

debug fax relay t30

To display debugging messages for T.30 real-time fax, use the **debug fax relay t30** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug fax relay t30 {all | calling-number string | called-number string}
```

```
no debug fax relay t30
```

Syntax Description		
all		Enables debugging for all incoming and outgoing calls.
calling-number		Enables debugging for incoming numbers that begin with a specified string of digits.
called-number		Enables debugging for outgoing numbers that begin with a specified string of digits.
<i>string</i>		Digits that specify the incoming or outgoing number.

Defaults No default behavior or values.

Command Modes Privileged EXEC

Command History	Release	Modification
	12.2(2)XB1	The debug fax relay t30 command was introduced on Cisco AS5350, Cisco AS5400, and Cisco AS5850 access servers.
	12.2(11)T	This command was integrated into Cisco IOS Release 12.2(11)T for the Cisco AS5350, Cisco AS5400, and Cisco AS5850 access servers.

Usage Guidelines The incoming or outgoing numbers must be a valid E.164 destination. The period symbol (.) as a wildcard should not be used. Instead of a wildcard, leave the space blank to indicate that any numbers can be valid.

There are no limits to the number of debug entries. The number entered generates a match if the calling or called number matches up to the final number of the debug entry. For example, the 408555 entry would match 408555, 4085551, 4085551212, or any other number starting with 408555.

Examples The following command enables debugging for any incoming calls that start with 408555:

```
Router# debug fax relay t30 calling-number 408555
```

```
Debugging fax relay t30 from 408555
```

The following command enables debugging for any calls received to a number starting with 555-1212:

```
Router# debug fax relay t30 called-number 4155551212
```

```
Debugging fax relay t30 to 4155551212
```

The following command displays all debug entries:

```
Router# debug fax relay t30 all
```

```
Debugging fax relay t30 from 408555
```

```
Debugging fax relay t30 to 4155551212
```

debug fddi smt-packets

To display information about Station Management (SMT) frames received by the router, use the **debug fddi smt-packets** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug fddi smt-packets

no debug fddi smt-packets

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Examples

The following is sample output from the **debug fddi smt-packets** command. In this example, an SMT frame has been output by FDDI 1/0. The SMT frame is a next station addressing (NSA) neighbor information frame (NIF) request frame with the parameters as shown.

```
Router# debug fddi smt-packets

SMT O: Fddi1/0, FC=NSA, DA=ffff.ffff.ffff, SA=00c0.eeee.be04,
class=NIF, type=Request, vers=1, station_id=00c0.eeee.be04, len=40
- code 1, len 8 -- 000000016850043F
- code 2, len 4 -- 00010200
- code 3, len 4 -- 00003100
- code 200B, len 8 -- 0000000100000000
```

[Table 61](#) describes the significant fields shown in the display.

Table 61 *debug fddi smt-packets Field Descriptions*

Field	Description
SMT O	SMT frame was sent from FDDI interface 1/0. Also, SMT I indicates that an SMT frame was received on the FDDI interface 1/0.
Fddi1/0	Interface associated with the frame.
FC	Frame control byte in the MAC header.
DA, SA	Destination and source addresses in FDDI form.
class	Frame class. Values can be echo frame (ECF), neighbor information frame (NIF), parameter management frame (PMF), request denied frame (RDF), status information frame (SIF), and status report frame (SRF).
type	Frame type. Values can be Request, Response, and Announce.
vers	Version identification. Values can be 1 or 2.
station_id	Station identification.

Table 61 *debug fddi smt-packets Field Descriptions (continued)*

Field	Description
len	Packet size.
code 1, len 8 -- 000000016850043F	Parameter type X'0001—upstream neighbor address (UNA), parameter length in bytes, and parameter value. SMT parameters are described in the SMT specification ANSI X3T9.

debug fmsp receive

To display debugging messages for Fax Media Services Provider (FMSP) receive, use the **debug fmsp receive** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug fmsp receive [t30 | t38]

no debug fmsp receive [t30 | t38]

Syntax Description

t30	(Optional) Specifies the T.30 fax protocol.
t38	(Optional) Specifies the T.38 fax protocol.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(3)XI	This command was introduced on the Cisco AS5300 access server.
12.2(8)T	This command was implemented on the Cisco 1751 access routers, Cisco 3725 access routers, and Cisco 3745 access routers.
12.2(13)T	Support for this command was implemented in Cisco 7200 series images.

Examples

The following is sample output from the **debug fmsp receive** command:

```
Router# debug fmsp receive

*Oct 16 08:31:33.243: faxmsp_call_setup_request: call id=28
*Oct 16 08:31:33.243: faxmsp_call_setup_request: ramp data dir=ONRAMP, conf dir=DEST
*Oct 16 08:31:33.243: faxmsp_bridge(): cfid=19, srccid=28, dstcid=27

*Oct 16 08:31:33.243: faxmsp_bridge(): ramp data dir=ONRAMP, conf dir=DEST
*Oct 16 08:31:33.243: faxmsp_bridge(): Explicit caps ind. done; will wait for registry
caps ind
*Oct 16 08:31:33.243: faxmsp_caps_ind: call id=28, src=27
*Oct 16 08:31:33.243: faxmsp_caps_ack: call id src=27
*Oct 16 08:31:33.279: faxmsp_call_setup_request: call id=29
*Oct 16 08:31:33.279: faxmsp_call_setup_request: ramp data dir=OFFRAMP, conf dir=SRC
*Oct 16 08:31:33.283: faxmsp_bridge(): cfid=20, srccid=29, dstcid=26

*Oct 16 08:31:33.283: faxmsp_bridge(): ramp data dir=OFFRAMP, conf dir=SRC
*Oct 16 08:31:33.283: faxmsp_bridge(): Explicit caps ind. done; will wait for registry
caps ind
*Oct 16 08:31:33.283: faxmsp_caps_ind: call id=29, src=26
*Oct 16 08:31:33.283: faxmsp_caps_ack: call id src=26
*Oct 16 08:31:33.635: faxmsp_codec_download_done: call id=29
*Oct 16 08:31:33.635: faxmsp_codec_download_done: call id=28
```

```
*Oct 16 08:31:33.643: faxmsp_xmit: callid src=26, dst=29
*Oct 16 08:31:33.643: faxmsp_xmit: callid src=27, dst=28
*Oct 16 08:31:33.643: faxmsp_process_rcv_data: call id src=26, dst=29
```

Related Commands

Command	Description
debug fmosp send	Displays debugging messages for FMSP send.

debug fmsp send

To display debugging messages for Fax Media Services Provider (FMSP) send, use the **debug fmsp send command** in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug fmsp send [t30 | t38]

no debug fmsp send [t30 | t38]

Syntax Description	t30	(Optional) Specifies the T.30 fax protocol.
	t38	(Optional) Specifies the T.38 fax protocol.

Defaults No default behavior or values.

Command Modes Privileged EXEC

Command History	Release	Modification
	12.1(3)XI	This command was introduced on the Cisco AS5300 access server.
	12.2(8)T	This command was implemented on the Cisco 1751 access routers, Cisco 3725 access routers, and Cisco 3745 access routers.
	12.2(13)T	This feature was implemented on the Cisco 7200 series routers.

Examples The following is sample output from the **debug fmsp send** command:

```
Router# debug fmsp send

Jan 1 05:02:56.782: faxmsp_call_setup_request: call id=21
Jan 1 05:02:56.782: faxmsp_call_setup_request: ramp data dir=OFFRAMP, conf dir=SRC
Jan 1 05:02:56.782: faxmsp_bridge(): cfid=7, srccid=21, dstcid=20

Jan 1 05:02:56.782: faxmsp_bridge(): ramp data dir=OFFRAMP, conf dir=SRC
Jan 1 05:02:56.782: faxmsp_bridge(): Explicit caps ind. done; will wait for registry caps
ind
Jan 1 05:02:56.782: faxmsp_caps_ind: call id=21, src=20
Jan 1 05:02:56.782: faxmsp_caps_ack: call id src=20
Jan 1 05:02:57.174: faxmsp_codec_download_done: call id=21
Jan 1 05:02:57.174: faxMsp_tx_buffer callID=21
Jan 1 05:02:57.178: faxMsp_tx_buffer callID=21
Jan 1 05:02:57.178: faxMsp_tx_buffer callID=21
Jan 1 05:02:57.178: faxMsp_tx_buffer callID=21
Jan 1 05:02:57.182: faxmsp_xmit: callid src=20, dst=21
Jan 1 05:02:57.182: faxmsp_process_rcv_data: call id src=20, dst=21
Jan 1 05:03:01.814: faxmsp_xmit: callid src=20, dst=21
Jan 1 05:03:01.814: faxmsp_process_rcv_data: call id src=20, dst=21
Jan 1 05:03:01.814: faxMsp_tx_buffer callID=21
Jan 1 05:03:02.802: faxmsp_xmit: callid src=20, dst=21
Jan 1 05:03:02.802: faxmsp_process_rcv_data: call id src=20, dst=21
Jan 1 05:03:02.822: faxmsp_xmit: callid src=20, dst=21
```

```
Jan 1 05:03:02.822: faxmsp_process_rcv_data: call id src=20, dst=21
Jan 1 05:03:02.854: faxmsp_xmit: callid src=20, dst=21
Jan 1 05:03:02.854: faxmsp_process_rcv_data: call id src=20, dst=21
```

Related Commands

Command	Description
debug fax relay t30	Displays debugging messages for FMSP receive.

debug foip off-ramp

To display debugging messages for off-ramp fax mail, use the **debug foip off-ramp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug foip off-ramp

no debug foip off-ramp

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(3)XI	This command was introduced on the Cisco AS5300 access server.
12.2(8)T	This command was introduced on the Cisco 1751 access routers, Cisco 3725 access routers, and Cisco 3745 access routers.
12.2(13)T	This feature was implemented on the Cisco 7200 series routers.

Examples

The following is sample output from the **debug foip off-ramp** command:

```
Router# debug foip off-ramp

Jan  1 02:31:17.539: lapp off: CC_EV_CALL_HANDOFF, cid(0xB)
Jan  1 02:31:17.539: loffHandoff: called number=5271714, callid=0xB
Jan  1 02:31:17.543: loffSetupPeer: cid1(0xB)
Jan  1 02:31:17.543:  destPat (5271714),matched(1),pref(5),tag(20),encap(1)
Jan  1 02:31:22.867: lapp off: CC_EV_CALL_CONNECTED, cid(0xC)
Jan  1 02:31:22.867:  st=CALL_SETTING cid(0xB,0x0,0x0,0xC),cfid(0x0,0x0,0x0)
Jan  1 02:31:22.867: loffConnected
Jan  1 02:31:22.867: loffFlushPeerTagQueue cid(11) peer list: (empty)
Jan  1 02:31:22.867: lapp off: CC_EV_CONF_CREATE_DONE, cid(0xC), cid2(0xD), cfid(0x1)
Jan  1 02:31:22.867:  st=CONFERENCING3 cid(0xB,0x0,0xD,0xC),cfid(0x0,0x0,0x1)
Jan  1 02:31:22.867: loffConfDone3
Jan  1 02:31:30.931: lapp off: CC_EV_FROM_FMSP_ON_CALL_DETAIL, cid(0xD)
Jan  1 02:31:30.931:  st=WAIT_SESS_INFO cid(0xB,0x0,0xD,0xC),cfid(0x0,0x0,0x1)
Jan  1 02:31:30.931: loffSessionInfo
Jan  1 02:31:30.931:  encd=2, res1=2, spd=26, min_scan_len=0, csid=          4085271714
Jan  1 02:31:30.931: lapp off: CC_EV_CONF_CREATE_DONE, cid(0xD), cid2(0xE), cfid(0x2)
Jan  1 02:31:30.931:  st=CONFERENCING2 cid(0xB,0xE,0xD,0xC),cfid(0x0,0x2,0x1)
Jan  1 02:31:30.931: loffConfDone2
```

Related Commands

Command	Description
debug foip on-ramp	Displays debugging messages for on-ramp fax mail.

debug foip on-ramp

To display debugging messages for on-ramp fax mail, use the **debug foip on-ramp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug foip on-ramp

no debug foip on-ramp

Syntax Description This command has no arguments or keywords.

Defaults No default behavior or values.

Command Modes Privileged EXEC

Command History	Release	Modification
	12.1(3)XI	This command was introduced on the Cisco AS5300 access server.
	12.2(8)T	This command was introduced on the Cisco 1751 access routers, Cisco 3725 access routers, and Cisco 3745 access routers.
	12.2(13)T	This feature was implemented on the Cisco 7200 series routers.

Examples The following is sample output from the **debug foip on-ramp** command:

```
Router# debug foip on-ramp

*Oct 16 08:07:01.947: lapp_on_application: Incoming Event: (15 = CC_EV_CALL_HANDOFF),
CID(11), DISP(0)
*Oct 16 08:07:01.947: lapp_on_call_handoff: Authentication enabled = FALSE
*Oct 16 08:07:01.947: lapp_on_call_handoff: Authentication ID = 0
*Oct 16 08:07:01.947: lapp_on_call_handoff: Authentication ID source = IVR or unknown
*Oct 16 08:07:01.947: lapp_on_call_handoff: Authentication status = SUCCESS
*Oct 16 08:07:01.947: lapp_on_call_handoff: Accounting enabled = FALSE
*Oct 16 08:07:01.947: lapp_on_call_handoff: Accounting method list = fax
*Oct 16 08:07:01.947: lapp_on_conference_vtsp_fmosp: Begin conferencing VTSP and FMSP...
*Oct 16 08:07:01.951: lapp_on_change_state: old state(0) new state(1)
*Oct 16 08:07:01.951: lapp_on_application: Incoming Event: (29 = CC_EV_CONF_CREATE_DONE),
CID(11), DISP(0)
*Oct 16 08:07:01.951: lapp_on_application: Current call state = 1
*Oct 16 08:07:01.951: lapp_on_conference_created: The VTSP and the FMSP are conferenced
*Oct 16 08:07:01.951: lapp_on_conference_created: Wait for FMSP call detail event
```

Related Commands	Command	Description
	debug foip off-ramp	Displays debugging messages for off-ramp fax mail.

debug frame-relay

To display debugging information about the packets received on a Frame Relay interface, use the **debug frame-relay** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug frame-relay

no debug frame-relay

Syntax Description This command has no arguments or keywords.

Defaults This command is disabled by default.

Command Modes Privileged EXEC

Command History	Release	Modification
	9.00	This command was introduced.
	12.2(13)T	Support for Banyan VINES was removed.

Usage Guidelines This command helps you analyze the packets that have been received. However, because the **debug frame-relay** command generates a substantial amount of output, use it only when the rate of traffic on the Frame Relay network is less than 25 packets per second.

To analyze the packets that have been *sent* on a Frame Relay interface, use the **debug frame-relay packet** command.

Examples The following is sample output from the **debug frame-relay** command:

```
Router# debug frame-relay

Serial0(i): dlcI 500(0x7C41), pkt type 0x809B, datagramsize 24
Serial1(i): dlcI 1023(0xFCF1), pkt type 0x309, datagramsize 13
Serial0(i): dlcI 500(0x7C41), pkt type 0x809B, datagramsize 24
Serial1(i): dlcI 1023(0xFCF1), pkt type 0x309, datagramsize 13
Serial0(i): dlcI 500(0x7C41), pkt type 0x809B, datagramsize 24
```

Table 62 describes the significant fields shown in the display.

Table 62 *debug frame-relay Field Descriptions*

Field	Description
Serial0(i):	Indicates that serial interface 0 has received this Frame Relay datagram as input.
dcli 500(0x7C41)	Indicates the value of the data-link connection identifier (DLCI) for this packet in decimal (and q922). In this case, 500 has been configured as the multicast DLCI.

Table 62 *debug frame-relay Field Descriptions (continued)*

Field	Description
pkt type 0x809B	<p>Indicates the packet type code.</p> <p>Possible supported signalling message codes are as follows:</p> <ul style="list-style-type: none"> • 0x308—Signalling message; valid only with a DLCI of 0 • 0x309—LMI message; valid only with a DLCI of 1023 <p>Possible supported Ethernet type codes are:</p> <ul style="list-style-type: none"> • 0x0201—IP on a 3 MB net • 0x0201—Xerox ARP on 10 MB networks • 0xCC—RFC 1294 (only for IP) • 0x0600—XNS • 0x0800—IP on a 10 MB network • 0x0806—IP ARP • 0x0808—Frame Relay Address Resolution Protocol (ARP) <p>Possible High-Level Data Link Control (HDLC) type codes are as follows:</p> <ul style="list-style-type: none"> • 0x6001—DEC Maintenance Operation Protocol (MOP) booting protocol • 0x6002—DEC MOP console protocol • 0x6003—DECnet Phase IV on Ethernet • 0x6004—DEC LAT on Ethernet • 0x8005—HP Probe • 0x8035—RARP • 0x8038—DEC spanning tree • 0x809b—Apple EtherTalk • 0x80f3—AppleTalk ARP • 0x8019—Apollo domain • 0x8137—Internetwork Packet Exchange (IPX) • 0x9000—Ethernet loopback packet IP • 0x1A58—IPX, standard form • 0xFEFE—Connectionless Network Service (CLNS) • 0xEFEF—End System-to-Intermediate System (ES-IS) • 0x1998—Uncompressed TCP • 0x1999—Compressed TCP • 0x6558—Serial line bridging
datagramsize 24	Indicates size of this datagram (in bytes).

debug frame-relay adjacency

To display information pertaining to an adjacent node that has one or more Frame Relay permanent virtual circuit (PVC) bundles, use the **debug frame-relay adjacency** command in privileged EXEC mode. To stop displaying the adjacent node information, use the **no** form of this command.

debug frame-relay adjacency { **pvc** [*dlci*] | **vc-bundle** [*vc-bundle-name*]}

no debug frame-relay adjacency { **pvc** [*dlci*] | **vc-bundle** [*vc-bundle-name*]}

Syntax Description

pvc	Displays information regarding the adjacent PVC only.
<i>dlci</i>	(Optional) Data-link connection identifier for a specific PVC.
vc-bundle	Displays information regarding the adjacent PVC bundle and its members.
<i>vc-bundle-name</i>	(Optional) Name of the PVC bundle.

Defaults

No default behaviors or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(13)T	This command was introduced.

Usage Guidelines

Use this command to monitor adjacency activity and status for an adjacent node.



Note

Debug messages that are prefixed with “FR_ADJ” (instead of “FR-ADJ”) indicate serious failures in the Frame Relay PVC bundle performance. Contact the Cisco Technical Assistance Center (TAC) if you see debugging messages with this prefix.

Examples

The following sample output from the **debug frame-relay adjacency vc-bundle** command shows PVC bundle “MP-4-dynamic” going down. Each bundle member PVC is marked for removal from the CEF adjacency table, and then the adjacency for the PVC bundle itself is marked for removal. The adjacencies are actually removed from the table later when a background clean-up process runs.

```
Router# debug frame-relay adjacency vc-bundle MP-4-dynamic

00:46:35: FR-ADJ: vcb MP-4-dynamic: ip 14.2.2.2: member 400: removing adj
00:46:35: FR-ADJ: vcb MP-4-dynamic: ip 14.2.2.2: member 401: removing adj
00:46:35: FR-ADJ: vcb MP-4-dynamic: ip 14.2.2.2: member 402: removing adj
00:46:35: FR-ADJ: vcb MP-4-dynamic: ip 14.2.2.2: member 403: removing adj
00:46:35: FR-ADJ: vcb MP-4-dynamic: ip 14.2.2.2: member 404: removing adj
00:46:35: FR-ADJ: vcb MP-4-dynamic: ip 14.2.2.2: member 405: removing adj
00:46:35: FR-ADJ: vcb MP-4-dynamic: ip 14.2.2.2: member 406: removing adj
00:46:35: FR-ADJ: vcb MP-4-dynamic: ip 14.2.2.2: member 407: removing adj
00:46:35: FR-ADJ: vcb MP-4-dynamic: ip 14.2.2.2: removing primary adj
```

Related Commands

Command	Description
debug frame-relay vc-bundle	Displays information pertaining to all the PVC bundles configured on the router.

debug frame-relay callcontrol

To display Frame Relay Layer 3 (network layer) call control information, use the **debug frame-relay callcontrol** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug frame-relay callcontrol

no debug frame-relay callcontrol

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

Usage Guidelines The **debug frame-relay callcontrol** command is used specifically for observing FRF.4/Q.933 signalling messages and related state changes. The FRF.4/Q.933 specification describes a state machine for call control. The signalling code implements the state machine. The debug statements display the actual event and state combinations.

The Frame Relay switched virtual circuit (SVC) signalling subsystem is an independent software module. When used with the **debug frame-relay networklayerinterface** command, the **debug frame-relay callcontrol** command provides a better understanding of the call setup and teardown sequence. The **debug frame-relay networklayerinterface** command provides the details of the interactions between the signalling subsystem on the router and the Frame Relay subsystem.

Examples State changes can be observed during a call setup on the calling party side. The **debug frame-relay networklayerinterface** command shows the following state changes or transitions:

```
STATE_NULL -> STATE_CALL_INITIATED -> STATE_CALL_PROCEEDING->STATE_ACTIVE
```

The following messages are samples of output generated during a call setup on the calling side:

```
6d20h: U0_SetupRequest: Serial0
6d20h: L3SDL: Ref: 1, Init: STATE_NULL, Rcvd: SETUP_REQUEST, Next: STATE_CALL_INITIATED
6d20h: U1_CallProceeding: Serial0
6d20h: L3SDL: Ref: 1, Init: STATE_CALL_INITIATED, Rcvd: MSG_CALL_PROCEEDING, Next:
STATE_CALL_PROCEEDING
6d20h: U3_Connect: Serial0
6d20h: L3SDL: Ref: 1, Init: STATE_CALL_PROCEEDING, Rcvd: MSG_CONNECT, Next: STATE_ACTIVE
6d20h:
```

The following messages are samples of output generated during a call setup on the called party side. Note the state transitions as the call goes to the active state:

```
STATE_NULL -> STATE_CALL_PRESENT-> STATE_INCOMING_CALL_PROCEEDING->STATE_ACTIVE
```

```
1w4d: U0_Setup: Serial2/3
1w4d: L3SDL: Ref: 32769, Init: STATE_NULL, Rcvd: MSG_SETUP, Next: STATE_CALL_PRESENT 1w4d:
L3SDL: Ref: 32769, Init: STATE_CALL_PRESENT, Rcvd: MSG_SETUP, Next:
STATE_INCOMING_CALL_PROC 1w4d: L3SDL: Ref: 32769, Init: STATE_INCOMING_CALL_PROC,
Rcvd: MSG_SETUP, Next: STATE_ACTIVE
```

Table 63 explains the possible call states.

Table 63 *Frame Relay Switched Virtual Circuit Call States*

Call State	Description
Null	No call exists.
Call Initiated	User has requested the network to establish a call.
Outgoing Call Proceeding	User has received confirmation from the network that the network has received all call information necessary to establish the call.
Call Present	User has received a request to establish a call but has not yet responded.
Incoming Call Proceeding	User has sent acknowledgment that all call information necessary to establish the call has been received (for an incoming call).
Active	On the called side, the network has indicated that the calling user has been awarded the call. On the calling side, the remote user has answered the call.
Disconnect Request	User has requested that the network clear the end-to-end call and is waiting for a response.
Disconnect Indication	User has received an invitation to disconnect the call because the network has disconnected the call.
Release Request	User has requested that the network release the call and is waiting for a response.

Related Commands

Command	Description
debug fax relay t30	Displays debugging information about the packets that are received on a Frame Relay interface.
debug frame-relay networklayerinterface	Displays NLI information.

debug frame-relay end-to-end keepalive

To display debug messages for the Frame Relay End-to-End Keepalive feature, use the **debug frame-relay end-to-end keepalive** command. Use the **no** form of this command to disable the display of debug messages.

debug frame-relay end-to-end keepalive { events | packet }

no debug frame-relay end-to-end keepalive { events | packet }

Syntax Description

events	Displays keepalive events.
packet	Displays keepalive packets sent and received.

Command History

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

Usage Guidelines

We recommend that both commands be enabled.

Examples

The following examples show typical output from the **debug frame-relay end-to-end keepalive packet** command. The following example shows output for an outgoing request packet:

```
EEK (o, Serial0.1 DLCI 200): 1 1 1 3 2 4 3
```

The seven number fields that follow the colon signify the following:

Field	Description
first (example value = 1)	Information Element (IE) type.
second (example value = 1)	IE length.
third (example value = 1)	Report ID. 1 = request, 2 = reply.
fourth (example value = 3)	Next IE type. 3 = LIV ID (Keepalive ID).
fifth (example value = 2)	IE length. (This IE is a Keepalive IE.)
sixth (example value = 4)	Send sequence number.
seventh (example value = 3)	Receive sequence number.

The following example shows output for an incoming reply packet:

```
EEK (i, Serial0.1 DLCI 200): 1 1 2 3 2 4 4
```

The seven number fields that follow the colon signify the following:

Field	Description
first (example value = 1)	Information Element (IE) type.
second (example value = 1)	IE length.
third (example value = 2)	Report ID. 1 = request, 2 = reply.

Field	Description
fourth (example value = 3)	Next IE type. 3 = LIV ID (Keepalive ID).
fifth (example value = 2)	IE length. (This IE is a Keepalive IE.)
sixth (example value = 4)	Send sequence number.
seventh (example value = 4)	Receive sequence number.

The following example shows typical output from the **debug frame-relay end-to-end keepalive events** command:

```
EEK SUCCESS (request, Serial0.2 DLCI 400)
EEK SUCCESS (reply, Serial0.1 DLCI 200)
EEK sender timeout (Serial0.1 DLCI 200)
```

debug frame-relay events

To display debugging information about Frame Relay Address Resolution Protocol (ARP) replies on networks that support a multicast channel and use dynamic addressing, use the **debug frame-relay events** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug frame-relay events

no debug frame-relay events

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

Command History	Release	Modification
	11.3	This command was introduced.
	12.0(23)S	This command was integrated into Cisco IOS Release 12.0(23)S for the Frame Relay over MPLS feature.
	12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.
	12.2(15)T	This command was integrated into Cisco IOS Release 12.2(15)T.

Usage Guidelines This command is useful for identifying the cause of end-to-end connection problems during the installation of a Frame Relay network or node.



Note

Because the **debug frame-relay events** command does not generate much output, you can use it at any time, even during periods of heavy traffic, without adversely affecting other users on the system.

Examples The following is sample output from the **debug frame-relay events** command:

```
Router# debug frame-relay events

Serial2(i): reply rcvd 172.16.170.26 126
Serial2(i): reply rcvd 172.16.170.28 128
Serial2(i): reply rcvd 172.16.170.34 134
Serial2(i): reply rcvd 172.16.170.38 144
Serial2(i): reply rcvd 172.16.170.41 228
Serial2(i): reply rcvd 172.16.170.65 325
```

As the output shows, the **debug frame-relay events** command returns one specific message type. The first line, for example, indicates that IP address 172.16.170.26 sent a Frame Relay ARP reply; this packet was received as input on serial interface 2. The last field (126) is the data-link connection identifier (DLCI) to use when communicating with the responding router.

For Frame Relay over MPLS, the following is sample output for the **debug frame-relay events** command. The command output shows the status of the VCs.

```
Router# debug frame-relay events
```

```
Frame Relay events debugging is on
```

This example shows the messages that are displayed when you shut the core-facing interface on a provider edge (PE) router:

```
04:40:38:%SYS-5-CONFIG_I: Configured from console by consolenf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# interface hssi2/0
Router(config-if)# shut

04:40:43:%OSPF-5-ADJCHG: Process 10, Nbr 12.12.12.12 on Hssi2/0 from FULL to DOWN,
Neighbor Down: Interface down or detached
04:40:43: FROMPLS [12.12.12.12, 100]: PW pvc_status set INACTIVE
04:40:43: FROMPLS [12.12.12.12, 100]: Setting pw segment DOWN
04:40:43: FROMPLS [12.12.12.12, 100]: Setting connection DOWN
04:40:43: FROMPLS [12.12.12.12, 101]: PW pvc_status set INACTIVE
04:40:43: FROMPLS [12.12.12.12, 101]: Setting pw segment DOWN
04:40:43: FROMPLS [12.12.12.12, 101]: Setting connection DOWN
04:40:45:%LINK-5-CHANGED: Interface Hssi2/0, changed state to administratively down
04:40:46:%LINEPROTO-5-UPDOWN: Line protocol on Interface Hssi2/0, changed state to down
```

This example shows the messages that are displayed when you enable the core-facing interface on a PE router:

```
Router(config-if)# no shut

04:40:56:%LINK-3-UPDOWN: Interface Hssi2/0, changed state to up
04:40:57:%LINEPROTO-5-UPDOWN: Line protocol on Interface Hssi2/0, changed state to up
04:41:06:%OSPF-5-ADJCHG: Process 10, Nbr 12.12.12.12 on Hssi2/0 from LOADING to FULL,
Loading Done
04:41:19: FROMPLS [12.12.12.12, 100]: PW pvc_status set ACTIVE
04:41:19: FROMPLS [12.12.12.12, 100]: Setting pw segment UP
04:41:19: FROMPLS [12.12.12.12, 101]: PW pvc_status set ACTIVE
04:41:19: FROMPLS [12.12.12.12, 101]: Setting pw segment UP
```

This example shows the messages that are displayed when you shut the edge-facing interface on a PE router:

```
Router(config)# interface pos4/0
Router(config-if)# shut

04:42:50: FROMPLS [12.12.12.12, 100]: acmgr_circuit_down
04:42:50: FROMPLS [12.12.12.12, 100]: Setting connection DOWN
04:42:50: FROMPLS [12.12.12.12, 100]: PW pvc_status set INACTIVE
04:42:52:%LINK-5-CHANGED: Interface POS4/0, changed state to administratively down
04:42:53:%LINEPROTO-5-UPDOWN: Line protocol on Interface POS4/0, changed state to down
```

This example shows the messages that are displayed when you enable the edge-facing interface on a PE router:

```
Router(config)# interface pos4/0
Router(config-if)# no shut

04:43:20:%LINK-3-UPDOWN: Interface POS4/0, changed state to up
c72-33-2(config-if)#
04:43:20: FROMPLS [12.12.12.12, 100]: Local up, sending acmgr_circuit_up
04:43:20: FROMPLS [12.12.12.12, 100]: PW nni_pvc_status set ACTIVE
04:43:20: FROMPLS [12.12.12.12, 100]: PW pvc_status set ACTIVE
04:43:20: FROMPLS [12.12.12.12, 100]: Setting pw segment UP
```

debug frame-relay foresight

To observe Frame Relay traces relating to traffic shaping with router ForeSight enabled, use the **debug frame-relay foresight** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug frame-relay foresight

no debug frame-relay foresight

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

Examples The following is sample output that shows the display message returned in response to the **debug frame-relay foresight** command:

```
Router# debug frame-relay foresight
```

```
FR rate control for DLCI 17 due to ForeSight msg
```

This message indicates the router learned from the ForeSight message that data-link connection identifier (DLCI) 17 is now experiencing congestion. The output rate for this circuit should be slowed down, and in the router this DLCI is configured to adapt traffic shaping in response to foresight messages.

Related Commands

Command	Description
show frame-relay pvc	Displays statistics about PVCs for Frame Relay interfaces.

debug frame-relay fragment

To display information related to Frame Relay fragmentation on a permanent virtual circuit (PVC), use the **debug frame-relay fragment** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug frame-relay fragment [**event** | **interface** *type number dlci*]

no debug frame-relay fragment [**event** | **interface** *type number dlci*]

Syntax Description

event	(Optional) Displays event or error messages related to Frame Relay fragmentation.
interface	(Optional) Displays fragments received or sent on the specified interface.
<i>type</i>	(Optional) The interface type for which you wish to display fragments received or sent.
<i>number</i>	(Optional) The Interface number.
<i>dlci</i>	(Optional) The data-link connection identifier (DLCI) value of the PVC for which you wish to display fragments received or sent.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(3)XG	This command was introduced.

Usage Guidelines

This command will display event or error messages related to Frame Relay fragmentation; it is only enabled at the PVC level on the selected interface.

This command is not supported on the Cisco MC3810 networking device for fragments received by a PVC configured via the **voice-encap** command.

Examples

The following is sample output from the **debug frame-relay fragment** command:

```
Router# debug frame-relay fragment interface serial 0/0 109
```

```
This may severely impact network performance.
You are advised to enable 'no logging console debug'. Continue?[confirm]
Frame Relay fragment/packet debugging is on
Displaying fragments/packets on interface Serial0/0 dlci 109 only
```

```
Serial0/0(i): dlci 109, rx-seq-num 126, exp_seq-num 126, BE bits set, frag_hdr 04 C0 7E
```

```
Serial0/0(o): dlci 109, tx-seq-num 82, BE bits set, frag_hdr 04 C0 52
```

The following is sample output from the **debug frame-relay fragment event** command:

```
Router# debug frame-relay fragment event

This may severely impact network performance.
You are advised to enable 'no logging console debug'. Continue?[confirm]
Frame Relay fragment event/errors debugging is on

Frame-relay reassembled packet is greater than MTU size, packet dropped on serial 0/0
dlci 109

Unexpected B bit frame rx on serial0/0 dlci 109, dropping pending segments

Rx an out-of-sequence packet on serial 0/0 dlci 109, seq_num_received 17
seq_num_expected 19
```

Related Commands

Command	Description
debug cccrf11 session	Displays the cccrf11 function calls during call setup and teardown.
debug ccsip all	Displays the ccsvoice function calls during call setup and teardown.
debug ccsvoice vofr-session	Displays the ccsvoice function calls during call setup and teardown.
debug voice vofr	Displays Cisco trunk and FRF.11 trunk call setup attempts; shows which dial peer is used in the call setup.
debug vpm error	Displays the behavior of the Holst state machine.
debug vtsp port	Displays the behavior of the VTSP state machine.
debug vtsp vofr subframe	Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.

debug frame-relay informationelements

To display information about Frame Relay Layer 3 (network layer) information element parsing and construction, use the **debug frame-relay informationelements** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug frame-relay informationelements

no debug frame-relay informationelements

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

Within the FRF.4/Q.933 signalling specification, messages are divided into subunits called information elements. Each information element defines parameters specific to the call. These parameters can be values configured on the router, or values requested from the network.

The **debug frame-relay informationelements** command shows the signalling message in hexadecimal format. Use this command to determine parameters being requested and granted for a call.



Note

Use the **debug frame-relay informationelements** command when the **debug frame-relay callcontrol** command does not explain why calls are not being set up.



Caution

The **debug frame-relay informationelements** command displays a substantial amount of information in bytes. You must be familiar with FRF.4/Q.933 to decode the information contained within the debug output.

Examples

The following is sample output from the **debug frame-relay informationelements** command. In this example, each information element has a length associated with it. For those with odd-numbered lengths, only the specified bytes are valid, and the extra byte is invalid. For example, in the message “Call Ref, length: 3, 0x0200 0x0100,” only “02 00 01” is valid; the last “00” is invalid.

```
l#w0d# debug frame-relay informationelements
Router: Outgoing MSG_SETUP

Router: Dir: U --> N, Type: Prot Disc, length: 1, 0x0800
Router: Dir: U --> N, Type: Call Ref, length: 3, 0x0200 0x0100
Router: Dir: U --> N, Type: Message type, length: 1, 0x0500
Router: Dir: U --> N, Type: Bearer Capability, length: 5, 0x0403 0x88A0 0xCF00
Router: Dir: U --> N, Type: DLCI, length: 4, 0x1902 0x46A0
Router: Dir: U --> N, Type: Link Lyr Core, length: 27, 0x4819 0x090B 0x5C0B 0xDC0A
Router:                               0x3140 0x31C0 0x0B21 0x4021
Router:                               0xC00D 0x7518 0x7598 0xE09
Router:                               0x307D 0x8000
Router: Dir: U --> N, Type: Calling Party, length: 12, 0x6C0A 0x1380 0x3837 0x3635
```

```

Router:                               0x3433 0x3231
Router: Dir: U --> N, Type: Calling Party Subaddr, length: 4, 0x6D02 0xA000
Router: Dir: U --> N, Type: Called Party, length: 11, 0x7009 0x9331 0x3233 0x3435
Router:                               0x3637 0x386E
Router: Dir: U --> N, Type: Called Party Subaddr, length: 4, 0x7102 0xA000
Router: Dir: U --> N, Type: Low Lyr Comp, length: 5, 0x7C03 0x88A0 0xCE65
Router: Dir: U --> N, Type: User to User, length: 4, 0x7E02 0x0000
    
```

Table 64 explains the information elements shown in the example.

Table 64 Information Elements in a Setup Message

Information Element	Description
Prot Disc	Protocol discriminator.
Call Ref	Call reference.
Message type	Message type such as setup, connect, and call proceeding.
Bearer Capability	Coding format such as data type, and Layer 2 and Layer 3 protocols.
DLCI	Data-link connection identifier.
Link Lyr Core	Link-layer core quality of service (QoS) requirements.
Calling Party	Type of source number (X121/E164) and the number.
Calling Party Subaddr	Subaddress that originated the call.
Called Party	Type of destination number (X121/E164) and the number.
Called Party Subaddr	Subaddress of the called party.
Low Lyr Comp	Coding format, data type, and Layer 2 and Layer 3 protocols intended for the end user.
User to User	Information between end users.

Related Commands

Command	Description
debug frame-relay callcontrol	Displays Frame Relay Layer 3 (network layer) call control information.

debug frame-relay ip tcp header-compression

To display debugging information about TCP/IP header compression on Frame Relay interfaces, use the **debug frame-relay ip tcp header-compression** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug frame-relay ip tcp header-compression

no debug frame-relay ip tcp header-compression

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
10.0	This command was introduced.

Usage Guidelines

The **debug frame-relay ip tcp header-compression** command shows the control packets that are passed to initialize IP header compression (IPHC) on a permanent virtual circuit (PVC). For Cisco IPHC, typically two packets are passed: one sent and one received per PVC. (Inverse Address Resolution Protocol (InARP) packets are sent on PVCs that do not have a mapping defined between a destination protocol address and the data-link connection identifier (DLCI) or Frame Relay PVC bundle that connects to the destination address.)

Debug messages are displayed only if the IPHC control protocol is renegotiated (for an interface or PVC state change or for a configuration change).

Examples

The following is sample output from the **debug frame-relay ip tcp header-compression** command when Cisco IPHC is configured in the IPHC profile:

```
Router# debug frame-relay ip tcp header-compression
```

```
*Nov 14 09:22:07.991: InARP REQ: Tx compr_flags 43 *Nov 14 09:22:08.103: InARP RSP: Rx
compr_flags: 43
```

Table 65 describes the significant fields shown in the display.

Table 65 *debug frame-relay ip tcp header-compression Field Descriptions*

Field	Description
InARP REQ: Tx	Indicates that an InARP request was sent or received. Following are the possible values: <ul style="list-style-type: none"> • InARP REQ Tx—An InARP request was sent. • InARP REQ Rx—An InARP request was received.
InARP RSP: Rx	Indicates that an InARP response was sent or received. Following are the possible values: <ul style="list-style-type: none"> • InARP REQ Tx—An InARP response was sent. • InARP REQ Rx—An InARP response was received.
compr_flags: 43	Compression flags that Frame Relay peers use to negotiate Cisco IPHC options. It consists of a bit mask, and the number is displayed in hexadecimal format. Following are the bits: <ul style="list-style-type: none"> • 0x0001—TCP IPHC • 0x0002—RTP IPHC • 0x0004—Passive TCP compression • 0x0008—Passive RTP compression • 0x0040—Frame Relay IPHC options

debug frame-relay lapf

To display Frame Relay switched virtual circuit (SVC) Layer 2 information, use the **debug frame-relay lapf** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug frame-relay lapf

no debug frame-relay lapf

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

Use the **debug frame-relay lapf** command to troubleshoot the data-link control portion of Layer 2 that runs over data-link connection identifier (DLCI) 0. Use this command only if you have a problem bringing up Layer 2. You can use the **show interface serial** command to determine the status of Layer 2. If it shows a Link Access Procedure, Frame Relay (LAPF) state of down, Layer 2 has a problem.

Examples

The following is sample output from the **debug frame-relay lapf** command. In this example, a line being brought up indicates an exchange of set asynchronous balanced mode extended (SABME) and unnumbered acknowledgment (UA) commands. A SABME is initiated by both sides, and a UA is the response. Until the SABME gets a UA response, the line is not declared to be up. The p/f value indicates the poll/final bit setting. TX means send, and RX means receive.

```
Router# debug frame-relay lapf

Router: *LAPF Serial0 TX -> SABME Cmd p/f=1
Router: *LAPF Serial0 Enter state 5
Router: *LAPF Serial0 RX <- UA Rsp p/f=1
Router: *LAPF Serial0 lapf_ua_5
Router: *LAPF Serial0 Link up!
Router: *LAPF Serial0 RX <- SABME Cmd p/f=1
Router: *LAPF Serial0 lapf_sabme_78
Router: *LAPF Serial0 TX -> UA Rsp p/f=1
```

In the following example, a line in an up LAPF state should see a steady exchange of RR (receiver ready) messages. TX means send, RX means receive, and N(R) indicates the receive sequence number.

```
Router# debug frame-relay lapf

Router: *LAPF Serial0 T203 expired, state = 7
Router: *LAPF Serial0 lapf_rr_7
Router: *LAPF Serial0 TX -> RR Rsp p/f=1, N(R)= 3
Router: *LAPF Serial0 RX <- RR Cmd p/f=1, N(R)= 3
Router: *LAPF Serial0 lapf_rr_7
Router: *LAPF Serial0 TX -> RR Rsp p/f=1, N(R)= 3
Router: *LAPF Serial0 RX <- RR Cmd p/f=1, N(R)= 3
Router: *LAPF Serial0 lapf_rr_7
```

debug frame-relay lmi

To display information on the local management interface (LMI) packets exchanged by the router and the Frame Relay service provider, use the **debug frame-relay lmi** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug frame-relay lmi [*interface name*]

no debug frame-relay lmi [*interface name*]

Syntax Description

interface name (Optional) The name of interface.

Command Modes

Privileged EXEC

Usage Guidelines

You can use this command to determine whether the router and the Frame Relay switch are sending and receiving LMI packets properly.



Note

Because the **debug frame-relay lmi** command does not generate much output, you can use it at any time, even during periods of heavy traffic, without adversely affecting other users on the system.

Examples

The following is sample output from the **debug frame-relay lmi** command:

```
router# debug frame-relay lmi
```

LMI exchange

```
Serial1(out): StEng, clock 20212760, myseq 206, mineseen 205, yourseen 136, DTE up
Serial1(in): Status, clock 20212764, myseq 206
RT IE 1, length 1, type 1
KA IE 3, length 2, yourseq 138, myseq 206
```

```
Serial1(out): StEng, clock 20222760, myseq 207, mineseen 206, yourseen 138, DTE up
Serial1(in): Status, clock 20222764, myseq 207
```

```
RT IE 1, length 1, type 1
```

```
KA IE 3, length 2, yourseq 140, myseq 207
```

```
Serial1(out): clock 20232760, myseq 208, mineseen 207, yourseen 140, line up
```

```
RT IE 1, length 1, type 1
```

```
KA IE 3, length 2, yourseq 142, myseq 208
```

Full LMI status message

```
Serial1(out): StEng, clock 20252760, myseq 210, mineseen 209, yourseen 144, DTE up
Serial1(in): Status, clock 20252764,
```

```
RT IE 1, length 1, type 0
```

```
KA IE 3, length 2, yourseq 146, myseq 210
```

```
PVC IE 0x7, length 0x6, dlci 400, status 0, bw 56000
```

```
PVC IE 0x7, length 0x6, dlci 401, status 0, bw 56000
```

92546

The first four lines describe an LMI exchange. The first line describes the LMI request the router has sent to the switch. The second line describes the LMI reply the router has received from the switch. The third and fourth lines describe the response to this request from the switch. This LMI exchange is followed by two similar LMI exchanges. The last six lines consist of a full LMI status message that includes a description of the two permanent virtual circuits (PVCs) of the router.

Table 66 describes significant fields shown in the first line of the display.

Table 66 *debug frame-relay lmi Field Descriptions*

Field	Description
Serial1(out)	Indicates that the LMI request was sent out on serial interface 1.
StEnq	Command mode of message, as follows: <ul style="list-style-type: none"> • StEnq—Status inquiry • Status—Status reply
clock 20212760	System clock (in milliseconds). Useful for determining whether an appropriate amount of time has transpired between events.
myseq 206	Myseq counter maps to the CURRENT SEQ counter of the router.
yourseen 136	Yourseen counter maps to the LAST RCVD SEQ counter of the switch.
DTE up	Line protocol up/down state for the DTE (user) port.

Table 67 describes the significant fields shown in the third and fourth lines of the display.

Table 67 *debug frame-relay lmi Field Descriptions*

Field	Description
RT IE 1	Value of the report type information element.
length 1	Length of the report type information element (in bytes).
type 1	Report type in RT IE.
KA IE 3	Value of the keepalive information element.
length 2	Length of the keepalive information element (in bytes).
yourseq 138	Yourseq counter maps to the CURRENT SEQ counter of the switch.
myseq 206	Myseq counter maps to the CURRENT SEQ counter of the router.

Table 68 describes the significant fields shown in the last line of the display.

Table 68 *debug frame-relay lmi Field Descriptions*

Field	Description
PVC IE 0x7	Value of the PVC information element type.
length 0x6	Length of the PVC IE (in bytes).
dldci 401	DLCI decimal value for this PVC.
status 0	Status value. Possible values include the following: <ul style="list-style-type: none">• 0x00—Added/inactive• 0x02—Added/active• 0x04—Deleted• 0x08—New/inactive• 0x0a—New/active
bw 56000	Committed information rate (in decimal) for the DLCI.

debug frame-relay multilink

To display debugging messages for multilink Frame Relay bundles and bundle links, use the **debug frame-relay multilink** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug frame-relay multilink [control [mfr number | serial number]]
```

```
no debug frame-relay multilink
```

Syntax Description

control	(Optional) Displays incoming and outgoing bundle link control messages and bundle link status changes.
mfr number	(Optional) Specific bundle interface for which information will be displayed.
serial number	(Optional) Specific bundle link interface for which information will be displayed.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(17)S	This command was introduced.
12.2(8)T	This command was integrated into Cisco IOS Release 12.2(8)T.

Usage Guidelines



Caution

Using the **debug frame-relay multilink** command without the **control** keyword could severely impact router performance and is not recommended.

Using the **debug frame-relay multilink** command without the **mfr** or **serial** keywords will display error conditions occurring at the bundle layer.

Examples

The following example shows output from the **debug frame-relay multilink** command for bundle “MFR0”, which has three bundle links:

```
Router# debug frame-relay multilink control MFR0

00:42:54:Serial5/3(o):msg=Add_link, Link=Serial5/3, Bundle=MFR0, Link id=Serial5/3,
BL state=Idle
E1 00 01 01 07 4D 46 52 30 00
00:42:54:Serial5/2(o):msg=Add_link, Link=Serial5/2, Bundle=MFR0, Link id=Serial5/2,
BL state=Idle
```

```

E1 00 01 01 07 4D 46 52 30 00
00:42:54:Serial5/1(o):msg=Add_link, Link=Serial5/1, Bundle=MFR0, Link id=Serial5/1,
BL state=Idle
E1 00 01 01 07 4D 46 52 30 00
00:42:54:%LINK-3-UPDOWN:Interface MFR0, changed state to down
00:42:54:Serial5/3(i):msg=Add_link_ack, Link=Serial5/3, Bundle=MFR0, Link id=Serial5/3,
BL state=Add_sent
E1 00 02 01 07 4D 46 52 30 00
00:42:54:Serial5/2(i):msg=Add_link_ack, Link=Serial5/2, Bundle=MFR0, Link id=Serial5/2,
BL state=Add_sent
E1 00 02 01 07 4D 46 52 30 00
00:42:54:Serial5/1(i):msg=Add_link_ack, Link=Serial5/1, Bundle=MFR0, Link id=Serial5/1,
BL state=Add_sent
E1 00 02 01 07 4D 46 52 30 00
00:42:54:%SYS-5-CONFIG_I:Configured from console by console
00:43:00:Serial5/1(i):msg=Add_link, Link=Serial5/1, Bundle=MFR0, Link id=Serial5/1,
BL state=Ack_rx
E1 00 01 01 07 4D 46 52 30 00
00:43:00:Serial5/1(o):msg=Add_link_ack, Link=Serial5/1, Bundle=MFR0, Link id=Serial5/1,
BL state=Ack_rx
E1 00 02 01 07 4D 46 52 30 00
00:43:00:%LINK-3-UPDOWN:Interface MFR0, changed state to up
00:43:00:Serial5/1(i):msg=Hello, Link=Serial5/1, Bundle=MFR0, Linkid=Serial5/1, BL
state=Up
E1 00 04 03 06 30 A7 E0 54 00
00:43:00:Serial5/1(o):msg=Hello_ack, Link=Serial5/1, Bundle=MFR0, Link id=Serial5/1, BL
state=Up
E1 00 05 03 06 90 E7 0F C2 06
00:43:01:Serial5/2(i):msg=Add_link, Link=Serial5/2, Bundle=MFR0, Link id=Serial5/2,
BL state=Ack_rx
E1 00 01 01 07 4D 46 52 30 00
00:43:01:Serial5/2(o):msg=Add_link_ack, Link=Serial5/2, Bundle=MFR0, Link id=Serial5/2,
BL state=Ack_rx
E1 00 02 01 07 4D 46 52 30 00
00:43:01:Serial5/2(i):msg=Hello, Link=Serial5/2, Bundle=MFR0, Linkid=Serial5/2, BL
state=Up
E1 00 04 03 06 30 A7 E0 54 00
00:43:01:Serial5/2(o):msg=Hello_ack, Link=Serial5/2, Bundle=MFR0, Link id=Serial5/2,
BL state=Up
E1 00 05 03 06 90 E7 0F C2 06
00:43:01:%LINEPROTO-5-UPDOWN:Line protocol on Interface Serial5/1, changed state to up
00:43:01:Serial5/3(i):msg=Add_link, Link=Serial5/3, Bundle=MFR0, Link id=Serial5/3,
BL state=Ack_rx
E1 00 01 01 07 4D 46 52 30 00
00:43:01:Serial5/3(o):msg=Add_link_ack, Link=Serial5/3, Bundle=MFR0, Link id=Serial5/3,
BL state=Ack_rx
E1 00 02 01 07 4D 46 52 30 00
00:43:01:Serial5/3(i):msg=Hello, Link=Serial5/3, Bundle=MFR0, Linkid=Serial5/3, BL
state=Up
E1 00 04 03 06 30 A7 E0 54 00
00:43:01:Serial5/3(o):msg=Hello_ack, Link=Serial5/3, Bundle=MFR0, Link id=Serial5/3,
BL state=Up
E1 00 05 03 06 90 E7 0F C2 06
00:43:02:%LINEPROTO-5-UPDOWN:Line protocol on Interface Serial5/2 , changed state to up
00:43:02:%LINEPROTO-5-UPDOWN:Line protocol on Interface Serial5/3 , changed state to up

```

Table 69 describes the significant fields shown in the display.

Table 69 *debug frame-relay multilink Field Descriptions*

Field	Description
msg	Type of bundle link control message that was sent or received.
Link	Interface number of the bundle link.
Bundle	Bundle with which the link is associated.
Link id	Bundle link identification name.
BL state	Operational state of the bundle link.

Related Commands

Command	Description
show frame-relay multilink	Displays configuration information and statistics about multilink Frame Relay bundles and bundle links.

debug frame-relay networklayerinterface

To display Network Layer Interface (NLI) information, use the **debug frame-relay networklayerinterface** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug frame-relay networklayerinterface

no debug frame-relay networklayerinterface

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

Usage Guidelines The Frame Relay switched virtual circuit (SVC) signaling subsystem is decoupled from the rest of the router code by means of the NLI intermediate software layer.

The **debug frame-relay networklayerinterface** command shows activity within the network-layer interface when a call is set up or torn down. All output that contains an *NL* relates to the interaction between the Q.933 signaling subsystem and the NLI.



Note

The **debug frame-relay networklayerinterface** command has no significance to anyone not familiar with the inner workings of the Cisco IOS software. This command is typically used by service personnel to debug problem situations.

Examples The following is sample output from the **debug frame-relay networklayerinterface** command. This example displays the output generated when a call is set up. The second example shows the output generated when a call is torn down.

```
Router# debug frame-relay networklayerinterface

Router: NLI STATE: L3_CALL_REQ, Call ID 1 state 0
Router: NLI: Walking the event table 1
Router: NLI: Walking the event table 2
Router: NLI: Walking the event table 3
Router: NLI: Walking the event table 4
Router: NLI: Walking the event table 5
Router: NLI: Walking the event table 6
Router: NLI: Walking the event table 7
Router: NLI: Walking the event table 8
Router: NLI: Walking the event table 9
Router: NLI: NL0_L3CallReq
Router: NLI: State: STATE_NL_NULL, Event: L3_CALL_REQ, Next: STATE_L3_CALL_REQ
Router: NLI: Enqueued outgoing packet on holdq
Router: NLI: Map-list search: Found maplist bermuda
Router: daddr.subaddr 0, saddr.subaddr 0, saddr.subaddr 0
Router: saddr.subaddr 0, daddr.subaddr 0, daddr.subaddr 0
Router: nli_parameter_negotiation
Router: NLI STATE: NL_CALL_CNF, Call ID 1 state 10
Router: NLI: Walking the event table 1
```

```

Router: NLI: Walking the event table 2
Router: NLI: Walking the event table 3
Router: NLI: NLx_CallCnf
Router: NLI: State: STATE_L3_CALL_REQ, Event: NL_CALL_CNF, Next: STATE_NL_CALL_CNF
Router: Checking maplist "junk"
Router: working with maplist "bermuda"
Router: Checking maplist "bermuda"
Router: working with maplist "bermuda"
Router: NLI: Emptying holdQ, link 7, dlci 100, size 104

Router# debug frame-relay networklayerinterface

Router: NLI: L3 Call Release Req for Call ID 1
Router: NLI STATE: L3_CALL_REL_REQ, Call ID 1 state 3
Router: NLI: Walking the event table 1
Router: NLI: Walking the event table 2
Router: NLI: Walking the event table 3
Router: NLI: Walking the event table 4
Router: NLI: Walking the event table 5
Router: NLI: Walking the event table 6
Router: NLI: Walking the event table 7
Router: NLI: Walking the event table 8
Router: NLI: Walking the event table 9
Router: NLI: Walking the event table 10
Router: NLI: NLx_L3CallRej
Router: NLI: State: STATE_NL_CALL_CNF, Event: L3_CALL_REL_REQ, Next: STATE_L3_CALL_REL_REQ
Router: NLI: junk: State: STATE_NL_NULL, Event: L3_CALL_REL_REQ, Next: STATE_NL_NULL
Router: NLI: Map-list search: Found maplist junk
Router: daddr.subaddr 0, saddr.subaddr 0, saddr.subaddr 0
Router: saddr.subaddr 0, daddr.subaddr 0, daddr.subaddr 0
Router: nli_parameter_negotiation
Router: NLI STATE: NL_REL_CNF, Call ID 1 state 0
Router: NLI: Walking the event table 1
Router: NLI: Walking the event table 2
Router: NLI: Walking the event table 3
Router: NLI: Walking the event table 4
Router: NLI: Walking the event table 5
Router: NLI: Walking the event table 6
Router: NLI: Walking the event table 7
Router: NLI: NLx_RelCnf
Router: NLI: State: STATE_NL_NULL, Event: NL_REL_CNF, Next: STATE_NL_NULL

```

Table 70 describes the significant states and events shown in the display.

Table 70 NLI State and Event Descriptions

State and Event	Description
L3_CALL_REQ	Internal call setup request. Network layer indicates that an SVC is required.
STATE_NL_NULL	Call in initial state—no call exists.
STATE_L3_CALL_REQ	Setup message sent out and waiting for a reply. This is the state the network-layer state machine changes to when a call request is received from Layer 3 but no confirmation has been received from the network.
NL_CALL_CNF	Message sent from the Q.933 signalling subsystem to the NLI asking that internal resources be allocated for the call.

Table 70 NLI State and Event Descriptions (continued)

State and Event	Description
STATE_L3_CALL_CNF	Q.933 state indicating that the call is active. After the network confirms a call request using a connect message, the Q.933 state machine changes to this state.
STATE_NL_CALL_CNF	Internal software state indicating that software resources are assigned and the call is up. After Q.933 changes to the STATE_L3_CALL_CNF state, it sends an NL_CALL_CNF message to the network-layer state machine, which then changes to the STATE_NL_CALL_CNF state.
L3_CALL_REL_REQ	Internal request to release the call.
STATE_L3_CALL_REL_REQ	Internal software state indicating the call is in the process of being released. At this point, the Q.933 subsystem is told that the call is being released and a disconnect message goes out for the Q.933 subsystem.
NL_REL_CNF	Indication from the Q.933 signalling subsystem that the signalling subsystem is releasing the call. After receiving a release complete message from the network indicating that the release process is complete, the Q.933 subsystem sends an NL_REL_CNF event to the network-layer subsystem.

Related Commands

Command	Description
debug frame-relay callcontrol	Displays Frame Relay Layer 3 (network layer) call control information.

debug frame-relay packet

To display information on packets that have been sent on a Frame Relay interface, use the **debug frame-relay packet** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug frame-relay packet [interface name [dlci value]]
```

```
no debug frame-relay packet [interface name [dlci value]]
```

Syntax Description

interface name	(Optional) Name of interface or subinterface.
dlci value	(Optional) Data-link connection identifier (DLCI) decimal value.

Command Modes

Privileged EXEC

Usage Guidelines

This command helps you analyze the packets that are sent on a Frame Relay interface. Because the **debug frame-relay packet** command generates a substantial amount of output, only use it when traffic on the Frame Relay network is fewer than 25 packets per second. Use the options to limit the debugging output to a specific DLCI or interface.

To analyze the packets *received* on a Frame Relay interface, use the **debug frame-relay** command.

Examples

The following is sample output from the **debug frame-relay packet** command:

```
router# debug frame-relay packets
```

```
Serial0: broadcast = 1, link 809B, addr 65535.255
Serial0(o):DLCI 500 type 809B size 24
Serial0: broadcast - 0, link 809B, addr 10.2
Serial0(o):DLCI 100 type 809B size 104
Serial0: broadcast search
Serial0(o):DLCI 300 type 809B size 24
Serial0(o):DLCI 400 type 809B size 24
```

Groups of output lines

S2547

The **debug frame-relay packet** output consists of groups of output lines; each group describes a Frame Relay packet that has been sent. The number of lines in the group can vary, depending on the number of DLCIs on which the packet was sent. For example, the first two pairs of output lines describe two different packets, both of which were sent out on a single DLCI. The last three lines describe a single Frame Relay packet that was sent out on two DLCIs.

Table 71 describes the significant fields shown in the display.

Table 71 *debug frame-relay packet Field Descriptions*

Field	Description
Serial0:	Interface that has sent the Frame Relay packet.
broadcast = 1	Destination of the packet. Possible values include the following: <ul style="list-style-type: none"> • broadcast = 1—Broadcast address • broadcast = 0—Particular destination • broadcast search—Searches all Frame Relay map entries for this particular protocol that include the broadcast keyword.
link 809B	Link type, as documented in the debug frame-relay command.
addr 65535.255	Destination protocol address for this packet. In this case, it is an AppleTalk address.
Serial0(o):	(o) indicates that this is an output event.
DLCI 500	Decimal value of the DLCI.
type 809B	Packet type, as documented under the debug frame-relay command.
size 24	Size of this packet (in bytes).

The following lines describe a Frame Relay packet sent to a particular address; in this case AppleTalk address 10.2:

```
Serial0: broadcast - 0, link 809B, addr 10.2
Serial0(o):DLCI 100 type 809B size 104
```

The following lines describe a Frame Relay packet that went out on two different DLCIs, because two Frame Relay map entries were found:

```
Serial0: broadcast search
Serial0(o):DLCI 300 type 809B size 24
Serial0(o):DLCI 400 type 809B size 24
```

The following lines do not appear. They describe a Frame Relay packet sent to a true broadcast address.

```
Serial1: broadcast search
Serial1(o):DLCI 400 type 800 size 288
```

debug frame-relay ppp

To display debugging information, use the **debug frame-relay ppp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug frame-relay ppp

no debug frame-relay ppp

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

Usage Guidelines This command displays error messages for link states and Local Management Interface (LMI) status changes for PPP over Frame Relay sessions.

To debug process-switched packets, use the **debug frame-relay packet** or **debug ppp packet** commands. To analyze the packets that have been *sent* on a Frame Relay interface, use the **debug frame-relay packet** command.

The **debug frame-relay ppp** command is generated from process-level switching only and is not CPU intensive.

Examples The following shows output from the **debug frame-relay ppp** command where the encapsulation failed for VC 100.

```
Router# debug frame-relay ppp

FR-PPP: encaps failed for FR VC 100 on Serial0 down
FR-PPP: input- Serial0 vc or va down, pak dropped
```

The following shows the output from the **debug frame-relay ppp** and **debug frame-relay packet** commands. This example shows a virtual interface (virtual interface 1) establishing a PPP connection over PPP.

```
Router# debug frame-relay ppp

Router# debug frame-relay packet

Vil LCP: O CONFREQ [Closed] id 1 len 10
Vil LCP:   MagicNumber 0xE0638565 (0x0506E0638565)
Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 16
Vil PPP: I pkt type 0xC021, datagramsize 14
Vil LCP: I CONFACK [REQsent] id 1 len 10
Vil LCP:   MagicNumber 0xE0638565 (0x0506E0638565)
Vil PPP: I pkt type 0xC021, datagramsize 14
Vil LCP: I CONFREQ [ACKrcvd] id 6 len 10
Vil LCP:   MagicNumber 0x000EAD99 (0x0506000EAD99)
Vil LCP: O CONFACK [ACKrcvd] id 6 len 10
Vil LCP:   MagicNumber 0x000EAD99 (0x0506000EAD99)
Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 16
Vil IPCP: O CONFREQ [Closed] id 1 len 10
Vil IPCP:   Address 170.100.9.10 (0x0306AA64090A)
```

```

Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 16
Vi1 PPP: I pkt type 0x8021, datagramsize 14
Vi1 IPCP: I CONFREQ [REQsent] id 1 len 10
Vi1 IPCP:   Address 170.100.9.20 (0x0306AA640914)
Vi1 IPCP: O CONFACK [REQsent] id 1 len 10
Vi1 IPCP:   Address 170.100.9.20 (0x0306AA640914)
Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 16
Vi1 PPP: I pkt type 0x8021, datagramsize 14
Vi1 IPCP: I CONFACK [ACKsent] id 1 len 10
Vi1 IPCP:   Address 170.100.9.10 (0x0306AA64090A)
Vi1 PPP: I pkt type 0xC021, datagramsize 16
Vi1 LCP: I ECHOREQ [Open] id 1 len 12 magic 0x000EAD99
Vi1 LCP: O ECHOREP [Open] id 1 len 12 magic 0xE0638565
Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 18
Vi1 LCP: O ECHOREQ [Open] id 1 len 12 magic 0xE0638565
Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 18
Vi1 LCP: echo_cnt 4, sent id 1, line up

```

The following shows the output for the **debug frame-relay ppp** and **debug frame-relay packet** commands that report a failed PPP over Frame Relay session. The problem is due to a challenge handshake authentication protocol (CHAP) failure.

```
Router# debug frame-relay ppp
```

```
Router# debug frame-relay packet
```

```

Vi1 LCP: O CONFREQ [Listen] id 24 len 10
Vi1 LCP:   MagicNumber 0xE068EC78 (0x0506E068EC78)
Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 16
Vi1 PPP: I pkt type 0xC021, datagramsize 19
Vi1 LCP: I CONFREQ [REQsent] id 18 len 15
Vi1 LCP:   AuthProto CHAP (0x0305C22305)
Vi1 LCP:   MagicNumber 0x0014387E (0x05060014387E)
Vi1 LCP: O CONFACK [REQsent] id 18 len 15
Vi1 LCP:   AuthProto CHAP (0x0305C22305)
Vi1 LCP:   MagicNumber 0x0014387E (0x05060014387E)
Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 21
Vi1 PPP: I pkt type 0xC021, datagramsize 14
Vi1 LCP: I CONFACK [ACKsent] id 24 len 10
Vi1 LCP:   MagicNumber 0xE068EC78 (0x0506E068EC78)
Vi1 PPP: I pkt type 0xC223, datagramsize 32
Vi1 CHAP: I CHALLENGE id 12 len 28 from "krishna"
Vi1 LCP: O TERMREQ [Open] id 25 len 4
Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 10
Vi1 PPP: I pkt type 0xC021, datagramsize 8
Vi1 LCP: I TERMACK [TERMsent] id 25 len 4
Serial2/1(i): dlci 201(0x3091), pkt type 0x2000, datagramsize 303
%SYS-5-CONFIG_I: Configured from console by console
Vi1 LCP: TIMEOUT: Time 0x199580 State Listen

```

debug frame-relay switching

To display debugging messages for switched Frame Relay permanent virtual circuits (PVCs), use the **debug frame-relay switching** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug frame-relay switching interface interface dci [interval interval]
```

```
no debug frame-relay switching
```

Syntax Description

interface <i>interface</i>	The name of the Frame Relay interface.
<i>dci</i>	The DLCI number of the switched PVC to be debugged.
interval <i>interval</i>	(Optional) Interval in seconds at which debugging messages will be updated.

Defaults

The default interval is 1 second.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(12)S	This command was introduced.
12.1(5)T	This command was integrated into Cisco IOS Release 12.1(5)T.

Usage Guidelines

The **debug frame-relay switching** command can be used only on switched Frame Relay PVCs, not terminated PVCs.

Debug statistics are displayed only if they have changed.



Note

Although statistics are displayed at configured intervals, there may be a delay between the occurrence of a debug event (such as a packet drop) and the display of that event. The delay may be as much as the configured interval plus 10 seconds.

Examples

The following is sample output from the **debug frame-relay switching** command:

```
Router# debug frame-relay switching interface s2/1 1000 interval 2
```

```
Frame Relay switching debugging is on
Display frame switching debug on interface Serial2/1 dlci 1000
1d02h: Serial2/1 dlci 1000: 32 packets switched to Serial2/0 dlci 1002
1d02h: Serial2/1 dlci 1000: 1800 packets output
1d02h: Serial2/1 dlci 1000: 4 packets dropped - outgoing PVC inactive
1d02h: Serial2/1 dlci 1000: Incoming PVC status changed to ACTIVE
1d02h: Serial2/1 dlci 1000: Outgoing PVC status changed to ACTIVE
1d02h: Serial2/1 dlci 1000: Incoming interface hardware module state changed to UP
1d02h: Serial2/1 dlci 1000: Outgoing interface hardware module state changed to UP
```

debug frame-relay vc-bundle

To display information about the Frame Relay permanent virtual circuit (PVC) bundles that are configured on a router, use the **debug frame-relay vc-bundle** command in privileged EXEC mode. To stop the display, use the **no** form of this command.

```
debug frame-relay vc-bundle {detail | state-change} [vc-bundle-name]
```

```
no debug frame-relay vc-bundle {detail | state-change} [vc-bundle-name]
```

Syntax Description

detail	Displays detailed information about the members of the bundle specified by <i>vc-bundle-name</i> . Displays detailed information about the members of all PVC bundles if <i>vc-bundle-name</i> is not specified.
state-change	Displays information pertaining only to the state changes of the PVC bundle and PVC bundle members specified by <i>vc-bundle-name</i> . Displays state-change information for all PVC bundles and bundle members if <i>vc-bundle-name</i> is not specified.
<i>vc-bundle-name</i>	(Optional) Specifies a particular PVC bundle.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(13)T	This command was introduced.

Usage Guidelines

Use this command to monitor state changes and Inverse ARP activity for one or all of the PVC bundles and bundle members configured on a router.



Note

Debugging messages that are prefixed with “FR_VCB” (instead of “FR-VCB”) indicate serious failures in the Frame Relay PVC bundle performance. Contact the Cisco Technical Assistance Center (TAC) if you see debugging messages with this prefix.

Examples

The following is sample output from the **debug frame-relay vc-bundle** command that shows Inverse ARP information for the PVC bundle. PVC bundle member 406 is the only PVC in the bundle to handle Inverse ARP packets. The Inverse ARP packets coming in on other bundle member PVCs are dropped.

```
Router# debug frame-relay vc-bundle

00:23:48:FR-VCB:MP-4-dynamic:inarp received on elected member 406
00:23:48:FR-VCB:MP-4-dynamic:installing dynamic map
00:23:48:FR-VCB:MP-4-dynamic:dropping inarp received on member 407
00:23:52:FR-VCB:MP-4-dynamic:sending inarp pkt on member 406
```

In the following example the PVC bundle goes down because the protected group goes down. All information about active transmission on each PVC is removed.

```
00:58:27:FR-VCB:MP-4-dynamic:member 402 state changed to DOWN
00:58:27:FR-VCB:MP-4-dynamic:protected group is DOWN
00:58:27:FR-VCB:MP-4-dynamic:state changed to DOWN
00:58:27:FR-VCB:MP-4-dynamic:active table reset
```

The following is sample output from the **debug frame-relay vc-bundle detail** command. State change and Inverse ARP activity is displayed for all PVC bundles and bundle members on the router.

```
Router# debug frame-relay adjacency vc-bundle detail

00:33:40: FR-VCB: MP-4-dynamic: member 404 state changed to UP
00:33:40: FR-VCB: MP-4-dynamic: active table update
00:33:40: FR-VCB: MP-3-static: sending inarp pkt on member 300
00:33:41: FR-VCB: MP-3-static: inarp received on elected member 300
00:33:48: FR-VCB: MP-3-static: inarp received on elected member 300
00:33:48: FR-VCB: MAIN-1-static: dropping inarp received on member 100
00:33:48: FR-VCB: MP-4-dynamic: dropping inarp received on member 404
00:33:48: FR-VCB: MP-4-dynamic: dropping inarp received on member 405
00:33:48: FR-VCB: P2P-5: dropping inarp received on member 507
00:33:48: FR-VCB: MP-3-static: dropping inarp received on member 303
00:33:48: FR-VCB: MAIN-2-dynamic: dropping inarp received on member 202
00:33:48: FR-VCB: MAIN-1-static: dropping inarp received on member 107
00:33:48: FR-VCB: MP-3-static: dropping inarp received on member 305
00:33:48: FR-VCB: MAIN-1-static: dropping inarp received on member 105
00:33:49: FR-VCB: P2P-5: dropping inarp received on member 505
00:33:49: FR-VCB: P2P-5: dropping inarp received on member 504
00:33:49: FR-VCB: P2P-5: dropping inarp received on member 503
00:33:49: FR-VCB: P2P-5: dropping inarp received on member 502
00:33:49: FR-VCB: P2P-5: dropping inarp received on member 501
```

Related Commands

Command	Description
debug frame-relay adjacency	Displays information pertaining to an adjacent node that has one or more Frame Relay PVC bundles.

debug frame-relay virtual

To display debugging messages for the virtual Frame Relay interface, use the **debug frame-relay virtual** command in privileged EXEC mode.

debug frame-relay virtual *destination interface*

Syntax Description	<i>destination interface</i> Enables the debugging messages for that specific interface.
---------------------------	--

Defaults	No default behavior or values.
-----------------	--------------------------------

Command Modes	Privileged EXEC
----------------------	-----------------

Command History	Release	Modification
	12.2(2)T	This command was introduced.

Usage Guidelines	Use the debug frame-relay virtual command to display debugging messages for the virtual Frame Relay interface. The debug frame-relay virtual command produces output only when problems occur.
-------------------------	--

Examples	The following example shows the output if one of the routers has not been configured. This output occurs when the other end is trying to send the receiving box Frame Relay packets.
-----------------	--

```
VFR: Radio1/0 has no VFR for 00:00:C068:6F:AA
```

Related Commands	Command	Description
	frame-relay over radio	Links the virtual Frame Relay interface to the specified radio interface and destination MAC address.
	interface virtual-framerelay	Defines the virtual interface and then associates the interface with a specific wireless connection.
	show virtual-framerelay	Shows the output of the interface virtual-frame relay command.

debug fras error

To display information about Frame Relay access support (FRAS) protocol errors, use the **debug fras error** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug fras error

no debug fras error

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

Usage Guidelines For complete information on the FRAS process, use the **debug fras message** along with the **debug fras error** command.

Examples The following is sample output from the **debug fras error** command. This example shows that no logical connection exists between the local station and remote station in the current setup.

```
Router# debug fras error

FRAS: No route, lmac 1000.5acc.7fb1 rmac 4fff.0000.0000, lSap=0x4, rSap=0x4
FRAS: Can not find the Setup
```

Related Commands

Command	Description
debug cls message	Displays information about CLS messages.
debug fras message	Displays general information about FRAS messages.
debug fras state	Displays information about FRAS data-link control state changes.

debug fras-host activation

To display the Logical Link Control, Type 2 (LLC2) session activation and deactivation frames (such as XID, SABME, DISC, UA) that are being handled by the Frame Relay access support (FRAS) host, use the **debug fras-host activation** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug fras-host activation

no debug fras-host activation

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

Usage Guidelines If many LLC2 sessions are being activated or deactivated at any time, this command may generate a substantial amount of output to the console.

Examples The following is sample output from the **debug fras-host activation** command:

```
Router# debug fras-host activation

FRHOST: Snd      TST C to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x00 SSAP =
0x04
FRHOST: Fwd  BNN  XID to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x04 SSAP =
0x04
FRHOST: Fwd HOST XID to  BNN, DA = 400f.dddd.001e SA = 4001.3745.1088 DSAP = 0x04 SSAP =
0x05
FRHOST: Fwd  BNN  XID to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x04 SSAP =
0x04
FRHOST: Fwd HOST SABME to  BNN, DA = 400f.dddd.001e SA = 4001.3745.1088 DSAP = 0x04 SSAP =
0x04
FRHOST: Fwd  BNN  UA to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x04 SSAP =
0x05
```

The first line indicates that the FRAS Host sent a TEST Command to the host. In the second line, the FRAS Host forwards an XID frame from a BNN device to the host. In the third line, the FRAS Host forwards an XID from the host to the BNN device.

[Table 72](#) describes the significant fields shown in the display.

Table 72 *debug fras-host activation Field Descriptions*

Field	Description
DA	Destination MAC address of the frame.
SA	Source MAC address of the frame.
DSAP	Destination SAP of the frame.
SSAP	Source SAP of the frame.

debug fras-host error

To enable the Frame Relay access support (FRAS) Host to send error messages to the console, use the **debug fras-host error** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug fras-host error

no debug fras-host error

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

Examples The following is sample output from the **debug fras-host error** command when the I-field in a TEST Response frame from a host does not match the I-field of the TEST Command sent by the FRAS Host:

```
Router# debug fras-host error
```

```
FRHOST: SRB TST R Protocol Violation - LLC I-field not maintained.
```

debug fras-host packet

To see which Logical Link Control, type 2 (LLC2) session frames are being handled by the Frame Relay access support (FRAS) Host, use the **debug fras-host packet** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug fras-host packet

no debug fras-host packet

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

Use this command with great care. If many LLC2 sessions are active and passing data, this command may generate a substantial amount of output to the console and impact device performance.

Examples

The following is sample output from the **debug fras-host packet** command:

```
Router# debug fras-host packet
```

```
FRHOST: Snd      TST C to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x00 SSAP =
0x04
FRHOST: Fwd  BNN  XID to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x04 SSAP =
0x04
FRHOST: Fwd  HOST  XID to  BNN, DA = 400f.dddd.001e SA = 4001.3745.1088 DSAP = 0x04 SSAP =
0x05
FRHOST: Fwd  BNN  XID to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x04 SSAP =
0x04
FRHOST: Fwd  HOST  SABME to  BNN, DA = 400f.dddd.001e SA = 4001.3745.1088 DSAP = 0x04 SSAP =
0x04
FRHOST: Fwd  BNN  UA to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x04 SSAP =
0x05
FRHOST: Fwd  HOST  LLC-2 to  BNN, DA = 400f.dddd.001e SA = 4001.3745.1088 DSAP = 0x04 SSAP =
0x04
FRHOST: Fwd  BNN  LLC-2 to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x04 SSAP =
0x05
FRHOST: Fwd  HOST  LLC-2 to  BNN, DA = 400f.dddd.001e SA = 4001.3745.1088 DSAP = 0x04 SSAP =
0x04
FRHOST: Fwd  BNN  LLC-2 to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x04 SSAP =
0x04
```

The **debug fras-host packet** output contains all of the output from the **debug fras-host activation** command and additional information. The first six lines of this sample display are the same as the output from the **debug fras-host activation** command. The last lines show LLC-2 frames being sent between the Frame Relay Boundary Network Node (BNN) device and the host.

Table 73 describes the significant fields shown in the display.

Table 73 *debug fras-host packet Field Descriptions*

Field	Description
DA	Destination MAC address of the frame.
SA	Source MAC address of the frame.
DSAP	Destination service access point (SAP) of the frame.
SSAP	Source SAP of the frame.

debug fras-host snmp

To display messages to the console describing Simple Network Management Protocol (SNMP) requests to the Frame Relay access support (FRAS) Host MIB, use the **debug fras-host snmp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug fras-host snmp

no debug fras-host snmp

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

Usage Guidelines Use of this command may result in a substantial amount of output to the screen. Only use this command for problem determination.

Examples The following is sample output from the **debug fras-host snmp** command. In this example, the MIB variable `k_frasHostConnEntry_get()` is providing SNMP information for the FRAS host.

```
Router# debug fras-host snmp

k_frasHostConnEntry_get(): serNum = -1, vRingIfIdx = 31, frIfIdx = 12
Hmac = 4001.3745.1088, frLocSap = 4, Rmac = 400f.dddd.001e, frRemSap = 4
```

[Table 74](#) describes the significant fields shown in the display.

Table 74 *debug fras-host snmp Field Descriptions*

Field	Description
serNum	Serial number of the SNMP request.
vRingIfIdx	Interface index of a virtual Token Ring.
frIfIdx	Interface index of a Frame Relay serial interface.
Hmac	MAC address associated with the host for this connection.
frLocSap	SAP associated with the host for this connection.
Rmac	MAC address associated with the FRAD for this connection.
frRemSap	LLC 2 SAP associated with the FRAD for this connection.

debug fras message

To display general information about Frame Relay access support (FRAS) messages, use the **debug fras message** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug fras message

no debug fras message

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

Usage Guidelines For complete information on the FRAS process, use the **debug fras error** command along with the **debug fras message** command.

Examples The following is sample output from the **debug fras message** command. This example shows incoming Cisco Link Services (CLS) primitives.

```
Router# debug fras message
```

```
FRAS: receive 4C23
```

```
FRAS: receive CC09
```

Related Commands

Command	Description
debug cls message	Limits output for some debugging commands based on the interfaces.
debug fras error	Displays information about FRAS protocol errors.
debug fras state	Displays information about FRAS data-link control state changes.

debug fras state

To display information about Frame Relay access support (FRAS) data-link control link-state changes, use the **debug fras state** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug fras state

no debug fras state

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Examples

The following is sample output from the **debug fras state** command. This example shows the state changing from a *request open station is sent* state to an *exchange XID* state.

Possible states are the following: reset, request open station is sent, exchange xid, connection request is sent, signal station wait, connection response wait, connection response sent, connection established, disconnect wait, and number of link states.

```
Router# debug fras state
```

```
FRAS: TR0 (04/04) oldstate=LS_RQOPNSTNSENT, input=RQ_OPNSTN_CNF
FRAS: newstate=LS_EXCHGXID
```

Related Commands

Command	Description
debug cls message	Limits output for some debug commands based on the interfaces.
debug fras error	Displays information about FRAS protocol errors.
debug fras message	Displays general information about FRAS messages.

debug ftpserver

To display information about the FTP server process, use the **debug ftpserver** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ftpserver

no debug ftpserver

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

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

```
Router# debug ftpserver

Mar  3 10:21:10: %FTPSERVER-6-NEWCONN: FTP Server - new connection made.
-Process= "TCP/FTP Server", ipl= 0, pid= 53
Mar  3 10:21:10: FTPSRV_DEBUG:FTP Server file path: 'disk0:'
Mar  3 10:21:10: FTPSRV_DEBUG:(REPLY)  220
Mar  3 10:21:10: FTPSRV_DEBUG:FTProuter IOS-FTP server (version 1.00) ready.
Mar  3 10:21:10: FTPSRV_DEBUG:FTP Server Command received: 'USER aa'
Mar  3 10:21:20: FTPSRV_DEBUG:(REPLY)  331
Mar  3 10:21:20: FTPSRV_DEBUG>Password required for 'aa'.
Mar  3 10:21:20: FTPSRV_DEBUG:FTP Server Command received: 'PASS aa'
Mar  3 10:21:21: FTPSRV_DEBUG:(REPLY)  230
Mar  3 10:21:21: FTPSRV_DEBUG:Logged in.
Mar  3 10:21:21: FTPSRV_DEBUG:FTP Server Command received: 'SYST'
Mar  3 10:21:21: FTPSRV_DEBUG:(REPLY)  215
Mar  3 10:21:21: FTPSRV_DEBUG:Cisco IOS Type: L8 Version: IOS/FTP 1.00
Mar  3 10:21:21: FTPSRV_DEBUG:FTP Server Command received: 'PWD'
Mar  3 10:21:35: FTPSRV_DEBUG:(REPLY)  257
Mar  3 10:21:39: FTPSRV_DEBUG:FTP Server Command received: 'CWD disk0:/syslogd.d'r/'
Mar  3 10:21:45: FTPSRV_DEBUG:FTP Server file path: 'disk0:/syslogd.dir'
Mar  3 10:21:45: FTPSRV_DEBUG:(REPLY)  250
Mar  3 10:21:45: FTPSRV_DEBUG:CWD command successful.
Mar  3 10:21:45: FTPSRV_DEBUG:FTP Server Command received: 'PORT 171,69,30,20,22',32
Mar  3 10:21:46: FTPSRV_DEBUG:(REPLY)  200
Mar  3 10:21:46: FTPSRV_DEBUG:PORT command successful.
Mar  3 10:21:46: FTPSRV_DEBUG:FTP Server Command received: 'LIST'
Mar  3 10:21:47: FTPSRV_DEBUG:FTP Server file path: 'disk0:/syslogd.dir/.'
Mar  3 10:21:47: FTPSRV_DEBUG:(REPLY)  220
Mar  3 10:23:11: FTPSRV_DEBUG:Opening ASCII mode data connection for file list.
Mar  3 10:23:11: FTPSRV_DEBUG:(REPLY)  226
Mar  3 10:23:12: FTPSRV_DEBUG:Transfer complete.
Mar  3 10:23:12: FTPSRV_DEBUG:FTP Server Command received: 'TYPE I'
Mar  3 10:23:14: FTPSRV_DEBUG:(REPLY)  200
Mar  3 10:23:14: FTPSRV_DEBUG:Type set to I.
Mar  3 10:23:14: FTPSRV_DEBUG:FTP Server Command received: 'PORT 171,69,30,20,22',51
Mar  3 10:23:20: FTPSRV_DEBUG:(REPLY)  200
Mar  3 10:23:20: FTPSRV_DEBUG:PORT command successful.
Mar  3 10:23:20: FTPSRV_DEBUG:FTP Server Command received: 'RETR syslogd.1'
Mar  3 10:23:21: FTPSRV_DEBUG:FTP Server file path: 'disk0:/syslogd.dir/syslogd.1'
Mar  3 10:23:21: FTPSRV_DEBUG:FTPSERVER: Input path passed Top-dir(disk0:/syslogd.dir/)
test.
```

```

Mar  3 10:23:21: FTPSRV_DEBUG:(REPLY) 150
Mar  3 10:23:21: FTPSRV_DEBUG:Opening BINARY mode data connection for syslogd.1 (607317
bytes).
Mar  3 10:23:21: FTPSRV_DEBUG:(REPLY) 226
Mar  3 10:23:29: FTPSRV_DEBUG:Transfer complete.

```

The sample output corresponds to the following FTP client session. In this example, the user connects to the FTP server, views the contents of the top-level directory, and gets a file.

```

FTPclient% ftp FTProuter
Connected to FTProuter.cisco.com.
220 FTProuter IOS-FTP server (version 1.00) ready.
Name (FTProuter:me): aa
331 Password required for 'aa'.
Password:
230 Logged in.
Remote system type is Cisco.
ftp> pwd
257 "disk0:/syslogd.dir/" is current directory.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
syslogd.1
syslogd.2
syslogd.3
syslogd.4
syslogd.5
syslogd.6
syslogd.7
syslogd.8
syslogd.9
syslogd.cur
226 Transfer complete.
ftp> bin
200 Type set to I.
ftp> get syslogd.1
200 PORT command successful.
150 Opening BINARY mode data connection for syslogd.1 (607317 bytes).
226 Transfer complete.
607317 bytes received in 7.7 seconds (77 Kbytes/s)
ftp>

```

The following **debug ftpserver** command output indicates that no top-level directory is specified. Therefore, the client cannot access any location on the FTP server. Use the **ftp-server topdir** command to specify the top-level directory.

```

Mar  3 10:29:14: FTPSRV_DEBUG:(REPLY) 550
Mar  3 10:29:14: FTPSRV_DEBUG:Access denied to 'disk0:'

```

debug gatekeeper gup

To display the Gatekeeper Update Protocol (GUP) events or Abstract Syntax Notation 1 (ASN.1) details, use the **debug gatekeeper gup** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug gatekeeper gup {events | asn1}

no debug gatekeeper gup {events | asn1}

Syntax Description

events	Displays a message whenever a GUP announcement is sent or received. GUP is the protocol used between individual gatekeepers in a cluster, which keeps all the gatekeepers synchronized with all endpoints registered on the cluster.
asn1	ASN.1 library. ASN.1 is an International Telecommunication Union (ITU) standard for protocol syntax and message encoding. Entering this keyword causes a packet dump of all GUP announcement messages.

Defaults

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(5)XM	This command was introduced.
12.2(2)T	This command was integrated into Cisco IOS Release 12.2(2)T.
12.2(2)XB1	This command was implemented on the Cisco AS5850 universal gateway.

Examples

The following example shows how to enable a packet dump of all GUP announcement messages:

```
Router# debug gatekeeper gup asn1

00:10:21:ENCODE BUFFER ::= 00 0A2A8648 86F70C0A 00000120 001E8001
86A08001 86A00547 656E6576 614E0000 00000142 80004700 65006E00 65007600
61080050 00610072 00690073 00000000 0000
00:10:21:
00:10:21:PDU ::=

value GUP_Information ::=

protocolIdentifier { 1 2 840 113548 10 0 0 1 }
message announcementIndication :

announcementInterval 30
endpointCapacity 100000
callCapacity 100000
hostName '47656E657661'H
percentMemory 39
```

```

percentCPU 0
currentCalls 0
currentEndpoints 0
zoneInformation

gatekeeperIdentifier {"Geneva"}
altGKIdentifier {"Paris"}
totalBandwidth 0
interzoneBandwidth 0
remoteBandwidth 0

RAW_BUFFER::=
00 0A2A8648 86F70C0A 00000120 001E800B 858A8001 86A00144 80007400 6F007200 6E006100
64006F00 2D006700 6B120063 00790063 006C006F 006E0065 002D0067 006B0000 00000000
*Mar 3 15:40:31:
*Mar 3 15:40:31:Sending GUP ANNOUNCEMENT INDICATION to 172.18.195.140RAW_BUFFER::=
00 0A2A8648 86F70C0A 00000120 001E800A EF8A8001 86A00144 80006300 79006300 6C006F00
6E006500 2D006700 6B120074 006F0072 006E0061 0064006F 002D0067 006B0000 00000000
*Mar 3 15:40:31:PDU DATA = 60EAB248

value GUP_Information ::=

protocolIdentifier { 1 2 840 113548 10 0 0 1 }
message announcementIndication :
{
announcementInterval 30
endpointCapacity 716682
callCapacity 100000
zoneInformation

gatekeeperIdentifier {"cyclone-gk"}
altGKIdentifier {"tornado-gk"}
totalBandwidth 0
interzoneBandwidth 0
remoteBandwidth 0

Mar 3 15:40:31:Received GUP ANNOUNCEMENT INDICATION from 172.18.195.140

u all
All possible debugging has been turned off

```

Related Commands

Command	Description
load-balance	Configures load balancing.

debug gatekeeper load

To display gatekeeper load-balancing debug events, use the **debug gatekeeper load** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug gatekeeper load {events}

no debug gatekeeper load {events}

Syntax Description	events	Displays a message whenever a load-balancing message is sent or received.
---------------------------	---------------	---

Defaults	Debugging is not enabled.
-----------------	---------------------------

Command Modes	Privileged EXEC
----------------------	-----------------

Command History	Release	Modification
	12.1(5)XM	This command was introduced.
	12.2(2)T	This command was integrated into Cisco IOS Release 12.2(2)T.
	12.2(2)XB1	This command was implemented on the Cisco AS5850 universal gateway.

Examples The following is sample output for the **debug gatekeeper load** command.



Note

The following output examples are independent of each other and would not ordinarily be seen at the same time.

```
Router# debug gatekeeper load

Router#
Router#

Router# show debugging

gk load-balancing debug level = Events
Router#

gk_load_overloaded:Overloaded, 5-second CPU utilization too high

gk_load_overloaded:Overloaded due to excessive calls/endpoints

gk_load_balance_endpt_request:load balance occurred. New load_balance_count=2
```

Related Commands	Command	Description
		load-balance

debug gatekeeper server

To trace all the message exchanges between the Cisco IOS Gatekeeper and the external applications, use the **debug gatekeeper server** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug gatekeeper server

no debug gatekeeper server

Syntax Description This command has no arguments or keywords.

Defaults Disabled

Command Modes Privileged EXEC

Command History	Release	Modification
	12.1(1)T	This command was introduced.

Usage Guidelines Use this command to see information about a Gatekeeper server. This command shows any errors that occur in sending messages to the external applications or in parsing messages from the external applications.

Examples The following example shows debugging information about a Gatekeeper server:

```
Router# debug gatekeeper servers
```

```
Router# show debug
```

```
Gatekeeper:
  Gatekeeper Server Messages debugging is on
```

To turn the Gatekeeper server debugging message off, see the following examples:

```
Router# no debug all
```

```
Router# no debug Gatekeeper servers
```

Related Commands	Command	Description
	show gatekeeper server	Displays information about the Gatekeeper servers configured on your network by ID.

debug glbp errors

To display debugging messages about Gateway Load Balancing Protocol (GLBP) error conditions, use the **debug glbp errors** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug glbp errors

no debug glbp errors

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

Command History	Release	Modification
	12.2(14)S	This command was introduced.
	12.2(15)T	This command was integrated into Cisco IOS Release 12.2(15)T.

Examples The following is sample output from the **debug glbp errors** command:

```
Router# debug glbp errors
```

```
GLBP Errors debugging is on
1d19h: GLBP: Fa0/0 API active virtual address 10.21.8.32 not found
1d19h: GLBP: Fa0/0 API active virtual address 10.21.8.32 not found
1d19h: GLBP: Fa0/0 API active virtual address 10.21.8.32 not found
```

Related Commands	Command	Description
	debug condition glbp	Displays debugging messages about GLBP that match specific conditions.

debug glbp events

To display debugging messages about Gateway Load Balancing Protocol (GLBP) events that are occurring, use the **debug glbp events** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug glbp events [all | detail | terse]
```

```
no debug glbp events [all | detail | terse]
```

Syntax Description

all	(Optional) Displays all debugging output about GLBP events.
detail	(Optional) Displays detailed debugging output about GLBP events.
terse	(Optional) Displays a limited range of debugging output about GLBP events.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(14)S	This command was introduced.
12.2(15)T	This command was integrated into Cisco IOS Release 12.2(15)T.

Examples

The following is sample output from the **debug glbp events** command when the **terse** keyword is specified:

```
Router# debug glbp events terse

GLBP Events debugging is on
  (protocol, redundancy, track)
```

Related Commands

Command	Description
debug condition glbp	Displays debugging messages about GLBP that match specific conditions.

debug glbp packets

To display summary information about Gateway Load Balancing Protocol (GLBP) packets being sent or received, use the **debug glbp packets** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug glbp packets [**all** | **detail** | **hello** | **reply** | **request** | **terse**]

no debug glbp packets [**all** | **detail** | **hello** | **reply** | **request** | **terse**]

Syntax Description

all	(Optional) Displays all debugging output about GLBP packets.
detail	(Optional) Displays detailed debugging output about GLBP packets.
hello	(Optional) Displays debugging output about GLBP hello packets.
reply	(Optional) Displays debugging output about GLBP reply packets.
request	(Optional) Displays debugging output about GLBP request packets.
terse	(Optional) Displays a limited range of debugging output about GLBP packets.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(14)S	This command was introduced.
12.2(15)T	This command was integrated into Cisco IOS Release 12.2(15)T.

Examples

The following sample output from the **debug glbp packets** command shows debugging output about GLBP hello packets:

```
Router# debug glbp packets hello
```

```
GLBP Packets debugging is on
(Hello)
1d19h: GLBP: Fa0/0 Grp 10 Hello out 10.21.8.32 VG Active pri 254 vIP 10.21.8.10 1
1d19h: GLBP: Fa0/0 Grp 10 Hello out 10.21.8.32 VG Active pri 254 vIP 10.21.8.10 1
1d19h: GLBP: Fa0/0 Grp 10 Hello out 10.21.8.32 VG Active pri 254 vIP 10.21.8.10 1
1d19h: GLBP: Fa0/0 Grp 10 Hello out 10.21.8.32 VG Active pri 254 vIP 10.21.8.10 1
```

Related Commands

Command	Description
debug condition glbp	Displays debugging messages about GLBP that match specific conditions.

debug glbp terse

To display a limited range of debugging messages about Gateway Load Balancing Protocol (GLBP) errors, events, and packets, use the **debug glbp terse** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug glbp terse

no debug glbp terse

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

Command History	Release	Modification
	12.2(14)S	This command was introduced.
	12.2(15)T	This command was integrated into Cisco IOS Release 12.2(15)T.

Examples The following is sample output from the **debug glbp terse** command:

```
Router# debug glbp terse

GLBP:
  GLBP Errors debugging is on
  GLBP Events debugging is on
    (protocol, redundancy, track)
  GLBP Packets debugging is on
    (Request, Reply)
```

Related Commands	Command	Description
	debug condition glbp	Displays debugging messages about GLBP that match specific conditions.
	debug glbp errors	Displays debugging messages about GLBP errors.
	debug glbp events	Displays debugging messages about GLBP events.
	debug glbp packets	Displays debugging messages about GLBP packets.

debug gprs charging

To display information about general packet radio service (GPRS) charging functions on the gateway GPRS support node (GGSN), use the **debug gprs charging events** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug gprs charging {events | packets}

no debug gprs charging {events | packets}

Syntax Description

events	Displays events related to GPRS charging processing on the GGSN.
packets	Displays GPRS charging packets that are sent between the GGSN and the charging gateway.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(1)GA	This command was introduced.
12.1(3)T	This command was integrated into Cisco IOS Release 12.1(3)T.

Usage Guidelines

This command is useful for system operators if problems are encountered with GPRS charging functions.



Caution

Because the **debug gprs charging** command generates a substantial amount of output, use it only when traffic on the GPRS network is low, so other activity on the system is not adversely affected.

Examples

The following example enables the display of events related to GPRS charging events on the GGSN:

```
Router# debug gprs charging events
```

The following example enables the display of GPRS charging packets sent between the GGSN and the charging gateway:

```
Router# debug gprs charging events
```

debug gprs gtp

To display information about the GPRS Tunneling Protocol (GTP), use the **debug gprs gtp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug gprs gtp { events | messages | packets }
```

```
no debug gprs gtp { events | messages | packets }
```

Syntax Description	events	Displays events related to GTP processing on the gateway GPRS support node (GGSN).
	messages	Displays GTP signaling messages that are sent between the SGSN and GGSN.
	packets	Displays GTP packets that are sent between the SGSN and GGSN.

Defaults No default behavior or values.

Command History	Release	Modification
	12.1(1)GA	This command was introduced.
	12.1(3)T	This command was integrated in Cisco IOS Release 12.1(3)T.

Usage Guidelines This command is useful for system operators and development engineers if problems are encountered with communication between the GGSN and the SGSN using GTP.



Caution

Because the **debug gprs gtp** command generates a significant amount of output, use it only when traffic on the GPRS network is low, so other activity on the system is not adversely affected.

Examples The following example enables the display of events related to GTP processing on the GGSN:

```
Router# debug gprs gtp events
```

The following example enables the display of GTP signaling messages:

```
Router# debug gprs gtp messages
```

The following example enables the display of GTP packets sent between the SGSN and GGSN:

```
Router# debug gprs gtp packets
```