table 56: debug condition match-list Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3C0B9468-15C8-11D4-8013-000A8A389BA8</td>
<td>Shows the global unique identifier (GUID).</td>
</tr>
<tr>
<td>VTSP:</td>
<td>Identifies the voice telephony service provider (VTSP) module.</td>
</tr>
<tr>
<td>(7/0:D):0:0:0</td>
<td>Shows the port name, channel number, DSP slot, and DSP channel number for the VTSP module.</td>
</tr>
</tbody>
</table>

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>call filter match-list voice</td>
<td>Creates a call filter match list for debugging voice calls.</td>
</tr>
<tr>
<td>debug call filter inout</td>
<td>Displays the debug trace inside the GCFM.</td>
</tr>
<tr>
<td>show call filter match-list</td>
<td>Displays call filter match lists.</td>
</tr>
</tbody>
</table>
debug condition standby

To filter the output of the `debug standby` command on the basis of interface and Hot Standby Router Protocol (HSRP) group number, use the `debug condition standby` command in privileged EXEC mode. To remove the specified filter condition, use the `no` form of this command.

```
debug condition standby interface group-number
no debug condition standby interface group-number
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>interface</code></td>
<td>Filters output on the basis of the interface.</td>
</tr>
<tr>
<td><code>group-number</code></td>
<td>Filters output on the basis of HSRP group number. The range is 0 to 255 for HSRP Version 1 and 0 to 4095 for HSRP Version 2.</td>
</tr>
</tbody>
</table>

### Command Default

All debugging messages for the `debug standby` command are regenerated.

### CommandModes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(2)</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

### Usage Guidelines

Use the `debug condition standby` command to restrict the debug output for the `debug standby` command. If the `debug condition standby` command is enabled, output is generated only for the interfaces and HSRP group numbers specified. The interface you specify must be a valid interface capable of supporting HSRP. The group can be any group (0 to 255 for HSRPv1 and 0 to 4095 for HSRPv2).

Use the `no` form of this command to remove the HSRP debug condition. If the last condition is removed, debugging output resumes for all interfaces. You will be asked for confirmation before removing the last condition or all conditions.

You can set debug conditions for groups that do not exist, which allows you to capture debug information during the initialization of a new group.

You must enable the debug standby command in order for any HSRP debug output to be produced. If you do not configure the `debug condition standby` command after entering the `debug standby` command, then debug output is produced for all groups on all interfaces.
In the following example, the router displays debugging messages only for Ethernet interface 0/0 that are part of HSRP group 23:

```plaintext
Router# debug standby
HSRP debugging is on
Router# debug condition standby ethernet0/0 23
Condition 1 set
00:27:39: HSRP: Et0/0 Grp 23 Hello out 10.0.0.1 Active pri 100 vIP 172.16.6.5
00:27:42: HSRP: Et0/0 Grp 23 Hello out 10.0.0.1 Active pri 100 vIP 172.16.6.5
00:27:45: HSRP: Et0/0 Grp 23 Hello out 10.0.0.1 Active pri 100 vIP 172.16.6.5
00:27:48: HSRP: Et0/0 Grp 23 Hello out 10.0.0.1 Active pri 100 vIP 172.16.6.5
00:27:51: HSRP: Et0/0 Grp 23 Hello out 10.0.0.1 Active pri 100 vIP 172.16.6.5
```

The following example shows how to remove an HSRP debug condition:

```plaintext
Router# no debug condition standby ethernet0/0 23
This condition is the last hsrp condition set.
Removing all conditions may cause a flood of debugging messages to result, unless specific debugging flags are first removed.
Proceed with removal? [yes/no]: Y
Condition 1 has been removed.
```

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug condition interface</td>
<td>Limits output for some debugging commands based on the interfaces.</td>
</tr>
<tr>
<td>debug standby</td>
<td>Displays HSRP state changes.</td>
</tr>
<tr>
<td>debug standby errors</td>
<td>Displays error messages related to HSRP.</td>
</tr>
<tr>
<td>debug standby events</td>
<td>Displays messages related to HSRP.</td>
</tr>
<tr>
<td>debug standby events icmp</td>
<td>Displays debugging messages for the HSRP ICMP redirects filter.</td>
</tr>
<tr>
<td>debug standby packets</td>
<td>Displays debugging information for packets related to HSRP.</td>
</tr>
</tbody>
</table>
debug condition voice-port

To display debug output for a specified port, use the `debug condition voice-port` command in privileged EXEC mode. To enable debugging messages for all voice ports, use the `no` form of this command.

```
debug condition voice-port port-number
no debug condition voice-port port-number
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>port-number</td>
<td>Voice port for which you want to display all enabled debugging messages.</td>
</tr>
</tbody>
</table>

**Note** Syntax for the `port-number` argument is platform-dependent; type `?` to determine available options for argument syntax.

**Command Default**

Debugging messages are enabled for all voice ports.

**Command Modes**

Privileged EXEC(#{

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(6)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

This command filters out debugging output for all voice ports except the port specified in the command. When this command is configured, the gateway displays debugging messages only for the specified port.

To display debugging output, you must first enable the `debug voip applications tcapp all` command. If no `debug` commands are turned on, no debugging messages are displayed even if the `debug condition voice-port` command is enabled.

The `debug condition voice-port` command can be configured multiple times to display output for more than one voice port. This command differs from the `debug voip applications tcapp port` command, which can be configured to display output for only one voice port.

To display which debug conditions have been set, use the `show debug` command.

Before disabling conditions, first disable any debugging commands; otherwise output for all ports could flood the logging buffer.

**Examples**

The following example filters debugging output so that output only for ports 2/1 and 2/3 is displayed:

```
Router# debug condition voice-port 2/1
Condition 1 set
*Mar 1 22:24:15.102: Debug: Condition 1, voice-port 2/1 triggered, count 1
```
Router# debug condition voice-port 2/3
Condition 2 set
Router# show debug
Condition 1: voice-port 2/1 (1 flags triggered)
  Flags: voice-port condition
Condition 2: voice-port 2/3 (1 flags triggered)
  Flags: voice-port condition

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug sccp all</td>
<td>Displays debugging information for SCCP.</td>
</tr>
<tr>
<td>debug voip application stcapp all</td>
<td>Displays debugging information for the components of the STCAPP.</td>
</tr>
<tr>
<td>debug voip application stcapp port</td>
<td>Enables STCAPP debugging for a specific port.</td>
</tr>
<tr>
<td>show debug</td>
<td>Displays the types of debugging and the debugging conditions that are enabled on your router.</td>
</tr>
</tbody>
</table>
debug condition vrf

To limit debug output to a specific Virtual Routing and Forwarding (VRF) instance, use the `debug condition vrf` command in privileged EXEC mode. To remove the debug condition, use the `undebug` version of the command.

```
debug condition vrf vrf-name
undebug condition vrf vrf-name
```

**Syntax Description**
- `vrf-name`  
  Name of a VRF.

**Command Modes**
- Privileged EXEC (`#`)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS XE Release 3.2S</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command to limit debug output to a single VRF.

**Note**

EIGRP does not support the `debug condition vrf` command.

**Examples**

The following example shows how to limit debugging output to VRF red:

```
Router# debug condition vrf red
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vrf definition</td>
<td>Defines a virtual routing and forwarding instance.</td>
</tr>
</tbody>
</table>
**debug condition xconnect**

To conditionally filter debug messages related to xconnect configurations, use the `debug condition xconnect` command in privileged EXEC configuration mode. To disable the filtering of xconnect debug messages, use the `no` form of this command.

```
debug condition xconnect {fib type| interface type number [dlci| vp number| vc number]| peer ip-address vcid vcid| segment segment-id}
```

```
no debug condition xconnect {fib type| interface type number [dlci| vp number| vc number]| peer ip-address vcid vcid| segment segment-id}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fib type</td>
<td>Filters control-plane and data-plane debug messages for the xconnect segment pair specified by matching against the Forwarding Information Base (FIB) Interface Descriptor Block (IDB) information associated with a particular interface on a line card.</td>
</tr>
<tr>
<td>interface type number</td>
<td>Filters control-plane and data-plane debug messages for the xconnect segment pair specified by the interface type and number on a Route Processor.</td>
</tr>
<tr>
<td>dlci</td>
<td>(Optional) The Frame Relay data-link connection identifier (DLCI) for the xconnect segment pair associated with a Frame Relay segment.</td>
</tr>
<tr>
<td>vp number</td>
<td>(Optional) The ATM virtual path (VP) number for the xconnect segment pair associated with an ATM segment.</td>
</tr>
<tr>
<td>vc number</td>
<td>(Optional) The ATM virtual circuit (VC) number for the xconnect segment pair associated with an ATM segment.</td>
</tr>
<tr>
<td>peer</td>
<td>Filters control-plane and data-plane debug messages for the xconnect segment pair specified by the remote peer IP address and the pseudowire virtual circuit ID (VCID).</td>
</tr>
<tr>
<td>ip-address</td>
<td>The IP address of the remote peer router.</td>
</tr>
<tr>
<td>vcid vcid</td>
<td>The VCID of the xconnect pseudowire.</td>
</tr>
<tr>
<td>segment</td>
<td>Filters data-plane debug messages for the xconnect segment pair specified by a segment ID.</td>
</tr>
<tr>
<td>segment-id</td>
<td>The segment ID. The segment ID value can be found in the output of the <code>show ss mid</code> command.</td>
</tr>
</tbody>
</table>
Command Default
Debug messages are not filtered.

Command Modes
Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(28)SB</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>15.3(1)S</td>
<td>This command was integrated in Cisco IOS Release 15.3(1)S.</td>
</tr>
</tbody>
</table>

Usage Guidelines
Use the debug condition xconnect command to specify conditions for filtering the debug messages displayed by related subscriber service switch (SSS), xconnect, and attachment circuit debug commands.

Examples
The following example sets filter conditions that allow related debug commands to display debug messages for only the xconnect segment pair specified by the remote peer IP address and the pseudowire VCID:

debug condition xconnect peer 10.0.0.1 vcid 100

The following example sets filter conditions that allow related debug commands to display debug messages for only the xconnect segment pair specified by the serial interface number and DLCI:

debug condition xconnect interface serial 0/0 100

The following example sets filter conditions that allow related debug commands to display debug messages for only the xconnect segment pair specified by the port mode ATM interface number:

debug condition xconnect interface atm 0/0

The following example sets filter conditions that allow related debug commands to display debug messages for only the xconnect segment pair specified by the VP mode ATM interface number:

debug condition xconnect interface atm 0/0 vp 1

The following example sets filter conditions that allow related debug commands to display debug messages for only the xconnect segment pair specified by the VC mode ATM interface number:

debug condition xconnect interface atm 0/0 vc 1/40

The following example finds the segment ID associated with an L2TPv3 xconnect segment pair and sets filter conditions that allow related debug commands to display debug messages for only that xconnect segment pair:

Router# show ssm id
Segment-ID: 8193 Type: L2TPv3[8]

Router# debug conditional xconnect segment 8193
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug acircuit</code></td>
<td>Displays errors and events that occur on the attachment circuits.</td>
</tr>
<tr>
<td><code>debug sss aaa authorization event</code></td>
<td>Displays messages about AAA authorization events that are part of normal call establishment.</td>
</tr>
<tr>
<td><code>debug sss aaa authorization fsm</code></td>
<td>Displays information about AAA authorization state changes.</td>
</tr>
<tr>
<td><code>debug sss error</code></td>
<td>Displays diagnostic information about errors that may occur during SSS call setup.</td>
</tr>
<tr>
<td><code>debug sss event</code></td>
<td>Displays diagnostic information about SSS call setup events.</td>
</tr>
<tr>
<td><code>debug sss fsm</code></td>
<td>Displays diagnostic information about the SSS call setup state.</td>
</tr>
<tr>
<td><code>debug xconnect</code></td>
<td>Displays debug messages related to an xconnect configuration.</td>
</tr>
<tr>
<td><code>show ssm id</code></td>
<td>Displays SSM information for switched Layer 2 segments.</td>
</tr>
</tbody>
</table>
**debug configuration lock**

To enable debugging of the Cisco IOS configuration lock, use the `debug configuration lock` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug configuration lock
no debug command lock
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(14)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.0(31)S</td>
<td>This command was integrated into Cisco IOS Release 12.0(31)S.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
<tr>
<td>12.2(33)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SB.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output with `debug configuration lock` enabled (assuming that the feature is enabled using the `configurationmodeexclusivemanual` command in global configuration mode):

```
Router# debug configuration lock
Session1 from console
-----------------------------
Router# configure terminal lock
Configuration mode locked exclusively. The lock will be cleared once you exit out of configuration mode using end/exit Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#
Parser : LOCK REQUEST in EXCLUSIVE mode
Parser: <configure terminal lock> - Config. Lock requested by process <3> client <PARSER Client>
Parser: <configure terminal lock> - Config. Lock acquired successfully !
Router(config)#
Session2 VTY (User from session2 is trying to enter single user config (exclusive) config mode)
-----------------------------
Router# c
configure terminal lock
Configuration mode locked exclusively by user 'unknown' process '3' from terminal '0'. Please try later.
```
Router# Session1 from console
------------------------
Router(config)#
Parser : LOCK REQUEST in EXCLUSIVE mode
Parser: <configure terminal lock> - Config. Lock requested by process <104> client <PARSER Client>
Parser: <configure terminal lock> - NON_BLOCKING : Config mode locked <EXCLUSIVE> owner <3>
Router(config)#
Router(config)# end
Router#
%SYS-5-CONFIG_I: Configured from console by console
Parser: <Configure terminal> - Config. EXC UnLock requested by user<3>
Parser: <Configure terminal> - Config UNLOCKED !
Router#

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>configuration mode exclusive</td>
<td>Enables single-user (exclusive) access functionality for the Cisco IOS CLI.</td>
</tr>
<tr>
<td>show configuration lock</td>
<td>Displays information about the lock status of the running configuration file during a configuration replace operation.</td>
</tr>
</tbody>
</table>
**debug confmodem**

To display information associated with the discovery and configuration of the modem attached to the router, use the `debug confmodem` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug confmodem
no debug confmodem
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

The `debug confmodem` command is used in debugging configurations that use the `modem autoconfig` command.

**Examples**

The following is sample output from the `debug confmodem` command. In the first three lines, the router is searching for a speed at which it can communicate with the modem. The remaining lines show the actual sending of the modem command.

```
Router# debug confmodem
TTY4: detection speed(115200) response ------
TTY4: detection speed(57600) response ------
TTY4: detection speed(38400) response ---OK---
TTY4: Modem command: --AT&F&C1&D2S180=3S190=1S0=1--
TTY4: Modem configuration succeeded
TTY4: Done with modem configuration
```
debug conn

To display information from the connection manager, time-division multiplexing (TDM) and digital signal processor (DSP) clients, use the `debug conn` command in privileged EXEC mode. To disable debugging output, use the `no debug conn` form of this command.

`debug conn`
`no debug conn`

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

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(5)XM</td>
<td>This command is supported on Cisco 3600 series routers.</td>
</tr>
<tr>
<td>12.2(4)T</td>
<td>This command is supported on Cisco 2600 series routers and was integrated into Cisco IOS Release 12.2(4)T.</td>
</tr>
</tbody>
</table>

**Examples**
The following example shows connection manager debugging output:

```
Router# debug conn
Connection Manager debugging is on
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# connect conn1 t1 3/0 i t1 4/0 i

Router(config-tdm-conn)# exit
*Mar 6 18:30:59:%CONN TDM:Segment attached to dsx1
*Mar 6 18:30:59:%CONN TDM:Parsed segment 1
*Mar 6 18:30:59:%CONN TDM:Segment attached to dsx1
*Mar 6 18:30:59:%CONN TDM:Parsed segment 2
*Mar 6 18:30:59:%CONN TDM:Creating new connection
Router(config)#
*Mar 6 18:31:01:%CONN TDM:Interwork Segments
*Mar 6 18:31:01:CONN TDM:Init Segment @ 61C26980
*Mar 6 18:31:01:CONN TDM:Init Segment @ 61C26A44
*Mar 6 18:31:01:CONN TDM:Activating Segment @ 61C26980
*Mar 6 18:31:01:CONN TDM:Activating Segment @ 61C26A44
*Mar 6 18:31:01:%CONN:Segment alarms for conn conn1 are 2
*Mar 6 18:31:01:%CONN:Segment alarms for conn conn1 are 0
*Mar 6 18:31:01:%CONN TDM:Connecting Segments
*Mar 6 18:31:01:%CONN TDM:MAKING CONNECTION
*Mar 6 18:31:01:%CONN:cm_activate_connection, stat = 5
Router(config)#
```
debug content-scan

Note: Effective with Cisco IOS Release 15.4(2)T, the `debug content-scan` command is replaced by the `debug cws` command. See the `debug cws` command for more information.

To enable content scan debugging, use the `debug content-scan` command in privileged EXEC mode. To disable content scan debugging, use the `no` form of this command.

```
debug content-scan {access-list {access-list-number | extended-access-list-number | access-list-name} | control-plane | errors | events | function-trace | packet-path}
no debug content-scan {access-list {access-list-number | extended-access-list-number | access-list-name} | control-plane | errors | events | function-trace | packet-path}
```

Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>access-list</code></td>
<td>Enables debugging of content scan access control lists (ACLs).</td>
</tr>
<tr>
<td><code>access-list-number</code></td>
<td>Access list number. The range is from 1 to 199.</td>
</tr>
<tr>
<td><code>extended-access-list-number</code></td>
<td>Extended access list number. The range is from 1300 to 2699.</td>
</tr>
<tr>
<td><code>access-list-name</code></td>
<td>Access list name.</td>
</tr>
<tr>
<td><code>control-plane</code></td>
<td>Enables debugging of content scan control plane messages.</td>
</tr>
<tr>
<td><code>errors</code></td>
<td>Enables debugging of content scan errors.</td>
</tr>
<tr>
<td><code>events</code></td>
<td>Enables debugging of content scan events.</td>
</tr>
<tr>
<td><code>function-trace</code></td>
<td>Enables debugging of content scan function trace.</td>
</tr>
<tr>
<td><code>packet-path</code></td>
<td>Enables debugging of content scan packet path.</td>
</tr>
</tbody>
</table>

Command Modes

Privileged EXEC (#)

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.2(1)T1</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>15.2(4)M</td>
<td>This command was modified. The <code>access-list</code> and <code>control-plane</code> keywords and the <code>access-list-number</code>, <code>extended-access-list-number</code>, and <code>access-list-name</code> arguments were added.</td>
</tr>
<tr>
<td>15.4(2)T</td>
<td>This command was replaced by the <code>debug cws</code> command.</td>
</tr>
</tbody>
</table>
### Usage Guidelines

The content scanning process redirects client web traffic to the ScanSafe web server. You must configure ACLs by using the `ip access-list extended` command before you enable the debugging of content-scan ACLs. The `debug content-scan access-list` command enables the conditional debugging for content scan. The conditional debugging does not work for existing sessions when you enable content scan; the ACL match occurs after the first packet is received.

### Examples

The following examples show how to enable extended ACL 149 and the debugging of content scan ACL 149:

```plaintext
Device(config)# ip access-list extended 149
Device(config-ext-nacl)# permit ip host 10.1.0.1
Device(config-ext-nacl)# end
Device(config)# debug content-scan access-list 149
Content-scan ACL based conditional debugging is on
```

The following is sample output from the `debug content-scan access-list 149` command:

```plaintext
Device# debug content-scan access-list 149
Content-scan ACL based conditional debugging is on
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH:Checking if it's a web trafficSrc IP:10.1.0.1(3646), Dst IP: 198.51.100.195(80),index:4
Feb 19 19:01:02.887: CONTENT.Scan-PAK-PATCH: Populating user/usergroup infoSrc IP:10.1.0.1(3646), Dst IP: 198.51.100.195(80)
Feb 19 19:01:02.887: CONTENT.Scan-EVE:Username Received: defpmuser Src IP:10.1.0.1(3646) Dst IP: 198.51.100.195(50) Pak:2033B0A
Feb 19 19:01:02.887: CONTENT.Scan-PAK-PATCH:Protocol not configuredSrc IP:10.1.0.1(3646), Dst IP: 198.51.100.195(80)
Feb 19 19:01:02.887: CONTENT.Scan-PAK-PATCH:Adding session into HashSrc IP:10.1.0.1(3646), Dst IP: 198.51.100.195(80)
Feb 19 19:01:02.887: CONTENT.Scan-PAK-PATCH:Free port equals source port for src_ip 10.1.0.1(3646) dst_ip 198.51.100.195(80) 2033BC0
Feb 19 19:01:02.887: CONTENT.Scan-PAK-PATCH:Session enqueued 21ED1DE0 for src_ip 10.1.0.1(3646) dst_ip 198.51.100.195(80)
Feb 19 19:01:02.887: CONTENT.Scan-PAK-PATCH:Inserting Ingress session 21ED1DE0 in hash table at 1646 Src IP:10.1.0.1(3646), Dst IP: 198.51.100.195(80)
Feb 19 19:01:02.887: CONTENT.Scan-PAK-PATCH:Inserting Egress session 21ED1DE0 in hash table at 1629 Src IP:10.1.0.1(3646), Dst IP: 198.51.100.195(80)
Feb 19 19:01:02.887: CONTENT.Scan-PAK-PATCH:timer wheel started for 21ED1DE0 for src_ip 10.1.0.1(3646)
Feb 19 19:01:02.887: CONTENT.Scan-EVE:1 Usergroups from auth proxy Src IP:10.1.0.1(3646), Dst IP: 198.51.100.195(50), cs_entry:21ED1DE0
Feb 19 19:01:02.887: CONTENT.Scan-PAK-PATCH:timer wheel started for 21ED1DE0 for src_ip 10.1.0.1(3646)
Feb 19 19:01:02.887: CONTENT.Scan-EVE:1 Usergroups found Src IP:10.1.0.1(3646), Dst IP: 198.51.100.195(50), cs_entry:21ED1DE0
```

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>content-scan out</td>
<td>Enables content scanning on an egress interface.</td>
</tr>
<tr>
<td>content-scan whitelisting</td>
<td>Enables whitelisting of incoming traffic and enters content-scan whitelisting configuration mode.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>show content-scan</td>
<td>Displays content scan information.</td>
</tr>
</tbody>
</table>
debug control-plane

To display debugging output from the control-plane routines, use the `debug control-plane` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug control-plane [all| host| port-filtering| queue-thresholding| log]
no debug control-plane [all| host| port-filtering| queue-thresholding| log]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>(Optional) Displays all events on all control-plane interfaces.</td>
</tr>
<tr>
<td>host</td>
<td>(Optional) Displays all events on the control-plane host interface.</td>
</tr>
<tr>
<td>port-filtering</td>
<td>(Optional) Displays TCP/IP protocol port-filtering events.</td>
</tr>
<tr>
<td>queue-thresholding</td>
<td>(Optional) Displays TCP/IP protocol queue-thresholding events.</td>
</tr>
<tr>
<td>log</td>
<td>(Optional) Displays control-plane logging events.</td>
</tr>
</tbody>
</table>

**Command Default**
The default is debugging off.

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(4)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(6)T</td>
<td>The <code>log</code> keyword was added to support control plane logging.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
Use the `debug control-plane` command if you want to display the debugging output from the control-plane routines.

**Examples**
The following example show a display from the `debug control-plane` command:

```
Router# debug control-plane
Control-plane infrastructure events debugging is on
Router# cp_receive_classify - marking pak host ingress pak marked cef-exception
```
The following example shows a display from the `debug control-plane` command using the port-filtering option:

```
Router# debug control-plane port-filtering
TCP/IP Port filtering events debugging is on
Dropped UDP dport 1243 sport 62134 saddr 209.165.200.225
```

The table below describes the significant fields shown in the display.

**Table 57: debug control plane field descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dport</td>
<td>UPD destination port.</td>
</tr>
<tr>
<td>sport</td>
<td>UPD source port.</td>
</tr>
<tr>
<td>saddr</td>
<td>Source address of the IP packets.</td>
</tr>
</tbody>
</table>

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear control-plane</td>
<td>Clears packet counters for control-plane interfaces and subinterfaces.</td>
</tr>
<tr>
<td>control-plane</td>
<td>Enters control-plane configuration mode, which allows users to associate or modify attributes or parameters that are associated with the control plane of the device.</td>
</tr>
<tr>
<td>show control-plane cef-exception counters</td>
<td>Displays the control plane packet counters for the control plane CEF-exception subinterface.</td>
</tr>
<tr>
<td>show control-plane cef-exception features</td>
<td>Displays the configured features for the control plane CEF-exception subinterface.</td>
</tr>
<tr>
<td>show control-plane counters</td>
<td>Displays the control-plane packet counters for the aggregate control-plane interface.</td>
</tr>
<tr>
<td>show control-plane features</td>
<td>Displays the configured features for the aggregate control-plane interface.</td>
</tr>
<tr>
<td>show control-plane host counters</td>
<td>Displays the control plane packet counters for the control plane host subinterface.</td>
</tr>
<tr>
<td>show control-plane host features</td>
<td>Displays the configured features for the control plane host subinterface.</td>
</tr>
<tr>
<td>show control-plane host open-ports</td>
<td>Displays a list of open TCP/UDP ports that are registered with the port-filter database.</td>
</tr>
</tbody>
</table>
**show control-plane transit counters**

Displays the control plane packet counters for the control plane transit subinterface.
**debug cops**

To display a one-line summary of each Common Open Policy Service (COPS) message sent from and received by the router, use the `debug cops` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```plaintext
debug cops [detail]
no debug cops [detail]
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>detail</td>
<td>(Optional) Displays additional debug information, including the contents of COPS and Resource Reservation Protocol (RSVP) messages.</td>
</tr>
</tbody>
</table>

### Command Default

COPS process debugging is not enabled.

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(1)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

### Usage Guidelines

To generate a complete record of the policy process, enter this command and, after entering a carriage return, enter the additional command `debug ip rsvp policy`.

### Examples

This first example displays the one-line COPS message summaries, as the router goes through six different events.

```
Router# debug cops
COPS debugging is on
```

The router becomes configured to communicate with a policy server:

```
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# ip rsvp policy cops servers 2.0.0.1
Router(config)#
15:13:45:COPS: Opened TCP connection to 2.0.0.1/3288
15:13:45:COPS: ** SENDING MESSAGE **
15:13:45:COPS: ** RECEIVED MESSAGE **
```
The router receives a Path message:

15:13:53:COPS:** SENDING MESSAGE **
15:13:53:COPS REQ message, Client-type:1, Length:216. Handle:[ 00 00 04 01]
15:13:53:COPS:** RECEIVED MESSAGE **
15:13:53:COPS DEC message, Client-type:1, Length:104. Handle:[ 00 00 04 01]

The router receives a unicast FF Resv message:

15:14:00:COPS:** SENDING MESSAGE **
15:14:00:COPS REQ message, Client-type:1, Length:148. Handle:[ 00 00 05 01]
15:14:00:COPS:** RECEIVED MESSAGE **
15:14:00:COPS DEC message, Client-type:1, Length:64. Handle:[ 00 00 05 01]
15:14:00:COPS:** SENDING MESSAGE **
15:14:00:COPS RPT message, Client-type:1, Length:24. Handle:[ 00 00 05 01]

The router receives a Resv tear:

15:14:06:COPS:** SENDING MESSAGE **
15:14:06:COPS DRQ message, Client-type:1, Length:24. Handle:[ 00 00 05 01]

The router receives a Path tear:

15:14:11:COPS:** SENDING MESSAGE **
15:14:11:COPS DRQ message, Client-type:1, Length:24. Handle:[ 00 00 04 01]

The router gets configured to cease communicating with the policy server:

Router(config)# no ip rsvp policy cops servers
15:14:23:COPS:** SENDING MESSAGE **
15:14:23:COPS:Closed TCP connection to 2.0.0.1/3288

This second example uses the `detail` keyword to display the contents of the COPS and RSVP messages, and additional debugging information:

Router# debug cops detail
COPS debugging is on
02:13:29:COPS:** SENDING MESSAGE **
    COPS HEADER:Version 1, Flags 0, Opcode 1 (REQ), Client-type:1, Length:216
    HANDLE (1/1) object. Length:8. 00 00 21 01
    CONTEXT (2/1) object. Length:8. R-type:5. M-type:1
    IN_IF (3/1) object. Length:12. Address:10.1.2.1. If_index:4
    OUT_IF (4/1) object. Length:12. Address:10.33.0.1. If_index:3
    CLIENT_SI (9/1) object. Length:168. CSI data:
    02:13:29: SESSION type 1 length 12:
    02:13:29: Destination 10.33.0.1, Protocol_Id 17, Don't Police, DstPort 44
    02:13:29: HOP type 1 length 12:0A010201
    02:13:29: TIME_VALUES type 1 length 8 :00000000
    02:13:29: SENDER_TEMPLATE type 1 length 12:
debug cops

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ip rsvp policy</td>
<td>Displays debugging messages for RSVP policy processing.</td>
</tr>
</tbody>
</table>

To see an example where the debugcops command is used along with the debugiprsvppolicy command, refer to the second example of the debugiprsvppolicy command.

deploy clns esis events through debug dbconn tcp
debug cot

To display information about the Continuity Test (COT) functionality, use the `debugcot` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cot {api| dsp| queue| detail}
no debug cot {api| dsp| queue| detail}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>api</td>
<td>Displays information about the COT application programming interface (API).</td>
</tr>
<tr>
<td>dsp</td>
<td>Displays information related to the COT/Digital Signal Processor configuration (DSP) interface. Typical DSP functions include data modems, voice codecs, fax modems and codecs, and low-level signaling such as channel-associated signaling (CAS)/R2.</td>
</tr>
<tr>
<td>queue</td>
<td>Display information related to the COT internal queue.</td>
</tr>
<tr>
<td>detail</td>
<td>Display information about COT internal detail; summary of the <code>debugcotapi</code>, <code>debugcotdsp</code>, and <code>debugcotqueue</code> commands.</td>
</tr>
</tbody>
</table>

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(7)</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

### Examples

The following is sample output from the `debugcotapi` command:

```
Router# debug cot api
COT API debugging is on
08:29:55: cot_request_handler(): CDB@0x60DEDE14, req(COT_CHECK_TONE_ON):
08:29:55: shelf 0 slot 0 appl_no 1 ds0 1
08:29:55: freqTX 2010 freqRX 1780 key 0xFFF1 duration 60000
```

The table below describes the significant fields shown in the display.
Table 58: debug cot api Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDB</td>
<td>Internal controller information.</td>
</tr>
<tr>
<td>req</td>
<td>Type of COT operation requested.</td>
</tr>
<tr>
<td>shelf</td>
<td>Shelf ID of the COT operation request.</td>
</tr>
<tr>
<td>slot</td>
<td>Designates the slot number, 1 to 4.</td>
</tr>
<tr>
<td>appl-no</td>
<td>Hardware unit that provides the external interface connections from a router to the network.</td>
</tr>
<tr>
<td>ds0</td>
<td>Number of the COT operation request.</td>
</tr>
<tr>
<td>key</td>
<td>COT operation identifier.</td>
</tr>
<tr>
<td>duration</td>
<td>Timeout duration of the COT operation.</td>
</tr>
<tr>
<td>freqTX</td>
<td>Requested transmit tone frequency.</td>
</tr>
<tr>
<td>freqRX</td>
<td>Requested receive tone frequency.</td>
</tr>
</tbody>
</table>

The following is sample output from the `debug cot dsp` command:

```
Router# debug cot dsp
Router# 00:10:42:COT:DSP (1/1) Allocated
00:10:43:In cot_callback
00:10:43: returned key 0xFFF1, status = 0
00:10:43:COT:Received DSP Q Event
00:10:43:COT:DSP (1/1) Done
00:10:43:COT:DSP (1/1) De-allocated
```

The table below describes the significant fields shown in the display.

Table 59: debug cot dsp Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP (1/1) Allocated</td>
<td>Slot and port of the DSP allocated for the COT operation.</td>
</tr>
<tr>
<td>Received DSP Q Event</td>
<td>Indicates the COT subsystem received an event from the DSP.</td>
</tr>
<tr>
<td>DSP (1/1) Done</td>
<td>Slot and port of the DSP transitioning to IDLE state.</td>
</tr>
<tr>
<td>DSP (1/1) De-allocated</td>
<td>Slot and port of the DSP de-allocated after the completion of the COT operation.</td>
</tr>
</tbody>
</table>
The following is sample output from the `debug cot queue` command:

```
Router# debug cot queue
Router# 00:11:26:COT(0x60EBB48C):Adding new request (0x61123DBC) to In Progress Q
00:11:26:COT(0x60EBB48C):Adding COT(0x61123DBC) to the Q head
00:11:27: in cot_callback
00:11:27: returned key 0xFFF1, status = 0
```

The table below describes the significant fields shown in the display.

**Table 60: debug cot api Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COT</td>
<td>Internal COT operation request.</td>
</tr>
<tr>
<td>Adding new request</td>
<td>Internal COT operation request queue.</td>
</tr>
</tbody>
</table>

The following is sample output from the `debug cot detail` command.

```
Router# debug cot detail
Router# 00:04:57: cot_request_handler(): CDB@0x60EBB48C, req(COT_CHECK_TONE_ON): 00:04:57: shelf 0 slot 0 appl_no 1 ds0 1 00:04:57: freqTX 1780 freqRX 2010 key 0xFFF1 duration 1000 00:04:57: COT: DSP (1/0) Allocated 00:04:57: COT: Request Transition to COT_WAIT_TD_ON 00:04:57: COT(0x60EBB48C): Adding new request (0x61123DBC) to In Progress Q 00:04:57: COT(0x60EBB48C): Adding COT(0x61123DBC) to the Q head 00:04:57: COT: Start Duration Timer for Check Tone Request 00:04:58: cot_request_handler(): CDB@0x60EBB48C, Params@0x61123EO8 00:04:58: state COT_WAIT_TD_ON CT 00:04:58: event proc(0x6093BB5C) 00:04:58: invoke NI2 callback to inform COT request status 00:04:58: in cot_callback 00:04:58: returned key 0xFFF1, status = 0 00:04:58: Return from NI2 callback 00:04:58: COT: Request Transition to IDLE 00:04:58: COT: DSP (1/0) Done 00:04:58: COT: DSP (1/0) De-allocated
```

Because the `debug cot detail` command is a summary of the `debug cot api`, `debug cot dsp`, and `debug cot queue` commands, the field descriptions are the same.
debug cpp event

Note
Effective with Release 12.3(4)T, the debugcpppevent command is no longer available in Cisco IOS 12.3(T) releases.

To display general Combinet Proprietary Protocol (CPP) events, use the debugcpppevent command in privileged EXEC mode. The no form of this command disables debugging output.

default no debug cpp event

Syntax Description
This command has no arguments or keywords.

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.2</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command is no longer available in Cisco IOS 12.3(T) releases.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

Usage Guidelines
CPP allows a router to engage in negotiation over an ISDN B channel to establish connections with a Combinet bridge.

The debugcpppevent command displays events such as CPP sequencing, group creation, and keepalives.

Examples
One or more of the messages in the table below appear when you use the debugcpppevent command. Each message begins with the short name of the interface the event occurred on (for example, SERIAL0:1 or BRI0:1) and might contain one or more packet sequence numbers or remote site names.

Table 61: debug cpp event Messages

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRI0:1: negotiation complete</td>
<td>Call was set up on the interface (in this example, BRI0:1).</td>
</tr>
<tr>
<td>BRI0:1: negotiation timed out</td>
<td>Call timed out.</td>
</tr>
<tr>
<td>BRI0:1: sending negotiation packet</td>
<td>Negotiation packet was sent to set up the call.</td>
</tr>
<tr>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>BRIO:1: out of sequence packet - got 10, range 1 8</td>
<td>Packet was received that was out of sequence. The first number displayed in the message is the sequence number received, and the following numbers are the range of valid sequence numbers.</td>
</tr>
<tr>
<td>BRIO:1: Sequence timer expired - Lost 11 Trying sequence 12</td>
<td>Timer expired before the packet was received. The first number displayed in the message is the sequence number of the packet that was lost, and the second number is the next sequence number.</td>
</tr>
<tr>
<td>BRIO:1: Line Integrity Violation</td>
<td>Router fails to maintain keepalives.</td>
</tr>
<tr>
<td>BRIO:1: create cpp group ber19 destroyed cpp group ber19</td>
<td>Dialer group is created on the remote site (in this example, ber19).</td>
</tr>
</tbody>
</table>

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cpp negotiation</td>
<td>Displays CPP negotiation events.</td>
</tr>
<tr>
<td>debug cpp packet</td>
<td>Displays CPP packets.</td>
</tr>
</tbody>
</table>
debug cpp negotiation

Note

Effective with Release 12.3(4)T, the **debugcppnegotiation** command is no longer available in Cisco IOS 12.3T releases.

To display Combinet Proprietary Protocol (CPP) negotiation events, use the **debugcppnegotiation** command in privileged EXEC mode. The **no** form of this command disables debugging output.

depcgppnegotiation

**no debug cpp negotiation**

Syntax Description

This command has no arguments or keywords.

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.2</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command is no longer available in Cisco IOS 12.3T releases.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

Usage Guidelines

CPP allows a router to engage in negotiation over an ISDN B channel to establish connections with a Combinet bridge.

The **debugcppnegotiation** command displays events such as the type of packet and packet size being sent.

Examples

The following is sample output from the **debugcppnegotiation** command. In this example, a sample connection is shown.

```
Router# debug cpp negotiation
%LINK-3-UPDOWN: Interface BRI0: B-Channel 2, changed state to down
%LINK-3-UPDOWN: Interface BRI0, changed state to up
%SYS-5-CONFIG_I: Configured from console by console
%LINK-3-UPDOWN: Interface BRI0: B-Channel 1, changed state to up
BR0:1:(I) NEG packet - len 77
   attempting proto:2
   ether id:0040.f902.c7b4
   port 1 number:5559876
   port 2 number:5559876
   origination port:1
   remote name:berl19
   password is correct
```

The table below describes the significant fields in the display.
### Table 62: debug CPP negotiation Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR0:1 (I) NEG packet - len 77</td>
<td>Interface name, packet type, and packet size.</td>
</tr>
<tr>
<td>attempting proto:</td>
<td>CPP protocol type.</td>
</tr>
<tr>
<td>ether id:</td>
<td>Ethernet address of the destination router.</td>
</tr>
<tr>
<td>port 1 number:</td>
<td>ISDN phone number of remote B channel #1.</td>
</tr>
<tr>
<td>port 2 number:</td>
<td>ISDN phone number of remote B channel #2.</td>
</tr>
<tr>
<td>origination port:</td>
<td>B channel 1 or 2 called.</td>
</tr>
<tr>
<td>remote name:</td>
<td>Remote site name to which this call is connecting.</td>
</tr>
<tr>
<td>password is correct</td>
<td>Password is accepted so the connection is established.</td>
</tr>
</tbody>
</table>

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cot</td>
<td>Displays information about the COT functionality.</td>
</tr>
<tr>
<td>debug cpp packet</td>
<td>Displays CPP packets.</td>
</tr>
</tbody>
</table>
**debug cpp packet**

**Note**
Effective with Release 12.3(4)T, the `debugcpppacket` command is no longer available in Cisco IOS 12.3T Releases.

To display Combinet Proprietary Protocol (CPP) packets, use the `debugcpppacket` command in privileged EXEC mode. The `no` form of this command disables debugging output.

```
debug cpp packet
no debug cpp packet
```

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

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.2</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command is no longer available in Cisco IOS 12.3T releases.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
CPP allows a router to engage in negotiation over an ISDN B channel to establish connections with a Combinet bridge.

The `debugcpppacket` command displays the hexadecimal values of the packets.

**Examples**
The following is sample output from the `debugcpppacket` command. This example shows the interface name, packet type, packet size, and the hexadecimal values of the packet.

```
Router# debug cpp packet
BR0:1:input packet - len 60
00 00 00 00 00 00 00 40 F9 02 C7 B4 08 01 01
08 00 06 04 00 02 00 40 F9 02 C7 B4 83 6C A1 01
Success rate is 80 percent (4/5), round-trip min/avg/max = 64/66/68 ms
BR0:1 output packet - len 116
06 00 00 40 F9 02 C7 B4 00 00 0C 3E 12 3A 08 00
45 00 00 64 00 01 00 00 FF 01 72 BB 83 6C A1 01
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cot</td>
<td>Displays information about the COT functionality.</td>
</tr>
<tr>
<td>debug cpp negotiation</td>
<td>Displays CPP negotiation events.</td>
</tr>
</tbody>
</table>
debug clns esis events through debug dbconn tcp

debug cpp packet
**debug credentials**

To set debugging on the credentials service that runs between the Cisco Unified Survivable Site Remote Telephony (Cisco Unified SRST) router and Cisco CallManager, use the `debug credentials` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug credentials
no debug credentials
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Cisco IOS Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(14)T</td>
<td>This command was introduced for Cisco SRST 3.3.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command to monitor Cisco CallManager while it requests certificates from the Cisco Unified SRST router.

**Examples**

The following is sample output showing the credentials service that runs between the Cisco Unified SRST router and Cisco CallManager. The credentials service provides Cisco CallManager with the certificate from the Cisco Unified SRST router.

```
Router# debug credentials
Credentials server debugging is enabled
Router#
May 25 12:08:17.944: Credentials service: Start TLS Handshake 1 10.5.43.174 4374
May 25 12:08:17.948: Credentials service: TLS Handshake returns OPSSLReadWouldBlockErr
May 25 12:08:18.948: Credentials service: TLS Handshake returns OPSSLReadWouldBlockErr
May 25 12:08:19.948: Credentials service: TLS Handshake returns OPSSLReadWouldBlockErr
May 25 12:08:20.964: Credentials service: TLS Handshake completes
```

The table below describes the significant fields shown in the display.

**Table 63: debug credentials Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start TLS Handshake 1 10.5.43.174 4374</td>
<td>Indicates the beginning of the TLS handshake between the secure SRST router and Cisco CallManager. In this example, 1 indicates the socket, 10.5.43.174 is the IP address, and 4374 is the port of Cisco CallManager.</td>
</tr>
<tr>
<td>TLS Handshake returns OPSSLReadWouldBlockErr</td>
<td>Indicates that the handshake is in process.</td>
</tr>
</tbody>
</table>
Indicate that the TLS handshake has finished and that the Cisco CallManager has received the secure SRST device certificate.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credentials (SRST)</td>
<td>Provides the Cisco Unified SRST router certificate to Cisco CallManager and enters credentials configuration mode.</td>
</tr>
<tr>
<td>ip source-address (credentials)</td>
<td>Enables the Cisco Unified SRST router to receive messages from Cisco CallManager through the specified IP address and port.</td>
</tr>
<tr>
<td>show credentials</td>
<td>Displays the credentials settings on the Cisco Unified SRST router that are supplied to Cisco CallManager for use during secure SRST fallback.</td>
</tr>
<tr>
<td>show debugging</td>
<td>Displays information about the types of debugging that are enabled for your router.</td>
</tr>
<tr>
<td>trustpoint (credentials)</td>
<td>Specifies the name of the trustpoint that is to be associated with the Cisco Unified SRST router certificate.</td>
</tr>
</tbody>
</table>
To troubleshoot the Carrier Resource Manager (CRM), use the `debugcrm` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debugcrm [all] [default] [detail] [error [call [informational]] [software [informational]]] [function] [inout]
```

### Syntax Description

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>all</code></td>
<td>(Optional) Displays all CRM debugging messages.</td>
</tr>
<tr>
<td><code>default</code></td>
<td>(Optional) Displays detail, error, and inout information. This option also runs if no keywords are added.</td>
</tr>
<tr>
<td><code>detail</code></td>
<td>(Optional) Displays non-inout information related to call processing, such as call updates or call acceptance checking.</td>
</tr>
<tr>
<td><code>error</code></td>
<td>(Optional) Displays CRM error messages.</td>
</tr>
<tr>
<td><code>call</code></td>
<td>(Optional) Displays call processing errors.</td>
</tr>
<tr>
<td><code>informational</code></td>
<td>(Optional) Displays minor errors and major errors. Without the <code>informational</code> keyword, only major errors are displayed.</td>
</tr>
<tr>
<td><code>software</code></td>
<td>(Optional) Displays software errors.</td>
</tr>
<tr>
<td><code>function</code></td>
<td>(Optional) Displays CRM function names and exit points from each function so that call processing can be traced within the CRM subsystem.</td>
</tr>
<tr>
<td><code>inout</code></td>
<td>(Optional) Displays information from the functions that form the external interfaces of CRM to other modules or subsystems.</td>
</tr>
</tbody>
</table>

### Command Default

Debugging is not enabled.

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(11)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>
Modification

Release 12.3(8)T
Modification: The all, default, detail, error, call, informational, software, function, and inout keywords were added.

Usage Guidelines
Disable console logging and use buffered logging before using the debugcrm command. Using the debugcrm command generates a large volume of debugs, which can affect router performance.

Examples
The following is sample output from the debugcrmall command for an incoming ISDN call on a trunk group:

```
Router# debug crm all
01:21:23: //1/xxxxxxxxxxxx/CRM/crm_send_periodic_update:
     RouteLabel=2023, CarrierType=TDM, EventType=Single RouteLabel Update, EventReason=Both Capacity Update,
        Max Capacity mask 0x0000001F, Current Capacity Mask 0x0000001F
01:21:23: //1/xxxxxxxxxxxx/CRM/print_event:
     RouteLabel=2023, CarrierType=TDM, EventType=Single RouteLabel Update, EventReason=Both Capacity Update,
        Max Capacity mask 0x0000001F, Current Capacity Mask 0x0000001F
01:21:23: //1/xxxxxxxxxxxx/CRM/crm_call_update:
     Increment the call count
        CarrierID=2023, TrunkGroupLabel=2023
        Update is for TrunkGroupLabel, Mask=0x00000001
01:21:23: //1/xxxxxxxxxxxx/CRM/crm_call_update:
     IncomingVoiceCalls=1
Router# 01:21:48: //1/xxxxxxxxxxxx/CRM/crm_send_periodic_update:
01:21:48: //1/xxxxxxxxxxxx/CRM/print_event:
     RouteLabel=2023, CarrierType=TDM, EventType=Single RouteLabel Update, EventReason=Both Capacity Update,
        Max Capacity mask 0x0000001F, Current Capacity Mask 0x0000001F
01:22:13: //1/xxxxxxxxxxxx/CRM/crm_send_periodic_update:
01:22:13: //1/xxxxxxxxxxxx/CRM/print_event:
     RouteLabel=2023, CarrierType=TDM, EventType=Single RouteLabel Update, EventReason=Both Capacity Update,
        Max Capacity mask 0x0000001F, Current Capacity Mask 0x0000001F
Router# 01:22:18: //1/xxxxxxxxxxxx/CRM/crm_call_update:
     Decrement the call count
        CarrierID=2023, TrunkGroupLabel=2023
        Update is for TrunkGroupLabel, Mask=0x00000001
01:22:18: //1/xxxxxxxxxxxx/CRM/crm_call_update:
     IncomingVoiceCalls=0
```

The table below describes the significant fields shown in the display.
### Table 64: debug crm all Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>//1/xxxxxxxxxxxxx/CRM/print_event:</td>
<td>The format of this message is //callid/GUID/CRM/function name:</td>
</tr>
<tr>
<td>• CallEntry ID is -1. This indicates that the CallEntry ID is unavailable.</td>
<td></td>
</tr>
<tr>
<td>• GUID is xxxxxxxxxxxxxxxx. This indicates that the GUID is unavailable.</td>
<td></td>
</tr>
<tr>
<td>• CRM is the module name.</td>
<td></td>
</tr>
<tr>
<td>• The print_event: field shows that the CRM is showing the print event function.</td>
<td></td>
</tr>
<tr>
<td>RouteLabel</td>
<td>Either the trunk group label or carrier ID.</td>
</tr>
<tr>
<td>CarrierType</td>
<td>Indicates the type of trunk.</td>
</tr>
<tr>
<td>EventType</td>
<td>Indicates if a single route or all routes are updated.</td>
</tr>
<tr>
<td>EventReason</td>
<td>Shows the reason for this event being sent.</td>
</tr>
</tbody>
</table>

#### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>max-calls</td>
<td>Specifies the maximum number of calls the trunk group can handle.</td>
</tr>
</tbody>
</table>
**debug crypto ace b2b**

To enable IPSec VPN SPA debugging for a Blade Failure Group, use the `debug crypto ace b2b` command in privileged EXEC mode.

**debug crypto ace b2b**

**Command Default**
No default behavior or values.

**Command Modes**
Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(18)SXE2</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>

**Examples**

The following example enables IPSec VPN SPA debugging for a Blade Failure Group:

```
Router# debug crypto ace b2b
ACE B2B Failover debugging is on
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>linecard-group feature card</td>
<td>Assigns a group ID to a Blade Failure Group.</td>
</tr>
<tr>
<td>show crypto ace redundancy</td>
<td>Displays information about a Blade Failure Group.</td>
</tr>
<tr>
<td>show redundancy linecard-group</td>
<td>Displays the components of a Blade Failure Group.</td>
</tr>
</tbody>
</table>
**debug crypto ace boot**

To enable ACE boot API debugging, use the `debug crypto ace boot` command in privileged EXEC mode. To disable ACE boot API debugging, use the `no` form of this command.

```
debug crypto ace boot
no debug crypto ace boot
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

None

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SXI5</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following example enables ACE boot API debugging:

```
Router# debug crypto ace boot
ACE Boot debugging is on
```
debug crypto ace cfgmon

To enable ACE CFGMON transactions debugging, use the debug crypto ace cfgmon command in privileged EXEC mode. To disable ACE CFGMON transactions debugging, use the no form of this command.

debug crypto ace cfgmon
no debug crypto ace cfgmon

Syntax Description
This command has no arguments or keywords.

Command Default
None

Command Modes
Privileged EXEC (#)

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SX15</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

Examples
The following example enables ACE CFGMON transaction debugging:

Router# debug crypto ace cfgmon
ACE CFGMON Transactions debugging is on
debug crypto ace congestion-mgr

To enable ACE Crypto engine congestion manager debugging, use the `debug crypto ace congestion-mgr` command in privileged EXEC mode. To disable ACE Crypto engine congestion manager debugging, use the `no` form of this command.

```
debug crypto ace congestion-mgr
no debug crypto ace congestion-mgr
```

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

**Command Default**
None

**Command Modes**
Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SXI5</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**
The following example enables ACE Crypto engine congestion manager debugging:

```
Router# debug crypto ace congestion-mgr
ACE Crypto Engine Congestion Manager debugging is on
```
debug crypto ace dpd

To enable ACE DPD debugging, use the debug crypto ace dpd command in privileged EXEC mode. To disable ACE DPD debugging, use the no form of this command.

```
debug crypto ace dpd
no debug crypto ace dpd
```

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

**Command Default**
None

**Command Modes**
Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SX15</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**
The following example enables ACE DPD debugging:

```
Router# debug crypto ace dpd
ACE-DPD debugging is on
```
debug crypto ace error

To enable error logging debugging, use the debug crypto ace error command in privileged EXEC mode. To disable ACE error logging debugging, use the no form of this command.

debug crypto ace error
no debug crypto ace error

Syntax Description
This command has no arguments or keywords.

Command Default
None

Command Modes
Privileged EXEC (#)

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SXI5</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

Examples
The following example enables error log debugging:

```
Router# debug crypto ace error
ACE Error logging debugging is on
```
debug crypto ace hapi

To enable ACE HAPI transactions debugging, use the debug crypto ace hapi command in privileged EXEC mode. To disable ACE HAPI transactions debugging, use the no form of this command.

debug crypto ace hapi
no debug crypto ace hapi

Syntax Description
This command has no arguments or keywords.

Command Default
None

Command Modes
Privileged EXEC (#)

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SX15</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

Examples
The following example enables ACE HAPI transaction debugging:

Router# debug crypto ace hapi
ACE HAPI transactions debugging is on
debug crypto ace ikea

To enable ACE IKEA transactions debugging, use the debug crypto ace ikea command in privileged EXEC mode. To disable ACE IKEA transactions debugging, use the no form of this command.

**debug crypto ace ikea**

**no debug crypto ace ikea**

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

**Command Default**
None

**Command Modes**
Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SXI5</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**
The following example enables ACE IKEA transaction debugging:

```
Router# debug crypto ace ikea
ACE IKEA transactions debugging is on
```
**debug crypto ace invspi**

To enable ACE invalid SPI debugging, use the `debug crypto ace invspi` command in privileged EXEC mode. To disable ACE invalid SPI debugging, use the `no` form of this command.

```
debug crypto ace invspi
no debug crypto ace invspi
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

None

**Command Modes**

Privileged EXEC (#)

**Command History**

```
<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SX1S</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>
```

**Examples**

The following example enables ACE invalid SPI debugging:

```
Router# debug crypto ace invspi
ACE Invalid SPI debugging is on
```
debug crypto ace ipd

To enable ACE IPD debugging, use the `debug crypto ace ipd` command in privileged EXEC mode. To disable ACE IPD debugging, use the `no` form of this command.

```
debug crypto ace ipd
no debug crypto ace ipd
```

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

**Command Default**
None

**Command Modes**
Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SXI5</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**
The following example enables ACE IPD debugging:

```
Router# debug crypto ace ipd
ACE IPD debugging is on
```
**debug crypto ace mace**

To enable multi-ACE messages debugging, use the `debug crypto ace mace` command in privileged EXEC mode. To disable multi-ACE messages debugging, use the `no` form of this command.

```
debug crypto ace mace
no debug crypto ace mace
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

None

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SX15</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following example enables multi-ACE debugging:

```
Router# debug crypto ace mace
MULTI-ACE messages debugging is on
```
debug crypto ace pcp

To enable ACE PCP transactions debugging, use the debug crypto ace pcp command in privileged EXEC mode. To disable ACE PCP transaction debugging, use the no form of this command.

```
debug crypto ace pcp
no debug crypto ace pcp
```

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

**Command Default**
None

**Command Modes**
Privileged EXEC (\#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SXI5</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**
The following example enables ACE PCP transactions debugging:

```
Router# debug crypto ace pcp
ACE PCP Transactions debugging is on
```
debug crypto ace polo

To enable ACE policy loader debugging, use the debug crypto ace polo command in privileged EXEC mode. To disable ACE policy loader debugging, use the no form of this command.

debug crypto ace polo
no debug crypto ace polo

Syntax Description
This command has no arguments or keywords.

Command Default
None

Command Modes
Privileged EXEC (#)

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SX15</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

Examples
The following example enables ACE policy loader debugging:

Router# debug crypto ace polo
ACE Policy Loader debugging is on
**debug crypto ace propcfg**

To enable ACE propagate configuration debugging, use the `debug crypto ace propcfg` command in privileged EXEC mode. To disable ACE propagate configuration debugging, use the `no` form of this command.

```
debug crypto ace propcfg
no debug crypto ace propcfg
```

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

**Command Default**  
None

**Command Modes**  
Privileged EXEC ("")

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SXI5</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**  
The following example enables ACE propagate configuration debugging:

```
Router# debug crypto ace propcfg
ACE Propagate Config debugging is on
```
**debug crypto ace qos**

To enable ACE QoS debugging, use the `debug crypto ace qos` command in privileged EXEC mode. To disable ACE QoS debugging, use the `no` form of this command.

```
debug crypto ace qos
no debug crypto ace qos
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

None

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SXIS</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following example enables ACE QoS debugging:

```
Router# debug crypto ace qos
ACE QoS debugging is on
```
**debug crypto ace rcon**

To enable ACE RCON messages debugging, use the `debug crypto ace rcon` command in privileged EXEC mode. To disable ACE RCON messages debugging, use the `no` form of this command.

```plaintext
debug crypto ace rcon
no debug crypto ace rcon
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

None

**Command Modes**

Privileged EXEC (\#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SX15</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following example enables ACE RCON messages debugging:

```plaintext
Router# debug crypto ace rcon
ACE RCON messages debugging is on
```
debug crypto ace redundancy

To enable ACE HA debugging, use the debug crypto ace redundancy command in privileged EXEC mode. To disable ACE HA debugging, use the no form of this command.

d debug crypto ace redundancy
no debug crypto ace redundancy

Syntax Description
This command has no arguments or keywords.

Command Default
None

Command Modes
Privileged EXEC (#)

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SX15</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

Examples
The following example enables ACE HA debugging:

Router# debug crypto ace redundancy
ACE HA debugging is on
**debug crypto ace spi**

To enable ACE SPI debugging, use the `debug crypto ace spi` command in privileged EXEC mode. To disable ACE SPI debugging, use the **no** form of this command.

```
debug crypto ace spi
no debug crypto ace spi
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

None

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SX15</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following example enables ACE SPI debugging:

```
Router# debug crypto ace spi
ACE SPI debugging is on
```
**debug crypto ace stats**

To enable ACE stats module debugging, use the `debug crypto ace stats` command in privileged EXEC mode. To disable ACE stats module debugging, use the `no` form of this command.

```
debug crypto ace stats
no debug crypto ace stats
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

None

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SXIS</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following example enables ACE stats module debugging:

```
Router# debug crypto ace stats
ACE Stats Module debugging is on
```
debug crypto ace syslog

To enable ACE syslog messages debugging, use the debug crypto ace syslog command in privileged EXEC mode. To disable ACE syslog messages debugging, use the no form of this command.

```
ddebug crypto ace syslog
no debug crypto ace syslog
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

None

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SXl5</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following example enables ACE syslog messages debugging:

```
Router# debug crypto ace syslog
ACE Syslog messages debugging is on
```
To enable ACE TFTP boot debugging, use the `debug crypto ace tftp` command in privileged EXEC mode.
To disable ACE TFTP boot debugging, use the `no debug crypto ace tftp` command.

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

**Command Default**
None

**Command Modes**
Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SX15</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**
The following example enables ACE TFTP boot debugging:

```
Router# debug crypto ace tftp
ACE TFTP Boot debugging is on
```
**debug crypto ace topn**

To enable ACE-TOPN-HELPER debugging, use the `debug crypto ace topn` command in privileged EXEC mode. To disable ACE-TOPN-HELPER debugging, use the `no` form of this command.

```plaintext
debug crypto ace topn
no debug crypto ace topn
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

None

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SX15</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following example enables ACE-TOPN-HELPER debugging:

```plaintext
Router# debug crypto ace topn
ACE-TOPN-HELPER debugging is on
```
debug crypto ace warning

To enable ACE warning log debugging, use the debug crypto ace warning command in privileged EXEC mode. To disable ACE warning logging debugging, use the no form of this command.

```
debug crypto ace warning
no debug crypto ace warning
```

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

**Command Default**
None

**Command Modes**
Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SX15</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**
The following example enables ACE warning log debugging:

```
Router# debug crypto ace warning
ACE Warning logging debugging is on
```
**debug crypto condition unmatched**

To display crypto conditional debug messages when context information is unavailable to check against debug conditions, use the `debug crypto condition unmatched` command in privileged EXEC mode. To disable debugging, use the no form of this command.

```
debug crypto condition unmatched [isakmp| ipsec| ikev2| engine| gdoi-group]
no debug crypto condition unmatched [isakmp| ipsec| ikev2| engine| gdoi-group]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>isakmp</th>
<th>ipsec</th>
<th>ikev2</th>
<th>engine</th>
<th>gdoi-group</th>
</tr>
</thead>
</table>
| (Optional) One or more of these keywords can be enabled to display debug messages for the specified areas. If none of these keywords are entered, debug messages for all crypto areas will be shown.

**Command Default**

Debug messages that do not have context information to match any debug conditions (filters) will not be printed.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(2)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(18)SXD</td>
<td>This command was integrated into Cisco IOS Release 12.2(18)SXD.</td>
</tr>
<tr>
<td>15.1(3)T</td>
<td>This command was modified. The <code>gdoi-group</code> keyword was replaced by the <code>group</code> keyword of the <code>debug crypto gdoigroup condition</code> command.</td>
</tr>
<tr>
<td>Cisco IOS XE Release 3.8S</td>
<td>This command was integrated into Cisco IOS XE Release 3.8S.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

After the `debug crypto condition` command has been enabled, you can use the `debug crypto condition unmatched` command to define whether the debug output is being printed when no context information is available in the code to check against the debug filters. For example, if the crypto engine’s connection-ID is the filter that the debug conditions are being checked against, the `debug crypto condition unmatched` command displays debug messages in the early negotiation phase when a connection-ID is unavailable to check against debug conditions.

**Examples**

The following example shows how to enable debug messages for all crypto-related areas:

```
Router# debug crypto condition unmatched
```
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug crypto gdoi condition</td>
<td>Defines conditional debug filters for GDOI.</td>
</tr>
<tr>
<td>debug crypto condition</td>
<td>Defines conditional debug filters.</td>
</tr>
<tr>
<td>show crypto debug-condition</td>
<td>Displays crypto debug conditions that have already been enabled in the router.</td>
</tr>
<tr>
<td>show crypto ipsec sa</td>
<td>Displays the settings used by current SAs.</td>
</tr>
<tr>
<td>show crypto isakmp sa</td>
<td>Displays all current IKE SAs at a peer.</td>
</tr>
</tbody>
</table>
**debug crypto ctcp**

To display information about a Cisco Tunnel Control Protocol (cTCP) session, use the `debug crypto ctcp` command in privileged EXEC mode. To turn off debugging, use the **no** form of this command.

```
debug crypto ctcp
no debug crypto ctcp
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Debugging is turned off.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(9)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

You can use this command if a cTCP session fails to come up.

**Examples**

The following example shows that debugging has been turned on for a cTCP session:

```
Router# debug crypto ctcp
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>crypto ctcp</code></td>
<td>Configures cTCP encapsulation for Easy VPN.</td>
</tr>
</tbody>
</table>
**debug crypto engine**

To display debugging messages about crypto engines, which perform encryption and decryption, use the `debug crypto engine` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
display crypto engine
no debug crypto engine
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use the `debug crypto engine` command to display information pertaining to the crypto engine, such as when Cisco IOS software is performing encryption or decryption operations.

The crypto engine is the actual mechanism that performs encryption and decryption. A crypto engine can be software or a hardware accelerator. Some platforms can have multiple crypto engines; therefore, the router will have multiple hardware accelerators.

**Examples**

The following is sample output from the `debug crypto engine` command. The first sample output shows messages from a router that successfully generates Rivest, Shamir, and Adelma (RSA) keys. The second sample output shows messages from a router that decrypts the RSA key during Internet Key Exchange (IKE) negotiation.

```
Router# debug crypto engine
00:25:13:CryptoEngine0:generate key pair
00:25:13:CryptoEngine0:CRYPTO_GEN_KEY_PAIR
00:25:13:CRYPTO_ENGINE:key process suspended and continued
00:25:14:CRYPTO_ENGINE:key process suspended and continuedcr
00:25:14:CryptoEngine0:generate alg parameter
00:25:14:CryptoEngine0:calculate pkey hmac for conn id 0
00:25:14:isakmp SKEYID for conn id 1
00:25:14:Crypto Engine 0:RSA decrypt with public key
00:27:52:CryptoEngine0:CRYPTO_RSA_PUB_DECRYPT
00:27:52:CryptoEngine0:generate hmac context for conn id 1
00:27:52:CryptoEngine0:generate hmac context for conn id 1
```
Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crypto key generate rsa</td>
<td>Generates RSA key pairs.</td>
</tr>
</tbody>
</table>
debug crypto engine accelerator logs

To enable logging of commands and associated parameters sent from the virtual private network (VPN) module driver to the VPN module hardware using a debug flag, use the debug crypto engine accelerator logs command in privileged EXEC mode. To disable debugging output, use the no form of this command.

Syntax Description

This command has no arguments or keywords.

Command Default

The logging of commands sent from the VPN module driver to the VPN module hardware is disabled.

Command Modes

Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(1)XC</td>
<td>This command was introduced on the Cisco 1720 and Cisco 1750 routers.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

Usage Guidelines

Use the debug crypto engine accelerator logs command when encryption traffic is sent to the router and a problem with the encryption module is suspected.

This command is intended only for Cisco TAC personnel to collect debugging information.

Examples

The debug crypto engine accelerator logs command uses a debug flag to log commands and associated parameters sent from the VPN module driver to the VPN module hardware as follows:

```
Router# debug crypto engine accelerator logs
encryption module logs debugging is on
```

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crypto engine accelerator</td>
<td>Enables or disables the crypto engine accelerator if it exists.</td>
</tr>
<tr>
<td>show crypto engine accelerator logs</td>
<td>Prints information about the last 32 CGX Library packet processing commands, and associated parameters sent from the VPN module driver to the VPN module hardware.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>show crypto engine accelerator sa-database</code></td>
<td>Prints active (in-use) entries in the platform-specific VPN module database.</td>
</tr>
<tr>
<td><code>show crypto engine configuration</code></td>
<td>Displays the Cisco IOS crypto engine of your router.</td>
</tr>
</tbody>
</table>
debug crypto engine ism-vpn

To enable debugging for a Cisco VPN Internal Service Module (ISM), use the `debug crypto engine ism-vpn` command in privileged EXEC mode. To disable debugging, use the `no` form of this command.

```
debug crypto engine ism-vpn [init| interrupt| polo| detail| dump] | shim| detail| tftp]
no debug crypto engine ism-vpn [init| interrupt| polo| detail| dump] | shim| detail| tftp]
```

**Syntax Description**

- **init**  
  (Optional) Enables initial state (INIT state) debugging for an Application Control Engine (ACE) appliance in Cisco VPN ISM.

- **interrupt**  
  (Optional) Enables interrupt service debugging.

- **polo**  
  (Optional) Enables VPN ISM polo debugging.

- **detail**  
  (Optional) Enables detailed debugging.

- **dump**  
  (Optional) Enables packet dump polo debugging.

- **shim**  
  (Optional) Enables VPN ISM shim layer debugging.

- **tftp**  
  (Optional) Enables debugging for TFTP download or upload on the Cisco VPN ISM.

**Command Default**

Cisco VPN ISM debugging is not enabled.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.3(2)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following example shows how to enable initial state (INIT state) debugging.

```
Device# debug crypto engine ism-vpn init
INIT debugging is on
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug crypto engine ism-vpn ssl</code></td>
<td>Enables debugging for Cisco VPN ISM SSL.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>debug crypto engine ism-vpn traffic</code></td>
<td>Enables debugging for Cisco VPN ISM traffic.</td>
</tr>
<tr>
<td><code>debug crypto engine ism-vpn traffic selector</code></td>
<td>Enables debugging for specific traffic in a Cisco VPN ISM.</td>
</tr>
<tr>
<td><code>show crypto engine accelerator statistic</code></td>
<td>Displays IPsec encryption statistics and error counters for the onboard hardware accelerator of a device, the IPsec VPN SPA, or the Cisco VPN ISM.</td>
</tr>
</tbody>
</table>
**debug crypto engine ism-vpn ssl**

To enable debugging for Secure Sockets Layer (SSL) in a Cisco VPN Internal Service Module (VPN ISM), use the `debug crypto engine ism-vpn ssl` command in privileged EXEC mode. To disable debugging, use the `no` form of this command.

```plaintext
debug crypto engine ism-vpn ssl {[context rule-number session-id] [direction number]}
no debug crypto engine ism-vpn ssl {[context rule-number session-id] [direction number]}
```

**Syntax Description**

- `context rule-number session-id` Enables debugging for a specific SSL depending on the rule or the session. The range for the rule number is from 1 to 10. The range for the session ID is 0 to 4294967294.

- `direction number` Enables debugging for packets that are in the inbound or outbound direction. The range for the direction number is from 0 to 3, where 0 indicates the inbound and outbound direction, 1 indicates encryption, 2 indicates decryption, and 3 indicates exceptions.

**Command Default**

Cisco VPN ISM SSL debugging is not enabled.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.3(2)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following example shows how to enable debugging of rule 3 and context 1:

```plaintext
Device# debug crypto engine ism-vpn ssl context 3 1
ISM-VPN Enable rule 3, ctx: 1
ISM-VPN SSL Context debugging is on
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug crypto engine ism-vpn</code></td>
<td>Enables debugging for a Cisco VPN ISM.</td>
</tr>
<tr>
<td><code>debug crypto engine ism-vpn traffic</code></td>
<td>Enables debugging for Cisco VPN ISM traffic.</td>
</tr>
<tr>
<td><code>debug crypto engine ism-vpn traffic selector</code></td>
<td>Enables debugging for specific traffic in a Cisco VPN ISM.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>show crypto engine accelerator statistic</td>
<td>Displays IPsec encryption statistics and error counters for the onboard hardware accelerator of a device, the IPsec VPN SPA, or Cisco VPN ISM.</td>
</tr>
</tbody>
</table>
debug crypto engine ism-vpn traffic

To enable debugging for Cisco VPN Internal Service Module (ISM) traffic, use the `debug crypto engine ism-vpn traffic` command in privileged EXEC mode. To disable debugging, use the `no` form of this command.

```
d debug crypto engine ism-vpn traffic {all| detail| exception number| inbound| outbound| selective| vam}
no debug crypto engine ism-vpn traffic {all| detail| exception number| inbound| outbound| selective| vam}
```

**Syntax Description**

- **all**
  - Enables debugging for all traffic that flows to the Cisco VPN ISM.

- **detail**
  - Enables detailed debugging for all traffic that flows to the Cisco VPN ISM.

- **exception number**
  - Enables debugging for specified exceptions in the traffic. The range is from 0 to 54. 0 enables debugging for all exceptions.

- **inbound**
  - Enables debugging for inbound traffic.

- **outbound**
  - Enables debugging for outbound traffic.

- **selective**
  - Enables selective rules for the traffic on the Cisco VPN ISM.

- **vam**
  - Enables debugging for the Cisco VPN Acceleration Module (VAM) in the Cisco VPN ISM.

**Command Default**

Cisco VPN ISM traffic debugging is not enabled.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.3(2)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following shows how to enable debugging for all exceptions in the traffic:

```
Device# debug crypto engine ism-vpn traffic exception 0
ISM-VPN Traffic Detail debugging is on
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug crypto engine ism-vpn</td>
<td>Enables debugging for a Cisco VPN ISM.</td>
</tr>
</tbody>
</table>
### Command Reference

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug crypto engine ism-vpn traffic selector</code></td>
<td>Enables debugging for specific traffic in a Cisco VPN ISM.</td>
</tr>
<tr>
<td><code>debug crypto engine ism-vpn ssl</code></td>
<td>Enables debugging for Cisco VPN ISM SSL.</td>
</tr>
<tr>
<td><code>show crypto engine accelerator statistic</code></td>
<td>Displays IPsec encryption statistics and error counters for the onboard hardware accelerator of a device, the IPsec VPN SPA, or the Cisco VPN ISM.</td>
</tr>
</tbody>
</table>
# debug crypto engine ism-vpn traffic selector

To enable debug rules for specific traffic in Cisco VPN Internal Service Module (ISM), use the **debug crypto engine ism-vpn traffic selector** command in privileged EXEC mode. To disable debugging, use the `no` form of this command.

```
debug crypto engine ism-vpn traffic selector {ipv4 | ipv6} {disable rule-number | enable rule-number source-ip-address source-address-mask destination-ip-address destination-address-mask protocols}
```

```
no debug crypto engine ism-vpn traffic selector {ipv4 | ipv6} {disable rule-number | enable rule-number source-ip-address source-address-mask destination-ip-address destination-address-mask protocols}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipv4</td>
<td>Enables debugging rules for IPv4 addresses.</td>
</tr>
<tr>
<td>ipv6</td>
<td>Enables debugging rules for IPv6 addresses.</td>
</tr>
<tr>
<td>disable</td>
<td>Enables debugging rules are disabled.</td>
</tr>
<tr>
<td>enable</td>
<td>Enables debugging rules are enabled.</td>
</tr>
<tr>
<td>rule-number</td>
<td>A specific rule. The range is from 1 to 10.</td>
</tr>
<tr>
<td>source-ip-address source-address-mask</td>
<td>Source IP address and network mask of the traffic for which rules must be enabled or disabled.</td>
</tr>
<tr>
<td>destination-ip-address destination-address-mask</td>
<td>Destination IP address and network mask of the traffic for which rules must be enabled or disabled.</td>
</tr>
<tr>
<td>protocols</td>
<td>Protocols in the source and destination IP addresses of the traffic for which rules must be enabled or disabled. The range is from 0 to 155. 0 denotes all protocols.</td>
</tr>
</tbody>
</table>

### Command Default

Cisco VPN ISM selective traffic debugging is not enabled.

### Command Modes

Privileged EXEC (`#`)

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.3(2)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>
The following example shows how to enable the debugging of rule 0 for all protocols from source address 10.0.0.1 255.255.255.0 and destination address 10.0.0.2 255.255.255.0:

```
Device# debug crypto engine ism-vpn traffic selector enable 0 10.0.0.1 255.255.255.0 10.0.0.2 255.255.255.0 1
ISM-VPN Enable rule 0, s: 10.0.0.1 d: 10.0.0.2 p: 1
```

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug crypto engine ism-vpn</code></td>
<td>Enables debugging for a Cisco VPN ISM.</td>
</tr>
<tr>
<td><code>debug crypto engine ism-vpn traffic</code></td>
<td>Enables debugging for Cisco VPN ISM traffic.</td>
</tr>
<tr>
<td><code>debug crypto engine ism-vpn ssl</code></td>
<td>Enables debugging for Cisco VPN ISM SSL.</td>
</tr>
<tr>
<td><code>show crypto engine accelerator statistic</code></td>
<td>Displays IPsec encryption statistics and error counters for the onboard hardware accelerator of a device, the IPsec VPN SPA, or Cisco VPN ISM.</td>
</tr>
</tbody>
</table>
debug crypto error

To enable error debugging for a crypto area, use the **debug crypto error** command in privileged EXEC mode. To disable crypto error debugging, use the **no** form of this command.

```
debug crypto {isakmp | ipsec | engine} error
no debug crypto {isakmp | ipsec | engine} error
```

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>isakmp</strong></td>
<td>Debug messages are shown for Internet Key Exchange (IKE)-related error operations only.</td>
</tr>
<tr>
<td><strong>ipsec</strong></td>
<td>Debug messages are shown for IP Security (IPSec)-related error operations only.</td>
</tr>
<tr>
<td><strong>engine</strong></td>
<td>Debug messages are shown for crypto engine-related error operations only.</td>
</tr>
</tbody>
</table>

**Command Default**
Crypto error debugging is not enabled.

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(2)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(18)SXD</td>
<td>This command was integrated into Cisco IOS Release 12.2(18)SXD.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
The **debug crypto error** command will display only error-related debug messages; that is, an error debug will not be shown if the operation is functioning properly.

This command should be used when debug conditions cannot be determined; for example, enable this command when a random, small subset of IKE peers is failing negotiation.

**Note**
The global crypto command-line interfaces (CLIs) (the **debug crypto isakmp**, **debug crypto ipsec**, and **debug crypto engine** commands) will override the **debug crypto error** command. Thus, this command should not be used in conjunction with the global CLIs because you may overwhelm the router.
Debug message filtering for crypto hardware engines is not supported.

Examples

The following example shows how to enable IPSec-related error messages:

```
Router# debug crypto error ipsec error
```
debug crypto gdoi

To display information about a Group Domain of Interpretation (GDOI) configuration, use the `debug crypto gdoi` command in privileged EXEC mode. To disable crypto GDOI debugging, use the `no` form of this command.

```
ddebug crypto gdoi [all-features | all-levels | detail | error | event | gm | infrastructure | ks | acls | coop] | packet | replay | terse]
```

```
no debug crypto gdoi [all-features | all-levels | detail | error | event | gm | infrastructure | ks | acls | coop] | packet | replay | terse]
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>all-features</code></td>
<td>(Optional) Displays debug data for all features in GDOI. This consists of rekey, cooperative key server, replay, and registration information (for KSs) and rekey, registration, and replay information (for GMs).</td>
</tr>
<tr>
<td><code>all-levels</code></td>
<td>(Optional) Displays all debug levels (detail, error, event, packet, and terse).</td>
</tr>
<tr>
<td><code>detail</code></td>
<td>(Optional) Displays detailed debug information.</td>
</tr>
<tr>
<td><code>error</code></td>
<td>(Optional) Displays information about error debugs.</td>
</tr>
<tr>
<td><code>event</code></td>
<td>(Optional) Displays user-level information.</td>
</tr>
<tr>
<td><code>gm</code></td>
<td>(Optional) Displays information about group members.</td>
</tr>
<tr>
<td><code>infrastructure</code></td>
<td>(Optional) Displays information about the GDOI infrastructure.</td>
</tr>
<tr>
<td><code>ks</code></td>
<td>(Optional) Displays information about key servers.</td>
</tr>
<tr>
<td><code>acls</code></td>
<td>(Optional) Displays information about implementation of ACLs.</td>
</tr>
<tr>
<td><code>coop</code></td>
<td>(Optional) Displays information about cooperative key servers.</td>
</tr>
<tr>
<td><code>packet</code></td>
<td>(Optional) Displays information about packet-level debugs (administrator-level information).</td>
</tr>
<tr>
<td><code>replay</code></td>
<td>(Optional) Displays information about the pseudotime stamp that is contained in a packet.</td>
</tr>
<tr>
<td><code>terse</code></td>
<td>(Optional) Displays lowest-level debugs (message-level information).</td>
</tr>
</tbody>
</table>
The detail, error, event, packet, and terse keywords can be used with the feature keywords (for example, gmerror, infrastructureerror, kscoopevent, replayerror).

Command Default
Debugging is turned off.

Command Modes
Privileged EXEC (#)

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(6)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(11)T</td>
<td>The detail, error, event, gm, infrastructure, ks, coop, packet, replay, and terse keywords were added.</td>
</tr>
<tr>
<td>Cisco IOS XE Release 2.3</td>
<td>This command was integrated into Cisco IOS XE Release 2.3.</td>
</tr>
<tr>
<td>15.1(3)T</td>
<td>This command was modified. The acls and replay keywords were removed. The all-features and all-levels keywords were added.</td>
</tr>
<tr>
<td>Cisco IOS XE Release 3.8S</td>
<td>This command was modified. The acls and replay keywords were removed. The all-features and all-levels keywords were added.</td>
</tr>
</tbody>
</table>

Usage Guidelines
Using this command displays various GDOI debugs. For debugging information for cooperative key servers, use the debugcryptogdoiko command.

If you do not specify a feature (using the all-features, registration, rekey, replay, coop, or infrastructure keywords), then messages for all features are displayed. If you do not specify a level (using the detail, error, event, packet, and terse keywords), then the terse level (which also includes the error level) is used.

You can use this command in conjunction with the debugcryptogdoicondition command. When these two commands are used together, only those messages that pass through any debug level or feature (specified by the debugcryptogdoi command) and pass through any condition (specified by the debugcryptogdoicondition command) are displayed.

Examples
The following example shows group member registration debug output:

Router# debug crypto gdoi
00:00:40: GDOI:(0:0:N/A:0):GDOI group diffint
00:00:40: %CRYPTO-5-GM_REGISTER: Start registration for group diffinit using address 10.0.3.1
00:00:40: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is ON
00:00:40: GDOI:(0:0101:HW:0:3333):beginning GDOI exchange, M-ID of 1167145075
00:00:40: GDOI: Group Number is 3333
00:00:40: GDOI:(0:0101:HW:0:3333):GDOI: GDOI ID sent successfully
The following output example shows key server registration debugs:

```plaintext
Router# debug crypto gdoi
00:00:40: GDOI:(0:1001:HW:0:3333):processing GDOI ID payload, message ID = 1167145075
00:00:40: GDOI:(0:1001:HW:0:3333):The GDOI ID is a Number: 3333
00:00:40: GDOI:(0:0:N/A:0):Adding KEK Policy to the current ks_group
00:00:40: GDOI:(0:0:N/A:0):Setting MULTICAST TEK rekey lifetime 30
00:00:40: GDOI:(0:0:N/A:0):GDOI SA sent successfully by KS
00:00:40: GDOI:(0:0:N/A:0):GDOI KD sent successfully by KS
The following output example shows group member rekey debugs:

Router# debug crypto gdoi
00:02:00: GDOI:(0:1002:HW:0:0):Received Rekey Message!
00:02:00: GDOI:(0:1002:HW:0:0):Signature Valid!
00:02:00: GDOI:received payload type 18
00:02:00: GDOI:(0:1002:HW:0:0):processing GDOI Seq Payload, message_id 0
00:02:00: GDOI:(0:1002:HW:0:0):Completed SEQ Processing for seq 0
00:02:00: GDOI:(0:1002:HW:0:0):processing GDOI SA Payload, message ID + 0
00:02:00: GDOI:(0:1002:HW:0:0):processing GDOI SA KEK Payload
00:02:00: GDOI:(0:1002:HW:0:0): KEK_ALGORITHM 5
00:02:00: GDOI:(0:1002:HW:0:0): KEY_LENGTH 24
00:02:00: GDOI:(0:1002:HW:0:0): KEY_LIFETIME 249
00:02:00: GDOI:(0:1002:HW:0:0): SIG_ALG I
00:02:00: GDOI:(0:1002:HW:0:0): SIG_KEY_LEN 94
00:02:00: GDOI:(0:1002:HW:0:0): Completed KEK Processing
00:02:00: GDOI:(0:1002:HW:0:0):processing GDOI SA TEK Payload
00:02:00: GDOI:(0:1002:HW:0:0):Completed TEK Processing
00:02:00: GDOI:(0:1002:HW:0:0):processing GDOI SA TEK Payload
00:02:00: GDOI:(0:1002:HW:0:0):Completed TEK Processing
00:02:00: GDOI:received payload type 17
00:02:00: GDOI:(0:1002:HW:0:0):processing GDOI KD Payload, message_id 0
00:02:00: GDOI:(0:1002:HW:0:0):processing GDOI Key Packet, message_id 38649336
00:02:00: GDOI:(0:1002:HW:0:0):procesing TEK KD: spi is 56165461, spi is 49195284, spi
```
00:02:00: GDOI:(0:1002:HW:0):TEK Integrity Key 20 bytes
00:02:00: GDOI:(0:1002:HW:0):Completed KeyPkt Processing
enc(config-if)#
00:02:00: GDOI:(0:1002:HW:0):processing GDOI Key Packet, message_id 38649336
00:02:00: GDOI:(0:1002:HW:0):processing TEK KD: spi is 49193345, spi
00:02:00: GDOI:(0:1002:HW:0):TEK Integrity Key 20 bytes
00:02:00: GDOI:(0:1002:HW:0):Completed KeyPkt Processing
00:02:00: GDOI:(0:1002:HW:0):Processing KEK KD
00:02:00: GDOI:(0:1002:HW:0):Completed KeyPkt Processing
debug crypto gdoi condition

To configure conditional filters (based on groups and peers) for GDOI debugging, use the debug crypto gdoi condition command in privileged EXEC mode. To disable conditional filters, use the no form of this command.

```
debug crypto gdoi condition {[group group-name] [peer {address ipv4 ipv4-address-of-peer| hostname ipv4 ipv4-hostname}] | reset| unmatched}
no debug crypto gdoi condition {[group group-name] [peer {address ipv4 ipv4-address-of-peer| hostname ipv4 ipv4-hostname}] | reset| unmatched}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group group-name</td>
<td>(Optional) Displays information for a specific group.</td>
</tr>
<tr>
<td>address ipv4 ipv4-address-of-peer</td>
<td>(Optional) Displays information for a specific IPv4 peer.</td>
</tr>
<tr>
<td>hostname ipv4 ipv4-hostname</td>
<td>(Optional) Displays information for a specific IPv4 host.</td>
</tr>
<tr>
<td>reset</td>
<td>(Optional) Removes all debugging conditions.</td>
</tr>
<tr>
<td>unmatched</td>
<td>(Optional) Displays debug messages if no context is available.</td>
</tr>
</tbody>
</table>

**Command Default**

Conditional debugging is turned off.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.1(3)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>Cisco IOS XE Release 3.8S</td>
<td>This command was integrated into Cisco IOS XE Release 3.8S.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

This command lets you filter the number of debug messages to make it easier to identify the problem of a particular group member (GM). This command is useful when many (such as thousands of) GMs are registered to a key server on which debugging is enabled.

You can use this command in conjunction with the debug crypto gdoi command. When these two commands are used together, only those messages that pass through any debug level or feature (specified by the
debugcryptogdoi command) and pass through any condition (specified by the debugcryptogdoicondition command) are displayed.

You can use this command multiple times to configure conditional debugging for any number of GMs. You can have more than one group configured and more than one GM registered to each group.

If no GM is specified, debug messages will appear for all GMs (this occurs automatically when all conditional filtering is removed).

Also, this command lets you filter per cooperative key server on each key server. This command is useful when there are many key servers in a system.

The **unmatched** keyword displays messages when there is insufficient information to perform conditional debugging. The **unmatched** keyword is enabled for the error, terse, and event debug levels.

To remove the debug messages for a GM or a key server, use the **no** form of the same command.

**Examples**

The following example shows how to configure a conditional filter for a GDOI group named group1:

```bash
Router# debug crypto gdoi condition group group1
```

The following output example shows how to configure a conditional filter for IPv4 peers:

```bash
Router# debug crypto gdoi condition peer
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug crypto condition</td>
<td>Defines conditional debug filters for IP Security (IPSec) tunnels.</td>
</tr>
</tbody>
</table>
**debug crypto ha**

To display crypto high availability debugging information, use the `debug crypto ha` command in privileged EXEC mode. To disable debugging messages, use the `no` form of this command.

```
debug crypto ha
no debug crypto ha
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(11)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following example is sample output from the `debug crypto ha` command:

```
Router# debug crypto ha
Active router: 
Router# show debug

Cryptographic Subsystem:
  Crypto High Availability Manager debugging is on
vrf-lite-R1#
*Sep 28 21:27:50.899:S:Sending IKE Add SA Message
*Sep 28 21:27:50.899:HA Message 0:flags=0x08 len=394 HA_IKE_MSG_ADD_SA (2)
*Sep 28 21:27:50.899: ID:04000003
*Sep 28 21:27:50.899: attr HA_IKE_ATT_MY_COOKIE (2) len 8
  9B 1A 76 AA 99 11 1A 1F
*Sep 28 21:27:50.899: attr HA_IKE_ATT_HIS_COOKIE (3) len 8
  E2 A2 A3 5F 53 1D EA 15
*Sep 28 21:27:50.899: attr HA_IKE_ATT_SRC (4) len 4
  04 00 00 05
*Sep 28 21:27:50.899: attr HA_IKE_ATT_DST (5) len 4
  04 00 00 03
*Sep 28 21:27:50.899: attr HA_IKE_ATT_PEER_PORT (6) len 2
  01 F4
*Sep 28 21:27:50.899: attr HA_IKE_ATT_F_VRF (7) len 1
  00
*Sep 28 21:27:50.899: attr HA_IKE_ATT_INIT_OR_RESP (8) len 1
  00
*Sep 28 21:27:50.899: attr HA_IKE_ATT_NAT_DISCOVERY (9) len 1
  00
*Sep 28 21:27:50.899: attr HA_IKE_ATT_IDTYPE (38) len 1
  02
*Sep 28 21:27:50.899: attr HA_IKE_ATT_IDTYPE (38) len 1
  01
*Sep 28 21:27:50.899: attr HA_IKE_ATT_PROTOCOL (39) len 1
  11
*Sep 28 21:27:50.899: attr HA_IKE_ATT_PORT (40) len 2
  01 F4
*Sep 28 21:27:50.899: attr HA_IKE_ATT_ADDR (41) len 4
  04 00 00 05
*Sep 28 21:27:50.899: attr HA_IKE_ATT_MASK (42) len 4
  00 00 00 00
```
*Sep 28 21:27:50.899: attr HA_IKE_ATT_ID_STR (44) len 4
*Sep 28 21:27:50.899: 00 00 00 00
*Sep 28 21:27:50.899: attr HA_IKE_ATT_PEERS_CAPABILITIES (11) len 4
*Sep 28 21:27:50.899: 00 00 07 7F
*Sep 28 21:27:50.899: attr HA_IKE_ATT_MY_CAPABILITIES (12) len 4
*Sep 28 21:27:50.899: 00 00 07 7F
*Sep 28 21:27:50.899: attr HA_IKE_ATT_STATE_MASK (13) len 4
*Sep 28 21:27:50.899: 00 00 27 FF
.
.
.

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug crypto ipsec ha</td>
<td>Enables debugging messages for IPSec high availability.</td>
</tr>
<tr>
<td>debug crypto isakmp ha</td>
<td>Enables debugging messages for ISAKMP high availability.</td>
</tr>
</tbody>
</table>
To display IP Security (IPSec) events for IPv6 networks, use the `debug crypto ipv6 ipsec` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug crypto ipv6 ipsec
no debug crypto ipv6 ipsec
```

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

**Command Default**
Debugging for IPv6 IPSec events is not enabled.

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(4)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>Cisco IOS XE Release 2.1</td>
<td>This command was introduced on Cisco ASR 1000 Series Routers.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
Use this command to display IPSec events while setting up or removing policy definitions during OSPF configuration.

**Examples**
The following example enables the display of IPSec events for IPv6 networks:

```
Router# debug crypto ipv6 ipsec
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug crypto engine</td>
<td>Displays debugging messages about crypto engines, which perform encryption and decryption.</td>
</tr>
<tr>
<td>debug crypto ipv6 packet</td>
<td>Displays debug messages for IPv6 packets allowing you to see the contents of packets outbound from a Cisco router when the remote node is not a Cisco node.</td>
</tr>
<tr>
<td>debug crypto socket</td>
<td>Displays communication between the client and IPSec during policy setup and removal processes.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>debug ipv6 ospf authentication</td>
<td>Displays the interaction between OSPF and IPSec, including creation or removal of policies.</td>
</tr>
</tbody>
</table>
debug crypto ipv6 packet

To display the contents of IPv6 packets, use the `debug crypto ipv6 packet` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug crypto ipv6 packet
no debug crypto ipv6 packet
```

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

**Command Default**
Debugging for IPv6 IPSec packets is not enabled.

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(4)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>Cisco IOS XE Release 2.1</td>
<td>This command was introduced on Cisco ASR 1000 Series Routers.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
Consult Cisco Technical Support before using this command.
Use this command to display the contents of IPv6 packets. This command is useful when the remote node is not a Cisco device and communication between the Cisco and non-Cisco router cannot be established. This command enables you to look at the contents of the packets outbound from the Cisco router.

This command examines the content of every IPv6 packet and may slow network performance.

**Examples**
This example shows the output of each packet when the `debug crypto ipv6 packet` command is enabled:

```
Router# debug crypto ipv6 packet
Crypto IPv6 IPSEC packet debugging is on
*Oct 30 16:57:06.330:
IPSECv6:before Encapsulation of IPv6 packet:
0E37A7C0: 6E000000 00285901 n....{Y.
0E37A7D0:FE800000 00000000 020A8BFF FED42C1D ..........~T,
0E37A7E0:FF020000 00000000 00000000 00000005 ...............
0E37A7F0:03010028 01010104 00000001 8AD80000 .........{..PO
0E37A810:01010104 0A0250CF ...........
*Oct 30 16:57:06.330:
IPSECv6:Encapsulated IPv6 packet
:
0E37A7B0:6E000000 00403301 FE800000 00000000 n....@3.~........
0E37A7C0:020A8BFF FED42C1D FF020000 00000000 ..........~T,
0E37A7D0:00000000 00000000 59040000 000022B8 ............Y........*B
0E37A7E0:0000001A 38AB1ED6 04C166FB FF1248CF .....8+.X.AF|..HO
0E37A7F0:03010028 01010104 00000001 8AD80000 ..........X..```

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>debug crypto engine</strong></td>
<td>Displays debugging messages about crypto engines, which perform encryption and decryption.</td>
</tr>
<tr>
<td><strong>debug crypto ipv6 ipsec</strong></td>
<td>Displays IPSec events for IPv6 networks.</td>
</tr>
<tr>
<td><strong>debug crypto socket</strong></td>
<td>Displays communication between the client and IPSec during policy setup and removal processes.</td>
</tr>
</tbody>
</table>
debug crypto ikev2

To enable Internet Key Exchange Version 2 (IKEv2) debug messages, use the debug crypto ikev2 command in privileged EXEC mode.

d debug crypto ikev2 [error| terse| event| packet| detail]
no debug crypto ikev2 [error| terse| event| packet| detail]

Syntax Description

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>(Optional) Enables debug messages capturing IKEv2 errors.</td>
</tr>
<tr>
<td>terse</td>
<td>(Optional) Enables debug messages capturing IKEv2 message exchanges.</td>
</tr>
<tr>
<td>event</td>
<td>(Optional) Enables debug messages capturing IKEv2 packet description, contents, and policy matching.</td>
</tr>
<tr>
<td>packet</td>
<td>(Optional) Enables debug messages capturing IKEv2 packet dump.</td>
</tr>
<tr>
<td>detail</td>
<td>(Optional) Enables debug messages capturing IKEv2 state machine events.</td>
</tr>
</tbody>
</table>

Command Modes

Privileged EXEC (#)

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.1(1)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

Cisco IOS XE Release 3.3S This command was integrated into Cisco IOS XE Release 3.3S.

Usage Guidelines

Use this command to enable IKEv2 debug messages. IKEv2 uses following debug levels.

- Level 1--error
- Level 2--terse
- Level 3--event
- Level 4--packet
- Level 5--detail
**Note**
Level 1 is the lowest level and is least verbose and level 5 is highest level. Enabling debug at a higher level enables debug at all lower levels. You can selectively disable a debug level and enable conditional debug using the `debug crypto condition` command.

**Examples**
The following example shows that the IKEv2 debugging is enabled:

```
Router# debug crypto ikev2
debugging is on
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug crypto condition</code></td>
<td>Enables conditional debugging.</td>
</tr>
</tbody>
</table>
debug crypto ipsec

To display IP security (IPsec) events, use the **debug crypto ipsec** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug crypto ipsec**

**no debug crypto ipsec**

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Cisco IOS XE Release 2.6</td>
<td>This command was integrated into Cisco IOS XE Release 2.6.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the **debug crypto ipsec** command. In this example, security associations (SAs) have been successfully established.

**Router# debug crypto ipsec**

IPsec requests SAs between 172.21.114.123 and 172.21.114.67, on behalf of the **permit ip host 172.21.114.123 host 172.21.114.67** command. IPsec is configured to first use the set esp-des w/esp-md5-hmac, but it will also use ah-sha-hmac on a secondary basis.

00:24:30: IPSEC (sa_request): ,
(key eng. msg.) src= 172.21.114.123, dest= 172.21.114.67,
src_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1),
dest_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1),
protocol= ESP, transform= esp-des esp-md5-hmac ,
lifedur= 120s and 4608000kb,
spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4
00:24:30: IPSEC (sa_request): ,
(key eng. msg.) src= 172.21.114.123, dest= 172.21.114.67,
src_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1),
dest_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1),
protocol= AH, transform= ah-sha-hmac ,
lifedur= 120s and 4608000kb,
spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4.

Internet Key Exchange (IKE) prompts for Service Provider Interfaces (SPIs) from IPsec. For inbound security associations, IPsec controls its own SPI space.

IKE will verify whether IPsec accepts the SA proposal. In this example, it is the SA proposal sent by the local IPsec that is accepted first.

00:24:34: IPSEC(validate_proposal_request): proposal part #1, (key eng. msg.) dest= 172.21.114.67, src= 172.21.114.123, dest_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1), src_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1), protocol= ESP, transform= esp-des esp-md5-hmac , lifedur= 0s and 0kb, spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4

After the proposal is accepted, IKE finishes the negotiations, generates the keying material, and then notifies IPsec of the new security associations (one security association for each direction).

00:24:35: IPSEC(key_engine): got a queue event...

The following output pertains to the inbound SA. The conn_id value references an entry in the crypto engine connection table.

00:24:35: IPSEC(initialize_sas): , (key eng. msg.) dest= 172.21.114.123, src= 172.21.114.67, dest_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1), src_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1), protocol= ESP, transform= esp-des esp-md5-hmac , lifedur= 120s and 4608000 kb, spi= 0x120f043c(302974012), conn_id= 29, keysize= 0, flags= 0x4

The following output pertains to the outbound SA:

00:24:35: IPSEC(initialize_sas): , (key eng. msg.) src= 172.21.114.123, dest= 172.21.114.67, src_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1), dest_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1), protocol= ESP, transform= esp-des esp-md5-hmac , lifedur= 120s and 4608000 kb, spi= 0x38914a4(59315364), conn_id= 30, keysize= 0, flags= 0x4

IPsec then installs the SA information into its SA database.

00:24:35: IPSEC(create_sa): sa created, (sa) sa_dest= 172.21.114.123, sa_prot= 50, sa_spi= 0x120f043c(302974012), sa_trans= esp-des esp-md5-hmac , sa_conn_id= 29 sa_lifetime(k/sec)= (4445923/500)

The following is sample output from the debugcryptoipsec command as seen on the peer router. In this example, IKE verifies whether IPsec will accept an SA proposal. Although the peer sent two proposals, IPsec accepted the first proposal.

00:26:15: IPSEC(validate_proposal_request): proposal part #1, (key eng. msg.) dest= 172.21.114.67, src= 172.21.114.123, dest_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1), src_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1), protocol= ESP, transform= esp-des esp-md5-hmac , lifedur= 0s and 0kb, spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4

IKE prompts for SPIs.

00:26:15: IPSEC(key_engine): got a queue event...
IKE does the remaining processing, completing the negotiation and generating keys. IKE then notifies IPsec about the new SAs.

00:26:15: IPSEC(key_engine): got a queue event...

The following output pertains to the inbound SA:

00:26:15: IPSEC(initialize_sas),
(key eng. msg.) dest= 172.21.114.67, src= 172.21.114.123,
dest_proxy= 172.21.114.67/0.0.0.0/0/0 (type-1),
src_proxy= 172.21.114.123/0.0.0.0/0/0 (type-1),
protocol= ESP, transform= esp-des esp-md5-hmac ,
lifedur= 120s and 4608000kb,
spi= 0x38914A4(59315364), conn id= 25, keysize= 0, flags= 0x4

The following output pertains to the outbound SA:

00:26:15: IPSEC(initialize_sas),
(key eng. msg.) src= 172.21.114.67, dest= 172.21.114.123,
src_proxy= 172.21.114.67/0.0.0.0/0/0 (type-1),
dest_proxy= 172.21.114.123/0.0.0.0/0/0 (type-1),
protocol= ESP, transform= esp-des esp-md5-hmac ,
lifedur= 120s and 4608000kb,
spi= 0x120F043C(302974012), conn id= 26, keysize= 0, flags= 0x4

IPsec then installs the SA information into its SA database:

00:26:15: IPSEC(create_sa), sa created,
(sa) sa_dest= 172.21.114.67, sa_prot= 50,
sa_spi= 0x38914A4(59315364),
sa_trans= esp-des esp-md5-hmac , sa_conn_id= 25
sa_lifetime(k/sec)= (4445923/500)
00:26:15: IPSEC(create_sa), sa created,
(sa) sa_dest= 172.21.114.123, sa_prot= 50,
sa_spi= 0x120F043C(302974012),
sa_trans= esp-des esp-md5-hmac , sa_conn_id= 26
sa_lifetime(k/sec)= (4445923/500)
**debug crypto ipsec client ezvpn**

To display information about voice control messages that have been captured by the Voice DSP Control Message Logger and about Cisco Easy VPN remote connections, use the `debug crypto ipsec client ezvpn` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug crypto ipsec client ezvpn
no debug crypto ipsec client ezvpn
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(4)YA</td>
<td>This command was introduced on Cisco 806, Cisco 826, Cisco 827, and Cisco 828 routers; Cisco 1700 series routers; and Cisco uBR905 and Cisco uBR925 cable access routers.</td>
</tr>
<tr>
<td>12.2(13)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(13)T.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was expanded to support the Easy VPN Remote feature.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS 12.2SX family of releases. Support in a specific 12.2SX release is dependent on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

To force the Voice DSP Contol Message Logger to reestablish the virtual private network (VPN) connections, use the `clearcryptoasandclearcryptoisakmp` commands to delete the IPSec security associations and Internet Key Exchange (IKE) connections, respectively.

**Examples**

The following example shows debugging messages when the Voice DSP Contol Message Logger is turned on and typical debugging messages that appear when the VPN tunnel is created:

```
Router# debug crypto ipsec client ezvpn

EzVPN debugging is on
router# 00:02:28: EZVPN(hw1): Current State: IPSEC_ACTIVE
```
The following example shows the typical display for a VPN tunnel that is reset with the `clearcryptoipsecclientezvpn` command:

```
3d17h: EZVPN: Current State: READY
3d17h: EZVPN: Event: RESET
3d17h: ezvpn_reconnect_request
3d17h: ezvpn_close
3d17h: ezvpn_disconnect
3d17h: EZVPN: No state change
3d17h: EZVPN: Current State: READY
3d17h: EZVPN: Event: MODE_CONFIG_REPLY
3d17h: EZVPN: Attributes sent in message:
3d17h: DNS Primary: 172.16.0.250
3d17h: DNS Secondary: 172.16.0.251
3d17h: NBMS/WINS Primary: 172.16.0.252
3d17h: NBMS/WINS Secondary: 172.16.0.253
3d17h: Split Tunnel List: 1
3d17h: Address: 172.16.0.128
3d17h: Mask: 255.255.255.128
3d17h: Protocol: 0x0
3d17h: Source Port: 0
3d17h: Dest Port: 0
3d17h: Split Tunnel List: 2
3d17h: Address: 172.16.1.128
3d17h: Mask: 255.255.255.128
3d17h: Protocol: 0x0
3d17h: Source Port: 0
3d17h: Dest Port: 0
3d17h: Default Domain: cisco.com
3d17h: ezvpn_nat_config
3d17h: EZVPN: Current State: SS_OPEN
3d17h: EZVPN: No state change
3d17h: EZVPN: Current State: SS_OPEN
3d17h: EZVPN: Event: SOCKET_READY
3d17h: EZVPN: No state change
3d17h: EZVPN: Current State: SS_OPEN
3d17h: EZVPN: Event: MTU_CHANGED
3d17h: EZVPN: No state change
3d17h: EZVPN: Current State: SS_OPEN
3d17h: EZVPN: Event: SOCKET_UP
3d17h: EZVPN: New State: IPSEC_ACTIVE
```
3d17h: EZVPN: Current State: IPSEC_ACTIVE
3d17h: EZVPN: Event: MTU_CHANGED
3d17h: EZVPN: No state change
3d17h: EZVPN: Current State: IPSEC_ACTIVE
3d17h: EZVPN: Event: SOCKET_UP
The following example shows the typical display for a VPN tunnel that is removed from the interface with the `nocryptoipsecclientezvpn` command:

4d16h: EZVPN: Current State: IPSEC ACTIVE
4d16h: EZVPN: Event: REMOVE INTERFACE CFG
4d16h: ezvpn_close_and_remove
4d16h: ezvpn_close
4d16h: ezvpn_remove
4d16h: EZVPN: New State: IDLE

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug crypto ipsec</code></td>
<td>Displays debugging messages for generic IPSec events.</td>
</tr>
<tr>
<td><code>debug crypto isakmp</code></td>
<td>Displays debugging messages for IKE events.</td>
</tr>
</tbody>
</table>
**debug crypto ipsec ha**

To enable debugging messages for IP Security (IPSec) high availability, use the `debug crypto ipsec ha` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
detail
update
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>detail</td>
<td>(Optional) Displays detailed debug information.</td>
</tr>
<tr>
<td>update</td>
<td>(Optional) Displays updates for inbound and outbound related data.</td>
</tr>
</tbody>
</table>

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(11)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

### Examples

The following example is sample output of the `debug crypto ipsec ha` command for both the active and stanby router:

**Active Router**

```
Router# debug crypto ipsec ha
```

Crypto IPSEC High Availability debugging is on  
*Sep 29 17:02:01.851:IPSec HA (crypto_ha_ipsec_notify_add_sa):called  
*Sep 29 17:02:01.851:IPSec HA (crypto_ha_ipsec_notify_add_sa):New IPsec SA added... notifying HA Mgr  
**Standby Router**

```
Router# debug crypto ipsec ha
```

Crypto IPSEC High Availability debugging is on  
*Sep 29 17:02:01.031:IPSec HA (crypto_ha_ipsec_mgr_recv_add_sas):HA mgr wants to insert the following bundle  
*Sep 29 17:02:01.031:IPSec HA (crypto_ha_ipsec_mgr_recv_add_sas):This SA Supports DPD  
*Sep 29 17:02:01.031:IPSec HA (crypto_ha_ipsec_gen_sa):Sending Kei to IPSec num_kei 2  
*Sep 29 17:02:01.039:IPSec HA (crypto_ha_ipsec_notify_add_sa):called  
*Sep 29 17:02:01.039:IPSec HA (crypto_ha_ipsec_notify_add_sa):operation not performed as standby ip 4.0.0.3

The following example is sample debug output with the `detail` keyword:

**Active Router**

```
*Sep 29 17:05:48.803:IPSec HA (crypto_ha_ipsec_mgr_set_state_common):called for vip 4.0.0.3  
*Sep 29 17:06:11.655:IPSec HA (crypto_ha_ipsec_mgr_recv_add_sas):Operation performed as standby for local addr 4.0.0.3  
*Sep 29 17:06:44.059:IPSec HA (crypto_ha_ipsec_notify_add_sa):called
```
The following example is sample debug output with the update keyword:

Active Router
*Sep 29 17:27:30.839:IPSec HA(check_and_send_replay_update):Replay triggered update seq_num 1000 last-sent 0 dir inbound
*Sep 29 17:27:30.839:IPSec HA(create_update_struct):Sending inbound update
*Sep 29 17:27:30.839:IPSec HA(send_update_struct):
  Outbound - New KB 0, New replay 0
  Inbound - New KB 3998772, New replay 1000
*Sep 29 17:29:30.883:IPSec HA(check_and_send_replay_update):Replay triggered update seq_num 2000 last-sent 1000 dir inbound
*Sep 29 17:29:30.883:IPSec HA(create_update_struct):Sending inbound update
*Sep 29 17:29:30.883:IPSec HA(send_update_struct):
  Outbound - New KB 0, New replay 0
  Inbound - New KB 3998624, New replay 2000
*Sep 29 17:30:30.899:IPSec HA(check_and_send_replay_update):Replay triggered update seq_num 3000 last-sent 2000 dir inbound
*Sep 29 17:30:30.899:IPSec HA(create_update_struct):Sending inbound update
*Sep 29 17:30:30.899:IPSec HA(send_update_struct):
  Outbound - New KB 0, New replay 0
  Inbound - New KB 3998476, New replay 3000
*Sep 29 17:32:30.943:IPSec HA(check_and_send_replay_update):Replay triggered update seq_num 4000 last-sent 3000 dir inbound
*Sep 29 17:32:30.943:IPSec HA(create_update_struct):Sending inbound update
*Sep 29 17:32:30.943:IPSec HA(send_update_struct):
  Outbound - New KB 0, New replay 0
  Inbound - New KB 3998327, New replay 4000

Standby Router
*Sep 29 17:27:28.887:IPSec HA(crypto_ha_ipsec_mgr_recv_update_sa):called
*Sep 29 17:27:28.887:IPSec HA(crypto_ha_ipsec_mgr_recv_update_sa):UPDATING INBOUND SA:ip = 4.0.0.3, protocol = 50, spi = B8A47EC9,
  NEW KB LIFE = 3998772,
  NEW REPLAY WINDOW START = 1000,
  NEW REPLAY WINDOW END = 3000,
*Sep 29 17:29:28.915:IPSec HA(crypto_ha_ipsec_mgr_recv_update_sa):called
*Sep 29 17:29:28.915:IPSec HA(crypto_ha_ipsec_mgr_recv_update_sa):UPDATING INBOUND SA:ip = 4.0.0.3, protocol = 50, spi = B8A47EC9,
  NEW KB LIFE = 3998624,
  NEW REPLAY WINDOW START = 2000,
  NEW REPLAY WINDOW END = 3000,
*Sep 29 17:30:28.939:IPSec HA(crypto_ha_ipsec_mgr_recv_update_sa):called
*Sep 29 17:30:28.939:IPSec HA(crypto_ha_ipsec_mgr_recv_update_sa):UPDATING INBOUND SA:ip = 4.0.0.3, protocol = 50, spi = B8A47EC9,
  NEW KB LIFE = 3998776,
  NEW REPLAY WINDOW START = 3000,
  NEW REPLAY WINDOW END = 4000,
*Sep 29 17:32:28.955:IPSec HA(crypto_ha_ipsec_mgr_recv_update_sa):called
*Sep 29 17:32:28.955:IPSec HA(crypto_ha_ipsec_mgr_recv_update_sa):UPDATING INBOUND SA:ip = 4.0.0.3, protocol = 50, spi = B8A47EC9,
  NEW KB LIFE = 3998327,
  NEW REPLAY WINDOW START = 4000,
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug crypto isakmp ha</td>
<td>Enables debugging messages for ISAKMP high availability.</td>
</tr>
</tbody>
</table>
debug crypto isakmp

To display messages about Internet Key Exchange (IKE) events, use the debug crypto isakmp command in privileged EXEC mode. To disable debugging output, use the no form of this command.

debug crypto isakmp [aaa| errors| kmi number]
no debug crypto isakmp [aaa| errors| kmi]

Syntax Description

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaa</td>
<td>(Optional) Specifies accounting events.</td>
</tr>
<tr>
<td>errors</td>
<td>(Optional) Enables error events.</td>
</tr>
<tr>
<td>kmi number</td>
<td>(Optional) Captures key management interface (KMI) messages.</td>
</tr>
</tbody>
</table>

Command Modes

Privileged EXEC (#)

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3 T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(15)T</td>
<td>The aaa keyword was added.</td>
</tr>
<tr>
<td>12.2(18)SXD</td>
<td>This command was integrated into Cisco IOS Release 12.2(18)SXD.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>Cisco IOS XE Release 2.6</td>
<td>This command was integrated into Cisco IOS XE Release 2.6.</td>
</tr>
<tr>
<td>15.3(2)T</td>
<td>This command was modified. The kmi number keyword-argument pair was added.</td>
</tr>
</tbody>
</table>

Usage Guidelines

IKE is a key management protocol standard that is used in conjunction with the IPsec to configure basic IPsec VPNs. IPsec can be configured without IKE, but IKE enhances IPsec by providing additional features, flexibility, and ease of configuration for the IPsec standard. IKE is a hybrid protocol that implements the Oakley key exchange and Skeme key exchange inside the Internet Security Association and Key Management Protocol (ISAKMP) framework.

The debug crypto isakmp kmi command must be enabled prior to any traffic that triggers sessions. This command enables capturing KMI sequence events along with timestamps for a particular IKE security association (SA) and corresponding IPsec SA. The number argument denotes the number of KMI messages to be captured. The KMI sequence is displayed in the output of the show crypto isakmp peers detail command.
The following is sample output from the `debug crypto isakmp` command for an IKE peer that initiates an IKE negotiation:

**IKE negotiates its own SA.**

```
Device# debug crypto isakmp
20:26:58: ISAKMP (8): beginning Main Mode exchange
20:26:58: ISAKMP (8): processing SA payload. message ID = 0
20:26:58: ISAKMP (8): Checking ISAKMP transform 1 against priority 10 policy
20:26:58: ISAKMP: encryption DES-CBC
20:26:58: ISAKMP: hash SHA
20:26:58: ISAKMP: default group 1
20:26:58: ISAKMP: auth pre-share
20:26:58: ISAKMP (8): atts are acceptable. Next payload is 0

When IKE finds a matching policy, each peer in the network uses the IKE SA to authenticate another peer.

```
20:26:58: ISAKMP (8): SA is doing pre-shared key authentication
20:26:58: ISAKMP (8): processing KE payload. message ID = 0
20:26:58: ISAKMP (8): processing NONCE payload. message ID = 0
20:26:58: ISAKMP (8): SKEYID state generated
20:26:58: ISAKMP (8): processing ID payload. message ID = 0
20:26:58: ISAKMP (8): processing HASH payload. message ID = 0
20:26:58: ISAKMP (8): SA has been authenticated

Next, IKE negotiates to set up the IPsec SA by searching for a matching transform set.

```
20:26:59: ISAKMP (8): beginning Quick Mode exchange, M-ID of 767162845
20:26:59: ISAKMP (8): processing SA payload. message ID = 767162845
20:26:59: ISAKMP (8): Checking IPSec proposal 1
20:26:59: ISAKMP: transform 1, ESP DES
20:26:59: ISAKMP: attributes in transform:
20:26:59: ISAKMP: encaps is 1
20:26:59: ISAKMP: SA life type in seconds
20:26:59: ISAKMP: SA life duration (basic) of 600
20:26:59: ISAKMP: SA life type in kilobytes
20:26:59: ISAKMP: SA life duration (VPI) of 0x0 0x46 0x50 0x0
20:26:59: ISAKMP: authenticator is HMAC-MD5
20:26:59: ISAKMP (8): atts are acceptable.

After finding a matching IPsec transform set for the two peers, an IPsec SA is created (one SA is created for each direction).

```
20:26:59: ISAKMP (8): processing NONCE payload. message ID = 767162845
20:26:59: ISAKMP (8): processing ID payload. message ID = 767162845
20:26:59: ISAKMP (8): processing ID payload. message ID = 767162845
20:26:59: ISAKMP (8): Creating IPsec SAs
20:26:59: inbound SA from 155.0.0.2 to 155.0.0.1 (proxy 155.0.0.2 to 155.0.0.1 )
20:26:59: has spi 454886490 and conn_id 9 and flags 4
20:26:59: lifetime of 600 seconds
20:26:59: lifetime of 4608000 kilobytes
20:26:59: outbound SA from 155.0.0.1 to 155.0.0.2 (proxy 155.0.0.1 to 155.0.0.2 )
20:26:59: has spi 75506225 and conn_id 10 and flags 4
20:26:59: lifetime of 600 seconds
20:26:59: lifetime of 4608000 kilobytes

The following is sample output from the `debug crypto isakmp` command using the `aaa` keyword:

```
Device# debug crypto isakmp aaa
01:38:55: ISAKMP AAA: Sent Accounting Message
The following example shows how to set the KMI sequence for a value of 10. The KMI sequence is displayed in the output of the `show crypto isakmp peers detail` command.

```
Device# debug crypto isakmp kmi 10
Crypto ISAKMP KMI debugging is on
Device# show crypto isakmp peers detail
Peer: 1.1.1.2 Port: 500 Local: 1.1.1.1
  Phase1 id: 1.1.1.2
  flags:
    NAS Port: 0 (Normal)
    IKE SAs: 1 IPSec SA bundles: 2
    Peer Handle: 0x80000005
    last_locker: 0xD198B32, last_last_locker: 0x0
    last_unlocker: 0x0, last_last_unlocker: 0x0

KMI messages (Time of capture - KMI message)
  *Mar 7 12:20:28.124  KEY_ENG_REQUEST_SAS
  *Mar 7 12:20:28.165  KEY_MGR_CREATE_IPSEC_SAS
  *Mar 7 12:20:28.165  KEY_ENG_NOTIFY_QOS_GROUP
  *Mar 7 12:20:28.165  KEY_ENG_NOTIFY_INCR_COUNT
  *Mar 7 12:20:58.124  KEY_ENG_REQUEST_SAS
  *Mar 7 12:20:58.126  KEY_MGR_CREATE_IPSEC_SAS
  *Mar 7 12:20:58.126  KEY_ENG_NOTIFY_QOS_GROUP
```

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>crypto isakmp profile</code></td>
<td>Defines an ISAKMP profile and audits IPsec user sessions.</td>
</tr>
<tr>
<td><code>crypto map (global IPsec)</code></td>
<td>Enters crypto map configuration mode and creates or modifies a crypto map entry, creates a crypto profile that provides a template for configuration of a dynamically created crypto map, or configures a client accounting list.</td>
</tr>
<tr>
<td><code>show crypto isakmp peers</code></td>
<td>Displays ISAKMP peer descriptions.</td>
</tr>
</tbody>
</table>
**debug crypto isakmp ha**

To enable debugging messages for Internet Security Association and Key Management Protocol (ISAKMP) high availability, use the `debug crypto isakmp ha` command in privileged EXEC mode. To disable debugging messages, use the `no` form of this command.

```
debug crypto isakmp ha [detail]
no debug crypto isakmp ha [detail]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>(Optional) Displays detailed debug information.</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>detail</code></td>
<td></td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(11)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output for a standby router from the `debug crypto isakmp ha` command:

Active Router
no debug message
Standby Router
Router# debug crypto isakmp ha

Crypto ISAKMP High Availability debugging is on
vrf-lite-R1#
*Sep 28 21:54:41.815:IKE HA:(4.0.0.3) Adding STANDBY IKE SA
*Sep 28 21:54:41.843:IKE HA:Create peer struct for local 4.0.0.3 remote 4.0.0.5 & locked
*Sep 28 21:54:41.843:IKE HA:IKE SA inserted on standby with src = 4.0.0.5, dst = 4.0.0.3

The following sample output is displayed when the `detail` keyword is issued. (Note that debug output without issuing the `detail` keyword is the same as the debug output with the `detail` keyword.)

Active Router
Router# debug crypto isakmp ha detail

Crypto ISAKMP High Availability detailed debugging is on
vrf-lite-R1#
*Sep 29 16:59:15.035:IKE HA:IKE SA is already failed over
Standby Router
Router# debug crypto isakmp ha detail

Crypto ISAKMP High Availability detailed debugging is on
vrf-lite-R2#
*Sep 29 16:59:14.371:IKE HA:(4.0.0.3) Adding STANDBY IKE SA
*Sep 29 16:59:14.411:IKE HA:Create peer struct for local 4.0.0.3 remote 4.0.0.5 & locked
*Sep 29 16:59:14.411:IKE HA:IKE SA inserted on standby with src = 4.0.0.5, dst = 4.0.0.3
## Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug crypto ha</td>
<td>Displays crypto high availability debugging information.</td>
</tr>
<tr>
<td>debug crypto ipsec ha</td>
<td>Enables debugging messages for IPSec high availability.</td>
</tr>
</tbody>
</table>
debug crypto key-exchange

To show Digital Signature Standard (DSS) public key exchange messages, use the debug crypto key-exchange command in privileged EXEC mode. To disable debugging output, use the no form of this command.

```
debug crypto key-exchange
no debug crypto key-exchange
```

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

Encryption and authentication are provided by a software service on the router called a crypto engine. The crypto engine performs authentication through DSS public and private keys when a connection is set up. DSS is a means of sending a "signature" at the end of a message that positively identifies the author of the message. The signature cannot be forged or duplicated by others, so whoever received a message with a DSS signature knows exactly who sent the message.

If the process of exchanging DSS public keys with a peer router by means of the config crypto key-exchange command is not successful, try to exchange DSS public keys again after enabling the debug crypto key-exchange command to help you diagnose the problem.

Examples

The following is sample output from the debug crypto key-exchange command. The first shows output from the initiating router in a key exchange. The second shows output from the passive router in a key exchange. The number of bytes received should match the number of bytes sent from the initiating side, although the number of messages can be different.

```
Router# debug crypto key-exchange
CRYPTO-KE: Sent 4 bytes.
CRYPTO-KE: Sent 2 bytes.
CRYPTO-KE: Sent 2 bytes.
CRYPTO-KE: Sent 64 bytes.
Router# debug crypto key-exchange
CRYPTO-KE: Received 4 bytes.
CRYPTO-KE: Received 2 bytes.
CRYPTO-KE: Received 2 bytes.
CRYPTO-KE: Received 49 bytes.
CRYPTO-KE: Received 15 bytes.
```

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug crypto sesmgmt</td>
<td>Displays connection setup messages and their flow through the router.</td>
</tr>
</tbody>
</table>
**debug crypto mib**

To display debug messages for the IP Security (IPsec) MIB subsystem, use the `debug crypto mib` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug crypto mib [detail] [error]
no debug crypto mib [detail] [error]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>detail</code></td>
<td>(Optional) Displays different events as they occur in the IPsec MIB subsystem.</td>
</tr>
<tr>
<td><code>error</code></td>
<td>(Optional) Displays error events in the MIB agent.</td>
</tr>
</tbody>
</table>

**Command Default**

Message notification debugging is not enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(4)E</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(4)T.</td>
</tr>
<tr>
<td>12.4(4)T</td>
<td>The <code>detail</code> and <code>error</code> keywords were added.</td>
</tr>
</tbody>
</table>

**Examples**

The following example shows IPsec MIB debug message notification being enabled:

```
Router# debug crypto mib
Crypto IPSec Mgmt Entity debugging is on
```

The following example shows that detailed information about events that are occurring in the subsystem has been requested:

```
Router# debug crypto mib detail
```

The following example shows that information has been requested about error events in the MIB agent:

```
Router# debug crypto mib error
```
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>show crypto mib ipsec flowmib history failure size</code></td>
<td>Displays the size of the IPSec failure history table.</td>
</tr>
<tr>
<td><code>show crypto mib ipsec flowmib history tunnel size</code></td>
<td>Displays the size of the IPSec tunnel history table.</td>
</tr>
<tr>
<td><code>show crypto mib ipsec flowmib version</code></td>
<td>Displays the IPSec Flow MIB version used by the router.</td>
</tr>
</tbody>
</table>
debug crypto pki messages

To display debugging messages for the details of the interaction (message dump) between the certification authority (CA) and the router, use the `debug crypto pki messages` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

**debug crypto pki messages**

**no debug crypto pki messages**

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Disabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The debug crypto pki messages command displays messages about the actual data being sent and received during public key infrastructure (PKI) transactions. This command is internal for use by Cisco support personnel.

You can use the show crypto ca certificates command to display information about your certificate.

**Examples**

The following is sample output from the `debug crypto pki messages` command:

```
Router# debug crypto pki messages
Fingerprint: 2CFC6265 77BA6496 3AEFCB50 29BC2BF2
00:48:23:Write out pkey#10 content:274
00:48:23:30 82 01 0E 30 81 B9 02 01 00 30 22 31 20 30 1E 06 09 2A 86
00:48:23:48 86 F7 0D 01 09 02 16 11 70 6B 69 2D 33 36 61 2E 63 69 73
00:48:23:63 6F 2E 63 6D 30 5C 30 0D 06 09 2A 86 48 86 F7 0D 01 01
00:48:23:01 05 00 03 4B 00 30 48 02 41 00 DD 2C C6 35 A5 3F 0F 97 6C
00:48:23:11 E2 81 95 01 6A 80 34 25 10 C4 5F 3D 8B 33 1C 19 50 FD 91
00:48:23:6C 2D 65 4C B6 A6 B0 02 1C B2 84 C1 C8 AC A4 28 6E EF 9D 3B
00:48:23:30 98 CB 36 A2 47 4E 7E 6F C9 3E B8 26 BE 15 02 03 01 00 01
00:48:23:0A 32 30 10 06 09 2A 86 48 86 F7 0D 01 09 07 31 03 13 01 63
00:48:23:30 1E 06 09 2A 86 48 86 F7 0D 01 09 07 31 11 14 0F 30 0D 30
00:48:23:0B 06 03 55 1D 0F 04 04 03 02 05 A0 30 0D 06 09 2A 86 48 86
00:48:23:FB 0D 01 03 04 05 00 03 41 00 2C FD 88 2C 8A 13 BE 81 88 EA
00:48:23:5C FD AE 52 8F 2C 13 95 9E 9D 8B A4 C9 48 32 84 BF 05 03 49
00:48:23:63 27 A3 AC 6D 74 EB 69 E3 06 E9 E4 9F 0A A8 FB 20 F0 02 03
00:48:23:BE 90 57 02 F2 75 8E 0F 16 60 10 6F BE 2B
00:48:23:Enveloped Data ...
```
debug crypto pki messages

00:48:23:30 80 06 09 2A 86 48 86 F7 0D 01 07 03 A0 80 30 80 02 01 00
00:48:23:31 80 30 82 01 0F 02 01 00 30 78 30 6A 31 0B 30 09 06 03 55
00:48:23:04 06 13 02 55 53 31 0B 30 09 06 03 55 04 08 13 02 43 41 31
00:48:23:13 30 11 06 03 55 04 07 13 0A 53 61 6E 74 74 70 73 73 73
00:48:23:31 15 30 13 06 03 55 04 0A 13 0C 43 69 63 6F 20 53 79 73
00:48:23:74 65 6D 31 0E 30 0C 06 03 55 04 0B 13 05 49 50 49 53 55 31
00:48:23:Signed Data 1382 bytes
00:48:23:30 80 06 09 2A 86 48 86 F7 0D 01 07 02 A0 80 30 80 02 01 01
00:48:23:31 0E 30 0C 06 08 2A 86 48 86 F7 0D 02 05 05 00 30 80 06 09
00:48:23:2A 86 48 86 F7 0D 01 07 01 A0 80 24 80 04 82 02 75 30 80 06
00:48:23:02 55 53 31 0B 30 09 06 03 55 04 08 13 02 43 41 31 13 30 11
00:48:23:33 34 5A 17 0D 31 30 31 31 31 31 35 31 30 35 34 33 34 5A 30 22
00:48:23:31 20 30 11 06 03 02 5A 17 0D 31 30 31 31 31 31 35 31 30 35 34 33 34 5A 30 22
00:48:23:2D 33 36 61 2E 63 69 73 63 6F 20 53 79 73 65 6D 31 0E 30 0C 06 03 55 04 0B 13 05 49 50 49 53 55 31
00:48:23:Received pki message:1778 types

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crypto ca enroll</td>
<td>Obtains the certificate of your router from the CA.</td>
</tr>
<tr>
<td>debug crypto pki messages</td>
<td>Displays debugging messages for the trace of interaction (message type) between the CA and the router.</td>
</tr>
<tr>
<td>show crypto ca certificates</td>
<td>Displays information about your certificate, the certificate of the CA, and any RA certificates.</td>
</tr>
</tbody>
</table>
**debug crypto pki server**

To enable debugging for a crypto public key infrastructure (PKI) certificate server, use the `debug crypto pki server` command in privileged EXEC mode. To disable debugging, use the `no` form of this command.

```
debug crypto pki server
no debug crypto pki server
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(4)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Examples**

The following example shows how to enable debugging for a certificate server. This example also contains sample debug messages, which allow users to troubleshoot the various certificate-request-related stages and tasks that are handled by the certificate server.

```
Router# debug crypto pki server
Crypto PKI Certificate Server debugging is on
Oct 15 19:50:41.000:CRYPTO_CS:old RA cert flag 0x4
Oct 15 19:50:41.000:CRYPTO_CS:new RA cert flag 0x1000C
Oct 15 19:50:41.000:CRYPTO_CS:nvram filesystem
Oct 15 19:50:41.279:CRYPTO_CS:serial number 0x1 written.
Oct 15 19:50:53.419:%SYS-5-CONFIG_I:Configured from console by console
Oct 15 19:50:53.731:CRYPTO_CS:received a SCEP GetCACert request
Oct 15 19:50:53.739:CRYPTO_CS:CA certificate sent
Oct 15 19:50:54.355:CRYPTO_CS:received a SCEP GetCACert request
Oct 15 19:50:57.791:CRYPTO_CS:received a SCEP request
Oct 15 19:50:58.689:CRYPTO_CS:received an enrollment request
Oct 15 19:50:58.691:CRYPTO_CS:request has been authorized, transaction id=3673CE2F0235A4AE6F26242B004BB4
Oct 15 19:50:58.699:CRYPTO_CS:byte 2 in key usage in PKCS#10 is 0x7
Oct 15 19:50:58.703:CRYPTO_CS:enrollment request with pendingID 1 sent to the CA
```
SCEP_WRTE_DB_8

....
Oct 15 19:53:08.403:CRYPTO_CS:enrollment request 2 granted
Oct 15 19:53:08.403:%CRYPTO-6-CERTRET:Certificate received from Certificate Authority
Oct 15 19:53:35.707:CRYPTO_CS:received a SCEP request
Oct 15 19:53:35.859:CRYPTO_CS:trans id - 4D774FFE2F7CA9991A7F6A7803E77
Oct 15 19:53:36.595:CRYPTO_CS:received an enrollment request

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crypto pki server</td>
<td>Enables a Cisco IOS certificate server and enters certificate server configuration mode.</td>
</tr>
</tbody>
</table>
**debug crypto pki transactions**

To display debugging messages for the trace of interaction (message type) between the certification authority (CA) and the router, use the `debug crypto pki transactions` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug crypto pki transactions
no debug crypto pki transactions
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Disabled

**Command Modes**

Privileged EXEC

**Command History**

**Release** | **Modification**
---|---
12.0 | This command was introduced.
12.2(33)SRA | This command was integrated into Cisco IOS Release 12.2(33)SRA.

**Usage Guidelines**

Use the `debug crypto pki transactions` command to display debugging messages pertaining to public key infrastructure (PKI) certificates. The messages will show status information during certificate enrollment and verification.

You can also use the `show crypto ca certificates` command to display information about your certificate.

**Examples**

The following example, which authenticates and enrolls a CA, contains sample output for the `debug crypto pki transactions` command:

```
Router(config)# crypto ca authenticate msca
Certificate has the following attributes:
Fingerprint:A5DE3C51 AD8B0207 B60BED6D 9356FB00
% Do you accept this certificate? [yes/no]:y
Router# debug crypto pki transactions
00:44:00:CRYPTO_PKI:Sending CA Certificate Request:
GET /certsrv/mscep/mscep.dll/pkiclient.exe?operation=GetCACert&message=msca HTTP/1.0
00:44:00:CRYPTO_PKI:http connection opened
00:44:01:CRYPTO_PKI:HTTP response header:
HTTP/1.1 200 OK
Server:Microsoft-IIS/5.0
Date:Fri, 17 Nov 2000 18:50:59 GMT
Content-Length:2693
Content-Type:application/x-x509-ca-ra-cert

Content-Type indicates we have received CA and RA certificates.

00:44:01:CRYPTO_PKI:WARNING:A certificate chain could not be constructed while selecting
```
Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crypto ca authenticate</td>
<td>Authenticates the CA (by getting the certificate of the CA).</td>
</tr>
<tr>
<td>crypto ca enroll</td>
<td>Obtains the certificate of your router from the CA.</td>
</tr>
<tr>
<td>debug crypto pki messages</td>
<td>Displays debugging messages for details of the interaction (message dump) between the CA and the router.</td>
</tr>
<tr>
<td>show crypto ca certificates</td>
<td>Displays information about your certificate, the certificate of the CA, and any RA certificates.</td>
</tr>
</tbody>
</table>
debug crypto provisioning

To display information about an easy secure device provisioning (SDP) operation, use the debug crypto provisioning command in privileged EXEC mode. To disable debugging output, use the no form of this command.

```
debug crypto provisioning
no debug crypto provisioning
```

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(8)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(14)T</td>
<td>This command replaced the debugcryptowui command.</td>
</tr>
</tbody>
</table>

Usage Guidelines

For more detailed information about authentication, authorization, and accounting (AAA), use the debugaaaauthorization command.

Examples

The following is sample output from the debug crypto provisioning command. The output includes explanations of the process.

```
Router# debug crypto provisioning
Petitioner device
! The user starts the Welcome phase.
Nov 7 03:15:48.171: CRYPTO_WUI_TTI: received welcome get request.
! The router generates a Rivest, Shamir, and Adelman (RSA) keypair for future enrollment.
Nov 7 03:15:48.279: CRYPTO_WUI_TTI: keyhash 'A506BE3B83C6F4B4A6EFCB3D5B4AACA'
! The TTI transaction is completed.
Nov 7 03:16:10.607: CRYPTO_WUI_TTI: received completion post request.
Registrar device
! During the introduction phase, the browser prompts for login information.
06:39:18: CRYPTO_WUI_TTI: received introduction post request.
06:39:18: CRYPTO_WUI_TTI: checking AAA authentication (ipsecca_script_aalalist, ttiuser)
! This happens if the user types in the wrong username or password.
06:39:19: CRYPTO_WUI_TTI: authentication declined by AAA, or AAA server not found - 0x3
06:39:19: CRYPTO_WUI_TTI: aaa query fails!
! The user re-enters login information.
06:39:19: CRYPTO_WUI_TTI: received introduction post request.
06:39:19: CRYPTO_WUI_TTI: checking AAA authentication (ipsecca_script_aalalist, ttiuser)
06:39:20: CRYPTO_WUI_TTI: checking AAA authorization (ipsecca_script_aalalist, ttiuser)
! The login attempt succeeds and authorization information is retrieved from the AAA database.
06:39:21: CRYPTO_WUI_TTI: aaa query ok!
```
These attributes are inserted into the configuration template.

```
06:39:21: CRYPTO_WUI_TTI: building TTI av pairs from AAA attributes
06:39:21: CRYPTO_WUI_TTI: "subjectname" = "CN=\user, O=company, C=\country"
06:39:21: CRYPTO_WUI_TTI: "$1" = "ntp server 192.0.2.1"
06:39:21: CRYPTO_WUI_TTI: "$2" = "hostname user-vpn"
```

The registrar stores this subject name and overrides the subject name in the subsequent enrollment request.

```
06:39:21: CRYPTO_WUI_TTI: subjectname=CN=\user, O=company, C=\country
```

The registrar stores this key information so that it may be used to automatically grant the subsequent enrollment request.

```
06:39:21: CRYPTO_WUI_TTI: key_hash=A506BE3B83C6F4B4A6EFCEB3D584AACA
```

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>authentication list (tti-registrar)</strong></td>
<td>Authenticates the introducer in an SDP operation.</td>
</tr>
<tr>
<td><strong>authorization list (tti-registrar)</strong></td>
<td>Specifies the appropriate authorized fields for the certificate subject name and list of template variables to be expanded into the Cisco IOS CLI snippet that is sent back to the petitioner in an SDP operation.</td>
</tr>
<tr>
<td><strong>debug aaa authorization</strong></td>
<td>Displays information on AAA TACACS+ authorization.</td>
</tr>
<tr>
<td><strong>template config</strong></td>
<td>Specifies a remote URL for a Cisco IOS CLI configuration template.</td>
</tr>
<tr>
<td><strong>template username</strong></td>
<td>Establishes a template username and password to access the configuration template on the file system.</td>
</tr>
</tbody>
</table>
debug crypto sesmgmt

To show connection setup messages and their flow through the router, use the `debug cryptosesmgmt` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```plaintext
debug crypto sesmgmt
no debug crypto sesmgmt
```

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

**Command Modes**
Privileged EXEC

**Usage Guidelines**
Encryption and authentication are provided by a software service on the router called a `cryptoengine`. The `cryptoengine` performs authentication through Digital Signature Standard (DSS) public and private keys when a connection is set up. DSS is a means of sending a "signature" at the end of a message that positively identifies the author of the message. The signature cannot be forged or duplicated by others, so whoever receives a message with a DSS signature knows exactly who sent the message.

When connections are not completing, use the `debug cryptosesmgmt` command to follow the progress of connection messages as a first step in diagnosing the problem. You see a record of each connection message as the router discovers it, and can track its progress through the necessary signing, verifying, and encryption session setup operations. Other significant connection setup events, such as the pre-generation of Diffie-Hellman public numbers, are also shown. For information on Diffie-Hellman public numbers, refer to the *Cisco IOS Security Configuration Guide*.

Also use the `show cryptoconnections` command to display additional information on connections.

**Examples**
The following is sample output from the `debug cryptosesmgmt` command. The first shows messages from a router that initiates a successful connection. The second shows messages from a router that receives a connection.

```plaintext
Router# debug crypto sesmgmt
CRYPTO: Dequeued a message: Initiate_Connection
CRYPTO: DH gen phase 1 status for conn_id 2 slot 0:OK
CRYPTO: Signing done. Status:OK
CRYPTO: ICMP message sent: s=172.21.114.163, d=172.21.114.162
CRYPTO-SDU: send_nnc_req: NNC Echo Request sent
CRYPTO: Dequeued a message: CRM
CRYPTO: DH gen phase 2 status for conn_id 2 slot 0:OK
CRYPTO: Verify done. Status:OK
CRYPTO: Signing done. Status:OK
CRYPTO: ICMP message sent: s=172.21.114.163, d=172.21.114.162
CRYPTO-SDU: act_on_nnc_req: NNC Echo Confirm sent
CRYPTO: Create encryption key for conn_id 2 slot 0:OK
CRYPTO: Replacing -2 in crypto maps with 2 (slot 0)
Router# debug crypto sesmgmt
CRYPTO: Dequeued a message: CIM
CRYPTO: Verify done. Status:OK
CRYPTO: DH gen phase 1 status for conn_id 1 slot 0:OK
CRYPTO: DH gen phase 2 status for conn_id 1 slot 0:OK
CRYPTO: Signing done. Status:OK
CRYPTO: ICMP message sent: s=172.21.114.162, d=172.21.114.163
CRYPTO-SDU: act_on_nnc_req: NNC Echo Reply sent
```
CRYPTO: Create encryption key for conn_id 1 slot 0:OK
CRYPTO: Replacing -2 in crypto maps with 1 (slot 0)
CRYPTO: Dequeued a message: CCM
CRYPTO: Verify done. Status=OK

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug crypto key-exchange</td>
<td>Displays DSS public key exchange messages.</td>
</tr>
</tbody>
</table>
debug csm neat

To turn on debugging for all Call Switching Module (CSM) Voice over IP (VoIP) calls, use the `debug csm neat` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug csm neat [slot| dspm| dsp| dsp-channel]
no debug csm neat [slot| dspm| dsp| dsp-channel]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slot</td>
<td>dspm</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(9)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The `debug csm neat` command turns on debugging for all CSM VoIP calls. If no arguments are specified, debugging is enabled for all voice calls.

**Note**

The `debug csm neat` command does not display any information if you try to debug ISDN voice calls. To view debugging information about ISDN calls, use the `debug cdapi` command.

The `nodebug csm neat` command turns off debugging information for all voice calls.

If the `slot, dspm, dsp, or dsp-channel` arguments are specified (if the specified DSP channel is engaged in a CSM call), CSM call-related debugging information is turned on for this channel. The `no` form of this command turns off debugging for that particular channel.

**Examples**

The following examples show sample output from the `debug csm neat` command. The following shows that CSM has received an event from ISDN.

```
Router# debug csm neat
March 18 04:05:07.052: EVENT_FROM_ISDN::dchan_idb=0x60D7B6B8, call_id=0xCF, ces=0x1 bchan=0x0, event=0x1, cause=0x0
```

The table below describes the significant fields shown in the display.

```sql
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dchan_idb</td>
<td>Dual channel identifier bus (DCHAN) ID</td>
</tr>
<tr>
<td>call_id</td>
<td>Call identifier (CALL-ID)</td>
</tr>
<tr>
<td>ces</td>
<td>Call setup event (CES)</td>
</tr>
<tr>
<td>bchan</td>
<td>Bus channel (BCHAN)</td>
</tr>
<tr>
<td>event</td>
<td>Event (EVENT)</td>
</tr>
<tr>
<td>cause</td>
<td>Cause (CAUSE)</td>
</tr>
</tbody>
</table>
```
Table 65: debug csm neat Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dchan_idb</td>
<td>Indicates the address of the hardware interface description block (IDB) for the D channel.</td>
</tr>
<tr>
<td>call_id</td>
<td>Indicates the call ID assigned by ISDN.</td>
</tr>
<tr>
<td>call-id</td>
<td>Indicates the call ID assigned by ISDN</td>
</tr>
<tr>
<td>bchan</td>
<td>Indicates the number of the B channel assigned for this call</td>
</tr>
<tr>
<td>cause</td>
<td>Indicates the ISDN event cause</td>
</tr>
</tbody>
</table>

The following example shows that CSM has allocated the CSM voice control block for the DSP device on slot 1 port 10 for this call.

March 18 04:05:07.052: VDEV_ALLOCATE: slot 1 and port 10 is allocated.

In this example, CSM must first allocate the CSM voice control block to initiate the state machine. After the voice control block has been allocated, CSM obtains from the DSP Resource Manager the actual DSP channel that will be used for the call. At that point, CSM will switch to the actual logical port number. The slot number in this example refers to the physical slot on the Cisco AS5400 access server. The port number is the logical DSP number interpreted as listed in the table below.

Table 66: Logical DSP Numbers

<table>
<thead>
<tr>
<th>Logical Port</th>
<th>Physical DSP Channel</th>
<th>DSP Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DSPRM 1</td>
<td>DSP 1</td>
</tr>
<tr>
<td>1</td>
<td>DSPRM 1</td>
<td>DSP 1</td>
</tr>
<tr>
<td>2</td>
<td>DSPRM 1</td>
<td>DSP 2</td>
</tr>
<tr>
<td>3</td>
<td>DSPRM 1</td>
<td>DSP 2</td>
</tr>
<tr>
<td>4</td>
<td>DSPRM 1</td>
<td>DSP 3</td>
</tr>
<tr>
<td>5</td>
<td>DSPRM 1</td>
<td>DSP 3</td>
</tr>
<tr>
<td>6</td>
<td>DSPRM 1</td>
<td>DSP 4</td>
</tr>
<tr>
<td>7</td>
<td>DSPRM 1</td>
<td>DSP 4</td>
</tr>
<tr>
<td>8</td>
<td>DSPRM 1</td>
<td>DSP 5</td>
</tr>
<tr>
<td>9</td>
<td>DSPRM 1</td>
<td>DSP 5</td>
</tr>
<tr>
<td>Logical Port</td>
<td>Physical DSP Channel</td>
<td>DSP Channel</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>10</td>
<td>DSPRM 1</td>
<td>DSP channel 1</td>
</tr>
<tr>
<td>11</td>
<td>DSPRM 1</td>
<td>DSP channel 2</td>
</tr>
<tr>
<td>12</td>
<td>DSPRM 2</td>
<td>DSP channel 1</td>
</tr>
<tr>
<td>13</td>
<td>DSPRM 2</td>
<td>DSP channel 2</td>
</tr>
<tr>
<td>14</td>
<td>DSPRM 2</td>
<td>DSP channel 1</td>
</tr>
<tr>
<td>15</td>
<td>DSPRM 2</td>
<td>DSP channel 2</td>
</tr>
<tr>
<td>16</td>
<td>DSPRM 2</td>
<td>DSP channel 1</td>
</tr>
<tr>
<td>17</td>
<td>DSPRM 2</td>
<td>DSP channel 2</td>
</tr>
<tr>
<td>18</td>
<td>DSPRM 2</td>
<td>DSP channel 1</td>
</tr>
<tr>
<td>19</td>
<td>DSPRM 2</td>
<td>DSP channel 2</td>
</tr>
<tr>
<td>20</td>
<td>DSPRM 2</td>
<td>DSP channel 1</td>
</tr>
<tr>
<td>21</td>
<td>DSPRM 2</td>
<td>DSP channel 2</td>
</tr>
<tr>
<td>22</td>
<td>DSPRM 2</td>
<td>DSP channel 1</td>
</tr>
<tr>
<td>23</td>
<td>DSPRM 2</td>
<td>DSP channel 2</td>
</tr>
<tr>
<td>48</td>
<td>DSPRM 5</td>
<td>DSP channel 1</td>
</tr>
<tr>
<td>49</td>
<td>DSPRM 5</td>
<td>DSP channel 2</td>
</tr>
<tr>
<td>50</td>
<td>DSPRM 5</td>
<td>DSP channel 1</td>
</tr>
<tr>
<td>51</td>
<td>DSPRM 5</td>
<td>DSP channel 2</td>
</tr>
<tr>
<td>52</td>
<td>DSPRM 5</td>
<td>DSP channel 1</td>
</tr>
<tr>
<td>53</td>
<td>DSPRM 5</td>
<td>DSP channel 2</td>
</tr>
<tr>
<td>54</td>
<td>DSPRM 5</td>
<td>DSP channel 1</td>
</tr>
<tr>
<td>55</td>
<td>DSPRM 5</td>
<td>DSP channel 2</td>
</tr>
<tr>
<td>56</td>
<td>DSPRM 5</td>
<td>DSP channel 1</td>
</tr>
<tr>
<td>57</td>
<td>DSPRM 5</td>
<td>DSP channel 2</td>
</tr>
</tbody>
</table>
The following example shows that the function `csm_vtsp_init_tdm()` has been called with a voice control block of address 0x60B8562C. This function will be called only when the call is treated as a voice call.

March 18 04:05:07.052: csm_vtsp_init_tdm (voice_vdev=0x60B8562C)

The following example shows that CSM has obtained a DSP channel from the DSP Resource Manager:

March 18 04:05:07.052: csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm slot 1, dspm 2, dsp 5, dsp_channel 1

March 18 04:05:07.052: csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm stream 5, channel 9, bank 0, bp_channel 10

The table below describes the significant fields shown in the DSP channel initialized TDM display.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDM slot 1, dspm 2, dsp 5, dsp_channel 1</td>
<td>Indicates the physical DSP channel that will be used for this call.</td>
</tr>
<tr>
<td>TDM stream 5, channel 9, bank 0, bp_channel_10</td>
<td>Indicates the on-chip and backplane TDM channel assigned to this DSP channel. Stream 5, channel 9 gives the on-chip TDM channel mapped to the DSP; bank 0, bp_channel_10 means that the backplane stream 0 and backplane channel #1 are assigned to this DSP.</td>
</tr>
</tbody>
</table>

The following shows that CSM received an incoming call event from ISDN:

March 18 04:05:07.052: EVENT_FROM_ISDN: (00CF): DEV_INCALL at slot 1 and port 20

Slot 1, port 20 means the logical DSP channel 20 (mapped to DSPRM 2, DSP 5, DSP channel 1).

The following shows that the DEV_INCALL message been translated into a CSM_EVENT_ISDN_CALL message:

March 18 04:05:07.052: CSM_PROC_IDLE: CSM_EVENT_ISDN_CALL at slot 1, port 20

This message is passed to the CSM central state machine while it is in the CSM_IDLE state and is in the CSM_PROC_IDLE procedure. The logical DSP channel port 20 on slot 1 is used to handle this call.

The following shows that CSM has invoked the vtsp_ic_notify() function with a CSM voice call control block 0x60B8562C.

March 18 04:05:07.052: vtsp_ic_notify : (voice_vdev= 0x60B8562C)

Inside this function, CSM will send a SETUP INDICATION message to the VTSP. This function will be invoked only if the call is a voice call.
The following shows that CSM received a SETUP INDICATION RESPONSE message from the VTSP as an acknowledgment.

March 18 04:05:07.056: csm_vtsp_call_setup_resp (vdev_info=0x60B8562C, vtsp_cdb=0x60FCA114)

This means that the VTSP received the CALL SETUP INDICATION message previously sent and has proceeded to process the call.

- **vdev_info**--Contains the address of the CSM voice data block.
- **vtsp_cdb**--Contains the address of the VTSP call control block.

The following shows that CSM received a CALL CONNECT message from the VTSP:

March 18 04:05:07.596: csm_vtsp_call_connect (vtsp_cdb=0x60FCA114, voice_vdev=0x60B8562C)

This indicates that the VTSP received a CONNECT message for the call leg initiated to the Internet side.

- **vtsp_cdb**--Contains the address of the VTSP call control block.
- **voice_vdev**--Contains the address of the CSM voice data block.

The following shows that while CSM is in the CSM_IC2_RING state, it receives a SETUP INDICATION RESPONSE from the VTSP. This message is translated into CSM_EVENT_MODEM_OFFHOOK and passed to the CSM central state machine.

March 18 04:05:07.596: CSM_PROC_IC2_RING: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 20

The following shows that CSM received a CONNECT message from ISDN for the call using the logical DSP channel on slot 1 and port 20:

March 18 04:05:07.616: EVENT_FROM_ISDN:(00CF): DEV_CONNECTED at slot 1 and port 20

The following shows that CSM translated the CONNECT event from ISDN into the CSM_EVENT_ISDN_CONNECTED message, which is then passed to the CSM central state machine:

March 18 04:05:07.616: CSM_PROC_IC4_WAIT_FOR_CARRIER: CSM_EVENT_ISDN_CONNECTED at slot 1, port 20

The following shows that CSM received a CALL SETUP REQUEST from the VTSP:

May 16 12:22:27.580: csm_vtsp_call_setup_request (vtsp_cdb=0x60FCFA20, vtsp_sdb=0x60DFB608)

This represents a request to make an outgoing call to the PSTN.

- **vtsp_cdb**--Contains the address of the VTSP call control block.
- **vtsp_sdb**--Contains the address of the signalling data block for the signalling interface to be used to send the outgoing call.

The following shows that the physical DSP channel has been allocated for this outgoing call:

May 16 12:22:27.580: csm_vtsp_call_setup_request: tdm slot 1, dspm 5, dsp 4, dsp_channel 1

The following shows the on-chip and backplane TDM channel assigned to this DSP channel:

May 16 12:22:27.580: csm_vtsp_call_setup_request: tdm stream 5, channel 25, bank 0, bp_channel 27

In this sample output, tdm stream 5, channel 25, bank 0, bp_channel 27 indicates the on-chip and backplane TDM channel assigned to this DSP channel. Stream 5, channel 25 gives the on-chip TDM channel mapped...
to the DSP; bank 0, bp_channel 27 means that the backplane stream 0 and backplane channel 1 are assigned to this DSP.

The following shows the calling number and the called number for this call:

May 16 12:22:27.580: csm_vtsp_call_setup_request: calling number: 10001, called number: 30001

The following shows that the CALL SETUP REQUEST from the VTSP has been translated into the 'CSM_EVENT_MODEM_OFFHOOK message and is passed to the CSM central state machine:

May 16 12:22:27.580: CSM_PROC_IDLE: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 54

The logical DSP channel number for the DSP (slot 1, port 54) is now displayed, which maps to the physical DSP channel slot 1, dspm 5, dsp 4, dsp_channel 1.

The following shows that CSM collected all the digits for dialing out:

May 16 12:22:27.580: CSM_PROC_OC3_COLLECT_ALL_DIGIT: CSM_EVENT_GET_ALL_DIGITS at slot 1, port 54

For PRI and for applications that do not require digit collection of outwardialing digits (for example, voice calls), the intermediate digit collection states are omitted and the CSM state machine moves to this state directly, pretending that the digit collection has been done.

The following shows an information message:

March 16 12:22:27.580: CSM_PROC_OC3_COLLECT_ALL_DIGIT: called party num: (30001) at slot 1, port 54

The following shows that CSM attempts to find a free signalling D channel to direct the outgoing call:

March 16 12:22:27.580: csm_vtsp_check_dchan (voice_vdev=0x60B8562C)
March 16 12:22:27.580: csm_vtsp_check_dchan (vtsp requested dchan=0x60D7ACB0, dchan_idb=0x60E8ACF0)
March 16 12:22:27.580: csm_vtsp_check_dchan (voice_vdev=0x60B8562C)
March 16 12:22:27.580: csm_vtsp_check_dchan (vtsp requested dchan=0x60D7ACB0, dchan_idb=0x60D7ACB0)

In the case of voice calls, the free signaling D channel must match the voice interface specified inside the signalling data block (vtsp_sdb) passed from the VTSP.

The following shows that CSM has received an event from ISDN:

March 16 12:22:27.624: EVENT_FROM_ISDN::dchan_idb=0x60D7ACB0, call_id=0xA121, ces=0x1, bchan=0x1E, event=0x3, cause=0x0

In this sample output:

- dchan_idb--Indicates the address of the hardware IDB for the D channel
- call_id--Indicates the call id assigned by ISDN
- bchan--Indicates the number of the B channel assigned for this call
- cause--Indicates the ISDN event cause

The following shows that CSM has received a CALL PROCEEDING message from ISDN.

March 16 12:22:27.624: EVENT_FROM_ISDN::(A121): DEV_CALL_PROC at slot 1 and port 54

The following shows that the CALL PROCEEDING event received from ISDN has been interpreted as a CSM_EVENT_ISDN_BCHAN_ASSIGNED message:

March 16 12:22:27.624: CSM_PROC_OC4_DIALING: CSM_EVENT_ISDN_BCHAN_ASSIGNED at slot 1, port 54
ISDN has assigned a B channel for this outgoing call. This B channel must be on the same PRI span as the signalling D channel allocated previously.

The following shows that the `csm_vtsp_setup_for_oc` function is called:

March 16 12:22:27.624: csm_vtsp_setup_for_oc (voice_vdev=0x60B8562C)
This is invoked when an outgoing call initiated by the VTSP receives a response from the ISDN stack.

The following shows that ISDN has sent a CONNECT message to CSM indicating that the call leg to the PSTN side has been established:

March 16 12:22:28.084: EVENT_FROM_ISDN::dchan_idb=0x60D7ACB0, call_id=0xA121, ces=0x1
bchan=0x1E, event=0x4, cause=0x0
March 16 12:22:28.084: EVENT_FROM_ISDN:(A121): DEV_CONNECTED at slot 1 and port 54

The following shows that while CSM is in the OC5_WAIT_FOR_CARRIER state, it has received the 'CONNECT' message from ISDN and has translated it into the CSM_EVENT_ISDN_CONNECTED message, which is passed to the CSM central state machine:

March 16 12:22:28.084: CSM_PROC_OC5_WAIT_FOR_CARRIER: CSM_EVENT_ISDN_CONNECTED at slot 1, port 54

The following shows that the function `vtsp_confirm_oc()` has been called:

March 16 12:22:28.084: vtsp_confirm_oc : (voice_vdev= 0x60B8562C)
This is invoked after CSM received the CONNECT message from ISDN. CSM sends a confirmation of the CONNECT to the VTSP.
debug csm tgrm

To view Call Switching Module (CSM) trunk group resource manager information, use the debug csm tgrm command in privileged EXEC mode. To disable debugging output, use the no form of this command.

ddebug csm tgrm
no debug csm tgrm

Syntax Description
This command has no arguments or keywords.

Command Default
Disabled

Command Modes
Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(11)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

Usage Guidelines
Disable console logging and use buffered logging before using the debug csm tgrm command. Using the debug csm tgrm command generates a large volume of debugs, which can affect router performance.

Examples
The following is sample output from the debug csm tgrm command. The output shows that the call type is voice, the direction is incoming, and the call is accepted by the CSM.

Router# debug csm tgrm

Router# 00:02:25:CSM-TGRM:csmtgrm_rx_cas_event_from_neat(EVENT_DIAL_IN) - c(T1 7/1:1:3) call_type=VOICE, dir=INCOMING
Router# 00:02:30:CSM-TGRM:csmtgrm Proc_ic3_wait_for_res_resp() c(T1 7/1:1:3) VOICE <ACCEPTED !!>

The table below describes the significant fields shown in the display.

Table 68: debug csm tgrm Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>call_type</td>
<td>Type of call: VOICE or MODEM.</td>
</tr>
<tr>
<td>dir</td>
<td>Direction of the call: INCOMING or OUTGOING.</td>
</tr>
</tbody>
</table>
debug clns esis events through debug dbconn tcp

debug csm tgrm
debug csm voice

To turn on debugging for all Call Switching Module (CSM) Voice over IP (VoIP) calls, use the debugcsmvoice command in privileged EXEC mode. To disable debugging output, use the no form of this command.

```
debug csm voice [slot| dspm| dsp| dsp-channel]
no debug csm voice [slot| dspm| dsp| dsp-channel]
```

**Syntax Description**

```
slot | dspm | dsp | dsp-channel
(Required) Identifies the location of a particular digital signal processor (DSP) channel.
```

**Command Modes**

Privileged EXEC

**Usage Guidelines**

The debugcsmvoice command turns on debugging for all CSM VoIP calls. If this command has no keyword specified, then debugging is enabled for all voice calls.

**Note**

The debugcsmvoice command does not display any information if you try to debug isdn voice calls. To view debugging information about isdn calls, use the debugcdapi command.

The nodebugcsmvoice command turns off debugging information for all voice calls.

If the slot|dspm|dsp|dsp-channel argument is specified, then (if the specified DSP channel is engaged in a CSM call) CSM call-related debugging information will be turned on for this channel. The no form of this command turns off debugging for that particular channel.

**Examples**

The following examples show sample output from the debugcsmvoice command. The following shows that CSM has received an event from ISDN.

```
Router# debug csm voice
Oct 18 04:09:07.052: EVENT_FROM_ISDN::dchan_idb=0x60D7B6B8, call_id=0xCF, ces=0x1
bchan=0x0, event=0x1, cause=0x0
```

In this example, terms are explained as follows:

- dchan_idb--Indicates the address of the hardware interface description block (IDB) for the D channel
- call_id--Indicates the call ID assigned by ISDN
- bchan--Indicates the number of the B channel assigned for this call
- cause--Indicates the ISDN event cause
The following shows that CSM has allocated the CSM voice control block for the DSP device on slot 1 port 10 for this call.

Oct 18 04:05:07.052: VDEV_ALLOCATE: slot 1 and port 10 is allocated. This AS5300 access server might not be actually used to handle this call. CSM must first allocate the CSM voice control block to initiate the state machine. After the voice control block has been allocated, CSM obtains from the DSP Resource Manager the actual DSP channel that will be used for the call. At that point, CSM will switch to the actual logical port number. The slot number refers to the physical slot on the AS5300 access server. The port number is the logical DSP number interpreted as listed in the table below.

**Table 69: Logical DSP Numbers**

<table>
<thead>
<tr>
<th>Logical Port Number</th>
<th>Physical DSP Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 0</td>
<td>DS CRM 1, DSP 1, DSP channel 1</td>
</tr>
<tr>
<td>Port 1</td>
<td>DS CRM 1, DSP 1, DSP channel 2</td>
</tr>
<tr>
<td>Port 2</td>
<td>DS CRM 1, DSP 2, DSP channel 1</td>
</tr>
<tr>
<td>Port 3</td>
<td>DS CRM 1, DSP 2, DSP channel 2</td>
</tr>
<tr>
<td>Port 4</td>
<td>DS CRM 1, DSP 3, DSP channel 1</td>
</tr>
<tr>
<td>Port 5</td>
<td>DS CRM 1, DSP 3, DSP channel 2</td>
</tr>
<tr>
<td>Port 6</td>
<td>DS CRM 1, DSP 4, DSP channel 1</td>
</tr>
<tr>
<td>Port 7</td>
<td>DS CRM 1, DSP 4, DSP channel 2</td>
</tr>
<tr>
<td>Port 8</td>
<td>DS CRM 1, DSP 5, DSP channel 1</td>
</tr>
<tr>
<td>Port 9</td>
<td>DS CRM 1, DSP 5, DSP channel 2</td>
</tr>
<tr>
<td>Port 10</td>
<td>DS CRM 1, DSP 6, DSP channel 1</td>
</tr>
<tr>
<td>Port 11</td>
<td>DS CRM 1, DSP 6, DSP channel 2</td>
</tr>
<tr>
<td>Port 12</td>
<td>DS CRM 2, DSP 1, DSP channel 1</td>
</tr>
<tr>
<td>Port 13</td>
<td>DS CRM 2, DSP 1, DSP channel 2</td>
</tr>
<tr>
<td>Port 14</td>
<td>DS CRM 2, DSP 2, DSP channel 1</td>
</tr>
<tr>
<td>Port 15</td>
<td>DS CRM 2, DSP 2, DSP channel 2</td>
</tr>
<tr>
<td>Port 16</td>
<td>DS CRM 2, DSP 3, DSP channel 1</td>
</tr>
<tr>
<td>Port 17</td>
<td>DS CRM 2, DSP 3, DSP channel 2</td>
</tr>
<tr>
<td>Port 18</td>
<td>DS CRM 2, DSP 4, DSP channel 1</td>
</tr>
<tr>
<td>Logical Port Number</td>
<td>Physical DSP Channel</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Port 19</td>
<td>DSPRM 2, DSP 4, DSP channel 2</td>
</tr>
<tr>
<td>Port 20</td>
<td>DSPRM 2, DSP 5, DSP channel 1</td>
</tr>
<tr>
<td>Port 21</td>
<td>DSPRM 2, DSP 5, DSP channel 2</td>
</tr>
<tr>
<td>Port 22</td>
<td>DSPRM 2, DSP 6, DSP channel 1</td>
</tr>
<tr>
<td>Port 23</td>
<td>DSPRM 2, DSP 6, DSP channel 2</td>
</tr>
<tr>
<td>Port 48</td>
<td>DSPRM 5, DSP 1, DSP channel 1</td>
</tr>
<tr>
<td>Port 49</td>
<td>DSPRM 5, DSP 1, DSP channel 2</td>
</tr>
<tr>
<td>Port 50</td>
<td>DSPRM 5, DSP 2, DSP channel 1</td>
</tr>
<tr>
<td>Port 51</td>
<td>DSPRM 5, DSP 2, DSP channel 2</td>
</tr>
<tr>
<td>Port 52</td>
<td>DSPRM 5, DSP 3, DSP channel 1</td>
</tr>
<tr>
<td>Port 53</td>
<td>DSPRM 5, DSP 3, DSP channel 2</td>
</tr>
<tr>
<td>Port 54</td>
<td>DSPRM 5, DSP 4, DSP channel 1</td>
</tr>
<tr>
<td>Port 55</td>
<td>DSPRM 5, DSP 4, DSP channel 2</td>
</tr>
<tr>
<td>Port 56</td>
<td>DSPRM 5, DSP 5, DSP channel 1</td>
</tr>
<tr>
<td>Port 57</td>
<td>DSPRM 5, DSP 5, DSP channel 2</td>
</tr>
<tr>
<td>Port 58</td>
<td>DSPRM 5, DSP 6, DSP channel 1</td>
</tr>
<tr>
<td>Port 59</td>
<td>DSPRM 5, DSP 6, DSP channel 2</td>
</tr>
</tbody>
</table>

The following shows that the function csm_vtsp_init_tdm() has been called with a voice control block of address 0x60B8562C. This function will be called only when the call is treated as a voice call.

Oct 18 04:05:07.052: csm_vtsp_init_tdm (voice_vdev=0x60B8562C)

The following shows that CSM has obtained a DSP channel from the DSP Resource Manager:

Oct 18 04:05:07.052: csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm slot 1, dspm 2, dsp 5, dsp_channel 1
    csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm stream 5, channel 9, bank 0, bp_channel 10

The DSP channel has the following initialized TDM channel information:

- TDM slot 1, dspm 2, dsp 5, dsp_channel 1 -- Indicates the physical DSP channel that will be used for this call.
• TDM stream 5, channel 9, bank 0, bp_channel 10--Indicates the on-chip and backplane TDM channel assigned to this DSP channel. Stream 5, channel 9 gives the on-chip TDM channel mapped to the DSP; bank 0, bp_channel 10 means that the backplane stream 0 and backplane channel #1 are assigned to this DSP.

The following shows that CSM has received an incoming call event from ISDN:

Oct 18 04:05:07.052: EVENT_FROM_ISDN:(00CF): DEV_INCALL at slot 1 and port 20
Slot 1, port 20 means the logical DSP channel 20 (mapped to DSPRM 2, DSP 5, DSP channel 1).

The following shows that the DEV_INCALL message has been translated into a CSM_EVENT_ISDN_CALL message:

Oct 18 04:05:07.052: CSM_PROC_IDLE: CSM_EVENT_ISDN_CALL at slot 1, port 20
This message is passed to the CSM central state machine while it is in the CSM_IDLE state and is in the CSM_PROC_IDLE procedure. The logical DSP channel port 20 on slot 1 is used to handle this call.

The following shows that CSM has invoked the vtsp_ic_notify() function with a CSM voice call control block 0x60B8562C.

Oct 18 04:05:07.052: vtsp_ic_notify : (voice_vdev= 0x60B8562C)
Inside this function, CSM will send a SETUP INDICATION message to the VTSP. This function will be invoked only if the call is a voice call.

The following shows that CSM has received a SETUP INDICATION RESPONSE message from the VTSP as an acknowledgment.

Oct 18 04:05:07.056: csm_vtsp_call_setup_resp (vdev_info=0x60B8562C, vtsp_cdb=0x60FCA114)
This means that the VTSP has received the CALL SETUP INDICATION message previously sent and has proceeded to process the call.

- vdev_info--Contains the address of the CSM voice data block.
- vtsp_cdb--Contains the address of the VTSP call control block.

The following shows that CSM has received a CALL CONNECT message from the VTSP:

Oct 18 04:05:07.596: csm_vtsp_call_connect (vtsp_cdb=0x60FCA114, voice_vdev=0x60B8562C)
This indicates that the VTSP has received a CONNECT message for the call leg initiated to the Internet side.

- vtsp_cdb--Contains the address of the VTSP call control block.
- voice_vdev--Contains the address of the CSM voice data block.

The following shows that while CSM is in the CSM_IC2_RING state, it receives a SETUP INDICATION RESPONSE from the VTSP. This message is translated into CSM_EVENT_MODEM_OFFHOOK and passed to the CSM central state machine.

Oct 18 04:05:07.596: CSM_PROC_IC2_RING: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 20
The following shows that CSM has received a CONNECT message from ISDN for the call using the logical DSP channel on slot 1 and port 20:

Oct 18 04:05:07.616: EVENT_FROM_ISDN:(00CF): DEV_CONNECTED at slot 1 and port 20
The following shows that CSM has translated the CONNECT event from ISDN into the CSM_EVENT_ISDN_CONNECTED message, which is then passed to the CSM central state machine:

Oct 18 04:05:07.616: CSM_PROC_IC4_WAIT_FOR_CARRIER: CSM_EVENT_ISDN_CONNECTED at slot 1, port 20

The following shows that CSM has received a CALL SETUP REQUEST from the VTSP:

May 16 12:22:27.580: csm_vtsp_call_setup_request (vtsp_cdb=0x60FCFA20, vtsp_sdb=0x60DFB608)

This represents a request to make an outgoing call to the PSTN.

- vtsp_cdb--Contains the address of the VTSP call control block.
- vtsp_sdb--Contains the address of the signalling data block for the signalling interface to be used to send the outgoing call.

The following shows that the physical DSP channel has been allocated for this outgoing call:

May 16 12:22:27.580: csm_vtsp_call_setup_request: tdm slot 1, dspm 5, dsp 4, dsp_channel 1

The following shows the on-chip and backplane TDM channel assigned to this DSP channel:

May 16 12:22:27.580: csm_vtsp_call_setup_request: tdm stream 5, channel 25, bank 0, bp_channel 27

In this sample output, tdm stream 5, channel 25, bank 0, bp_channel 27 indicates the on-chip and backplane TDM channel assigned to this DSP channel. Stream 5, channel 25 gives the on-chip TDM channel mapped to the DSP; bank 0, bp_channel 27 means that the backplane stream 0 and backplane channel 1 are assigned to this DSP.

The following shows the calling number and the called number for this call:

May 16 12:22:27.580: csm_vtsp_call_setup_request: calling number: 10001, called number: 30001

The following shows that the CALL SETUP REQUEST from the VTSP has been translated into the CSM_EVENT_MODEM_OFFHOOK message and is passed to the CSM central state machine:

May 16 12:22:27.580: CSM_PROC_IDLE: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 54

The logical DSP channel number for the DSP (slot 1, port 54) is now displayed, which maps to the physical DSP channel slot 1, dspm 5, dsp 4, dsp_channel 1.

The following shows that CSM has collected all the digits for dialing out:

May 16 12:22:27.580: CSM_PROC_OC3_COLLECT_ALL_DIGIT: CSM_EVENT_GET_ALL_DIGITS at slot 1, port 54

For PRI and for applications that do not require digit collection of outdialing digits (for example, voice calls), the intermediate digit collection states are omitted and the CSM state machine moves to this state directly, pretending that the digit collection has been done.

The following shows an information message:

May 16 12:22:27.580: CSM_PROC_OC3_COLLECT_ALL_DIGIT: called party num: (30001) at slot 1, port 54

The following shows that CSM attempts to find a free signalling D channel to direct the outgoing call:

May 16 12:22:27.580: csm_vtsp_check_dchan (voice_vdev=0x60B8562C)
May 16 12:22:27.580: csm_vtsp_check_dchan (vtsp_requested_dchan=0x60D7ACB0, dchan_idb=0x60E8ACF0)
May 16 12:22:27.580: csm_vtsp_check_dchan (voice_vdev=0x60B8562C)
May 16 12:22:27.580: csm_vtsp_check_dchan (vtsp_requested_dchan=0x60D7ACB0, dchan_idb=0x60D7ACB0)
In the case of voice calls, the free signaling D channel must match the voice interface specified inside the signalling data block (vtsp_sdb) passed from the VTSP.

The following shows that CSM has received an event from ISDN:

May 16 12:22:27.624: EVENT_FROM_ISDN::dchan_idb=0x60D7ACB0, call_id=0xA121, ces=0x1 bchan=0x1E, event=0x3, cause=0x0

In this sample output:

- dchan_idb--indicates the address of the hardware IDB for the D channel
- call_id--Indicates the call id assigned by ISDN
- bchan--Indicates the number of the B channel assigned for this call
- cause--Indicates the ISDN event cause

The following shows that CSM has received a CALL PROCEEDING message from ISDN.

May 16 12:22:27.624: EVENT_FROM_ISDN:(A121): DEV_CALL_PROC at slot 1 and port 54

The following shows that the CALL PROCEEDING event received from ISDN has been interpreted as a CSM_EVENT_ISDN_BCHAN_ASSIGNED message:

*May 16 12:22:27.624: CSM_PROC_OC4_DIALING: CSM_EVENT_ISDN_BCHAN_ASSIGNED at slot 1, port 54

ISDN has assigned a B channel for this outgoing call. This B channel must be on the same PRI span as the signalling D channel allocated previously.

The following shows that the csm_vtsp_setup_for_oc function is called:

May 16 12:22:27.624: csm_vtsp_setup_for_oc (voice_vdev=0x60B8562C)

This is invoked when an outgoing call initiated by the VTSP receives a response from the ISDN stack.

The following shows that ISDN has sent a CONNECT message to CSM indicating that the call leg to the PSTN side has been established:

May 16 12:22:28.084: EVENT_FROM_ISDN::dchan_idb=0x60D7ACB0, call_id=0xA121, ces=0x1 bchan=0x1E, event=0x4, cause=0x0

May 16 12:22:28.084: EVENT_FROM_ISDN:(A121): DEV_CONNECTED at slot 1 and port 54

The following shows that while CSM is in the OC5_WAIT_FOR_CARRIER state, it has received the 'CONNECT' message from ISDN and has translated it into the CSM_EVENT_ISDN_CONNECTED message, which is passed to the CSM central state machine:

May 16 12:22:28.084: CSM_PROC_OC5_WAIT_FOR_CARRIER: CSM_EVENT_ISDN_CONNECTED at slot 1, port 54

The following shows that the function vtsp_confirm_oc() has been called:

May 16 12:22:28.084: vtsp_confirm_oc : (voice_vdev= 0x60B8562C)

This is invoked after CSM received the CONNECT message from ISDN. CSM sends a confirmation of the CONNECT to the VTSP.
debug ctl-client

To collect debug information about the CTL client, use the debugctl-client command in privileged EXEC configuration mode. To disable collection of debug information, use the no form of this command.

```
debug ctl-client
no debug ctl-client
```

Syntax Description

This command has no arguments or keywords.

Command Default

Collection of CTL client debug information is disabled.

Command Modes

Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Cisco IOS Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(4)XC</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(9)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(9)T.</td>
</tr>
</tbody>
</table>

Usage Guidelines

This command is used with Cisco Unified CME phone authentication.

Examples

The following example shows debug messages for the CTL client:

```
Router# debug ctl-client
```
**debug ctunnel**

To display debugging messages for the IP over a Connectionless Network Service (CLNS) Tunnel feature, use the `debug ctunnel` command in privileged EXEC mode. To disable debugging messages, use the `no` form of this command.

```
debug ctunnel
no debug ctunnel
```

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

**Command Default**
No default behavior or values.

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(5)</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Examples**
As packets are sent over the virtual interface, the following type of output will appear on the console when the `debug ctunnel` command is used:

```
Router# debug ctunnel
4d21h: CTunnel1: IPCLNP encapsulated 49.0001.1111.1111.1111.00->49.0001.2222.2222.2222.00 (linktype=7, ien=89)
```
debug custom-queue

To enable custom queueing output, use the `debug custom-queue` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

`debug custom-queue`
`no debug custom-queue`

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Examples**

The following is an example of enabling custom queueing output:

```
Router# debug custom-queue
Custom output queueing debugging is on
```

The following is sample output from the `debug custom-queue` command:

```
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 2
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 2
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 2 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug priority</code></td>
<td>Enables priority queueing output.</td>
</tr>
</tbody>
</table>
**debug cwmp**

To debug the TR-069 Agent, use the `debug cwmp` command in privileged EXEC mode. To disable, use the `no` form of this command.

```
debug cwmp {all| error| profile| trace}
no debug cwmp {all| error| profile| trace}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Specifies all debug messages.</td>
</tr>
<tr>
<td>error</td>
<td>Specifies error messages.</td>
</tr>
<tr>
<td>profile</td>
<td>Specifies the error and trace messages in the Cisco WAN Management Protocol (CWMP) profiles.</td>
</tr>
<tr>
<td>trace</td>
<td>Specifies trace message.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(20)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug cwmp error` command:

```
Device# debug cwmp error
CWMP ERROR: cwmp_session_response_data -> HTTPC Response failed with httpc error
```

The following is sample trace message output from the `debug cwmp profile` command:

```
Device# debug cwmp profile
CWMP PROFILE: cwmp_generate_connection_request_URL -> ConnectionRequestURL = http://172.27.116.170/00000C/FTX1101ALXH/cwmp
```

The following is sample error message output from the `debug cwmp profile` command:

```
Device# debug cwmp profile
CWMP PROFILE ERROR: validate_dhcp_pool_min_max_address -> Missing InternetGatewayDevice.LANDevice.5.LANHostConfigManagement.MaxAddress
```

The following is sample output from the `debug cwmp trace` command:

```
Device# debug cwmp trace
CWMP: cwmp_process -> CWMP Engine started
```
debug cws

To enable Cloud Web Security debugging, use the debug cws command in privileged EXEC mode. To disable Cloud Web Security debugging, use the no form of this command.

```
debug cws {access-list {access-list-number | extended-access-list-number | access-list-name} | control-plane | errors | events | function-trace | packet-path}
```

```
no debug cws {access-list {access-list-number | extended-access-list-number | access-list-name} | control-plane | errors | events | function-trace | packet-path}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access-list</td>
<td>Enables debugging of Cloud Web Security access control lists (ACLs).</td>
</tr>
<tr>
<td>access-list-number</td>
<td>Access list number. The range is from 1 to 199.</td>
</tr>
<tr>
<td>extended-access-list-number</td>
<td>Extended access list number. The range is from 1300 to 2699.</td>
</tr>
<tr>
<td>access-list-name</td>
<td>Access list name.</td>
</tr>
<tr>
<td>control-plane</td>
<td>Enables debugging of Cloud Web Security control plane messages.</td>
</tr>
<tr>
<td>errors</td>
<td>Enables debugging of Cloud Web Security errors.</td>
</tr>
<tr>
<td>events</td>
<td>Enables debugging of Cloud Web Security events.</td>
</tr>
<tr>
<td>function-trace</td>
<td>Enables debugging of Cloud Web Security function trace.</td>
</tr>
</tbody>
</table>

### Command Modes

Privileged EXEC (#)

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.4(2)T</td>
<td>This command was introduced. This command replaces the debug content-scan command.</td>
</tr>
</tbody>
</table>

### Usage Guidelines

The content scanning process redirects client web traffic to the Cloud Web Security server.

You must configure ACLs by using the `ip access-list extended` command before you enable the debugging of Cloud Web Security ACLs. The `debug cws access-list` command enables conditional debugging for Cloud Web Security. The conditional debugging does not work for existing sessions when you enable content scan; the ACL match occurs after the first packet is received.
The following example shows how to enable extended ACL 149 and the debugging of Cloud Web Security ACL 149:

Device(config)# ip access-list extended 149
Device(config-ext-nacl)# permit ip host 10.1.0.1
Device(config-ext-nacl)# end
Device# debug cws access-list 149

Content-scan ACL based conditional debugging is on

The following is sample output from the debug cws access-list 149 command:

Device# debug cws access-list 149

Content-scan ACL based conditional debugging is on

Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH:Checking if it's a web trafficSrc IP:10.1.0.1(3646), Dst IP: 198.51.100.195(80),index:4
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH: Populating user/usergroup infoSrc IP:10.1.0.1(3646), Dst IP: 198.51.100.195(80)
Feb 19 19:01:02.887: CONT-SCAN-EVE: Username Received: defpmuser Src IP:10.1.0.1(E3E) Dst IP: 198.51.100.195(50) Pak:2033BCA0
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH: Adding session into HashSrc IP:10.1.0.1(3646), Dst IP: 198.51.100.195(80)
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH: Free port equals source port for src_ip 10.1.0.1(3646) dst_ip 198.51.100.195(80) 2033BCA0
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH: session enqueued 21ED1DE0 for src_ip 10.1.0.1(3646) dst_ip 198.51.100.195(80)
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH: Inserting Ingress session 21ED1DE0 in hash table at 1646 Src IP:10.1.0.1(3646), Dst IP: 198.51.100.195(80)
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH: Inserting Egress session 21ED1DE0 in hash table at 1629 Src IP:10.1.0.1(3646), Dst IP: 198.51.100.195(80)
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH: timer wheel started for 21ED1DE0 for src_ip 10.1.0.1(3646)
Feb 19 19:01:02.887: CONT-SCAN-EVE: 1 Usergroups from auth proxy Src IP:10.1.0.1(E3E), Dst IP: 198.51.100.195(50), cs_entry:21ED1DE0
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH: conn 21EE2024 ingress_fd 1073741847 0x2, payload_len = 0
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH: CONT SCAN: received fd1: 1073741847 and tcp flags

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cws out</td>
<td>Enables Cloud Web Security content scanning on an egress interface.</td>
</tr>
<tr>
<td>cws whitelisting</td>
<td>Enables whitelisting of incoming traffic and enters Cloud Web Security whitelisting configuration mode.</td>
</tr>
<tr>
<td>show cws</td>
<td>Displays Cloud Web Security information.</td>
</tr>
</tbody>
</table>
debug dampening

To display debug trace information messages for interface dampening, use the `debug dampening` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

`debug dampening [all interface]`
`no debug dampening [all interface]`

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>all</code></td>
<td>(Optional) Enables trace debugging for all dampening features.</td>
</tr>
<tr>
<td><code>interface</code></td>
<td>(Optional) Enables trace debugging for IP event dampening.</td>
</tr>
</tbody>
</table>

**Command Default**

No default behavior or values.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(22)S</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(13)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(13)T.</td>
</tr>
<tr>
<td>12.2(14)S</td>
<td>This command was integrated into Cisco IOS Release 12.2(14)S.</td>
</tr>
<tr>
<td>12.2(18)SXD</td>
<td>This command was integrated into Cisco IOS Release 12.2(18)SXD.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2(31)SB2</td>
<td>This command was integrated into Cisco IOS Release 12.2(31)SB2.</td>
</tr>
</tbody>
</table>

**Examples**

The following sample output is similar to the output that will be displayed when the `debug dampening` command is entered with the `interface` keyword. The sample output shows the following information:

- Ethernet interface 1/1 is configured with the IP Event Dampening feature. The half-life period is set to 30 seconds, the reuse threshold to 500, the suppress threshold to 1000, and the restart penalty to 90.
- The `shutdown` command and then the `no shutdown` command was entered on Ethernet interface 1/1. The interface was suppressed and then reused by the IP Event Damping feature.

```
Router# debug dampening interface
```
00:07:17:IF-EvD(Ethernet1/1): CLNS Routing reports state transition from UP to DOWN
00:07:17:EvD(Ethernet1/1): charge penalty 1000, new accum. penalty 1000, flap count 1
00:07:17:EvD(Ethernet1/1): accum. penalty 1000, now suppressed with a reuse intervals of 30
00:07:17:IF-EvD(Ethernet1/1): update CLNS Routing state to DOWN, interface is suppressed
00:07:17:IF-EvD(Ethernet1/1): IP Routing reports state transition from DOWN to DOWN
00:07:17:IF-EvD(Ethernet1/1): update IP Routing state to DOWN, interface is suppressed
00:07:17:IF-EvD(Ethernet1/1): CLNS Routing reports state transition from DOWN to UP
00:07:17:IF-EvD(Ethernet1/1): accum. penalty decayed to 1000 after 0 second(s)
00:07:17:EvD(Ethernet1/1): charge penalty 1000, new accum. penalty 1000, flap count 1
00:07:17:EvD(Ethernet1/1): accum. penalty 1000, now suppressed with a reuse intervals of 30
00:07:17:IF-EvD(Ethernet1/1): CLNS Routing reports state transition from DOWN to UP
00:07:17:EvD(Ethernet1/1): accum. penalty decayed to 1000 after 0 second(s)
00:07:17: IF-EvD(Ethernet1/1): charge penalty 1000, new accum. penalty 2000, flap count 2
00:07:17:EvD(Ethernet1/1): accum. penalty 2000, now suppressed with a reuse intervals of 60
00:07:17:IF-EvD(Ethernet1/1): CLNS Routing reports state transition from DOWN to UP
00:07:17:IF-EvD(Ethernet1/1): accum. penalty decayed to 2000 after 0 second(s)
00:07:17:EvD(Ethernet1/1): charge penalty 1000, new accum. penalty 3000, flap count 3
00:07:17:EvD(Ethernet1/1): accum. penalty 3000, now suppressed with a reuse intervals of 78
00:07:17:IF-EvD(Ethernet1/1): CLNS Routing reports state transition from UP to UP
00:07:17:IF-EvD(Ethernet1/1): IP Routing reports state transition from UP to UP
00:07:17:IF-EvD(Ethernet1/1): IP Routing reports state transition from UP to UP
00:07:20:IF-EvD(Ethernet1/1): CLNS Routing reports state transition from UP to UP
00:07:20:IF-EvD(Ethernet1/1): IP Routing reports state transition from UP to UP
00:07:47:IF-EvD: unsuppress interfaces
00:08:36:IF-EvD: unsuppress interfaces
00:08:36:EvD(Ethernet1/1): accum. penalty decayed to 483 after 79 second(s)
00:08:36:EvD(Ethernet1/1): accum. penalty 483, now unsuppressed
00:08:36:IF-EvD(Ethernet1/1): update IP Routing state to UP, interface is not suppressed
00:08:36:IF-EvD(Ethernet1/1): update CLNS Routing state to UP, interface is not suppressed
00:08:36:IF-EvD(Ethernet1/1): CLNS Routing reports state transition from UP to UP

The table below describes the significant fields shown in the display.

### Table 70: debug dampening Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing reports state transition from UP to DOWN</td>
<td>Displays the status of the specified interface from the perspective of the specified protocol. Interface state changes are displayed. The interface is specified within parentheses. The protocol is specified at the beginning of the message.</td>
</tr>
<tr>
<td>charge penalty 1000, new accum. penalty 1000, flap count 1</td>
<td>Displays the penalty assigned to the flapping interface and amount of penalty that is added to the accumulated penalty. The interface flap count is also displayed.</td>
</tr>
<tr>
<td>accum. penalty 1000, now suppressed with a reuse intervals of 30</td>
<td>Displays the status of the interface, accumulated penalty, and configured reuse threshold.</td>
</tr>
<tr>
<td>update CLNS Routing state to DOWN, interface is suppressed</td>
<td>Displays the status of the specified interface. Interface state changes and suppression status are displayed.</td>
</tr>
<tr>
<td>accum. penalty decayed to 1000 after 0 second(s)</td>
<td>Displays the decay rate of the accumulated penalty.</td>
</tr>
<tr>
<td>unsuppress interfaces</td>
<td>Indicates that dampened interfaces have been unsuppressed.</td>
</tr>
</tbody>
</table>
**debug data-store**

To display persistant storage device (PSD)-related debugging messages for the gateway GPRS support node (GGSN), use the `debug data-store` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug data-store
no debug data-store
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(14)YU</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(2)XB</td>
<td>This command was integrated into Cisco IOS Release 12.4(2)XB.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

This command displays PSD-related debugging messages for the GGSN.

⚠️ **Caution**

Because debugging output is assigned high priority in the CPU process, it can render the system unusable. For this reason, use debug commands only to troubleshoot specific problems or during troubleshooting sessions with Cisco technical support staff. Moreover, it is best to use debug commands during periods of lower network flows and fewer users. Debugging during these periods reduces the effect these commands have on other users on the system.

**Examples**

The following example configures a debugging session to check PSD-related parameters:

```
Router# debug data-store
```
debug data-store detail

To display extended details for persistent storage device (PSD)-related debugging information, use the debug data-store detail command in privileged EXEC mode. To disable debugging output, use the no form of this command.

```plaintext
debug data-store detail
no debug data-store detail
```

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

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(14)YU</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(2)XB</td>
<td>This command was integrated into Cisco IOS Release 12.4(2)XB.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
This command displays PSD-related debugging messages for the GGSN.

⚠️ **Caution**
Because debugging output is assigned high priority in the CPU process, it can render the system unusable. For this reason, use debug commands only to troubleshoot specific problems or during troubleshooting sessions with Cisco technical support staff. Moreover, it is best to use debug commands during periods of lower network flows and fewer users. Debugging during these periods reduces the effect these commands have on other users on the system.

**Examples**
The following example configures a detailed PSD-related debugging session:

```plaintext
Router# debug data-store details
```
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto-retrieve</td>
<td>Configures the GGSN to automatically initiate a retrieval of G-CDRs from PSDs defined in a PSD server group.</td>
</tr>
<tr>
<td>clear data-store statistics</td>
<td>Clears PSD-related statistics.</td>
</tr>
<tr>
<td>show data-store</td>
<td>Displays the status of the PSD client and PSD server-related information.</td>
</tr>
<tr>
<td>show data-store statistics</td>
<td>Displays statistics related to the PSD client.</td>
</tr>
</tbody>
</table>
debug dbconn all

To turn on all debug flags for Database Connection, use the `debug dbconn all` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug dbconn all
no debug dbconn all
```

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

**Command Default**  
Debugging is not enabled for Database Connection.

**Command Modes**  
Privileged EXEC

**Usage Guidelines**  
The `debug dbconn all` command displays debug output for Advanced Program-to-Program Communication (APPC), Database Connection configuration, Distributed Relational Database Architecture (DRDA), error messages, event traces, and TCP. The Database Connection debug flags are `appc`, `config`, `drda`, `event`, and `tcp`.

**Examples**  
See the sample output provided for the `debug dbconn appc`, `debug dbconn config`, `debug dbconn drda`, `debug dbconn event`, and `debug dbconn tcp` commands.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug dbconn appc</code></td>
<td>Displays APPC-related trace or error messages.</td>
</tr>
<tr>
<td><code>debug dbconn config</code></td>
<td>Displays trace or error messages for Database Connection configuration and control blocks.</td>
</tr>
<tr>
<td><code>debug dbconn drda</code></td>
<td>Displays error messages and stream traces for DRDA.</td>
</tr>
<tr>
<td><code>debug dbconn event</code></td>
<td>Displays trace or error messages for CTRC events related to DB2 communications.</td>
</tr>
<tr>
<td><code>debug dbconn tcp</code></td>
<td>Displays error messages and traces for TCP.</td>
</tr>
</tbody>
</table>
debug dbconn appc

To display Advanced Program-to-Program Communication (APPC)-related trace or error messages, use the debug dbconn appc command in privileged EXEC mode. To disable debugging output, use the no form of this command.

d debug dbconn appc
no debug dbconn appc

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

In a router with stable Database Connection, the alias_cp_name field in the trace message should not be blank. There should be no other APPC error message. You can use Advanced Peer-to-Peer Networking (APPN) debug commands with this debug command to track APPN-related errors.

Examples

The following is sample output from the debug dbconn appc command. In a normal situation, only the following message is displayed:

DBCONN-APPC: alias_cp_name is "ASH"

The following error messages are displayed if there is a network configuration error or other APPN-related problem:

DBCONN-APPC-612C2B28: APPC error: opcode 0x1, primary_rc 0x0003, secondary_rc 0x0004 0020 100C
DBCONN-APPC-612C2B28: Verb block =
DBCONN-APPC-612C2B28: 0001 0200 0000 0000 0000 0000 0000 0000
DBCONN-APPC-612C2B28: 0000 0000 8014 0003 0000 0000 0000 0000
DBCONN-APPC-612C2B28: D3E4 F6F2 E2E3 C1D9 C4C2 F240 4040 4040
DBCONN-APPC-612C2B28: 610A 828B 0000 0000 0000 0000 0000 0000
DBCONN-APPC-612C2B28: D4C5 D9D9 C9C5 4040 4040

You can use the debug appn command to obtain more information.
The following message is displayed if a database connection is manually cleared and an outstanding APPC verb is pending:

DBCONN-APPC-%612C2B28: Canceling pending APPC verb 0x1

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug dbconn all</td>
<td>Turns on all debug flags for Database Connection.</td>
</tr>
<tr>
<td>debug dbconn config</td>
<td>Displays trace or error messages for Database Connection configuration and control blocks.</td>
</tr>
<tr>
<td>debug dbconn drda</td>
<td>Displays error messages and stream traces for DRDA.</td>
</tr>
<tr>
<td>debug dbconn event</td>
<td>Displays trace or error messages for Database Connection events.</td>
</tr>
<tr>
<td>debug dbconn tcp</td>
<td>Displays error messages and traces for TCP.</td>
</tr>
</tbody>
</table>
debug dbconn config

To display trace or error messages for Database Connection configuration and control blocks, use the debug dbconn config command in privileged EXEC mode. To disable debugging output, use the no form of this command.

d debug dbconn config
no debug dbconn config

Syntax Description
This command has no arguments or keywords.

Command Modes
Privileged EXEC

Usage Guidelines
Most of the messages for Database Connection and control blocks do not report any errors. If a connection is inactive and cannot be cleared, use this command with the debug dbconn appc, debug dbconn tcp, and debug appn commands to locate the problem. The alias_cp_name field must match the configured APPN cpname.

Examples
The following is sample output from the debug dbconn config command:

Router# debug dbconn config
DBCONN-CONFIG: alias_cp_name is "ASH"
DBCONN-CONFIG: connection 612BDAAC matching server on 198.147.235.5:0 with rdbname=STELLA
DBCONN-CONFIG: APPN shutdown; clearing connection 1234abcd
DBCONN-CONFIG: created server 612C2720
DBCONN-CONFIG: server 612C2720 (listen 60F72E94) is active
DBCONN-CONFIG: server 612C2720 (listen 60F72E94) is active
DBCONN-CONFIG: new connection 612BDAAC
DBCONN-CONFIG: listen 60F72E94 accepts connection 612BDAAC
DBCONN-CONFIG: server 60F74614 takes connection 612BDAAC
DBCONN-CONFIG: listen 60F72E94 releases connection 612BDAAC
DBCONN-CONFIG: server 60F74614 releases connection 612BDAAC
DBCONN-CONFIG: listen 60F72E94 abandons connection 612BDAAC
DBCONN-CONFIG: server 60F74614 abandons connection 612BDAAC
DBCONN-CONFIG: deleting server 612C2720
DBCONN-CONFIG: daemon 60361738 takes zombie connection 612BDAAC
DBCONN-CONFIG: daemon 60361738 releases zombie connection 612BDAAC

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug dbconn all</td>
<td>Turns on all debug flags for Database Connection.</td>
</tr>
<tr>
<td>debug dbconn appc</td>
<td>Displays APPC-related trace or error messages.</td>
</tr>
<tr>
<td>debug dbconn drda</td>
<td>Displays error messages and stream traces for DRDA.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>debug dbconn event</td>
<td>Displays trace or error messages for Database Connection events.</td>
</tr>
<tr>
<td>debug dbconn tcp</td>
<td>Displays error messages and traces for TCP.</td>
</tr>
</tbody>
</table>
**debug dbconn drda**

To display error messages and stream traces for Distributed Relational Database Architecture (DRDA), use the `debug dbconn drda` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug dbconn drda
no debug dbconn drda
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Debugging is not enabled for the dbconn subsystem.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(2)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug dbconn drda` command:

```
Router# debug dbconn drda
*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: DSS X'006CD0410001', length 108, in chain, REQDSS, correlator 1
*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: OBJECT X'00661041', length 98, code point X'1041'
*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: OBJECT X'0020115E' in COLLECTION X'1041', length 28, code point X'115E'
*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: OBJECT X'0013115A' in COLLECTION X'1041', length 15, code point X'115A' (skipping...)
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug dbconn all</td>
<td>Turns on all debug flags for Database Connection.</td>
</tr>
<tr>
<td>debug dbconn appc</td>
<td>Displays APPC-related trace or error messages.</td>
</tr>
<tr>
<td>debug dbconn config</td>
<td>Displays trace or error messages for Database Connection configuration and control blocks.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>debug dbconn event</td>
<td>Displays trace or error messages for CTRC events related to DB2 communications.</td>
</tr>
<tr>
<td>debug dbconn tcp</td>
<td>Displays error messages and traces for TCP.</td>
</tr>
</tbody>
</table>
debug dbconn event

To display trace or error messages for Cisco Transaction Connection (CTRC) events related to DATABASE2 (DB2) communications, use the debug dbconn event command in privileged EXEC mode. To disable debugging output, use the no form of this command.

**debug dbconn event**

**no debug dbconn event**

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Debugging is not enabled for the dbconn subsystem.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(2)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Examples**

The following examples display output from the debug dbconn event command in a variety of situations. A normal trace for the debug dbconn event displays as follows:

```
Router# debug dbconn event
DBCONN-EVENT: Dispatch to 60FD6C00, from 0, msg 60F754CC, msgid 6468 'dh', buffer 0.
DBCONN-EVENT: [*] Post to 61134240(cn), from 60EC5470(tc), msg 611419E4, msgid 0x6372 'cr', buffer 612BF68C.
DBCONN-EVENT: Flush events called for pto 61182742, pfrom 61239837.
DBCONN-EVENT: Event discarded: to 61182742 (cn), from 61239837(ap), msg 61339273, msgid 0x6372 'cr' buffer 0.
DBCONN-EVENT: == Send to 1234abcd, from 22938acd, msg 72618394, msgid 0x6372 'cr', buffer 0.
```

If the following messages are displayed, contact Cisco technical support personnel:

```
DBCONN-TCPFSM-1234abcd: Cannot occur in state 2 on input 6363 ('cc')
DBCONN-APPFSM-1234abcd: Cannot occur in state 3 on input 6363 ('cc')
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug dbconn all</td>
<td>Turns on all debug flags for Database Connection.</td>
</tr>
<tr>
<td>debug dbconn appc</td>
<td>Displays APPC-related trace or error messages.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>debug dbconn config</td>
<td>Display trace or error messages for Database Connection configuration and control blocks.</td>
</tr>
<tr>
<td>debug dbconn drda</td>
<td>Displays error messages and stream traces for DRDA.</td>
</tr>
<tr>
<td>debug dbconn tcp</td>
<td>Displays error messages and traces for TCP.</td>
</tr>
<tr>
<td>show debugging</td>
<td>Displays the state of each debugging option.</td>
</tr>
</tbody>
</table>
**debug dbconn tcp**

To display error messages and traces for TCP, use the `debug dbconn tcp` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```plaintext
debug dbconn tcp
no debug dbconn tcp
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Debugging is not enabled for the dbconn subsystem.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(2)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug dbconn tcp` command:

```plaintext
Router# debug dbconn tcp
DBCONN-TCP-63528473: tcpdriver_passive_open returned NULL
DBCONN-TCP-63528473: (no memory) tcp_reset(63829482) returns 4
DBCONN-TCP: tcp_accept(74625348, error) returns tcb 63829482, error 4
DBCONN-TCP: (no memory) tcp_reset(63829482) returns 4
DBCONN-TCP-63528473: (open) tcp_create returns 63829482, error - 4
DBCONN-TCP-63528473: tcb_connect(63829482, 1.2.3.4, 2010) returns 4
DBCONN-TCP-63528473: (open error) tcp_reset(63829482) returns 4
DBCONN-TCP-63528473: tcp_create returns 63829482, error - 4
DBCONN-TCP-63528473: tcb_bind(63829482, 0.0.0.0, 2001) returns 4
DBCONN-TCP-63528473: tcp_listen(63829482, ) returns 4
DBCONN-TCP-63528473: (errors) Calling tcp_close (63829482)
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug dbconn all</code></td>
<td>Turns on all debug flags for Database Connection.</td>
</tr>
<tr>
<td><code>debug dbconn appc</code></td>
<td>Displays APPC-related trace or error messages.</td>
</tr>
<tr>
<td><code>debug dbconn config</code></td>
<td>Displays trace or error messages for Database Connection configuration and control blocks.</td>
</tr>
<tr>
<td><code>debug dbconn drda</code></td>
<td>Displays error messages and stream traces for DRDA.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>debug dbconn event</code></td>
<td>Displays trace or error messages for CTRC events related to DB2 communications.</td>
</tr>
<tr>
<td><code>show debugging</code></td>
<td>Displays the state of each debugging option.</td>
</tr>
</tbody>
</table>
debug decnet adj through debug dss ipx event

- debug decnet adj through debug dss ipx event, page 581
The **debug decnet adj** command is not available in Cisco IOS Release 12.2(33)SXH and later Cisco IOS 12.2SX releases.

To display debugging information on DECnet adjacencies, use the **debug decnet adj** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug decnet adj
no debug decnet adj
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC (#)

**Examples**

The following is sample output from the **debug decnet adj** command:

```
Router# debug decnet adj
DNET-ADJ: Level 1 hello from 1.3
DNET-ADJ: sending hellos
DNET-ADJ: Sending hellos to all routers on interface Ethernet0, blksize 1498
DNET-ADJ: Level 1 hello from 1.3
DNET-ADJ: 1.5 adjacency initializing
DNET-ADJ: Sending hellos to all routers on interface Ethernet0, blksize 1498
DNET-ADJ: Level 1 hello from 1.3
DNET-ADJ: 1.5 adjacency up
DNET-ADJ: Level 1 hello from 1.5
DNET-ADJ: 1.5 adjacency down, listener timeout
```

The following line indicates that the router is sending hello messages to all routers on this segment, which in this case is Ethernet 0:

```
DNET-ADJ: Sending hellos to all routers on interface Ethernet0, blksize 1498
```

The following line indicates that the router has heard a hello message from address 1.5 and is creating an adjacency entry in its table. The initial state of this adjacency will be **initializing**.

```
DNET-ADJ: 1.5 adjacency initializing
```

The following line indicates that the router is sending an unscheduled (triggered) hello message as a result of some event, such as new adjacency being heard:

```
DNET-ADJ: sending triggered hellos
```

The following line indicates that the adjacency with 1.5 is now up, or active:

```
DNET-ADJ: 1.5 adjacency up
```

The following line indicates that the adjacency with 1.5 has timed out, because no hello message has been heard from adjacency 1.5 in the time interval originally specified in the hello message from 1.5:

```
DNET-ADJ: 1.5 adjacency down, listener timeout
```
The following line indicates that the router is sending an unscheduled hello message, as a result of some event, such as the adjacency state changing:

DNET-ADJ: hello update triggered by state changed in dn_add_adjacency
debug decnet connects

The **debug decnet connects** command is not available in Cisco IOS Release 12.2(33)SXH and later Cisco IOS 12.2SX releases.

To display debugging information of all connect packets that are filtered (permitted or denied) by DECnet access lists, use the **debug decnet connects** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug decnet connects
no debug decnet connects
```

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

**Command Modes**
Privileged EXEC (#)

**Usage Guidelines**
When you use connect packet filtering, it may be helpful to use the **decnets access-group** configuration command to apply the following basic access list:

```
access-list 300 permit 0.0 63.1023 eq any
```
You can then log all connect packets sent on interfaces to which you applied this list, in order to determine those elements on which your connect packets must be filtered.

**Note**
Packet password and account information is not logged in the **debug decnet connects** message, nor is it displayed by the **show access** EXEC command. If you specify **password** or **account** information in your access list, they can be viewed by anyone with access to the configuration of the router.

**Examples**
The following is sample output from the **debug decnet connects** command:

```
Router# debug decnet connects
DNET-CON: list 300 item #2 matched src=19.403 dst=19.309 on Ethernet0: permitted
srcreterr"=RICK" srcuic=[0,017] dstobj=42 id="USER"
The table below describes significant fields shown in the output.
```

**Table 71: debug decnet connects Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNET-CON:</td>
<td>Indicates that this is a <strong>debug decnet connects</strong> packet.</td>
</tr>
<tr>
<td>list 300 item #2 matched</td>
<td>Indicates that a packet matched the second item in access list 300.</td>
</tr>
<tr>
<td>src=19.403</td>
<td>Indicates the source DECnet address for the packet.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dst=19.309</td>
<td>Indicates the destination DECnet address for the packet.</td>
</tr>
<tr>
<td>on Ethernet0:</td>
<td>Indicates the router interface on which the access list filtering the packet was applied.</td>
</tr>
<tr>
<td>permitted</td>
<td>Indicates that the access list permitted the packet.</td>
</tr>
<tr>
<td>srcname = &quot;RICK&quot;</td>
<td>Indicates the originator user of the packet.</td>
</tr>
<tr>
<td>srcuic=[0,017]</td>
<td>Indicates the source UIC of the packet.</td>
</tr>
<tr>
<td>dstobj=42</td>
<td>Indicates that DECnet object 42 is the destination.</td>
</tr>
<tr>
<td>id=&quot;USER&quot;</td>
<td>Indicates the access user.</td>
</tr>
</tbody>
</table>
The `debug decnet events` command is not available in Cisco IOS Release 12.2(33)SXH and later Cisco IOS 12.2SX releases.

To display debugging information on DECnet events, use the `debug decnet events` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
dbg decnet events
no dbg decnet events
```

### Syntax Description

This command has no arguments or keywords.

### Command Modes

Privileged EXEC (#)

### Examples

The following is sample output from the `debug decnet events` command:

```
Router# debug decnet events
DNET: Hello from area 50 rejected - exceeded 'max area' parameter (45)
DNET: Hello from area 50 rejected - exceeded 'max area' parameter (45)
```

The following line indicates that the router received a hello message from a router whose area was greater than the max-area parameter with which this router was configured:

```
DNET: Hello from area 50 rejected - exceeded 'max area' parameter (45)
```

The following line indicates that the router received a hello message from a router whose node ID was greater than the max-node parameter with which this router was configured:

```
DNET: Hello from node 1002 rejected - exceeded 'max node' parameter (1000)
```
debug decnet packet

**Note**
The `debugdecnetpacket` command is not available in Cisco IOS Release 12.2(33)SXH and later Cisco IOS 12.2SX releases.

To display debugging information on DECnet packet events, use the `debugdecnetpacket` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
depedt decnet packet
        no debug decnet packet
```

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

**Command Modes**
Privileged EXEC (#)

**Examples**
The following is sample output from the `debugdecnetpacket` command:

```
Router# debug decnet packet
DNET-PKT: src 1.4 dst 1.5 sending to PHASEV
        DNET-PKT: Packet fwded from 1.4 to 1.5, via 1.5, snpa 0000.3080.cf90, TokenRing0
The following line indicates that the router is sending a converted packet addressed to node 1.5 to Phase V:

DNET-PKT: src 1.4 dst 1.5 sending to PHASEV
        The following line indicates that the router forwarded a packet from node 1.4 to node 1.5. The packet is being sent to the next hop of 1.5 whose subnetwork point of attachment (MAC address) on that interface is 0000.3080.cf90.

DNET-PKT: Packet fwded from 1.4 to 1.5, via 1.5, snpa 0000.3080.cf90, TokenRing0
```
The `debug decnet routing` command is not available in Cisco IOS Release 12.2(33)SXH and later Cisco IOS 12.2SX releases.

To display all DECnet routing-related events occurring at the router, use the `debug decnet routing` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
decnet routing
no decnet routing
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC (#)

**Examples**

The following is sample output from the `debug decnet routing` command:

```
Router# debug decnet routing
DNET-RT: Received level 1 routing from 1.3 on Ethernet0 at 1:16:34
DNET-RT: Sending routes
DNET-RT: Sending normal routing updates on Ethernet0
DNET-RT: Sending level 1 routing updates on interface Ethernet0
DNET-RT: Level1 routes from 1.5 on Ethernet0: entry for node 5 created
DNET-RT: route update triggered by after split route pointers in dn_rt_input
DNET-RT: Received level 1 routing from 1.5 on Ethernet 0 at 1:18:35
DNET-RT: Sending L1 triggered routes
DNET-RT: Sending L1 triggered routing updates on Ethernet0
DNET-RT: removing route to node 5
```

The following line indicates that the router has received a level 1 update on Ethernet interface 0:

```
DNET-RT: Received level 1 routing from 1.3 on Ethernet0 at 1:16:34
```

The following line indicates that the router is sending its scheduled updates on Ethernet interface 0:

```
DNET-RT: Sending normal routing updates on Ethernet0
```

The following line indicates that the router will send an unscheduled update on this interface as a result of some event. In this case, the unscheduled update is a result of a new entry created in the routing table of the interface.

```
DNET-RT: route update triggered by after split route pointers in dn_rt_input
```

The following line indicates that the router sent the unscheduled update on Ethernet 0:

```
DNET-RT: Sending L1 triggered routes
DNET-RT: Sending L1 triggered routing updates on Ethernet0
```

The following line indicates that the router removed the entry for node 5 because the adjacency with node 5 timed out, or the route to node 5 through a next-hop router was disconnected:

```
DNET-RT: removing route to node 5
```
**debug device-sensor**

To enable debugging for device sensor, use the `debug device-sensor` command in privileged EXEC mode.

```
debugeevice-sensor {errors | events}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>errors</td>
<td>Displays device sensor error messages.</td>
</tr>
<tr>
<td>events</td>
<td>Displays messages for events such as protocol packet arrivals, identity updates, and release events sent to the session manager.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.0(1)SE1</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>15.1(1)SG</td>
<td>This command was integrated into Cisco IOS Release 15.1(1)SG.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use the `debug device-sensor` command in conjunction with the `debug authentication all` command to troubleshoot scenarios where device sensor cache entries are not being created for the connected devices.

**Examples**

The following is sample output from the `debug device-sensor events` command. The debug output shows how Cisco Discovery Protocol packets and Type-Length-Values (TLVs) are received from the device connected to the Gigabit Ethernet interface 2/1.

```
Device# debug device-sensor events
Device# *Nov 30 23:58:45.811: DSensor: Received cdp packet from GigabitEthernet2/1:00d0.2bdf.08a5
*Nov 30 23:58:45.811: DSensor: SM returned no or invalid session label for GigabitEthernet2/1:00d0.2bdf.08a5
*Nov 30 23:58:45.811: DSensor: Updating SM with identity attribute list
  cdp-tlv 0 00 00 00 0B 4A 41 45 30 37 34 31 31 53 52 5A
  cdp-tlv 0 00 00 04 04 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 05 00 68 57 53 2D 43 32 39 34 38 20 53 6F 66 74 77 61 72 65
  cdp-tlv 0 00 06 00 08 57 53 2D 43 32 39 34 38 20 53 6F 66 74 77 61 72 65
  cdp-tlv 0 00 07 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 08 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 09 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 0A 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 0B 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 0C 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 0D 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 0E 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 0F 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 10 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 11 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 12 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 13 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 14 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 15 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 16 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 17 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 18 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 19 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 1A 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 1B 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 1C 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 1D 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 1E 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 1F 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 20 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 21 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 22 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 23 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 24 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 25 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 26 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 27 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 28 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 29 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 2A 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 2B 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 2C 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 2D 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 2E 00 04 00 00 00 00 00 00 00 00 00 00 00
  cdp-tlv 0 00 2F 00 04 00 00 00 00 00 00 00 00 00 00 00
```
Routed Protocols

cdp-tlv 0 00 13 00 01 00
cdp-tlv 0 00 14 00 00
cdp-tlv 0 00 15 00 0A 06 08 2B 06 01 04 01 09 05 2A
cdp-tlv 0 00 16 00 16 00 00 00 02 01 01 CC 00 04 00 00 00 00 00 00 00 01 CC 00 04 01 01 01
cdp-tlv 0 00 17 00 01 00
swidb 0 604702240 (0x240B0620)
clid-mac-addr 0 00 D0 2B DF 08 A5

*Nov 30 23:58:46.831: DSensor: Received cdp packet from GigabitEthernet2/1:00d0.2bdf.08a5exi
Switch#
*Nov 30 23:58:51.171: %SYS-5-CONFIG_I: Configured from console by console

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug authentication all</td>
<td>Displays all debugging information about the Authentication Manager and all features.</td>
</tr>
<tr>
<td>device-sensor accounting</td>
<td>Adds the device sensor protocol data to the accounting records and generates additional accounting events when new sensor data is detected.</td>
</tr>
</tbody>
</table>
debug dhcp

To display debugging information about the Dynamic Host Configuration Protocol (DHCP) client activities and to monitor the status of DHCP packets, use the debug dhcp command in privileged EXEC mode. To disable debugging output, use the no form of this command.

```
default dhcp [detail]
no default dhcp [detail]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>detail</th>
<th>(Optional) Displays additional debugging information.</th>
</tr>
</thead>
</table>

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(8)T</td>
<td>The output of this command was enhanced to display default static routes.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

You can also use the debug dhcp command to monitor the subnet allocation and releasing for on-demand address pools.

For debugging purposes, the debug dhcp detail command provides the most useful information such as the lease entry structure of the client and the state transitions of the lease entry. The debug output shows the scanned option values from received DHCP messages that are replies to a router request. The values of the op, htype, hlen, hops, server identifier option, xid, secs, flags, ciaddr, yiaddr, siaddr, and giaddr fields of the DHCP packet are shown in addition to the length of the options field.

**Examples**

The following examples show and explain some of the typical debugging messages you may see when using the debug dhcp detail command.

The following sample output shows when a DHCP client sends a DHCPDISCOVER broadcast message to find its local DHCP server:

```
Router# debug dhcp detail
00:07:16:DHCP:DHCN:DHCP client process started:10
00:07:16:RAC:Starting DHCP discover on Ethernet2
00:07:16:DHCP:Try 1 to acquire address for Ethernet2
00:07:16:%SYS-5-CONFIG_I:Configured from console by console
00:07:19:DHCP:Shutting down from get_netinfo()
00:07:19:DHCP:Attempting to shutdown DHCP Client
00:07:21:DHCP:allocate request
```
The first seven lines of the following output show the current values stored in the lease entry structure for the client:

```
00:07:21:DHCP:new entry. add to queue
00:07:21:DHCP:SDiscover attempt # 1 for entry:
00:07:21:Temp IP addr:0.0.0.0 for peer on Interface:Ethernet2
00:07:21:Temp sub net mask:0.0.0.0
00:07:21: DHCP Lease server:0.0.0.0, state:1 Selecting
00:07:21: DHCP transaction id:582
00:07:21: Lease:0 secs, Renewal:0 secs, Rebind:0 secs
00:07:21: Next timer fires after:00:00:03
00:07:21: Retry count:1 Client-ID:cisco-0010.7b6e.afd8-Et2
00:07:21:DHCP:SDiscover: sending 308 byte length DHCP packet
00:07:21:DHCP:SDiscover 308 bytes
00:07:21: B'cast on Ethernet2 interface from 0.0.0.0
```

The following output shows the offered addresses and parameters sent to the DHCP client by the DHCP server via a DHCPOFFER message. The messages containing the Scan field indicate the options that were scanned from the received BOOTP packet and the corresponding values:

```
00:07:23:DHCP:Received a BOOTREP pkt
00:07:23:DHCP:Scan:Message type:DHCP Offer
00:07:23:DHCP:Scan:Server ID Option:10.1.1.1 = A010101
00:07:23:DHCP:Scan:Lease Time:180
00:07:23:DHCP:Scan:Renewal time:90
00:07:23:DHCP:Scan:Rebind time:157
00:07:23:DHCP:Scan:Subnet Address Option:255.255.255.0
```

The following output shows selected fields in the received BOOTP packet:

```
00:07:23:DHCP:rcvd pkt source:10.1.1.1, destination: 255.255.255.255
00:07:23: UDP sport:43, dport:44, length:308
00:07:23: DHCP op:2, htype:1, hlen:6, hops:0
00:07:23: DHCP server identifier:10.1.1.1
00:07:23: xid:582, secs:0, flags:8000
00:07:23: client:0.0.0.0, your:10.1.1.2
00:07:23: srvr: 0.0.0.0, gw:0.0.0.0
00:07:23: options block length:60
00:07:23:DHCP Offer Message Offered Address:10.1.1.2
00:07:23:DHCP:Lease Seconds:180 Renewal secs: 90 Rebind secs:157
00:07:23:DHCP:Server ID Option:10.1.1.1
00:07:23:DHCP:offer received from 10.1.1.1
```

The following output shows when the DHCP client sends a DHCPREQUEST broadcast message to the DHCP server to accept the offered parameters:

```
00:07:23:DHCP:SRequest attempt # 1 for entry:
00:07:23:Temp IP addr:10.1.1.2 for peer on Interface:Ethernet2
00:07:23:Temp sub net mask:255.255.255.0
00:07:23: DHCP Lease server:10.1.1.1, state:2 Requesting
00:07:23: DHCP transaction id:582
00:07:23: Lease:180 secs, Renewal:0 secs, Rebind:0 secs
00:07:23: Next timer fires after:00:00:02
00:07:23: Retry count:1 Client-ID:cisco-0010.7b6e.afd8-Et2
00:07:23:DHCP:SRequest- Server ID option:10.1.1.1
00:07:23:DHCP:SRequest- Requested IP addr option:10.1.1.2
00:07:23:DHCP:SRequest placed lease len option:180
00:07:23:DHCP:SRequest:326 bytes
00:07:23:DHCP:SRequest:326 bytes
00:07:23: B'cast on Ethernet2 interface from 0.0.0.0
```

The following output shows when the DHCP server sends a DHCPACK message to the client with the full set of configuration parameters:

```
00:07:23:DHCP:Received a BOOTREP pkt
00:07:23:DHCP:Scan:Message type:DHCP Ack
00:07:23:DHCP:Scan:Server ID Option:10.1.1.1 = A010101
00:07:23:DHCP:Scan:Lease Time:180
00:07:23:DHCP:Scan:Renewal time:90
00:07:23:DHCP:Scan:Rebind time:157
```
The following output shows when a default gateway (option 3) is assigned a static IP address that is the default route and that static routes were added from the DHCP server:

Oct 2 06:22:24: Setting default_gateway to 68.8.8.1! This is the option 3 default gateway.
Oct 2 06:22:24: Adding default route 68.8.8.1
Oct 2 06:22:24: DHCP: Adding static route to 4.3.2.1 255.255.255.255 via 68.8.8.1
Oct 2 06:22:24: DHCP: Adding static route to 1.1.1.1 255.255.255.255 via 68.8.8.1
Oct 2 06:22:24: DHCP: Adding static route to 67.2.2.2 255.255.255.255 via 68.8.8.1

Most fields are self-explanatory; however, fields that may need further explanation are described in the table below.

Table 72: debug dhcp Field Descriptions

<table>
<thead>
<tr>
<th>Fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DHCP:Scan:Subnet Address Option:255.255.255.0</td>
<td>Subnet mask option (option 1).</td>
</tr>
<tr>
<td>DHCP server identifier:1.1.1.1</td>
<td>Value of the DHCP server ID option (option 54). Note that this is not the same as the siaddr field, which is the server IP address.</td>
</tr>
<tr>
<td>srvr:0.0.0.0, gw:0.0.0.0</td>
<td>srvr is the value of the siaddr field. gw is the value of the giaddr field.</td>
</tr>
</tbody>
</table>

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ip ddns update</td>
<td>Enables debugging for DDNS updates.</td>
</tr>
<tr>
<td>debug ip dhcp server</td>
<td>Enables DHCP server debugging.</td>
</tr>
<tr>
<td>host (host-list)</td>
<td>Specifies a list of hosts that will receive DDNS updates of A and PTR RRs.</td>
</tr>
<tr>
<td>ip ddns update hostname</td>
<td>Enables a host to be used for DDNS updates of A and PTR RRs.</td>
</tr>
<tr>
<td>ip ddns update method</td>
<td>Specifies a method of DDNS updates of A and PTR RRs and the maximum interval between the updates.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>ip dhcp client update dns</code></td>
<td>Enables DDNS updates of A RRs using the same hostname passed in the hostname and FQDN options by a client.</td>
</tr>
<tr>
<td><code>ip dhcp-client update dns</code></td>
<td>Enables DDNS updates of A RRs using the same hostname passed in the hostname and FQDN options by a client.</td>
</tr>
<tr>
<td><code>ip dhcp update dns</code></td>
<td>Enables DDNS updates of A and PTR RRs for most address pools.</td>
</tr>
<tr>
<td><code>ip host-list</code></td>
<td>Specifies a list of hosts that will receive DDNS updates of A and PTR RRs.</td>
</tr>
<tr>
<td><code>show ip ddns update</code></td>
<td>Displays information about the DDNS updates.</td>
</tr>
<tr>
<td><code>show ip ddns update method</code></td>
<td>Displays information about the DDNS update method.</td>
</tr>
<tr>
<td><code>show ip dhcp server pool</code></td>
<td>Displays DHCP server pool statistics.</td>
</tr>
<tr>
<td><code>show ip host-list</code></td>
<td>Displays the assigned hosts in a list.</td>
</tr>
<tr>
<td><code>update dns</code></td>
<td>Dynamically updates a DNS with A and PTR RRs for some address pools.</td>
</tr>
</tbody>
</table>
**debug dhcp redundancy**

To display debugging information about DHCP proxy client redundancy events, use the `debug dhcp redundancy` command in privileged EXEC mode. To disable the display of debugging output, use the `no` form of this command.

```
d debug dhcp redundancy
no debug dhcp redundancy
```

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

**Command Default**  Debugging output is disabled for DHCP redundancy events.

**Command Modes**  Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(31)SB2</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(31)SRB1</td>
<td>This command was integrated into Cisco IOS Release 12.2(31)SRB1.</td>
</tr>
</tbody>
</table>

**Examples**

The following example displays debug messages regarding DHCP redundancy events. The last line is output when the `debug dhcp redundancy` command is enabled. The line indicates that the active Route Processor has sent a dynamic lease synchronization message for IP address 10.1.1.1:

```
Router# debug dhcp redundancy
*Mar 15 10:32:21: DHCPD: assigned IP address 10.1.1.1 to client
*Mar 15 10:32:21: DHCPD: dynamic sync sent for 10.1.1.1
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug ip dhcp server redundancy</code></td>
<td>Displays debugging information about DHCP server and relay agent redundancy events.</td>
</tr>
</tbody>
</table>
debug dialer events

To display debugging information about the packets received on a dialer interface, use the `debug dialer events` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug dialer events
no debug dialer events
```

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

**Command Modes**  
Privileged EXEC

**Usage Guidelines**  
When dial-on-demand routing (DDR) is enabled on the interface, information concerning the cause of any call (called the Dialing cause) is displayed.

**Examples**  
In the following example, the line of output for an IP packet lists the name of the DDR interface and the source and destination addresses of the packet:

```
Router# debug dialer events
Dialing cause: Serial0: ip (s=172.16.1.111 d=172.16.2.22)
```

The following line of output for a bridged packet lists the DDR interface and the type of packet (in hexadecimal). For information on these packet types, see the "Ethernet Type Codes" appendix of the Cisco IOS Bridging and IBM Networking Command Reference publication.

```
Dialing cause: Serial1: Bridge (0x6005)
```

Most messages are self-explanatory; however, messages that may need some explanation are described in the table below.

**Table 73: debug dialer events Message Descriptions**

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dialer0: Already xxx call(s) in progress on Dialer0, dialing not allowed</td>
<td>Number of calls in progress (xxx) exceeds the maximum number of calls set on the interface.</td>
</tr>
<tr>
<td>Dialer0: No free dialer - starting fast idle timer</td>
<td>All the lines in the interface or rotary group are busy, and a packet is waiting to be sent to the destination.</td>
</tr>
<tr>
<td>BRI0: rotary group to xxx overloaded (yyy)</td>
<td>Number dialer (xxx) exceeds the load set on the interface (yyy).</td>
</tr>
<tr>
<td>BRI0: authenticated host xxx with no matching dialer profile</td>
<td>No dialer profile matches xxx, the Challenge Handshake Authentication Protocol (CHAP) name or remote name of the remote host.</td>
</tr>
</tbody>
</table>
The table below describes the messages that the `debug dialer events` command can generate for a serial interface used as a V.25bis dialer for DDR.

### Table 74: debug dialer events Command Message Descriptions for DDR

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial 0: Dialer result = xxxxxxxxxxxxxx</td>
<td>Result returned from the V.25bis dialer. It is useful in debugging if calls are failing. On some hardware platforms, this message cannot be displayed due to hardware limitations. Possible values for the xxxxxxxxxxx variable depend on the V.25bis device with which the router is communicating.</td>
</tr>
<tr>
<td>Serial 0: No dialer string defined. Dialing cannot occur.</td>
<td>Packet is received that should cause a call to be placed. However, no dialer string is configured, so dialing cannot occur. This message usually indicates a configuration problem.</td>
</tr>
<tr>
<td>Serial 0: Attempting to dial xxxxxxxxxxxxxx</td>
<td>Packet has been received that passes the dial-on-demand access lists. That packet causes phone number xxxxxxxxxxx to be dialed.</td>
</tr>
<tr>
<td>Serial 0: Unable to dial xxxxxxxxxxxxxx</td>
<td>Phone call to xxxxxxxxxxx cannot be placed. This failure might be due to a lack of memory, full output queues, or other problems.</td>
</tr>
<tr>
<td>Serial 0: disconnecting call</td>
<td>Router hangs up a call.</td>
</tr>
</tbody>
</table>

Serial 0: idle timeout
Serial 0: re-enable timeout
Serial 0: wait for carrier timeout

One of these three messages is displayed when a dialer timer expires. These messages are mostly informational, but are useful for debugging a disconnected call or call failure.

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug decnet packet</td>
<td>Displays debugging information about the packets received on a dialer interface.</td>
</tr>
</tbody>
</table>
**debug dialer forwarding**

To display debugging information about the control plane at the home gateway (HGW) for Layer 2 Tunneling Protocol (L2TP) dialout, use the `debug dialer forwarding` command in privileged EXEC mode. The `no` form of this command disables debugging output.

```
debug dialer forwarding
no debug dialer forwarding
```

**Syntax Description**

This command has no keywords or arguments.

**Command Default**

This command is disabled by default.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2 T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use the `debug dialer forwarding` command to configure a virtual private dialout network (VPDN) on the HGW and a network access server (NAS) to dial from the HGW to the client.

An L2TP tunnel is created between the HGW and the NAS and the packets are forwarded transparently at the NAS.

**Examples**

The following is sample output from the `debug dialer forwarding` command for dialing from the HGW to the client.

```
Router# debug dialer forwarding
Dialer forwarding events debugging is on
Router# ping
Protocol [ip]:
Target IP address:1.1.1.3
Repeat count [5]:1
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 1.1.1.3, timeout is 2 seconds:
```

**Note**

DDR-FWD is `debug dialer forwarding` information. (DDR= dial-on-demand routing.)
router# debug dialer forwarding

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug dialer events</td>
<td>Displays debugging information about events on a dialer interface.</td>
</tr>
<tr>
<td>debug dialer packets</td>
<td>Displays debugging information about packets received on a dialer interface.</td>
</tr>
</tbody>
</table>
**debug dialer map**

To display debugging information about the creation and deletion of dynamic dialer maps, use the `debug dialer map` command in privileged EXEC mode. The `no` form of this command disables debugging output.

```
ddebug dialer map
no debug dialer map
```

**Syntax Description**

This command has no keywords or arguments.

**Command Default**

This command is disabled by default.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(5.1)</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use the `debug dialer map` command to track large-scale dialout (LSDO) and incoming calls that use dynamic dialer maps. This command shows the whole trace including when the map is created and removed.

If an interface is configured for dial-on-demand routing (DDR), and a map to a specified address does not exist, then a dynamic dialer map is created and when the call disconnects, the dialer map is removed.

**Note**

Do not configure a dialer string or a dialer map on the incoming interface.

**Examples**

In the following sample output from the `debug dialer map` command, a dialer map is created when an incoming call is connected and removed when that call is disconnected:

```
Router# debug dialer map
Dial on demand dynamic dialer maps debugging is on
Incoming call connected:

Router#
*Mar 22 12:19:15.597:%LINK-3-UPDOWN:Interface BRIO/0:1, changed state to up
*Mar 22 12:19:17.748:BR0/0:1 DDR:dialer_create_dynamic_map map created for 11.0.0.1
*Mar 22 12:19:18.734:%LINEPROTO-5-UPDOWN:Line protocol on Interface BRIO/0:1, changed state to up
*Mar 22 12:19:21.598:%ISDN-6-CONNECT:Interface BRIO/0:1 is now connected to unknown R2604
```
Incoming call disconnected:

Router#
*Mar 22 12:21:15.597:%ISDN-6-DISCONNECT:Interface BRI0/0:1 disconnected from R2604, call lasted 120 seconds
*Mar 22 12:21:15.645:%LINK-3-UPDOWN:Interface BRI0/0:1, changed state to down
*Mar 22 12:21:15.649:BR0/0:1 DDR:dialer_remove_dynamic_map map 11.0.0.1 removed
*Mar 22 12:21:16.647:%LINEPROTO-5-UPDOWN:Line protocol on Interface BRI0/0:1, changed state to down

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug dialer events</td>
<td>Displays debugging information about events on a dialer interface.</td>
</tr>
<tr>
<td>debug dialer packets</td>
<td>Displays debugging information about packets received on a dialer interface.</td>
</tr>
</tbody>
</table>
debug dialpeer

**Note**

Effective with release 12.3(8)T, the `debug dialpeer` command is replaced by the `debug voip dialpeer` command. See the `debug voip dialpeer` command for more information.

To view dial peer information, use the `debug dialpeer` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug dialpeer
no debug dialpeer
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Disabled

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(11)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(8)T</td>
<td>This command was replaced by the <code>debug voip dialpeer</code> command.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Disable console logging and use buffered logging before using the `debug dialpeer` command. Using the `debug dialpeer` command generates a large volume of debugging messages, which can affect router performance.

**Examples**

The following is sample output for the `debug dialpeer` command. The output shows the destination pattern configured on the matched dial-peer. Expanded string is the string after applying number translation to the original number. It shows that dial-peer 1311 was an incoming dial-peer match. It also shows that routing label was att1. It shows that dial-peer 5108888 and 111399 are an outgoing dial-peer match.

```
Router# debug dialpeer
```
00:22:28: Inside dpMatchCore:
00:22:28: destination pattn:5108880101 expanded string:5108880101
00:22:28: dpAssociateIncomingPeer_T:Matching peer with src route label att1 failed
00:22:28: Inside dpMatchCore:
00:22:28: destination pattn:5108880101 expanded string:5108880101
00:22:28:MatchNextPeer:Peer 1311 matched
00:22:28:dpMatchPeersMoreArg:Result=0 after MATCH_ORIGINATE
The table below describes the significant fields shown in the display.

**Table 75: debug dialpeer Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>destination pattn</td>
<td>Destination pattern configured on the dial peer.</td>
</tr>
<tr>
<td>expanded string</td>
<td>The string after applying number translation to the original number.</td>
</tr>
<tr>
<td>Match Dest. pattern; called</td>
<td>Indicates that dial-peer match is going to match destination pattern against the called number.</td>
</tr>
<tr>
<td>Matching route label</td>
<td>The trunk group label or carrier id that is used for matching a dial peer.</td>
</tr>
<tr>
<td>MatchNextPeer</td>
<td>Indicates the dial peer tag that matched.</td>
</tr>
<tr>
<td>Result</td>
<td>Indicates the result of dial peer matching algorithm:</td>
</tr>
<tr>
<td></td>
<td>0 = Successful 1 = More digits needed for a possible match -1 = No match (match failed) -2 = The digits matched, but the destination address could not be obtained</td>
</tr>
</tbody>
</table>

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>call-block (dial peer)</td>
<td>Enables blocking of incoming calls on the dial peer.</td>
</tr>
<tr>
<td>carrier-id (dial-peer)</td>
<td>Identifies the carrier handling the incoming call.</td>
</tr>
<tr>
<td>session target (ENUM)</td>
<td>Specifies the ENUM search table for the target session.</td>
</tr>
<tr>
<td>show dial-peer voice</td>
<td>Displays the configuration of the dial peer.</td>
</tr>
<tr>
<td>translation-profile (dial-peer)</td>
<td>Assigns a translation profile to the dial peer.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>trunkgroup (dial-peer)</td>
<td>Assigns a trunk group to the dial peer.</td>
</tr>
<tr>
<td>trunk-group-label (dial-peer)</td>
<td>Identifies the trunk group handling the incoming call.</td>
</tr>
</tbody>
</table>
To display information about the Diameter Protocol, use the `debug diameter` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug diameter [dcca| connection| error| packet| event| fsm| failover]
no debug diameter [dcca| connection| error| packet| event| fsm| failover]
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dcca</code></td>
<td>(Optional) Enables debugging for Diameter-Credit Control Accounting.</td>
</tr>
<tr>
<td><code>connection</code></td>
<td>(Optional) Enables debugging output for the connection between two Diameter nodes.</td>
</tr>
<tr>
<td><code>error</code></td>
<td>(Optional) Enables debugging output for Diameter errors.</td>
</tr>
<tr>
<td><code>packet</code></td>
<td>(Optional) Enables debugging output for Diameter data packets.</td>
</tr>
<tr>
<td><code>event</code></td>
<td>(Optional) Enables debugging output for Diameter events.</td>
</tr>
<tr>
<td><code>fsm</code></td>
<td>(Optional) Enables debugging output for the finite state machine.</td>
</tr>
<tr>
<td><code>failover</code></td>
<td>(Optional) Enables debugging output for Diameter redundancy.</td>
</tr>
</tbody>
</table>

### Command Modes

Privileged EXEC (#)

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(9)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>

### Usage Guidelines

Use this command to display information about any of the listed classes of information about the Diameter Protocol.
The following examples show output from the `debug diameter` command:

```
Router# debug diameter all
*May 9 17:58:14.832: Dia Base: Allocate the peer connection context 50F63888, handle C000000C
*May 9 17:58:14.832: Dia Base: (C000000C): Received peer configuration event
*May 9 17:58:14.832: Dia Peer FSM (50F63888): Input event START in state CLOSED
*May 9 17:58:14.832: Dia Peer FSM (50F63888): Event START, state CLOSED --> WAIT_CONN_ACK

*May 9 17:58:14.836: Dia Transport: socket 0 - connecting to 9.113.33.6 (3868)

*May 9 17:58:14.836: Dia Transport: socket 0 - connection in progress
*May 9 17:58:14.836: Dia Transport: socket 0 - local address 9.113.33.5 (49214)

*May 9 17:58:14.836: Dia Transport: socket 0 - resume socket write - nothing to write
*May 9 17:58:14.836: Dia Peer FSM (50F63888): Received peer connection event from transport
*May 9 17:58:14.836: Dia Peer FSM (50F63888): Event RCV_CONN_ACK in state WAIT_CONN_ACK

*May 9 17:58:14.836: Dia Base: Sending diameter message to peer "Unknown"
*May 9 17:58:14.836: DIAMETER: CER message, ver=1, len=120, app=0, [2328318322/2328318322]

*May 9 17:58:14.836: DIAMETER: Origin-host-name [264] "host" (M)
*May 9 17:58:14.836: DIAMETER: Origin-Realm [296] "cisco" (M)
*May 9 17:58:14.836: DIAMETER: Product-name [269] "C7200-G8IS-M" (M)
*May 9 17:58:14.836: DIAMETER: Firmware-Revision [267] 150D0B710: 01000000 00000000 ...x........
*May 9 17:58:14.836: DIAMETER: Firmware-Revision [267] 150D0B720: 8AC75172 8AC75172 00000108 4000000C .GQr.GQr....@...
*May 9 17:58:14.836: DIAMETER: Firmware-Revision [267] 150D0B730: 686F7374 00000128 4000000D 63697363 host...@...cisco
*May 9 17:58:14.836: DIAMETER: Firmware-Revision [267] 150D0B750: 21050000 0000010A 4000000C 00000009 !......@......q
*May 9 17:58:14.836: DIAMETER: Firmware-Revision [267] 150D0B770: 49532D44 00000128 4000000C 00000004 IS-M....@......q
*May 9 17:58:14.836: DIAMETER: Firmware-Revision [267] 150D0B780: 0000010B 0000000C 00000001 00 .............

*May 9 17:58:14.840: Dia Base: Request message hash ctx created for [2328318322/2328318322]

*May 9 17:58:14.840: Diams: Starting CER timer

*May 9 17:58:14.840: Dia Transport write message event
*May 9 17:58:14.836: Dia Transport: Dia Transport: Dia Transport write message event
*May 9 17:58:14.836: Dia Transport: Dia Transport: Dia Transport write message event
*May 9 17:58:14.836: Dia Transport: Dia Transport write message event
*May 9 17:58:14.836: Dia Base: Diameter message received from the peer "Unknown"

*May 9 17:58:14.840: DIAMETER: CEA message, ver=1, len=192, app=0, [2328318322/2328318322]

*May 9 17:58:14.840: DIAMETER: Origin-host-name [264] "diameter2.cisco.com" (M)
*May 9 17:58:14.840: DIAMETER: Host-IP-address [257] 10.77.154.80 (M)
*May 9 17:58:14.840: DIAMETER: Product-name [269] "Diameter-Server" (M)
```

65940780: 01000000 00000101 00000000 ...@........
65940790: 8AC75172 8AC75172 0000010C 4000000C .GQr.GQr....@...
659407A0: 000007D1 00000108 4000001B 6469616D ...Q....@...diam
659407B0: 65746572 322E6369 73636F2E 636F6D00 eteter2.cisco.com.
659407C0: 00000128 40000011 63697363 6F2E636F ...(@...cisco.co
659407D0: 65757465 322E6369 73636F2E 636F6D00 eter2.cisco.com.
659407E0: 0000010A 4000000E 00000009 .P......@.......
659407F0: 0000010D 00000017 4469616D 65746572 ........Diameter
65940800: 65746572 322E6369 73636F2E 636F6D00 eteter2.cisco.com.
65940810: 00000128 40000011 63697363 6F2E636F ...(@...cisco.co
65940820: 65757465 322E6369 73636F2E 636F6D00 eter2.cisco.com.
65940830: 00000109 4000000C 00000009 00000109 ...@...........
65940840: 00000004 00 ....

May 9 17:58:14.840: Dia Base: Request message hash ctx removed for [2328318322/2328318322]

May 9 17:58:14.840: Di Base: Sending diameter message to peer "diameter2.cisco.com"

May 9 17:59:14.840: Dia Peer FSM (50F63888): input event RCV_CEA in state WAIT_CEA

May 9 17:59:14.840: Dia Peer FSM (50F63888): Starting Watchdog timer

%DIABASE-4-DIA_PEER_UP: Diameter peer 9.113.33.6 port 3868 TCP UP

May 9 17:59:14.840: Dia Peer FSM (50F63888): event RCV_CEA, state WAIT_CEA-->OPEN

Examples

May 9 17:59:14.840: Dia Peer FSM (50F63888): input event TIMEOUT in state OPEN

May 9 17:59:14.840: Dia Peer FSM (50F63888): Sending diameter message to peer "diameter2.cisco.com"

May 9 17:59:14.840: DIAMETER: DWR message, ver=1, len=48, app=0,


50D0B710: 01000030 80000118 00000000 ...0........
50D0B720: 8AC75173 8AC75173 0000010C 4000000C .GQs.GQs....@...
50D0B730: 000007D1 00000108 4000001B 6469616D ...Q....@...diam
50D0B740: 65746572 322E6369 73636F2E 636F6D00 eteter2.cisco.com.
50D0B7A0: 00000128 40000011 63697363 6F2E636F ...(@...cisco.co

May 9 17:59:14.840: Dia Base: Request message hash ctx created for [2328318323/2328318323]

May 9 17:59:14.840: Dia Peer FSM (50F63888): Starting Watchdog timer, [60] left for next timeout

May 9 17:59:14.840: Dia Peer FSM (50F63888): event TIMEOUT, state OPEN-->OPEN

May 9 17:59:14.840: Dia Peer FSM (50F63888): input event TIMEOUT in state OPEN

May 9 17:59:14.840: Dia Peer FSM (50F63888): Sending diameter message to peer "diameter2.cisco.com"

May 9 17:59:14.840: DIAMETER: DWR message, ver=1, len=48, app=0,


50D0B710: 01000030 80000118 00000000 ...0........
50D0B720: 8AC75173 8AC75173 0000010C 4000000C .GQs.GQs....@...
50D0B730: 000007D1 00000108 4000001B 6469616D ...Q....@...diam
50D0B740: 65746572 322E6369 73636F2E 636F6D00 eteter2.cisco.com.
50D0B7A0: 00000128 40000011 63697363 6F2E636F ...(@...cisco.co

Cisco IOS Debug Command Reference - Commands A through D
May 9 17:59:14.840: Dia Base: Request message hash ctx removed for [2328318323/2328318323]
May 9 17:59:14.840: Dia Peer FSM (50F63888): input event RCV_DWA in state OPEN
May 9 17:59:14.840: Dia Peer FSM (50F63888): Starting Watchdog timer
May 9 17:59:14.840: Dia Peer FSM (50F63888): event RCV_DWA, state OPEN->OPEN

May 9 18:07:18.472: Dia Transport: socket 0 READ event: UP->CLOSE due to bytes read = 0
May 9 18:07:18.472: Dia Base: (8600000E): Received peer disconnection event from transport
May 9 18:07:18.472: %DIABASE-4-DIA_PEER_DOWN: Diameter peer 9.113.33.6 port 3868 TCP DOWN
May 9 18:07:18.472: Dia Peer FSM (2068FF44): input event PEER_DISC in state OPEN
May 9 18:07:18.472: Dia Peer FSM (2068FF44): Starting Reconnect timer
May 9 18:07:18.472: Dia Peer FSM (2068FF44): event PEER_DISC, state OPEN-->CLOSED
May 9 18:07:48.472: Dia Peer FSM (2068FF44): input event PEER_DISC in state WAIT_CONN_ACK
May 9 18:07:48.472: Dia Transport: socket 0 - connecting to 9.113.33.6 (3868)
May 9 18:07:48.472: Dia Transport: socket 0 - connection in progress
May 9 18:07:48.472: Dia Transport: socket 0 - local address 9.113.33.5 (61122)
May 9 18:07:48.472: Dia Base: (8600000E): Received peer disconnection event from transport
May 9 18:07:48.472: %DIABASE-4-DIA_PEER_DOWN: Diameter peer 9.113.33.6 port 3868 TCP DOWN
May 9 18:07:48.472: Dia Peer FSM (2068FF44): input event PEER_DISC in state OPEN
May 9 18:07:48.472: Dia Peer FSM (2068FF44): Starting Reconnect timer
May 9 18:07:48.472: Dia Peer FSM (2068FF44): event PEER_DISC, state WAIT_CONN_ACK-->CLOSED

Examples

Ginger(config)#no diameter peer watch
Ginger(config)#

May 9 18:05:02.812: Dia Base: Peer unconfigured, start peer disconnection
May 9 18:05:02.812: Dia Base: (C000000C): Received peer unconfiguration event
May 9 18:05:02.812: Dia Peer FSM (50F63888): input event STOP in state OPEN
May 9 18:05:02.812: Dia Base: Sending diameter message to peer "diameter2.cisco.com"
May 9 18:05:02.812: DIAMETER: DPR message, ver=1, len=60, app=0, [2328318329/2328318329]
May 9 18:05:02.812: "host" (M)
May 9 18:05:02.812: DIAMETER: Origin-host-name [264] "host" (M)
May 9 18:05:02.812: DIAMETER: Origin-Realm [296] "cisco" (M)
653D1810: 0100003C 8000011A ...<....
653D1820: 00000000 8AC75179 8AC75179 00000010 .....GQy.GQy....
653D1830: 4000000C 686F7374 00000128 4000000D @...host...(@...
653D1840: 63697363 6F000000 00000111 4000000C cisco.......@...
653D1850: 00000002 00 ......
May 9 18:05:02.812: Dia Base: Request message hash ctx created for [2328318329/2328318329]
May 9 18:05:02.816: Dia Peer FSM (50F63888): Starting DPR timer
*May 9 18:05:02.816: Dia Peer FSM (50F63888): event STOP, state OPEN-->CLOSING
*May 9 18:05:02.816: Dia Transport: Dia Transport write message event
*May 9 18:05:02.816: Dia Transport: socket 0 - complete msg sent
*May 9 18:05:02.816: Dia Transport: socket 0 - complete read of 20 bytes
*May 9 18:05:02.816: Dia Transport: complete header read from socket 0
*May 9 18:05:02.816: Dia Transport: read msg (60) bytes from socket 0
*May 9 18:05:02.816: Dia Transport: socket 0 - complete read of 60 bytes
*May 9 18:05:02.816: Dia Base: Diameter message received from the peer "diameter2.cisco.com"
*May 9 18:05:02.816: DIAMETER: DPA message, ver=1, len=80, app=0, [2328318329/2328318329]
*May 9 18:05:02.816: DIAMETER: Result-code [268] 2001 (M)
*May 9 18:05:02.816: DIAMETER: Origin-host-name [264] "diameter2.cisco.com" (M)
*May 9 18:05:02.816: DIAMETER: Origin-Realm [296] "cisco.com" (M)

65913A20: 01000050 ...P
65913A30: 0000011A 00000000 8AC75179 8AC75179 ........GQy.GQy
65913A40: 0000010C 4000000C 00000007D1 000000108 ........Q....
65913A50: 4000001B 6469616D 65746572 322E6369 73636F2E 636F6D00 00000128 40000011 sco.com....([^)
65913A60: 63697363 6F2E636F 6D000000 00 cisco.com....

*May 9 18:05:02.816: Dia Base: Request message hash ctx removed for [2328318329/2328318329]
*May 9 18:05:02.816: Dia Base: (C000000C): Received msg event from message i/o
*May 9 18:05:02.816: Dia Peer FSM (50F63888): input event RCV_DPA in state CLOSING
*May 9 18:05:02.816: Dia Base: (C000000C): Free the peer connection context 50F63888

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>show diameter peer</td>
<td>Displays Diameter peer configuration information.</td>
</tr>
</tbody>
</table>
## debug dlsw

To enable debugging of data-link switching plus (DLSw+), use the `debug dlsw` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug dlsw border-peers [interface interface | ip address ip-address] core [flow-control messages | state | xid] [circuit-number] local-circuit circuit-number peers [interface interface | ip address ip-address] ip address ip-address [fast-errors | fast-paks | fst-seq | udp] reachability [error | verbose] [sna | netbios]

no debug dlsw border-peers [interface interface | ip address ip-address] core [flow-control messages | state | xid] [circuit-number] local-circuit circuit-number peers [interface interface | ip address ip-address] ip address ip-address [fast-errors | fast-paks | fst-seq | udp] reachability [error | verbose] [sna | netbios]
```

### Syntax Description

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>border-peers</code></td>
<td>(Optional) Enables debugging output for border peer events.</td>
</tr>
<tr>
<td><code>interface interface</code></td>
<td>(Optional) Specifies a remote peer to debug by a direct interface.</td>
</tr>
<tr>
<td><code>ip address ip-address</code></td>
<td>(Optional) Specifies a remote peer to debug by its IP address.</td>
</tr>
<tr>
<td><code>core</code></td>
<td>(Optional) Enables debugging output for DLSw core events.</td>
</tr>
<tr>
<td><code>flow-control</code></td>
<td>(Optional) Enables debugging output for congestion in the WAN or at the remote end station.</td>
</tr>
<tr>
<td><code>messages</code></td>
<td>(Optional) Enables debugging output of core messages--specific packets received by DLSw either from one of its peers or from a local medium via the Cisco link services interface.</td>
</tr>
<tr>
<td><code>state</code></td>
<td>(Optional) Enables debugging output for state changes on the circuit.</td>
</tr>
<tr>
<td><code>xid</code></td>
<td>(Optional) Enables debugging output for the exchange identification state machine.</td>
</tr>
<tr>
<td><code>circuit-number</code></td>
<td>(Optional) Specifies the circuit for which you want core debugging output to reduce the output.</td>
</tr>
<tr>
<td><code>local-circuit circuit-number</code></td>
<td>(Optional) Enables debugging output for circuits performing local conversion. Local conversion occurs when both the input and output data-link connections are on the same local peer and no remote peer exists.</td>
</tr>
<tr>
<td><code>peers</code></td>
<td>(Optional) Enables debugging output for peer events.</td>
</tr>
</tbody>
</table>
### fast-errors
(Optional) Debugs errors for fast-switched packets.

### fast-paks
(Optional) Debugs fast-switched packets.

### fst-seq
(Optional) Debugs Fast-Sequenced Transport (FST) sequence numbers on fast switched packets.

### udp
(Optional) Debugs User Datagram Protocol (UDP) packets.

### reachability
(Optional) Enables debugging output for reachability events (explorer traffic). If no options are specified, event-level information is displayed for all protocols.

### error | verbose
(Optional) Specifies how much reachability information you want displayed. The verbose keyword displays everything, including errors and events. The error keyword displays error information only. If no option is specified, event-level information is displayed.

### sna | netbios
(Optional) Specifies that reachability information be displayed for only Systems Network Architecture (SNA) or Network Basic Input/Output System (NetBIOS) protocols. If no option is specified, information for all protocols is displayed.

---

**Usage Guidelines**

When you specify no optional keywords, the debug **dlsw** command enables all available DLSW debugging output.

Normally you need to use only the **error** or **verbose** option of the **debug dlsw reachability** command to help identify problems. The **error** option is recommended for use by customers and provides a subset of the messages from the normal event-level debugging. The **verbose** option provides a very detailed view of events, and is typically used only by service personnel.

To reduce the amount of debug information displayed, use the **sna** or **netbios** option with the **debug dlsw reachability** command if you know that you have an SNA or NetBIOS problem.

The DLSw core is the engine that is responsible for the establishment and maintenance of remote circuits. If possible, specifying the index of the specific circuit you want to debug reduces the amount of output displayed. However, if you want to watch a circuit initially come up, do not use the **circuit-number** option with the **core** keyword.

The **coreflow-control** option provides information about congestion in the WAN or at the remote end station. In these cases, DLSw sends Receiver Not Ready (RNR) frames on its local circuits, slowing data traffic on established sessions and giving the congestion an opportunity to clear.

The **corestate** option allows you to see when the circuit changes state. This capability is especially useful for determining why a session cannot be established or why a session is being disconnected.

The **coreXID** option allows you to track the exchange identification (XID)-state machine. The router tracks XID commands and responses used in negotiations between end stations before establishing a session.
The following examples show and explain some of the typical DLsw debugging messages you might see when using the `debug dlsw` command.

The following example enables UDP packet debugging for a specific remote peer:

```
Router# debug dlsw peers ip-address 1.1.1.6 udp
```

The following message is sample output from the `debug dlsw border-peers` command:

```
*Mar 10 17:39:56: CSM: delete group mac cache for group 0
*Mar 10 17:39:56: CSM: delete group name cache for group 0
*Mar 10 17:40:19: CSM: update group cache for mac 0000.3072.1070, group 10
*Mar 10 17:40:22: DLsw: send_to_group_members(): copy to peer 10.19.32.5
```

The following message is from a router that initiated a TCP connection:

```
DLsw: START-TPFSM (peer 10.3.8.7(2065)): event:ADMIN-OPEN CONNECTION state:DISCONN
DLsw: dtp_action_a() attempting to connect peer 10.3.8.7(2065)
DLsw: End Open Callback 10.3.8.7(2065) -> 11002
DLsw: START-TPFSM (peer 10.3.8.7(2065)): event:TCP-WR PIPE OPENED state:WAIT_WR
DLsw: dtp_action_f() start read open timer for peer 10.3.8.7(2065)
DLsw: START-TPFSM (peer 10.3.8.7(2065)): state:WAIT_WR->WAIT_RD
DLsw: dtp_action_g() start read open timer for peer 10.3.8.7(2065)
DLsw: dtp_action_i() write pipe opened for peer 10.3.8.7(2065)
DLsw: CapExId Msg sent to peer 10.3.8.7(2065)
DLsw: dtp_action_j() cap msg rcvd from peer 10.3.8.7(2065)
DLsw: Recv CapExId Msg from peer 10.3.8.7(2065)
DLsw: Pos CapExResp sent to peer 10.3.8.7(2065)
DLsw: dtp_action_k() cap xchged for peer 10.3.8.7(2065)
DLsw: Closing read pipe tcp connection for peer 10.3.8.7(2065)
DLsw: END-TPFSM (peer 10.3.8.7(2065)): state:CONNECT->CONNECT
```

The following message is from a router that received a TCP connection:

```
DLsw: passive open 10.10.10.4(11002) -> 2065
DLsw: START-TPFSM (peer 10.10.10.4(2065)): event:TCP-PEER PIPE OPENED state:DISCONN
DLsw: dtp_action_c() opening write pipe for peer 10.10.10.4(2065)
DLsw: END-TPFSM (peer 10.10.10.4(2065)): state:DISCONN->WWR_RDOP
DLsw: Async Open Callback 10.10.10.4(2065) -> 11004
DLsw: START-TPFSM (peer 10.10.10.4(2065)): event:TCP-WR PIPE OPENED state:WAIT_WR
DLsw: dtp_action_f() start read open timer for peer 10.10.10.4(2065)
DLsw: START-TPFSM (peer 10.10.10.4(2065)): state:WAIT_WR->WAIT_RD
DLsw: dtp_action_g() start read open timer for peer 10.10.10.4(2065)
DLsw: dtp_action_i() write pipe opened for peer 10.10.10.4(2065)
DLsw: CapExId Msg sent to peer 10.10.10.4(2065)
DLsw: dtp_action_j() cap msg rcvd from peer 10.10.10.4(2065)
DLsw: Recv CapExId Msg from peer 10.10.10.4(2065)
DLsw: Pos CapExResp sent to peer 10.10.10.4(2065)
DLsw: dtp_action_k() cap xchged for peer 10.10.10.4(2065)
DLsw: Closing read pipe tcp connection for peer 10.10.10.4(2065)
DLsw: END-TPFSM (peer 10.10.10.4(2065)): state:CONNECT->CONNECT
```

The following message is from a router that initiated a TCP connection:

```
DLsw: passive open 10.3.8.7(11004) -> 2065
DLsw: START-TPFSM (peer 10.3.8.7(2065)): event:TCP-PEER PIPE OPENED state:DISCONN
DLsw: dtp_action_c() opening write pipe for peer 10.3.8.7(2065)
DLsw: END-TPFSM (peer 10.3.8.7(2065)): state:DISCONN->WWR_RDOP
DLsw: Async Open Callback 10.3.8.7(2065) -> 11004
DLsw: START-TPFSM (peer 10.3.8.7(2065)): event:TCP-WR PIPE OPENED state:WAIT_WR
DLsw: dtp_action_f() start read open timer for peer 10.3.8.7(2065)
DLsw: START-TPFSM (peer 10.3.8.7(2065)): state:WAIT_WR->WAIT_RD
DLsw: dtp_action_g() start read open timer for peer 10.3.8.7(2065)
DLsw: dtp_action_i() write pipe opened for peer 10.3.8.7(2065)
DLsw: CapExId Msg sent to peer 10.3.8.7(2065)
DLsw: dtp_action_j() cap msg rcvd from peer 10.3.8.7(2065)
DLsw: Recv CapExId Msg from peer 10.3.8.7(2065)
DLsw: Pos CapExResp sent to peer 10.3.8.7(2065)
DLsw: dtp_action_k() cap xchged for peer 10.3.8.7(2065)
DLsw: Closing read pipe tcp connection for peer 10.3.8.7(2065)
DLsw: END-TPFSM (peer 10.3.8.7(2065)): state:CONNECT->CONNECT
```

The following message is from a router that received a TCP connection:

```
DLsw: passive open 10.10.10.4(11002) -> 2065
DLsw: START-TPFSM (peer 10.10.10.4(2065)): event:TCP-RD PIPE OPENED state:DISCONN
DLsw: dtp_action_c() opening write pipe for peer 10.10.10.4(2065)
DLsw: END-TPFSM (peer 10.10.10.4(2065)): state:DISCONN->CONNECT
DLsw: Async Open Callback 10.10.10.4(2065) -> 11004
DLsw: START-TPFSM (peer 10.10.10.4(2065)): event:TCP-WR PIPE OPENED state:WAIT_WR
DLsw: dtp_action_f() start read open timer for peer 10.10.10.4(2065)
DLsw: START-TPFSM (peer 10.10.10.4(2065)): state:WAIT_WR->WAIT_RD
DLsw: dtp_action_g() start read open timer for peer 10.10.10.4(2065)
DLsw: dtp_action_i() write pipe opened for peer 10.10.10.4(2065)
DLsw: CapExId Msg sent to peer 10.10.10.4(2065)
DLsw: dtp_action_j() cap msg rcvd from peer 10.10.10.4(2065)
DLsw: Recv CapExId Msg from peer 10.10.10.4(2065)
DLsw: Pos CapExResp sent to peer 10.10.10.4(2065)
DLsw: dtp_action_k() cap xchged for peer 10.10.10.4(2065)
DLsw: Closing read pipe tcp connection for peer 10.10.10.4(2065)
DLsw: END-TPFSM (peer 10.10.10.4(2065)): state:CONNECT->CONNECT
```

The following message is from a router that initiated a TCP connection:

```
DLsw: passive open 10.10.10.4(11004) -> 2065
DLsw: START-TPFSM (peer 10.10.10.4(2065)): event:TCP-PEER PIPE OPENED state:DISCONN
DLsw: dtp_action_c() opening write pipe for peer 10.10.10.4(2065)
DLsw: END-TPFSM (peer 10.10.10.4(2065)): state:DISCONN->CONNECT
DLsw: Async Open Callback 10.10.10.4(2065) -> 11004
DLsw: START-TPFSM (peer 10.10.10.4(2065)): event:TCP-WR PIPE OPENED state:WAIT_WR
DLsw: dtp_action_f() start read open timer for peer 10.10.10.4(2065)
DLsw: START-TPFSM (peer 10.10.10.4(2065)): state:WAIT_WR->WAIT_RD
DLsw: dtp_action_g() start read open timer for peer 10.10.10.4(2065)
DLsw: dtp_action_i() write pipe opened for peer 10.10.10.4(2065)
DLsw: CapExId Msg sent to peer 10.10.10.4(2065)
DLsw: dtp_action_j() cap msg rcvd from peer 10.10.10.4(2065)
DLsw: Recv CapExId Msg from peer 10.10.10.4(2065)
```
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:WAIT_CAP->WAIT_CAP
DLSw: Processing delayed event:SSP-CAP EXCHANGED - prev state:WAIT_CAP
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:SSP-CAP EXCHANGED state:WAIT_CAP
DLSw: dtp_action_k() cap xchged for peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:WAIT_CAP->PCONN_WT
DLSw: dlsw_tcpd_fini() for peer 10.10.10.4(2065)
DLSw: dlsw_tcpd_fini() closing write pipe for peer 10.10.10.4
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:TCP-CLOSE WR PIPE state:PCONN_WT
DLSw: dtp_action_l() close write pipe for peer 10.10.10.4(2065)
DLSw: closing write pipe tcp connection for peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:PCONN_WT->PCONN_WT
DLSw: Processing delayed event:TCP-PEER CONNECTED - prev state:PCONN_WT
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:TCP-PEER CONNECTED state:PCONN_WT
DLSw: dtp_action_m() peer connected for peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:PCONN_WT->CONNECT
DLSw-LLC2: Sending enable port ; port no : 0
DLSw: Peer Received : CLSI Msg : ENABLE.Req dlen: 20
DLSw: Peer Received : CLSI Msg : ENABLE.Cfm CLS_OK dlen: 20
DLSw: Sending activate sap for Serial1 - port_id = 887C3C
DLSw: Peer Received : CLSI Msg : ACTIVATE_SAP.Req dlen: 60
DLSw: Peer Received : CLSI Msg : ACTIVATE_SAP.Cfm CLS_OK dlen: 60
DLSw: Got ActSapcnf back for Serial1 - port_id = 8978204, port_type = 7, psap_id = 0
DLSw: Peer Received : CLSI Msg : REQUEST OPNSAP Cnf dlen: 106
DLSw: Peer Received : CLSI Msg : REQUEST OPNSAP.Req dlen: 106
dlc2p_action_a() attempting to connect peer on interface Serial1
DLSw: Peer Received : CLSI Msg : REQUEST OPNSAP.Cfm CLS_OK dlen: 106
DLSw: END-LLC2PFSM (peer on interface Serial1): state:DICOMM->ROS_SENT
DLSw: Peer Received : CLSI Msg : REQUEST OPNSAP.Cnf CLS_OK dlen: 106
DLSw: END-LLC2PFSM (peer on interface Serial1): state:DICOMM->ROS_SENT
DLSw: Peer Received : CLSI Msg : REQUEST OPNSAP.Req dlen: 106
DLSw: Peer Received : CLSI Msg : REQUEST OPNSAP.Cfm CLS_OK dlen: 106
DLSw: END-LLC2PFSM (peer on interface Serial1): state:DICOMM->ROS_SENT
The following message is from a router that initiated an FST connection:
DLSw: START-FSTPFSM (peer 10.10.10.4(0)): event:ADMIN-OPEN CONNECTION state:DISCONN
DLSw: dftp_action_a() attempting to connect peer 10.10.10.4(0)
DLSw: Connection opened for peer 10.10.10.4(0)
DLSw: CapExId Msg sent to peer 10.10.10.4(0)
DLSw: END-FSTPFSM (peer 10.10.10.4(0)): state:DISCONN->WAIT_CAP
DLSw: START-FSTPFSM (peer 10.10.10.4(0)): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dftp_action_e() cap msg rcvd for peer 10.10.10.4(0)
DLSw: Recv CapExPosRsp Msg from peer 10.10.10.4(0)
DLSw: END-FSTPFSM (peer 10.10.10.4(0)): state:WAIT_CAP->WAIT_CAP
DLSw: START-FSTPFSM (peer 10.10.10.4(0)): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dftp_action_e() cap msg rcvd for peer 10.10.10.4(0)
DLSw: Recv CapExId Msg from peer 10.10.10.4(0)
DLSw: Pos CapExResp sent to peer 10.10.10.4(0)
DLSw: END-FSTPFSM (peer 10.10.10.4(0)): state:WAIT_CAP->CONNECT
The following message is from a router that received an FST connection:
DLSw: START-FSTPFSM (peer 10.10.10.4(0)): event:ADMIN-OPEN CONNECTION state:DISCONN
DLSw: dftp_action_a() attempting to connect peer 10.10.10.4(0)
DLSw: Connection opened for peer 10.10.10.4(0)
DLSw: CapExId Msg sent to peer 10.10.10.4(0)
DLSw: END-FSTPFSM (peer 10.10.10.4(0)): state:DISCONN->WAIT_CAP
DLSw: START-FSTPFSM (peer 10.10.10.4(0)): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dftp_action_e() cap msg rcvd for peer 10.10.10.4(0)
DLSw: Recv CapExPosRsp Msg from peer 10.10.10.4(0)
DLSw: END-FSTPFSM (peer 10.10.10.4(0)): state:WAIT_CAP->WAIT_CAP
DLSw: Processing delayed event:SSP-CAP EXCHANGED - prev state:WAIT_CAP
DLSw: START-FSTPFSM (peer 10.10.10.4(0)): event:SSP-CAP EXCHANGED state:WAIT_CAP
DLSw: dftp_action_f() cap xchged for peer 10.10.10.4(0)
DLSw: END-FSTPFSM (peer 10.10.10.4(0)): state:WAIT_CAP->CONNECT
The following message is from a router that initiated an LLC2 connection:
DLSw-LLC2: Sending enable port ; port no : 0
PEER-DISP Sent : CLSI Msg : ENABLE.Req dlen: 20
DLSw: Peer Received : CLSI Msg : ENABLE.Cfm CLS_OK dlen: 20
DLSw-LLC2: Sending activate sap for Serial1 - port_id = 887C3C
port_type = 7 dgra(UsapID) = 952458
PEER-DISP Sent : CLSI Msg : ACTIVATE_SAP.Req dlen: 60
DLSw: Peer Received : CLSI Msg : ACTIVATE_SAP.Cfm CLS_OK dlen: 60
DLSw: Peer Received : CLSI Msg : ACTIVATE_SAP.Req dlen: 60
DLSw: Peer Received : CLSI Msg : ACTIVATE_SAP.Cfm CLS_OK dlen: 60
DLSw: Got ActSapcnf back for Serial1 - port_id = 8978204, port_type = 7, psap_id = 0
DLSw: Peer Received : CLSI Msg : REQUEST OPNSAP Cnf dlen: 106
DLSw: Peer Received : CLSI Msg : REQUEST OPNSAP.Req dlen: 106
dlc2p_action_a() attempting to connect peer on interface Serial1
DLSw: Peer Received : CLSI Msg : REQUEST OPNSAP.Cfm CLS_OK dlen: 106
DLSw: END-LLC2PFSM (peer on interface Serial1): state:DICOMM->ROS_SENT
DLSw: Peer Received : CLSI Msg : REQUEST OPNSAP.Cnf CLS_OK dlen: 106
DLSw: END-LLC2PFSM (peer on interface Serial1): state:DICOMM->ROS_SENT
DLSw: Peer Received : CLSI Msg : REQUEST OPNSAP.Req dlen: 106
DLSw: Peer Received : CLSI Msg : REQUEST OPNSAP.Cfm CLS_OK dlen: 106
DLSw: END-LLC2PFSM (peer on interface Serial1): state:DICOMM->ROS_SENT
The following message is from a router that initiated an LLC2 connection:
The following message is from a router that received a Logical Link Control, type 2 (LLC2) connection:

```
DLSw: dllc2p_action_c()  
PEER-DISP Sent : CLSI Msg : CONNECT.Req dlen: 16  
DLSw: END-LLC2FFSM (peer on interface Serial1): state:ROS_SENT->CON_PEND  
DLSw: dllc2p_action_e() send capabilities to peer on interface Serial1  
PEER-DISP Sent : CLSI Msg : SIGNAL_STN.Req dlen: 8  
PEER-DISP Sent : CLSI Msg : DATA.Req dlen: 418  
DLSw: CapExId Msg sent to peer on interface Serial1  
DLSw: END-LLC2FFSM (peer on interface Serial1): state:CON_PEND->WAIT_CAP  
DLSw: Peer Received : CLSI Msg : DATA.Ind dlen: 418  
DLSw: dllc2p_action_k() cap msg rcvd for peer on interface Serial1  
DLSw: Peer Received : CLSI Msg : DATA.Ind dlen: 96  
DLSw: dllc2p_action_l() cap xchgd for peer on interface Serial1  
DLSw: END-LLC2FFSM (peer on interface Serial1): state:WAIT_CAP->CONNECT  
```

The following message is from a router that received a Logical Link Control, type 2 (LLC2) connection:

```
DLSw-LLC2: Sending enable port ; port no : 0  
PEER-DISP Sent : CLSI Msg : ENABLE.Req dlen: 20  
DLSw: Peer Received : CLSI Msg : ENABLE.Cfm CLS_OK dlen: 20  
DLSw-LLC2 : Sending activate sap for Serial0 - port_id = 887C3C  
port_type = 7 dgra(UsapID) = 93AB34  
PEER-DISP Sent : CLSI Msg : ACTIVATE_SAP.Req dlen: 60  
DLSw: Peer Received : CLSI Msg : ACTIVATE_SAP.Cfm CLS_OK dlen: 60  
DLSw: Peer Received : CLSI Msg : CONNECT_STN.Ind dlen: 39  
DLSw: Peer Received : CLSI Msg : CONNECTED.Ind dlen: 8  
DLSw: Peer Received : CLSI Msg : DATA.Ind dlen: 418  
DLSw: dllc2p_action_h() send capabilities to peer on interface Serial10  
PEER-DISP Sent : CLSI Msg : DATA.Req dlen: 96  
DLSw: Peer Received : CLSI Msg : DATA.Req dlen: 106  
DLSw: dllc2p_action_k() cap msg rcvd for peer on interface Serial10  
DLSw: dllc2p_action_l() cap xchgd for peer on interface Serial10  
DLSw: END-LLC2FFSM (peer on interface Serial10): state:WAIT_CAP->CONNECT  ```
The following messages occur when a CUR_ex (CANUREACH explorer) frame is received from other peers, and the peer statements or the promiscuous keyword have not been enabled so that the router is not configured correctly:

```
22:42:44: DLSw: Not promiscuous - Rej conn from 172.20.96.1(2065)
```

In the following messages, the router sends a keepalive message every 30 seconds to keep the peer connected. If three keepalive messages are missed, the peer is torn down. These messages are displayed only if keepalives are enabled (by default, keepalives are disabled):

```
22:44:03: DLSw: Keepalive Request sent to peer 172.20.98.1(2065) (168243148)
22:44:03: DLSw: Keepalive Response from peer 172.20.98.1(2065) (168243176)
22:44:34: DLSw: Keepalive Request sent to peer 172.20.98.1(2065) (168274148)
22:44:34: DLSw: Keepalive Response from peer 172.20.98.1(2065) (168274172)
```

The following peer debugging messages indicate that the local peer is disconnecting from the specified remote peer because of missed peer keepalives:

```
0:03:24: DLSw: keepalive failure for peer on interface Serial0
0:03:24: DLSw: action_d(): for peer on interface Serial0
0:03:24: DLSW: DIRECT aborting connection for peer on interface Serial0, old state CONNECT, new state DISCONN
```

The following peer debugging messages result from an attempt to connect to an IP address that does not have DLSw enabled. The local router attempts to connect in 30-second intervals:

```
23:13:22: action_a() attempting to connect peer 172.20.100.1(2065)
23:13:52: DLSw: peer on interface Serial0, old state DISCONN, new state DISCONN
```

The following peer debugging messages that indicates a remote peer statement is missing on the router (address 172.20.100.1) to which the connection attempt is sent:

```
23:14:52: DLSw: action_a() attempting to connect peer 172.20.100.1(2065)
23:14:52: DLSw: action_a(): Write pipe opened for peer 172.20.100.1(2065)
23:14:52: DLSw: peer 172.20.100.1(2065), old state DISCONN, new state WAIT_RD
23:14:52: DLSw: dlsw_tcpd_fini() closing connection for peer 172.20.100.1(2065)
23:14:52: DLSw: aborting tcp connection for peer 172.20.100.1(2065)
23:14:52: DLSw: peer 172.20.100.1(2065), old state WAIT_RD, new state DISCONN
```

The following messages show a peer connection opening with no errors or abnormal events:

```
23:16:37: action_a() attempting to connect peer 172.20.100.1(2065)
23:16:37: DLSw: action_a(): Write pipe opened for peer 172.20.100.1(2065)
23:16:37: DLSw: peer 172.20.100.1(2065), old state DISCONN, new state WAIT_RD
23:16:37: DLSw: passive open 172.20.100.1(2065)
23:16:37: DLSw: action_c(): for peer 172.20.100.1(2065)
23:16:37: DLSw: action_c(): for peer 172.20.100.1(2065)
23:16:37: DLSw: dlsw_tcpd_fini() closing connection for peer 172.20.100.1(2065)
23:16:37: DLSw: peer 172.20.100.1(2065), old state WAIT_RD, new state DISCONN
```

The following messages indicate the local peer is disconnecting from the specified remote peer because of missed peer keepalives:
The following two messages show that an information frame is passing through the router:

DLSw: dlsw_tr2fct() lmac:c000.a400.0000 rmac:0800.5a29.75fe ls:5 rs:4 i:34
DLSw: dlsw_tr2fct() lmac:c000.a400.0000 rmac:0800.5a29.75fe ls:4 rs:4 i:34

Examples

The messages in this section are based on the following criteria:

- Reachability is stored in cache. DLSw+ maintains two reachability caches: one for MAC addresses and one for NetBIOS names. Depending on how long entries have been in the cache, they are either fresh or stale.
- If a router has a fresh entry in the cache for a certain resource, it answers a locate request for that resource without verifying that it is still available. A locate request is typically a TEST frame for MAC addresses or a FIND_NAME_QUERY for NetBIOS.
- If a router has a stale entry in the cache for a certain resource, it verifies that the entry is still valid before answering a locate request for the resource by sending a frame to the last known location of the resource and waits for a resource. If the entry is a REMOTE entry, the router sends a CUR_ex frame to the remote peer to verify. If the entry is a LOCAL entry, it sends either a TEST frame or a NetBIOS FIND_NAME_QUERY on the appropriate local port.
- By default, all reachability cache entries remain fresh for 4 minutes after they are learned. For MAC addresses, you can change this time with the `dlswtimersna-verify-interval` command. For NetBIOS names, you can change this time with the `dlswtimernetbios-verify-interval` command.
- By default, all reachability cache entries age out of the cache 16 minutes after they are learned. For MAC addresses, you can change this time with the `dlswtimersna-cache-timeout` command. For NetBIOS names, you can change this time with the `dlswtimernetbios-cache-timeout` command.

The table below describes the debug output indicating that the DLSW router received an SSP message that is flow controlled and should be counted against the window of the sender.

Dec 6 11:26:49: CSM: Received SSP CUR csex flags = 80, mac 4000.90b1.26cf,
The csex flags = 80 means that this is an CUR_ex (explorer).
Dec 5 10:48:33: DLSw: 1620175180 decr r - s:27 so:0 r:27 ro:0

Table 76: debug dlsw Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>decr r</td>
<td>Decrement received count.</td>
</tr>
<tr>
<td>s</td>
<td>This DLSW router’s granted units for the circuit.</td>
</tr>
<tr>
<td>so</td>
<td>0=This DLSW router does not owe a flow control acknowledgment. 1=This router owes a flow control acknowledgment.</td>
</tr>
<tr>
<td>r</td>
<td>Partner’s number of granted units for the circuit.</td>
</tr>
<tr>
<td>ro</td>
<td>Indicates whether the partner owes flow control acknowledgment.</td>
</tr>
</tbody>
</table>
The following message shows that DLSW is sending an I frame to a LAN:


The following message shows that DLSW received the I frame from the LAN:

Dec 5 10:48:35: DLSW Received-disp : CLSI Msg : DATA.Ind  dlen: 4

The following messages show that the reachability cache is cleared:

Router# clear dlsw rea
23:44:11: CSM: Clearing CSM cache
23:44:11: CSM: delete local mac cache for port 0
23:44:11: CSM: delete local name cache for port 0
23:44:11: CSM: delete remote mac cache for peer 0
23:44:11: CSM: delete remote name cash dlsw rea

The next group of messages show that the DLSW reachability cache is added, and that a name query is perform from the router MARIAN:

23:45:11: CSM: core_to_csm CLSI_MSG_PROC - port_id 5EFBB4
23:45:11: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:11: CSM: update local cache for mac 0800.5a30.7a9b, port 5EFBB4
23:45:11: CSM: update local cache for name MARIAN, port 5EFBB4
23:45:11: CSM: Received CLS_UDATA_STN from Core
23:45:11: CSM: Received netbios frame type A
23:45:11: CSM: Processing Name Query
23:45:11: CSM: Netbios Name Query: ws_status = 6
23:45:11: CSM: Write to peer 0 ok.
23:45:11: CSM: Freeing clsi message
23:45:11: CSM: core_to_csm CLSI_MSG_PROC - port_id 658AB4
23:45:11: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:11: CSM: update local cache for mac 0800.5a30.7a9b, port 658AB4
23:45:11: CSM: update local cache for name MARIAN, port 658AB4
23:45:11: CSM: Received CLS_UDATA_STN from Core
23:45:11: CSM: Received netbios frame type A
23:45:11: CSM: Processing Name Query
23:45:11: CSM: Netbios Name Query: ws_status = 5
23:45:11: CSM: DLXNR_PEND match found.... drop name query
23:45:11: CSM: Freeing clsi message
23:45:12: CSM: core_to_csm CLSI_MSG_PROC - port_id 5EFBB4
23:45:12: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:12: CSM: update local cache for mac 0800.5a30.7a9b, port 5EFBB4
23:45:12: CSM: update local cache for name MARIAN, port 5EFBB4
23:45:12: CSM: Received CLS_UDATA_STN from Core
23:45:12: CSM: Received netbios frame type A
23:45:12: CSM: Processing Name Query
23:45:12: CSM: Netbios Name Query: ws_status = 5
23:45:12: CSM: DLXNR_PEND match found.... drop name query
23:45:12: CSM: Freeing clsi message
23:45:12: CSM: core_to_csm CLSI_MSG_PROC - port_id 658AB4
23:45:12: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:12: CSM: update local cache for mac 0800.5a30.7a9b, port 658AB4
23:45:12: CSM: update local cache for name MARIAN, port 658AB4
23:45:12: CSM: Received CLS_UDATA_STN from Core
23:45:12: CSM: Received netbios frame type A
23:45:12: CSM: Processing Name Query
23:45:12: CSM: Netbios Name Query: ws_status = 5
23:45:12: CSM: DLXNR_PEND match found.... drop name query
23:45:12: CSM: Freeing clsi message
23:45:12: CSM: core_to_csm CLSI_MSG_PROC - port_id 5EFBB4
23:45:12: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:12: CSM: update local cache for mac 0800.5a30.7a9b, port 5EFBB4
23:45:12: CSM: update local cache for name MARIAN, port 5EFBB4
23:45:12: CSM: Received CLS_UDATA_STN from Core
23:45:12: CSM: Received netbios frame type A
23:45:12: CSM: Processing Name Query
23:45:12: CSM: Netbios Name Query: ws_status = 5
23:45:12: CSM: DLXNR_PEND match found.... drop name query
23:45:12: CSM: Freeing clsi message
23:45:12: CSM: core_to_csm CLSI_MSG_PROC - port_id 658AB4
The following messages show that the router named MARIAN is added to the network:

```
23:45:12: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:12: CSM: update local cache for mac 0800.5a30.7a9b, port 658AB4
23:45:12: CSM: Received CLS_UDATA_STN from Core
23:45:12: CSM: update local cache for name MARIAN , port 658AB4
23:45:12: CSM: Freeing clsi message
23:45:18: CSM: Deleting Reachability cache
23:45:18: CSM: Deleting DLX NR pending record....
23:45:38: CSM: core_to_csm CLSI_MSG_PROC - port_id 5EFBB4
23:45:38: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:38: CSM: update local cache for mac 0800.5a30.7a9b, port 5EFBB4
23:45:38: CSM: update local cache for name MARIAN , port 5EFBB4
23:45:38: CSM: Freeing clsi message
```

In the next group of messages, an attempt is made to add the router named GINGER on the Ethernet interface:

```
0:07:44: CSM: core_to_csm CLSI_MSG_PROC - port_id 658AB4
0:07:44: CSM: 0004.f545.24e6 passes local mac excl. filter
0:07:44: CSM: update local cache for mac 0004.f545.24e6, port 658AB4
0:07:44: CSM: update local cache for name GINGER , port 658AB4
0:07:44: CSM: Freeing clsi message
```

In the following example, the output from the `show dlsw reachability` command indicates that GINGER is on the Ethernet interface and MARIAN is on the Token Ring interface:

```
Router# show dlsw reachability
DLSw MAC address reachability cache list
Mac Addr status Loc. peer/port rif
0004.f545.24e6 FOUND LOCAL P007-S000 --no rif--
0800.5a30.7a9b FOUND LOCAL P000-S000 06C0.0621.7D00
0800.5a30.7a9b FOUND LOCAL P007-S000 F0F8.0006.A6FC.005F.F100.0000.0000.0000
DLSw NetBIOS Name reachability cache list
NetBIOS Name status Loc. peer/port rif
GINGER FOUND LOCAL P007-S000 --no rif--
MARIAN FOUND LOCAL P000-S000 06C0.0621.7D00
P007-S000 --no rif--
```
debug dmsp doc-to-fax

Note

In release 12.3(8)T, the debug dmsp doc-to-fax command is replaced by the debug fax dmsp command. See the debug fax dmsp command for more information.

To display debugging messages for the doc Media Service Provider (docMSP) TIFF or text2Fax engine, use the debug dm sp doc-to-fax command in privileged EXEC mode. To disable debugging output, use the no form of this command.

debug dmsp doc-to-fax [text-to-fax] tiff-reader
no debug dmsp doc-to-fax [text-to-fax] tiff-reader

Syntax Description

| text-to-fax | (Optional) Displays debugging messages that occur while the DocMSP Component is receiving text packets and producing T4 fax data. |
| tiff-reader | (Optional) Displays debugging messages that occur while the DocMSP Component is receiving TIFF packets and producing T4 fax data. |

Command Default
No default behavior or values.

Command Modes
Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XI</td>
<td>This command was introduced on the Cisco AS5300 access server.</td>
</tr>
<tr>
<td>12.3(8)T</td>
<td>This command was replaced by the debug fax dm sp command in the Cisco IOS 12.3T release.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

Examples

The following is sample output from the debug dm sp doc-to-fax command:

Router# debug dm sp doc-to-fax
Jan 1 04:58:39.898: docmsp_call_setup_request: callid=18
Jan 1 04:58:39.902: docmsp_call_setup_request(): ramp data dir=OFFRAMP, conf dir=SRC
Jan 1 04:58:39.902: docmsp_caps_ind: call id=18, src=17
Jan 1 04:58:39.902: docmsp_bridge cfid=5, srcclid=18, dstcid=17
Jan 1 04:58:39.902: docmsp_bridge(): ramp data dir=OFFRAMP, conf dir=SRC, encode out=2
Jan 1 04:58:39.902: docmsp_rcv_msp_ev: call id =18, evID = 42
Jan 1 04:58:39.902: docmsp_bridge cfid=6, srccid=18, dstcid=15
Jan 1 04:58:39.902: docmsp_bridge(): ramp data dir=OFFRAMP, conf dir=DEST, encode out=2
Jan 1 04:58:39.902: docmsp_process_rcv_data: call id src=0, dst=18
Jan 1 04:58:39.902: docmsp_generate_page:
Jan 1 04:58:39.902: docmsp_generate_page: new context for Call 18
Jan 1 04:58:39.922: docmsp_get_msp_event_buffer:
Jan 1 04:58:42.082: docmsp_xmit: call id src=15, dst=18
Jan 1 04:58:42.082: docmsp_process_rcv_data: call id src=15, dst=18
Jan 1 04:58:42.082: docmsp_process_rcv_data: call id src=15, dst=18
Jan 1 04:58:42.082: docmsp_process_rcv_data: call id src=15, dst=18
Jan 1 04:58:42.102: docmsp_xmit: call id src=15, dst=18
Jan 1 04:58:42.102: docmsp_xmit: call id src=15, dst=18
Jan 1 04:58:42.106: docmsp_process_rcv_data: call id src=15, dst=18
Jan 1 04:58:42.106: docmsp_process_rcv_data: call id src=15, dst=18
Jan 1 04:58:42.122: docmsp_xmit: call id src=15, dst=18
Jan 1 04:58:42.122: docmsp_xmit: call id src=15, dst=18
Jan 1 04:58:42.126: docmsp_xmit: call id src=15, dst=18
Jan 1 04:58:42.126: offramp_data_process:
Jan 1 04:58:42.142: docmsp_xmit: call id src=15, dst=18
Jan 1 04:58:42.146: docmsp_xmit: call id src=15, dst=18

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug dmsp fax-to-doc</td>
<td>Displays debugging messages for doc MPS fax-to-doc.</td>
</tr>
</tbody>
</table>
debug dmsp fax-to-doc

**Note**
In release 12.3(8)T, the `debugdmspfax-to-doc` command is replaced by the `debugfaxdmsp` command. See the `debugfaxdmsp` command for more information.

To display debugging messages for doc MSP (docMSP) fax-to-doc, use the `debugdmspfax-to-doc` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

`debug dmsp fax-to-doc [tiff-writer]
no debug dmsp fax-to-doc [tiff-writer]`

**Syntax Description**

| tiff-writer | (Optional) Displays debug messages that occur while the DocMSP Component is receiving T4 fax data and producing TIFF packets. |

**Command Default**
No default behavior or values.

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XI</td>
<td>This command was introduced on the Cisco AS5300 access server.</td>
</tr>
<tr>
<td>12.3(8)T</td>
<td>This command was replaced by the <code>debugfaxdmsp</code> command in the Cisco IOS 12.3T release.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Examples**
The following is sample output from the `debugdmspfax-to-doc` command:

```
Router# debug dmsp fax-to-doc
*Oct 16 08:29:54.487: docmsp_call_setup_request: callid=22
*Oct 16 08:29:54.487: docmsp_call_setup_request(): ramp data dir=OFFRAMP, conf dir=SRC
*Oct 16 08:29:54.487: docmsp_caps_ind: call id=22, src=21
*Oct 16 08:29:54.487: docmsp_bridge cfid=15, srcclid=22, dstclid=21
*Oct 16 08:29:54.487: docmsp_bridge(): ramp data dir=OFFRAMP, conf dir=SRC, encode out=2
*Oct 16 08:29:54.487: docmsp_bridge cfid=16, srcclid=22, dstclid=17
*Oct 16 08:29:54.487: docmsp_bridge(): ramp data dir=OFFRAMP, conf dir=DEST, encode out=2
*Oct 16 08:29:54.487: docmsp_xmit: call id src=17, dst=22
*Oct 16 08:29:54.487: docmsp_process_rcv_data: call id src=17, dst=22
*Oct 16 08:29:54.487: offramp_data_process:  
```
*Oct 16 08:29:54.515: docmsp_get_msp_event_buffer:
*Oct 16 08:29:56.115: docmsp_call_setup_request: callid=24
*Oct 16 08:29:56.115: docmsp_call_setup_request(): ramp data dir=ONRAMP, conf dir=DEST
*Oct 16 08:29:56.115: docmsp_caps_ind: call id=24, src=20
*Oct 16 08:29:56.115: docmsp_bridge cfid=17, srccid=24, dstcid=20

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug dmsp doc-to-fax</td>
<td>Displays debugging messages for the doc Media Service Provider TIFF or text2Fax engine.</td>
</tr>
</tbody>
</table>
# debug dmvpn

To display debug Dynamic Multipoint VPN (DMVPN) session information, use the **debug dmvpn** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug dmvpn {all| error| detail| packet} {all| debug-type}
no debug dmvpn {all| error| detail| packet} {all| debug-type}
```

## Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>all</code></td>
<td>Enables all levels of debugging.</td>
</tr>
<tr>
<td><code>error</code></td>
<td>Enables error-level debugging.</td>
</tr>
<tr>
<td><code>detail</code></td>
<td>Enables detail-level debugging.</td>
</tr>
<tr>
<td><code>packet</code></td>
<td>Enables packet-level debugging.</td>
</tr>
<tr>
<td><code>debug-type</code></td>
<td>The type of debugging that you want to enable. The following keywords can be specified for the <code>debug-type</code> argument:</td>
</tr>
<tr>
<td><code>nhrp</code></td>
<td>Enables Next Hop Resolution Protocol (NHRP) debugging only.</td>
</tr>
<tr>
<td><code>crypto</code></td>
<td>Enables crypto Internet Key Exchange (IKE) and IPsec debugging.</td>
</tr>
<tr>
<td><code>tunnel</code></td>
<td>Enables tunnel protection debugging.</td>
</tr>
<tr>
<td><code>socket</code></td>
<td>Enables crypto secure socket debugging.</td>
</tr>
</tbody>
</table>

The keywords can be used alone, or in any combination with each other, but each keyword can be used only once.

## Command Default

DMVPN debugging is disabled.

## Command Modes

Privileged EXEC (#)

## Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(9)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>
This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.

This command was modified. This command was integrated into Cisco IOS XE Release 2.5.

**Usage Guidelines**

You must specify both the level and the type of debugging that you want to enable. The debugging levels are all, error, detail, or packet. You can enable NHRP, crypto Internet Key Exchange (IKE) and IPsec, tunnel protection, and crypto secure socket debugging at any of the four debugging levels.

To enable conditional DMVPN debugging, you must first specify the level and type of debugging that you want to enable, and then use the `debug dmvpn condition` command to specify the conditions that you want to enable.

**Error-Level Debugging**

When error-level debugging is enabled with the `debug dmvpn error` command, the following debugging commands are enabled by default:

- `debug crypto ipsec error`
- `debug crypto isakmp error`
- `debug nhrp error`

**Detail-Level Debugging**

When detail-level debugging is enabled with the `debug dmvpn detail` command, the following debugging commands are enabled by default:

- `debug crypto ipsec`
- `debug crypto isakmp`
- `debug crypto sockets`
- `debug nhrp`
- `debug nhrp cache`
- `debug nhrp rate`
- `debug tunnel protection`

**Packet-Level Debugging**

When packet-level debugging is enabled with the `debug dmvpn packet` command, the following debugging commands are enabled by default:

- `debug nhrp extension`
- `debug nhrp packet`
Executing the `debug dmvpn all` command with a high number of active sessions may result in high CPU utilization and large data output.

**NHRP Shortcut Route Debugging**

When shortcut switching is enabled on the router, the system looks up the NHRP shortcut route in the Routing Information Base (RIB) in order to forward the packet to the next-hop in the DMVPN cloud.

The table below describes the debug messages displayed by the router when shortcut switching and NHRP debugging are both enabled.

<table>
<thead>
<tr>
<th>Event</th>
<th>Sample Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>NHRP successfully adds a route to the RIB</td>
<td>*Feb 21 13:11:24.043: NHRP: Adding route entry for 172.16.99.0 to RIB</td>
</tr>
<tr>
<td></td>
<td>*Feb 21 13:11:24.043: NHRP: Route addition to RIB successful</td>
</tr>
<tr>
<td>NHRP is unable to add a route to the RIB</td>
<td>*Feb 21 13:11:24.043: NHRP: Adding route entry for 172.16.99.0 to RIB</td>
</tr>
<tr>
<td></td>
<td>*Feb 21 13:11:24.043: NHRP: Route addition to RIB failed</td>
</tr>
<tr>
<td>NHRP removes a route from the RIB</td>
<td>*Feb 21 13:11:24.043: NHRP: Deleting route entry for 172.16.99.0 from RIB</td>
</tr>
<tr>
<td>NHRP evicts a route from the RIB</td>
<td>*Mar 1 18:24:29.371: NHRP: Route entry 172.16.22.0/24 clobbered by distance</td>
</tr>
<tr>
<td>NHRP changes the administrative distance</td>
<td>*Mar 1 00:14:16.799: NHRP: Administrative distance changed to 240</td>
</tr>
</tbody>
</table>

**Examples**

The following example shows how to enable all debugging levels for DMVPN tunnel debugging:

`Router# debug dmvpn all tunnel`

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug crypto error</td>
<td>Enables error debugging for a crypto area.</td>
</tr>
<tr>
<td>debug crypto ipsec</td>
<td>Displays IPsec events.</td>
</tr>
<tr>
<td>debug crypto isakmp</td>
<td>Displays messages about IKE events.</td>
</tr>
<tr>
<td>debug dmvpn condition</td>
<td>Display conditional debug DMVPN session information.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>debug nhrp condition</td>
<td>Enables NHRP conditional debugging.</td>
</tr>
<tr>
<td>debug nhrp error</td>
<td>Displays NHRP error-level debugging information.</td>
</tr>
</tbody>
</table>
To display conditional debug Dynamic Multipoint VPN (DMVPN) session information, use the `debug dmvpn condition` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
default dmvpn condition {unmatched|peer {nbma|tunnel {ipv4-address|ipv6-address}}|vrf vrf-name|interface tunnel tunnel-interface}
no default dmvpn condition [unmatched|peer {nbma|tunnel {ipv4-address|ipv6-address}}|vrf vrf-name|interface tunnel number]
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unmatched</td>
<td>Specifies debugging when context information is not available.</td>
</tr>
<tr>
<td>peer</td>
<td>Specifies information for a specific DMVPN peer.</td>
</tr>
<tr>
<td>nbma</td>
<td>Displays DMVPN information based on the peer mapping nonbroadcast access (NBMA) address.</td>
</tr>
<tr>
<td>tunnel</td>
<td>Displays DMVPN information based on the peer Virtual Private Network (VPN) address.</td>
</tr>
<tr>
<td>ipv4-address</td>
<td>The DMVPN peer IPv4 address.</td>
</tr>
<tr>
<td>ipv6-address</td>
<td>The DMVPN peer IPv6 address. <strong>Note</strong> Cisco IOS XE Release 2.5 does not support the ipv6-address argument.</td>
</tr>
<tr>
<td>vrf</td>
<td>Displays information based on the specified virtual routing and forwarding (VRF) name.</td>
</tr>
<tr>
<td>vrf-name</td>
<td>The VRF name.</td>
</tr>
<tr>
<td>interface</td>
<td>Displays DMVPN information based on a specific interface.</td>
</tr>
<tr>
<td>tunnel</td>
<td>Specifies the tunnel address for a DMVPN peer.</td>
</tr>
<tr>
<td>number</td>
<td>The tunnel interface number.</td>
</tr>
</tbody>
</table>

### Command Default

DMVPN conditional debugging is disabled.

### Command Modes

Privileged EXEC (#)
Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(9)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
<tr>
<td>12.4(20)T</td>
<td>The ipv6-address argument was added.</td>
</tr>
<tr>
<td>Cisco IOS XE Release 2.5</td>
<td>This command was modified. It was integrated into Cisco IOS XE Release 2.5.</td>
</tr>
</tbody>
</table>

Usage Guidelines

Conditional debugging is enabled only after the DMVPN debugging type and level have been specified using the `debug dmvpn` command.

Console Output

The following `debug dmvpn` commands do not have any console output on the Cisco 3845 and Cisco 7200 series routers:

- `debug dmvpn condition interface`
- `debug dmvpn condition peer`
- `debug dmvpn condition unmatched`
- `debug dmvpn condition vrf`

Note

When the `debug dmvpn condition unmatched` command is enabled on the Cisco 3845 and Cisco 7200 series routers, issuing the `show debugging` command does not produce any console output.

Examples

The following example shows how to enable conditional DMVPN debugging for a specific peer NBMA address:

```
Router# debug dmvpn condition peer nbma 192.0.2.1
```

The following example shows how to enable conditional DMVPN debugging when context is not available to check against debugging conditions:

```
Router# debug dmvpn condition unmatched
```

The following example shows how to disable conditional debugging for a specific tunnel interface:

```
Router# no debug dmvpn condition interface tunnel 1
```
The following example shows how to disable all conditional debugging:

```
Router# no debug dmvpn condition
```

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug crypto error</td>
<td>Enables error debugging for a crypto area.</td>
</tr>
<tr>
<td>debug crypto ipsec</td>
<td>Displays IPsec events.</td>
</tr>
<tr>
<td>debug crypto isakmp</td>
<td>Displays messages about IKE events.</td>
</tr>
<tr>
<td>debug dmvpn</td>
<td>Displays debug DMVPN session information.</td>
</tr>
<tr>
<td>debug nhrp condition</td>
<td>Enables NHRP conditional debugging.</td>
</tr>
<tr>
<td>debug nhrp error</td>
<td>Displays NHRP error-level debugging information.</td>
</tr>
</tbody>
</table>
debug dot11

To enable debugging of radio functions, use the debug dot11 command in privileged EXEC mode. To stop or disable the debug operation, use the no form of this command.

debug dot11 {events|forwarding|mgmt|packets|syslog|virtual-interface}
no debug dot11 {events|forwarding|mgmt|packets|syslog|virtual-interface}

Syntax Description

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>events</td>
<td>Displays information about all radio-related events.</td>
</tr>
<tr>
<td>forwarding</td>
<td>Displays information about radio-forwarded packets.</td>
</tr>
<tr>
<td>mgmt</td>
<td>Displays information about radio access point management activity.</td>
</tr>
<tr>
<td>packets</td>
<td>Displays information about received or transmitted radio packets.</td>
</tr>
<tr>
<td>syslog</td>
<td>Displays information about the radio system log.</td>
</tr>
<tr>
<td>virtual-interface</td>
<td>Displays information about radio virtual interfaces.</td>
</tr>
</tbody>
</table>

Command Default

Debugging is disabled.

Command Modes

Privileged EXEC (#)

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(4)JA</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(2)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(2)T.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>

Usage Guidelines

Use this command to display debugging information about radio functions.
Examples

The following example shows how to enable debugging of all radio-related events:

Router# debug dot11 events

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug dot11 aaa</td>
<td>Enables debugging of dot11 AAA operations.</td>
</tr>
<tr>
<td>debug dot11 dot11radio</td>
<td>Enables radio debug options.</td>
</tr>
</tbody>
</table>
# debug dot11 aaa

To enable debugging of dot11 authentication, authorization, and accounting (AAA) operations, use the `debug dot11 aaa` command in privileged EXEC mode. To disable or stop the debug operation, use the `no` form of this command.

```
debug dot11 aaa {accounting| authenticator {all| dispatcher| mac-authen| process| rxdata| state-machine| txdata}| dispatcher| manager {all| dispatcher| keys| rxdata| state-machine| supplicant| txdata}}
no debug dot11 aaa {accounting| authenticator {all| dispatcher| mac-authen| process| rxdata| state-machine| txdata}| dispatcher| manager {all| dispatcher| keys| rxdata| state-machine| supplicant| txdata}}
```

## Syntax Description

<table>
<thead>
<tr>
<th>syntax</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>accounting</strong></td>
<td>Provides information about 802.11 AAA accounting packets.</td>
</tr>
</tbody>
</table>
| **authenticator** | Provides information about MAC and Extensible Authentication Protocol (EAP) authentication packets. Use the following options to activate authenticator debugging:  
  - `all`--Activates debugging for all authenticator packets  
  - `dispatcher`--Activates debugging for authentication request handler packets  
  - `mac-authen`--Activates debugging for MAC authentication packets  
  - `process`--Activates debugging for authenticator process packets  
  - `rxdata`--Activates debugging for EAP over LAN (EAPOL) packets from client devices  
  - `state-machine`--Activates debugging for authenticator state-machine packets  
  - `txdata`--Activates debugging for EAPOL packets sent to client devices |
| **dispatcher** | Provides information about 802.11 AAA dispatcher (interface between association and manager) packets. |
### manager

Provides information about the AAA manager. Use these options to activate AAA manager debugging:

- **all** -- Activates all AAA manager debugging
- **dispatcher** -- Activates debug information for AAA manager authenticator dispatch traffic
- **keys** -- Activates debug information for AAA manager key processing
- **rxdata** -- Activates debugging for AAA manager packets received from client devices
- **state-machine** -- Activates debugging for AAA manager state-machine packets
- **supplicant** -- Activates debugging for Light Extensible Authentication Protocol (LEAP) supplicant packets
- **txdata** -- Activates debugging for AAA manager packets sent to client devices.

#### Command Default

Debugging is disabled.

#### Command Modes

Privileged EXEC (#)

#### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(4)JA</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(15)JA</td>
<td>This command was modified to include the accounting, authenticator, dispatcher, and manager debugging options.</td>
</tr>
<tr>
<td>12.4(2)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(2)T.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>

#### Usage Guidelines

Use this command to display debugging information about dot11 AAA operations.

#### Examples

The following example shows how to activate debugging for 802.11 AAA accounting packets:

```
Router# debug dot11 aaa accounting
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug dot11</td>
<td>Enables debugging of radio functions.</td>
</tr>
<tr>
<td>debug dot11 dot11radio</td>
<td>Enables radio debug options.</td>
</tr>
</tbody>
</table>
debug dot11 cac

Use the debug dot11 cac privileged EXEC command to begin debugging of admission control radio functions. Use the no form of this command to stop the debug operation.

[no] debug dot11 cac {events|unit}

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>events</td>
<td>Activates debugging of radio admission control events.</td>
</tr>
<tr>
<td>unit</td>
<td>Activates verbose debugging of radio admission control events.</td>
</tr>
</tbody>
</table>

**Command Default**

Debugging is not enabled.

**Command Modes**

Privileged EXEC ("")

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(8)JA</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>

**Examples**

This example shows how to begin debugging of all admission control radio-related events:

```
SOAP-AP# debug dot11 cac events
```

This example shows how to begin verbose debugging of all admission control radio-related events:

```
SOAP-AP# debug dot11 cac unit
```

This example shows how to stop debugging of all admission control radio-related events:

```
SOAP-AP# debug dot11 cac events
```

This example shows how to stop verbose debugging of all admission control radio-related events:

```
SOAP-AP# no debug dot11 cac unit
```
This command is not supported on repeaters.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin-traffic (SSID configuration mode)</td>
<td>Enables CAC admission control for an SSID on the access point.</td>
</tr>
<tr>
<td>admit-traffic (QoS Class interface configuration mode)</td>
<td>Configures CAC admission control on the access point.</td>
</tr>
<tr>
<td>show debugging</td>
<td>Displays all debug settings and the debug packet headers</td>
</tr>
<tr>
<td>show dot11 ids eap</td>
<td>Displays all CAC radio events on the access point.</td>
</tr>
<tr>
<td>traffic-stream</td>
<td>Configures CAC traffic data rates and priorities for a radio interface on the access point.</td>
</tr>
</tbody>
</table>
debug dot11 dot11radio

To enable radio debug options, use the debug dot11 dot11radio command in privileged EXEC mode. To disable debug options, use the no form of this command.

download dot11 dot11radio interface {accept-radio-firmware| dfs simulate [ channel ]| monitor {ack| address| beacon| crc| lines| plcp| print| probe| store}| print {hex| if| iv| lines| mic| plcp| printf| raw| shortadr}| stop-on-failure| trace {off| print| store}}

no debug dot11 dot11radio interface {accept-radio-firmware| dfs simulate [ channel ]| monitor {ack| address| beacon| crc| lines| plcp| print| probe| store}| print {hex| if| iv| lines| mic| plcp| printf| raw| shortadr}| stop-on-failure| trace {off| print| store}}

Syntax Description

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interface</td>
<td>The radio interface. The 2.4-GHz radio is 0. The 5-GHz radio is 1.</td>
</tr>
<tr>
<td>accept-radio-firmware</td>
<td>Configures the access point to disable checking the radio firmware version.</td>
</tr>
<tr>
<td>dfs simulate</td>
<td>Configures the access point to simulate radar generation as part of Dynamic Frequency Selection (DFS).</td>
</tr>
<tr>
<td>channel</td>
<td>(Optional) Radio channel to move to. Range is from 24 to 161.</td>
</tr>
<tr>
<td>monitor</td>
<td>Enables RF monitor mode. Use these options to turn on monitor modes:</td>
</tr>
<tr>
<td></td>
<td>• ack -- Displays ACK packets. ACK packets acknowledge receipt of a signal, information, or packet.</td>
</tr>
<tr>
<td></td>
<td>• address -- Displays packets to or from the specified IP address</td>
</tr>
<tr>
<td></td>
<td>• beacon -- Displays beacon packets</td>
</tr>
<tr>
<td></td>
<td>• crc -- Displays packets with CRC errors</td>
</tr>
<tr>
<td></td>
<td>• lines -- Specifies a print line count</td>
</tr>
<tr>
<td></td>
<td>• plcp -- Displays Physical Layer Control Protocol (PLCP) packets</td>
</tr>
<tr>
<td></td>
<td>• print -- Enables RF monitor printing mode</td>
</tr>
<tr>
<td></td>
<td>• probe -- Displays probe packets</td>
</tr>
<tr>
<td></td>
<td>• store -- Enables RF monitor storage mode</td>
</tr>
</tbody>
</table>
### print

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hex</td>
<td>Prints entire packets without formatting</td>
</tr>
<tr>
<td>if</td>
<td>Prints the in and out interfaces for packets</td>
</tr>
<tr>
<td>iv</td>
<td>Prints the packet Wired Equivalent Privacy (WEP) IV</td>
</tr>
<tr>
<td>lines</td>
<td>Prints the line count for the trace</td>
</tr>
<tr>
<td>mic</td>
<td>Prints the Cisco Message Integrity Check (MIC)</td>
</tr>
<tr>
<td>plcp</td>
<td>Displays the PLCP</td>
</tr>
<tr>
<td>printf</td>
<td>Prints using printf instead of buginf</td>
</tr>
<tr>
<td>raw</td>
<td>Prints without formatting data</td>
</tr>
<tr>
<td>shortadr</td>
<td>Prints MAC addresses in short form</td>
</tr>
</tbody>
</table>

### stop-on-failure

Configures the access point to not restart when the radio driver fails.

### trace

Enables trace mode. Use these options to turn on trace modes:

- **off** -- Turns off traces
- **print** -- Enables trace printing
- **store** -- Enables trace storage

---

**Command Default**

Debugging is disabled.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(4)JA</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(2)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(2)T.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>
Usage Guidelines

Use this command to display debugging information about radio options.

Examples

This example shows how to begin monitoring of all packets with CRC errors:

```
Router# debug dot11 dot11radio 0 monitor crc
```

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug dot11</td>
<td>Enables debugging of radio functions.</td>
</tr>
<tr>
<td>debug dot11 aaa</td>
<td>Enables debugging of dot11 AAA operations.</td>
</tr>
</tbody>
</table>
debug dot11 ids

Use the `debug dot11 ids eap` privileged EXEC command to enable debugging for wireless IDS monitoring. Use the `no` form of the command to disable IDS debugging.

```
[no] debug dot11 ids {eap | cipher-errors}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eap</td>
<td>Activates debugging of IDS authentication events</td>
</tr>
<tr>
<td>cipher-errors</td>
<td>Activates debugging of cipher errors detected by IDS</td>
</tr>
</tbody>
</table>

### Command Default

Debugging is not enabled.

### Command Modes

Privileged EXEC (#)

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(4)JA</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>

### Examples

This example shows how to activate wireless IDS debugging for authentication events:

```
SOAP-AP# debug dot11 ids eap
```

### Related Commands

This command is not supported on 1400 series bridges.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dot11 ids eap attempts</td>
<td>Configures limits on authentication attempts and EAPOL flooding on scanner access points in monitor mode</td>
</tr>
<tr>
<td>show debugging</td>
<td>Displays all debug settings and the debug packet headers</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td><strong>show dot11 ids eap</strong></td>
<td>Displays wireless IDS statistics</td>
</tr>
</tbody>
</table>
debug dot11 ids mfp

Use the debug dot11 ids mfp privileged EXEC command to debug Management Frame Protection (MFP) operations on the access point.

```
{[no] debug dot11 ids mfp [ap | all] [detectors | events | generators] [io | reporting] [wds | all] [detectors | events | generators] [reporting | statistics] [wlccp]}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ap</td>
<td>Debugs MFP events on the access point.</td>
</tr>
<tr>
<td>all</td>
<td>Debugs all MFP events.</td>
</tr>
<tr>
<td>detectors</td>
<td>Debugs MFP detector key management events.</td>
</tr>
<tr>
<td>events</td>
<td>Debugs high level MFP events.</td>
</tr>
<tr>
<td>generators</td>
<td>Debugs MFP generator key management events.</td>
</tr>
<tr>
<td>io</td>
<td>Debugs MFP IO (generate or detect frame) events.</td>
</tr>
<tr>
<td>reporting</td>
<td>Debugs MFP reporting events.</td>
</tr>
<tr>
<td>statistics</td>
<td>Debugs MFP WDS statistics received from the detectors.</td>
</tr>
<tr>
<td>wds</td>
<td>Debugs MFP WDS events.</td>
</tr>
<tr>
<td>wlccp</td>
<td>Debugs MFP WLCCP messages.</td>
</tr>
</tbody>
</table>

### Command Default

There are no defaults for this command.

### Command Modes

Privileged EXEC (#)

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(8)JA</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>
**Examples**

This example shows how to debug the MFP detectors on the access point:

```
ap(config)# debug dot11 ids mfp ap detectors
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dot11 ids mfp</td>
<td>Configures MFP parameters on the access point.</td>
</tr>
<tr>
<td>show dot11 ids mfp</td>
<td>Displays MFP parameters on the access point.</td>
</tr>
</tbody>
</table>
debug dot1x

To display 802.1X debugging information, use the **debug dot1x** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
dot1x [all| errors| events| feature| packets| redundancy| registry| state-machine]  
no debug dot1x [all| errors| events| feature| packets| redundancy| registry| state-machine]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>all</strong></td>
<td>(Optional) Enables all 802.1X debugging messages.</td>
</tr>
<tr>
<td><strong>errors</strong></td>
<td>(Optional) Provides information about all 802.1X errors.</td>
</tr>
<tr>
<td><strong>events</strong></td>
<td>(Optional) Provides information about all 802.1X events.</td>
</tr>
<tr>
<td><strong>feature</strong></td>
<td>(Optional) Provides information about 802.1X features.</td>
</tr>
<tr>
<td><strong>packets</strong></td>
<td>(Optional) Provides information about all 802.1X packets.</td>
</tr>
<tr>
<td><strong>redundancy</strong></td>
<td>(Optional) Provides information about 802.1X redundancy.</td>
</tr>
<tr>
<td><strong>registry</strong></td>
<td>(Optional) Provides information about 802.1X registries.</td>
</tr>
<tr>
<td><strong>state-machine</strong></td>
<td>(Optional) Provides information regarding the 802.1X state.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(11)AX</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.1(14)EA1</td>
<td>The <strong>authsm</strong>, <strong>backend</strong>, <strong>besm</strong>, <strong>core</strong>, and <strong>reauthsm</strong> keywords were removed. The <strong>errors</strong>, <strong>events</strong>, <strong>packets</strong>, <strong>registry</strong>, and <strong>state-machine</strong> keywords were added.</td>
</tr>
<tr>
<td>12.3(2)XA</td>
<td>This command was integrated into Cisco IOS Release 12.3(2)XA.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>
### Examples

The following is sample output for the `debug dot1x` command:

```plaintext
Router# debug dot1x

Router-871# debug dot1x all
*Nov 7 13:07:56.872: dot1x-ev:dot1x_mgr_pre_process_eapol_pak: Role determination not required on FastEthernet1.
*Nov 7 13:07:56.876: dot1x-packet:dot1x_mgr_process_eapol_pak: queuing an EAPOL pkt on Authenticator Q
*Nov 7 13:07:56.876: dot1x-ev:Enqueued the eapol packet to the global authenticator queue
*Nov 7 13:07:56.876: dot1x-packet:Received an EAPOL frame on interface FastEthernet1
*Nov 7 13:07:56.876: dot1x-ev:Received pkt saddr = 000f.23c4.a401 , daddr = 0180.c200.0003, pae-ether-type = 888e.0202.0000
*Nov 7 13:07:56.876: dot1x-packet:Received an EAPOL-Logoff packet on interface FastEthernet1
*Nov 7 13:07:56.876: dot1x-sm:Posting EAPOL_LOGOFF on Client=82AC85CC
*Nov 7 13:07:56.876: dot1x_auth Fa1: during state auth_authorizing, got event 7(eapolLogoff)

The fields in the output are self-explanatory.
```

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear dot1x</td>
<td>Clears 802.1X interface information.</td>
</tr>
<tr>
<td>identity profile default</td>
<td>Creates an identity profile and enters identity profile configuration mode.</td>
</tr>
<tr>
<td>show dot1x</td>
<td>Displays details for an identity profile.</td>
</tr>
</tbody>
</table>
**debug dot1x (EtherSwitch)**

To enable debugging of the 802.1x protocol when an Ethernet switch network module is installed, use the `debug dot1x` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```plaintext
debug dot1x {all| authsm| backend| besm| core| reauthsm}
no debug dot1x {all| authsm| backend| besm| core| reauthsm}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Enables debugging of all conditions.</td>
</tr>
<tr>
<td>authsm</td>
<td>Enables debugging of the authenticator state machine, which is responsible for controlling access to the network through 802.1x-enabled ports.</td>
</tr>
<tr>
<td>backend</td>
<td>Enables debugging of the interaction between the 802.1x process and the router RADIUS client.</td>
</tr>
<tr>
<td>besm</td>
<td>Enables debugging of the backend state machine, which is responsible for relaying authentication request between the client and the authentication server.</td>
</tr>
<tr>
<td>core</td>
<td>Enables debugging of the 802.1x process, which includes 802.1x initialization, configuration, and the interaction with the port manager module.</td>
</tr>
<tr>
<td>reauthsm</td>
<td>Enables debugging of the reauthentication state machine, which manages periodic reauthentication of the client.</td>
</tr>
</tbody>
</table>

**Command Default**

Debugging is disabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(6)EA2</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(15)ZJ</td>
<td>This command was implemented on the following platforms: Cisco 2600 series, Cisco 3600 series, and Cisco 3700 series routers.</td>
</tr>
</tbody>
</table>
This command was integrated into Cisco IOS Release 12.3(4)T on the following platforms: Cisco 2600 series, Cisco 3600 series, and Cisco 3700 series routers.

12.2(33)SRA  This command was integrated into Cisco IOS Release 12.2(33)SRA.

**Usage Guidelines**

The `undebugdot1x` command is the same as the `nodebugdot1x` command.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>show debugging</code></td>
<td>Displays information about the types of debugging that are enabled.</td>
</tr>
<tr>
<td><code>show dot1x</code></td>
<td>Displays 802.1x statistics, administrative status, and operational status for the router or for the specified interface.</td>
</tr>
</tbody>
</table>
**debug drip event**

To display debugging messages for Duplicate Ring Protocol (DRiP) events, use the `debugdripevent` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug drip event
no debug drip event
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Debugging is disabled for DRiP events.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(4)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

When a TrBRF interface is configured on the Remote Switch Module (RSM), the DRiP protocol is activated. The DRiP protocol adds the VLAN ID specified in the router command to its database and recognizes the VLAN as a locally configured, active VLAN.

**Examples**

The following is sample output from the `debugdripevent` command:

```
Router# debug drip event
DRiP gets a packet from the network:

612B92C0: 01000000 00000000 0C501900 0000AAAA ........P....**
612B92D0: 0300000C 00020000 00000100 0CCCCCCC ............LLL
612B92E0: 00000C50 19000020 AAA0300 000C0102 ..........P......
612B92F0: 01010114 00000002 00000002 00000C50 ............P
612B9300: 19000001 04C00064 04 .......0.d.
DRiP gets a packet from the network:

Recvd. pak
DRiP recognizes that the VLAN ID it is getting is a new one from the network:

6116C840: 0100 0CCCCCCC .......LLL
6116C850: 00102F72 CBFB0024 AAA0300 000C0102 ..../K/.**......
6116C860: 01FF0214 0002E254 00015003 00102F72 ......bT.P....../r
6116C870: C8010010 04C00014 044003EB 14 ....H...@...@.k.
DRIP : remote update - Never heard of this vlan
```
DRiP attempts to resolve any conflicts when it discovers a new VLAN. The value action = 1 means to notify the local platform of change in state.

DRIP : resolve remote for vlan 20 in VLAN0
DRIP : resolve remote - action = 1
The local platform is notified of change in state:

DRIP Change notification active vlan 20
Another new VLAN ID was received in the packet:

DRIP : resolve remote for vlan 1003 in Vlan0
No action is required:

DRIP : resolve remote - action = 0
Thirty seconds have expired, and DRiP sends its local database entries to all its trunk ports:

DRIP : local timer expired
DRIP : transmit on 0000.0c50.1900, length = 24
612B92C0: 01000C00 00000000 0C501900 0000AAAA .........P....**
612B92D0: 0300000C 00020000 00000100 0CCCCCCC .............LLL
612B92E0: 00000C50 19000020 AAAA0300 000C0102 ...P... **.....
612B92F0: 01FF0114 00000003 00000002 00000C50 ...............P
612B9300: 19000001 04C00064 04 ......@.d.
**debug drip packet**

To display debugging messages for Duplicate Ring Protocol (DRiP) packets, use the `debugdrippacket` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```plaintext
debug drip packet
no debug drip packet
```

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

**Command Default**
Debugging is not enabled for DRiP packets.

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(4)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
Before you use this command, you can optionally use the `cleardrip` command first. As a result the DRiP counters are reset to 0. If the DRiP counters begin to increment, the router is receiving packets.

**Examples**
The following is sample output from the `debugdrippacket` command:

```
Router# debug drip packet
The following type of output is displayed when a packet is entering the router and you use the `showdebug` command:

039E5FC0: 0100 00CC00CC 00E0A39B 3FFB0028 ...LLL."?.{( (039E5FD0: AAAA0300 000C0102 01FF0314 0000A5F6 **............%\n039E5FE0: 00008805 00E0A39B 3C000000 04C00028 .......#.<....@.(
039E5FF0: 04C00032 044003EB 0F .@.2.@.k.
039FBD20: 01000C00 00000010 ........
039FBD30: A6ABE450 0000AAAA 03000000 00020000 &.4P..**........
039FBD40: 00000100 0CC0CC00 0100A6AE B4500020 .......LLL.&.4P.
039FBD50: AAAA0300 000C0102 01FF0114 00000003 **............
039FBD60: 00000002 0010A6AE B4500001 04C00064 ......&.4P...@.d
039FBD70: 04
```

The following type of output is displayed when a packet is sent by the router:

```plaintext
039FBD30: A6ABE450 0000AAAA 03000000 00020000 &.4P..**........
039FBD40: 00000100 0CC0CC00 0100A6AE B4500020 .......LLL.&.4P.
039FBD50: AAAA0300 000C0102 01FF0114 00000003 **............
039FBD60: 00000002 0010A6AE B4500001 04C00064 ......&.4P...@.d
039FBD70: 04
```
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug drip event</td>
<td>Displays debugging messages for DRiP events.</td>
</tr>
</tbody>
</table>
debug dsc clock

To display debugging output for the time-division multiplexing (TDM) clock-switching events on the dial shelf controller (DSC), use the `debug dsc clock` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```plaintext
[execute-on] debug dsc clock
[execute-on] no debug dsc clock
```

**Syntax Description**

This command has no arguments or keywords; however, it can be used with the `execute-on` command.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(2)AA</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

To perform this command from the router shelf on the Cisco AS5800 series platform, use the `execute-on slot slot-number debug dsc clock` form of this command.

The `debug dsc clock` command displays TDM clock-switching events on the dial shelf controller. The information displayed includes the following:

- Clock configuration messages received from trunks via NBUS
- Dial shelf controller clock configuration messages from the router shelf over the dial shelf interface link
- Clock switchover algorithm events

**Examples**

The following example shows that the `debug dsc clock` command has been enabled, and that trunk messages are received, and that the configuration message has been received:

```plaintext
AS5800# debug dsc clock
Dial Shelf Controller Clock debugging is on
AS5800# 00:02:55: Clock Addition msg of len 12 priority 8 from slot 1 port 1 on line 0
00:02:55: Trunk 1 has reloaded
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>execute-on</td>
<td>Executes commands remotely on a line card.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>show dsc clock</td>
<td>Displays information about the dial shelf controller clock.</td>
</tr>
</tbody>
</table>
debug dsip

To display debugging output for Distributed System Interconnect Protocol (DSIP) used between a router shelf and a dial shelf, use the `debug dsip` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug dsip {all| api| boot| console| trace| transport}

no debug dsip {all| api| boot| console| trace| transport}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>View all DSIP debugging messages.</td>
</tr>
<tr>
<td>api</td>
<td>View DSIP client interface (API) debugging messages.</td>
</tr>
<tr>
<td>boot</td>
<td>View DSIP booting messages that are generated when a download of the feature board image is occurring properly.</td>
</tr>
<tr>
<td>console</td>
<td>View DSIP console operation while debugging.</td>
</tr>
<tr>
<td>trace</td>
<td>Enable logging of header information concerning DSIP packets entering the system into a trace buffer. This logged information can be viewed with the <code>show dsip tracing</code> command.</td>
</tr>
<tr>
<td>transport</td>
<td>Debug the DSIP transport layer, the module that interacts with the underlying physical media driver.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(2)AA</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The `debug dsip` command is used to enable the display of debugging messages for DSIP between the router shelf and the dial shelf. Using this command, you can display booting messages generated when the download of an image occurs, view console operation, and trace logging of MAC header information and DSIP transport layer information as modules interact with the underlying physical media driver. This command can be applied to a single modem or a group of modems.
Once the `debug dsip trace` command has been enabled, you can read the information captured in the trace buffer using the `show dsip tracing` command.

### Examples

The following example indicates the `debug dsip trace` command logs MAC headers of the various classes of DSIP packets. To view the logged information, use the `show dsip tracing` command:

```
AS5800# debug dsip trace
NIP tracing debugging is on
AS5800# show dsip tracing
NIP Control Packet Trace
-----------------------------------------------------------------------
Dest:00e0.b093.2238 Src:0007.4c72.0058 Type:200B SrcShelf:1 SrcSlot:11
MsgType:0 MsgLen:82 Timestamp: 00:49:14
-----------------------------------------------------------------------
Dest:00e0.b093.2238 Src:0007.4c72.0028 Type:200B SrcShelf:1 SrcSlot:5
MsgType:0 MsgLen:82 Timestamp: 00:49:14
-----------------------------------------------------------------------
```

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug modem</code></td>
<td>Displays information about the dial shelf, including clocking information.</td>
</tr>
<tr>
<td><code>show dsip tracing</code></td>
<td>Displays DSIP media header information logged using the <code>debug dsip trace</code> command.</td>
</tr>
</tbody>
</table>
debug dspapi

Effective with release 12.3(8)T, the `debug dspapi` command is replaced by the `debug voip dspapi` command. See the `debug voip dspapi` command for more information.

To enable debugging for Digital Signal Processor (DSP) application programming interface (API) message events, use the `debug dspapi` command in privileged EXEC mode. To reset the default value for this feature, use the `no` form of this command.

```
debug dspapi {all | command | detail | error | notification | response}
no debug dspapi {all | command | detail | error | notification | response}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Enables all <code>debug dspapi</code> options (command, detail, error, notification and response).</td>
</tr>
<tr>
<td>command</td>
<td>Displays commands sent to the DSPs.</td>
</tr>
<tr>
<td>detail</td>
<td>Displays additional detail for the DSP API debugs enabled.</td>
</tr>
<tr>
<td>error</td>
<td>Displays any DSP API errors.</td>
</tr>
<tr>
<td>notification</td>
<td>Displays notification messages sent from the DSP (for example, tone detection notification).</td>
</tr>
<tr>
<td>response</td>
<td>Displays responses sent by the DSP (for example, responses to statistic requests).</td>
</tr>
</tbody>
</table>

### Command Default

This command is not enabled.

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(5)XM</td>
<td>This command was introduced on the Cisco AS5300 and Cisco AS5800.</td>
</tr>
<tr>
<td>12.1(5)XM1</td>
<td>This command was implemented on the Cisco AS5350 and Cisco AS5400.</td>
</tr>
<tr>
<td>12.2(2)T</td>
<td>This command was implemented on the Cisco 1700, Cisco 2600 series, Cisco 3600 series, and the Cisco 3810.</td>
</tr>
</tbody>
</table>
This command was integrated into Cisco IOS Release 12.2(8)T.

This command was replaced by the `debugvoipdspapi` command.

This command was integrated into Cisco IOS Release 12.2(33)SRA.

### Usage Guidelines
DSP API message events used to communicate with DSPs are intended for use with Connexant (Nextport) and Texas Instrument (54x) DSPs. This command severely impacts performance and should be used only for single-call debug capture.

### Examples
The following example shows how to enable debugging for all DSP API message events:

```
Router# debug dspapi all
```

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug hpi</td>
<td>Enables debugging for HPI message events.</td>
</tr>
</tbody>
</table>
debug dspfarm

To display digital signal processor (DSP) farm service debugging information, use the debug dspfarm command in privileged EXEC mode. To disable debugging output, use the no form of this command.

depass debug dspfarm {all| errors| events| packets}
no debug dspfarm

Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>All DSP-farm debug-trace information.</td>
</tr>
<tr>
<td>errors</td>
<td>DSP-farm errors.</td>
</tr>
<tr>
<td>events</td>
<td>DSP-farm events.</td>
</tr>
<tr>
<td>packets</td>
<td>DSP-farm packets.</td>
</tr>
</tbody>
</table>

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(5)YH</td>
<td>This command was introduced on the Cisco VG200.</td>
</tr>
<tr>
<td>12.2(13)T</td>
<td>This command was implemented on the Cisco 2600 series, Cisco 3620, Cisco 3640, Cisco 3660, and Cisco 3700 series.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

Usage Guidelines

The router on which this command is used must be equipped with one or more digital T1/E1 packet voice trunk network modules (NM-HDVs) or high-density voice (HDV) transcoding/conferencing DSP farms (NM-HDV-FARMS) to provide DSP resources.

Debugging is turned on for all DSP-farm-service sessions. You can debug multiple sessions simultaneously, with different levels of debugging for each.

Examples

The following is sample output from the debug dspfarm events command:

Router# debug dspfarm events
DSP Farm service events debugging is on
*Mar 1 00:45:51: Sent 180 bytes to DSP 4 channel 2
*Mar 1 00:45:53: Sent 180 bytes to DSP 4 channel 3
*Mar 1 00:46:05: dspfarm_delete_stream: sess_id 26, conn_id 2721, stream 63122004, in
sess 631143CC is freed

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug frame-relay vc-bundle</code></td>
<td>Sets debugging for SCCP and its applications at one of four levels.</td>
</tr>
<tr>
<td><code>dspfarm (DSP farm)</code></td>
<td>Enables DSP-farm service.</td>
</tr>
<tr>
<td><code>sccp</code></td>
<td>Enables SCCP and its associated transcoding and conferencing applications.</td>
</tr>
<tr>
<td><code>show dspfarm</code></td>
<td>Displays summary information about DSP resources.</td>
</tr>
</tbody>
</table>
debug dspu activation

To display information on downstream physical unit (DSPU) activation, use the `debug dspu activation` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug dspu activation [name]
no debug dspu activation [name]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>(Optional) The host or physical unit (PU) name designation.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC

**Usage Guidelines**

The `debug dspu activation` command displays all DSPU activation traffic. To restrict the output to a specific host or PU, include the host or PU `name` argument. You cannot turn off debugging output for an individual PU if that PU has not been named in the `debug dspu activation` command.

**Examples**

The following is sample output from the `debug dspu activation` command. Not all intermediate numbers are shown for the "activated" and "deactivated" logical unit (LU) address ranges.

```
Router# debug dspu activation
DSPU: LS HOST3745 connected
DSPU: PU HOST3745 activated
DSPU: LU HOST3745-2 activated
DSPU: LU HOST3745-3 activated
...
DSPU: LU HOST3745-253 activated
DSPU: LU HOST3745-254 activated
DSPU: LU HOST3745-2 deactivated
DSPU: LU HOST3745-3 deactivated
...
DSPU: LU HOST3745-253 deactivated
DSPU: LU HOST3745-254 deactivated
DSPU: LS HOST3745 disconnected
DSPU: PU HOST3745 deactivated
```

The table below describes the significant fields shown in the display.

**Table 78: debug dspu activation Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSPU</td>
<td>Downstream PU debugging message.</td>
</tr>
<tr>
<td>LS</td>
<td>Link station (LS) event triggered the message.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>PU</td>
<td>PU event triggered the message.</td>
</tr>
<tr>
<td>LU</td>
<td>LU event triggered the message.</td>
</tr>
<tr>
<td>HOST3745</td>
<td>Host name or PU name.</td>
</tr>
<tr>
<td>HOST3745-253</td>
<td>Host name or PU name and the LU address, separated by a dash.</td>
</tr>
<tr>
<td>connected activated disconnected deactivated</td>
<td>Event that occurred to trigger the message.</td>
</tr>
</tbody>
</table>

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug dspu packet</td>
<td>Displays information on a DSPU packet.</td>
</tr>
<tr>
<td>debug dspu state</td>
<td>Displays information on DSPU FSM state changes.</td>
</tr>
<tr>
<td>debug dspu trace</td>
<td>Displays information on DSPU trace activity.</td>
</tr>
</tbody>
</table>
**debug dspu packet**

To display information on a downstream physical unit (DSPU) packet, use the **debug dspu packet** command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

### Syntax Description

```plaintext
debug dspu packet [ name ]
no debug dspu packet [ name ]
```

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>name</code></td>
<td>(Optional) The host or PU name designation.</td>
</tr>
</tbody>
</table>

### Command Modes

Privileged EXEC

### Usage Guidelines

The **debug dspu packet** command displays all DSPU packet data flowing through the router. To restrict the output to a specific host or physical unit (PU), include the host or PU `name` argument. You cannot turn off debugging output for an individual PU if that PU has not been named in the debug dspu packet command.

### Examples

The following is sample output from the **debug dspu packet** command:

```
Router# debug dspu packet
DSPU: Rx: PU HOST3745 data length 12 data: 2D0003002BE16B80 000D0201
DSPU: Tx: PU HOST3745 data length 25 data: 2D000032BE1EB80 000D020100850000 000C060000010000 00
DSPU: Rx: PU HOST3745 data length 12 data: 2D0004002BE26B80 000D0201
DSPU: Tx: PU HOST3745 data length 25 data: 2D0000042BE2EB80 000D020100850000 000C060000010000 00
```

The table below describes the significant fields shown in the display.

### Table 79: debug dspu packet Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSPU: Rx:</td>
<td>Received frame (packet) from the remote PU to the router PU.</td>
</tr>
<tr>
<td>DSPU: Tx:</td>
<td>Transmitted frame (packet) from the router PU to the remote PU.</td>
</tr>
<tr>
<td>PU HOST3745</td>
<td>Host name or PU associated with the transmit or receive.</td>
</tr>
<tr>
<td>data length 12 data:</td>
<td>Number of bytes of data, followed by up to 128 bytes of displayed data.</td>
</tr>
</tbody>
</table>
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug drip event</code></td>
<td>Displays debugging messages for DRiP packets.</td>
</tr>
<tr>
<td><code>debug dspu state</code></td>
<td>Displays information on DSPU FSM state changes.</td>
</tr>
<tr>
<td><code>debug dspu trace</code></td>
<td>Displays information on DSPU trace activity.</td>
</tr>
</tbody>
</table>
**debug dspu state**

To display information on downstream physical unit (DSPU) finite state machine (FSM) state changes, use the `debug dspu state` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug dspu state [ name ]
no debug dspu state [ name ]
```

### Syntax Description

| name | (Optional) The host or physical unit (PU) name designation. |

### Command Modes

Privileged EXEC

### Usage Guidelines

Use the `debug dspu state` command to display only the FSM state changes. To see all FSM activity, use the `debug dspu trace` command. You cannot turn off debugging output for an individual PU if that PU has not been named in the `debug dspu state` command.

### Examples

The following is sample output from the `debug dspu state` command. Not all intermediate numbers are shown for the "activated" and "deactivated" logical unit (LU) address ranges.

```
Router# debug dspu state
DSPU: LS HOST3745: input=StartLs, Reset -> PendConOut
DSPU: LS HOST3745: input=ReqOpn.Cnf, PendConOut -> Xid
DSPU: LS HOST3745: input=Connect.Ind, Xid -> ConnIn
DSPU: LS HOST3745: input=Connected.Ind, ConnIn -> Connected
DSPU: PU HOST3745: input=Actpu, Reset -> Active
DSPU: LU HOST3745-2: input=uActlu, Reset -> upLuActive
DSPU: LU HOST3745-3: input=uActlu, Reset -> upLuActive
.. DSPU: LU HOST3745-253: input=uActlu, Reset -> upLuActive
DSPU: LU HOST3745-254: input=uActlu, Reset -> upLuActive
DSPU: LS HOST3745: input=PuStopped, Connected -> PendDisc
DSPU: LS HOST3745: input=Disc.Cnf, PendDisc -> PendClose
DSPU: LS HOST3745: input=Close.Cnf, PendClose -> Reset
DSPU: PU HOST3745: input=T2ResetPu, Active -> Reset
DSPU: LU HOST3745-2: input=uStopLu, upLuActive -> Reset
DSPU: LU HOST3745-3: input=uStopLu, upLuActive -> Reset
.. DSPU: LU HOST3745-253: input=uStopLu, upLuActive -> Reset
DSPU: LU HOST3745-254: input=uStopLu, upLuActive -> Reset
```

The table below describes the significant fields shown in the display.
Table 80: debug dspu state Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSPU</td>
<td>Downstream PU debug message.</td>
</tr>
<tr>
<td>LS</td>
<td>Link station (LS) event triggered the message.</td>
</tr>
<tr>
<td>PU</td>
<td>PU event triggered the message.</td>
</tr>
<tr>
<td>LU</td>
<td>LU event triggered the message.</td>
</tr>
<tr>
<td>HOST3745-253</td>
<td>Host name or PU name and LU address.</td>
</tr>
<tr>
<td>input=input,</td>
<td>Input received by the FSM.</td>
</tr>
<tr>
<td>previous-state, -&gt; current-state</td>
<td>Previous state and current new state as seen by the FSM.</td>
</tr>
</tbody>
</table>

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug drip event</td>
<td>Displays debugging messages for DRiP packets.</td>
</tr>
<tr>
<td>debug drip packet</td>
<td>Displays information on DSPU packet.</td>
</tr>
<tr>
<td>debug dspu trace</td>
<td>Displays information on DSPU trace activity.</td>
</tr>
</tbody>
</table>
debug dspu trace

To display information on downstream physical unit (DSPU) trace activity, which includes all finite state machine (FSM) activity, use the `debug dspu trace` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug dspu trace [ name ]
no debug dspu trace [ name ]
```

**Syntax Description**

<table>
<thead>
<tr>
<th><code>name</code></th>
<th>(Optional) The host or physical unit (PU) name designation.</th>
</tr>
</thead>
</table>

**Command Modes**

Privileged EXEC

**Usage Guidelines**

Use the `debug dspu trace` command to display all FSM state changes. To see FSM state changes only, use the `debug dspustate` command. You cannot turn off debugging output for an individual PU if that PU has not been named in the `debug dspu trace` command.

**Examples**

The following is sample output from the `debug dspu trace` command:

```
Router# debug dspu trace
DSPU: LS HOST3745 input = 0 ->(1,a1)
DSPU: LS HOST3745 input = 5 ->(5,a6)
DSPU: LS HOST3745 input = 7 ->(5,a9)
DSPU: LS HOST3745 input = 9 ->(5,a28)
DSPU: LU HOST3745-2 in:0 s:0->(2,a1)
DSPU: LS HOST3745 input = 19 ->(8,a20)
DSPU: LS HOST3745 input = 18 ->(8,a17)
DSPU: LU HOST3745-3 in:0 s:0->(2,a1)
DSPU: LS HOST3745 input = 19 ->(8,a20)
DSPU: LS HOST3745 input = 18 ->(8,a17)
DSPU: LU HOST3745-252 in:0 s:0->(2,a1)
DSPU: LS HOST3745 input = 19 ->(8,a20)
DSPU: LS HOST3745 input = 18 ->(8,a17)
DSPU: LU HOST3745-254 in:0 s:0->(2,a1)
DSPU: LS HOST3745 input = 19 ->(8,a20)
```

The table below describes significant fields shown in the output.

**Table 81: debug dspu trace Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:23:57</td>
<td>Time stamp.</td>
</tr>
<tr>
<td>DSPU</td>
<td>Downstream PU debug message.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>LS</td>
<td>Link station (LS) event triggered the message.</td>
</tr>
<tr>
<td>PU</td>
<td>A PU event triggered the message.</td>
</tr>
<tr>
<td>LU</td>
<td>LU event triggered the message.</td>
</tr>
<tr>
<td>HOST3745-253</td>
<td>Host name or PU name and LU address.</td>
</tr>
<tr>
<td>in:&lt;inputs&gt;:state-&gt;(new-state, action)</td>
<td>String describing the following:</td>
</tr>
<tr>
<td></td>
<td>• input --LU FSM input</td>
</tr>
<tr>
<td></td>
<td>• state --Current FSM state</td>
</tr>
<tr>
<td></td>
<td>• new-state --New FSM state</td>
</tr>
<tr>
<td></td>
<td>• action --FSM action</td>
</tr>
<tr>
<td>input=input-&gt;(new-state,action)</td>
<td>String describing the following:</td>
</tr>
<tr>
<td></td>
<td>• input --PU or LS FSM input</td>
</tr>
<tr>
<td></td>
<td>• new-state --New PU or LS FSM state</td>
</tr>
<tr>
<td></td>
<td>• action --PU or LS FSM action</td>
</tr>
</tbody>
</table>

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug drip event</td>
<td>Displays debugging messages for DRiP packets.</td>
</tr>
<tr>
<td>debug drip packet</td>
<td>Displays information on DSpU packet.</td>
</tr>
<tr>
<td>debug dspu state</td>
<td>Displays information on DSpU FSM state changes.</td>
</tr>
</tbody>
</table>
debug dss ipx event

To display debugging messages for route change events that affect Internetwork Packet Exchange (IPX) Multilayer Switching (MLS), use the debug dss ipx event command in privileged EXEC mode. To disable debugging output, use the no form of this command.

debug dss ipx event
no debug dss ipx event

Syntax Description
This command has no arguments or keywords.

Command Default
Debugging is not enabled.

Command Modes
Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(5)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

Examples

The following is sample output from the debug dss ipx event command:

Router#

    debug dss ipx event
    DSS IPX events debugging is on
    Router# configure terminal
    Enter configuration commands, one per line. End with CNTL/Z.
    Router(config)# interface vlan 22
    Router(config-if)# ipx access-group 800 out
    05:51:36:DSS-feature:dss_ipxcache_version():idb:NULL, reason:42,
    prefix:0, mask:FFFFFFFF
    05:51:36:DSS-feature:dss_ipx_access_group():idb:Vlan22
    05:51:36:DSS-feature:dss_ipx_access_list()
    05:51:36:DSS-feature:dss_ipx_access_list()
    05:51:36:DSS-feature:dss_ipx_access_list()
    05:51:36:DSS-feature:dss_ipx_access_list()
    05:51:36:DSS-feature:dss_ipx_flowmask_reg 2
    05:51:36:%IPX mls flowmask transition from 1 to 2 due to new status of
    simple IPX access list on interfaces

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug mls rp</td>
<td>Displays various MLS debugging elements.</td>
</tr>
</tbody>
</table>