debug lane config

To display information about a LANE configuration server, use the **debug lane config** privileged EXEC command. The **no** form of this command disables debugging output.

debug lane config {**all** | **events** | **packets**}

no debug lane config {**all** | **events** | **packets**}

Syntax Description

all	Displays all debug messages related to the LANE configuration server. The output includes both the **events** and **packets** types of output.
events	Displays only messages related to significant LANE configuration server events.
packets	Displays information on each packet sent or received by the LANE configuration server.

Usage Guidelines

The **debug lane config** output is intended to be used primarily by a Cisco technical support representative.

Examples

The following is sample output from the **debug lane config all** command when an interface with LECS, an LES/BUS, and an LEC is shut down:

```
Router# debug lane config all

LECS EVENT ATM1/0: processing interface down transition
LECS EVENT ATM1/0: placed de-register address 0x60E8A824
(47.00918100000000613E5A2F01.006070174823.00) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: placed de-register address 0x60EC4F28
(47.007900000000000000000000.00A03E000001.00) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: placed de-register address 0x60EC5C08
(47.00918100000000613E5A2F01.006070174823.99) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: tearing down all connexions
LECS EVENT ATM1/0: elan 'xxx' LES 47.00918100000000613E5A2F01.006070174821.01 callId
0x60CE0F58 deliberately being disconnected
LECS EVENT ATM1/0: sending RELEASE for call 0x60CE0F58 cause 31
LECS EVENT ATM1/0: elan 'yyy' LES 47.00918100000000613E5A2F01.006070174821.02 callId
0x60CE2104 deliberately being disconnected
LECS EVENT ATM1/0: sending RELEASE for call 0x60CE2104 cause 31
LECS EVENT ATM1/0: elan 'zzz' LES 47.00918100000000613E5A2F01.006070174821.03 callId
0x60CE2DC8 deliberately being disconnected
LECS EVENT ATM1/0: sending RELEASE for call 0x60CE2DC8 cause 31
LECS EVENT ATM1/0: All calls to/from LECSs are being released
LECS EVENT ATM1/0: placed de-register address 0x60EC4F28
(47.007900000000000000000000.00A03E000001.00) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: ATM_RELEASE_COMPLETE received: callId 0x60CE0F58 cause 0
LECS EVENT ATM1/0: call 0x60CE0F58 cleaned up
LECS EVENT ATM1/0: ATM_RELEASE_COMPLETE received: callId 0x60CE2104 cause 0
LECS EVENT ATM1/0: call 0x60CE2104 cleaned up
```

```
LECS EVENT ATM1/0: ATM_RELEASE_COMPLETE received: callId 0x60CE2DC8 cause 0
LECS EVENT ATM1/0: call 0x60CE2DC8 cleaned up
LECS EVENT ATM1/0: UNKNOWN/UNSET: signalling DE-registered
LECS EVENT: UNKNOWN/UNSET: signalling DE-registered
LECS EVENT ATM1/0: UNKNOWN/UNSET: signalling DE-registered
LECS EVENT ATM1/0: placed de-register address 0x60E8A824
(47.00918100000000613E5A2F01.006070174823.00) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: placed de-register address 0x60EC5C08
(47.00918100000000613E5A2F01.006070174823.99) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: tearing down all connexions
LECS EVENT ATM1/0: All calls to/from LECSs are being released
LECS EVENT: config server 56 killed
```

debug lane finder

To display information about the finder internal state machine, use the **debug lane finder** privileged EXEC command. The **no** form of this command disables debugging output.

debug lane finder

no debug lane finder

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug lane finder** command output is intended to be used primarily by a Cisco technical support representative.

Examples

The following is sample output from the **debug lane finder** command when an interface with LECS, LES/BUS, and LEC is shut down:

```
Router# debug lane finder

LECS FINDER ATM1/0.3: user request 1819 of type GET_MASTER_LECS_ADDRESS queued up
LECS FINDER ATM1/0: finder state machine started
LECS FINDER ATM1/0: time to perform a getNext on the ILMI
LECS FINDER ATM1/0: LECS 47.00918100000000613E5A2F01.006070174823.00 deleted
LECS FINDER ATM1/0: ilmi_client_request failed, answering all users
LECS FINDER ATM1/0: answering all requests now
LECS FINDER ATM1/0: responded to user request 1819
LECS FINDER ATM1/0: number of remaining requests still to be processed: 0
LECS FINDER ATM1/0.2: user request 1820 of type GET_MASTER_LECS_ADDRESS queued up
LECS FINDER ATM1/0: finder state machine started
LECS FINDER ATM1/0: time to perform a getNext on the ILMI
LECS FINDER ATM1/0: ilmi_client_request failed, answering all users
LECS FINDER ATM1/0: answering all requests now
LECS FINDER ATM1/0: responded to user request 1820
LECS FINDER ATM1/0: number of remaining requests still to be processed: 0
LECS FINDER ATM1/0.1: user request 1821 of type GET_MASTER_LECS_ADDRESS queued up
LECS FINDER ATM1/0: finder state machine started
LECS FINDER ATM1/0: time to perform a getNext on the ILMI
LECS FINDER ATM1/0: ilmi_client_request failed, answering all users
LECS FINDER ATM1/0: answering all requests now
LECS FINDER ATM1/0: responded to user request 1821
LECS FINDER ATM1/0: number of remaining requests still to be processed: 0
```

debug lane server

To display information about a LANE server, use the **debug lane server** privileged EXEC command. The **no** form of this command disables debugging output.

debug lane server [**interface** *interface*]

no debug lane server [**interface** *interface*]

Syntax Description

interface *interface*	(Optional) Limits the debugging output to messages relating to a specific interface or subinterface. If you use this command multiple times with different interfaces, the last interface entered is the one used to filter debug messages.

Usage Guidelines

The **debug lane server** command output is intended to be used primarily by a Cisco technical support representative. The **debug lane server** command can generate a substantial amount of output. Specify a subinterface to decrease the amount of output and focus on the information you need.

Examples

The following is sample output from the **debug lane server** command when an interface with LECS, LES/BUS, and LEC is shut down:

```
Router# debug lane server

LES ATM1/0.1: lsv_lecsAccessSigCB called with callId 0x60CE124C, opcode
ATM_RELEASE_COMPLETE
LES ATM1/0.1: disconnected from the master LECS
LES ATM1/0.1: should have been connected, will reconnect in 3 seconds
LES ATM1/0.2: lsv_lecsAccessSigCB called with callId 0x60CE29E0, opcode
ATM_RELEASE_COMPLETE
LES ATM1/0.2: disconnected from the master LECS
LES ATM1/0.2: should have been connected, will reconnect in 3 seconds
LES ATM1/0.3: lsv_lecsAccessSigCB called with callId 0x60EB1940, opcode
ATM_RELEASE_COMPLETE
LES ATM1/0.3: disconnected from the master LECS
LES ATM1/0.3: should have been connected, will reconnect in 3 seconds
LES ATM1/0.2: elan yyy client 1 lost control distribute
LES ATM1/0.2: elan yyy client 1: lsv_kill_client called
LES ATM1/0.2: elan yyy client 1 state change Oper -> Term
LES ATM1/0.3: elan zzz client 1 lost control distribute
LES ATM1/0.3: elan zzz client 1: lsv_kill_client called
LES ATM1/0.3: elan zzz client 1 state change Oper -> Term
LES ATM1/0.2: elan yyy client 1 lost MC forward
LES ATM1/0.2: elan yyy client 1: lsv_kill_client called
LES ATM1/0.3: elan zzz client 1 lost MC forward
LES ATM1/0.3: elan zzz client 1: lsv_kill_client called
LES ATM1/0.1: elan xxx client 1 lost control distribute
LES ATM1/0.1: elan xxx client 1: lsv_kill_client called
LES ATM1/0.1: elan xxx client 1 state change Oper -> Term
LES ATM1/0.1: elan xxx client 1 lost MC forward
LES ATM1/0.1: elan xxx client 1: lsv_kill_client called
LES ATM1/0.2: elan yyy client 1 released control direct
LES ATM1/0.2: elan yyy client 1: lsv_kill_client called
LES ATM1/0.3: elan zzz client 1 released control direct
LES ATM1/0.3: elan zzz client 1: lsv_kill_client called
LES ATM1/0.2: elan yyy client 1 MC forward released
```

```
LES ATM1/0.2: elan yyy client 1: lsv_kill_client called
LES ATM1/0.2: elan yyy client 1: freeing client structures
LES ATM1/0.2: elan yyy client 1 unregistered 0060.7017.4820
LES ATM1/0.2: elan yyy client 1 destroyed
LES ATM1/0.3: elan zzz client 1 MC forward released
LES ATM1/0.3: elan zzz client 1: lsv_kill_client called
LES ATM1/0.3: elan zzz client 1: freeing client structures
LES ATM1/0.3: elan zzz client 1 unregistered 0060.7017.4820
LES ATM1/0.3: elan zzz client 1 destroyed
LES ATM1/0.1: elan xxx client 1 released control direct
LES ATM1/0.1: elan xxx client 1: lsv_kill_client called
LES ATM1/0.1: elan xxx client 1 MC forward released
LES ATM1/0.1: elan xxx client 1: lsv_kill_client called
LES ATM1/0.1: elan xxx client 1: freeing client structures
LES ATM1/0.1: elan xxx client 1 unregistered 0060.7017.4820
LES ATM1/0.1: elan xxx client 1 destroyed
LES ATM1/0.1: elan xxx major interface state change
LES ATM1/0.1: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.1: shutting down
LES ATM1/0.1: elan xxx: lsv_kill_lesbus called
LES ATM1/0.1: elan xxx: LES/BUS state change operational -> terminating
LES ATM1/0.1: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.2: elan yyy major interface state change
LES ATM1/0.2: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.2: shutting down
LES ATM1/0.2: elan yyy: lsv_kill_lesbus called
LES ATM1/0.2: elan yyy: LES/BUS state change operational -> terminating
LES ATM1/0.2: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.3: elan zzz major interface state change
LES ATM1/0.3: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.3: shutting down
LES ATM1/0.3: elan zzz: lsv_kill_lesbus called
LES ATM1/0.3: elan zzz: LES/BUS state change operational -> terminating
LES ATM1/0.3: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.1: elan xxx: lsv_kill_lesbus called
LES ATM1/0.1: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.1: elan xxx: lsv_kill_lesbus called
LES ATM1/0.1: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.1: elan xxx: stopped listening on addresses
LES ATM1/0.1: elan xxx: all clients killed
LES ATM1/0.1: elan xxx: multicast groups killed
LES ATM1/0.1: elan xxx: addresses de-registered from ilmi
LES ATM1/0.1: elan xxx: LES/BUS state change terminating -> down
LES ATM1/0.1: elan xxx: administratively down
LES ATM1/0.2: elan yyy: lsv_kill_lesbus called
LES ATM1/0.2: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.2: elan yyy: lsv_kill_lesbus called
LES ATM1/0.2: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.2: elan yyy: stopped listening on addresses
LES ATM1/0.2: elan yyy: all clients killed
LES ATM1/0.2: elan yyy: multicast groups killed
LES ATM1/0.2: elan yyy: addresses de-registered from ilmi
LES ATM1/0.2: elan yyy: LES/BUS state change terminating -> down
LES ATM1/0.2: elan yyy: administratively down
LES ATM1/0.3: elan zzz: lsv_kill_lesbus called
LES ATM1/0.3: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.3: elan zzz: lsv_kill_lesbus called
LES ATM1/0.3: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.3: elan zzz: stopped listening on addresses
LES ATM1/0.3: elan zzz: all clients killed
LES ATM1/0.3: elan zzz: multicast groups killed
LES ATM1/0.3: elan zzz: addresses de-registered from ilmi
LES ATM1/0.3: elan zzz: LES/BUS state change terminating -> down
LES ATM1/0.3: elan zzz: administratively down
```

```
LES ATM1/0.3: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.2: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.1: cleanupLecsAccess: discarding all validation requests
```

debug lane signaling

To display information about LANE Server (LES) and BUS switched virtual circuits (SVCs), use the **debug lane signaling** privileged EXEC command. The **no** form of this command disables debugging output.

debug lane signaling [**interface** *interface*]

no debug lane signaling [**interface** *interface*]

Syntax Description

interface *interface*	(Optional) Limits the debugging output to messages relating to a specific interface or subinterface. If you use this command multiple times with different interfaces, the last interface entered is the one used to filter debug messages.

Usage Guidelines

The **debug lane signaling** command output is intended to be used primarily by a Cisco technical support representative. The **debug lane signaling** command can generate a substantial amount of output. Specify a subinterface to decrease the amount of output and focus on the information you need.

Examples

The following is sample output from the **debug lane signaling** command when an interface with LECS, LES/BUS, and LEC is shut down:

```
Router# debug lane signaling

LANE SIG ATM1/0.2: received ATM_RELEASE_COMPLETE callid 0x60EB565C cause 0 lv 0x60E8D348
lvstate LANE_VCC_CONNECTED
LANE SIG ATM1/0.2: lane_sig_mc_release: breaking lv 0x60E8D348 from mcg 0x60E97E84
LANE SIG ATM1/0.2: timer for lv 0x60E8D348 stopped
LANE SIG ATM1/0.2: sent ATM_RELEASE request for lv 0x60E8D468 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.2: sent ATM_RELEASE request for lv 0x60E8D3D8 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.2: sent ATM_RELEASE request for lv 0x60E8D2B8 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.3: received ATM_RELEASE_COMPLETE callid 0x60EB5CA0 cause 0 lv 0x60E8BEF4
lvstate LANE_VCC_CONNECTED
LANE SIG ATM1/0.3: lane_sig_mc_release: breaking lv 0x60E8BEF4 from mcg 0x60E9A37C
LANE SIG ATM1/0.3: timer for lv 0x60E8BEF4 stopped
LANE SIG ATM1/0.3: sent ATM_RELEASE request for lv 0x60E8C014 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.3: sent ATM_RELEASE request for lv 0x60E8BF84 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.3: sent ATM_RELEASE request for lv 0x60E8BE64 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.2: received ATM_RELEASE_COMPLETE callid 0x60EB9040 cause 0 lv 0x60E8D468
lvstate LANE_VCC_DROP_SENT
LANE SIG ATM1/0.2: lane_sig_mc_release: breaking lv 0x60E8D468 from mcg 0x60E97EC8
LANE SIG ATM1/0.2: timer for lv 0x60E8D468 stopped
LANE SIG ATM1/0.3: received ATM_RELEASE_COMPLETE callid 0x60EB97D4 cause 0 lv 0x60E8C014
lvstate LANE_VCC_DROP_SENT
LANE SIG ATM1/0.3: lane_sig_mc_release: breaking lv 0x60E8C014 from mcg 0x60E9A3C0
LANE SIG ATM1/0.3: timer for lv 0x60E8C014 stopped
LANE SIG ATM1/0.1: received ATM_RELEASE_COMPLETE callid 0x60EBCEB8 cause 0 lv 0x60EBBAF0
lvstate LANE_VCC_CONNECTED
LANE SIG ATM1/0.1: lane_sig_mc_release: breaking lv 0x60EBBAF0 from mcg 0x60E8F51C
LANE SIG ATM1/0.1: timer for lv 0x60EBBAF0 stopped
LANE SIG ATM1/0.1: sent ATM_RELEASE request for lv 0x60EBBC10 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.1: sent ATM_RELEASE request for lv 0x60EBBB80 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.1: sent ATM_RELEASE request for lv 0x60EBBA60 in state LANE_VCC_CONNECTED
```

```
LANE SIG ATM1/0.1: received ATM_RELEASE_COMPLETE callid 0x60EBEB00 cause 0 lv 0x60EBBC10
lvstate LANE_VCC_DROP_SENT
LANE SIG ATM1/0.1: lane_sig_mc_release: breaking lv 0x60EBBC10 from mcg 0x60E8F560
LANE SIG ATM1/0.1: timer for lv 0x60EBBC10 stopped
LANE SIG ATM1/0.2: received ATM_RELEASE_COMPLETE callid 0x60E8B174 cause 0 lv 0x60E8D2B8
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.2: timer for lv 0x60E8D2B8 stopped
LANE SIG ATM1/0.3: received ATM_RELEASE_COMPLETE callid 0x60E8B990 cause 0 lv 0x60E8BE64
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.3: timer for lv 0x60E8BE64 stopped
LANE SIG ATM1/0.2: received ATM_RELEASE_COMPLETE callid 0x60EB7FE0 cause 0 lv 0x60E8D3D8
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.2: timer for lv 0x60E8D3D8 stopped
LANE SIG ATM1/0.3: received ATM_RELEASE_COMPLETE callid 0x60EB8554 cause 0 lv 0x60E8BF84
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.3: timer for lv 0x60E8BF84 stopped
LANE SIG ATM1/0.1: received ATM_RELEASE_COMPLETE callid 0x60EBB6D4 cause 0 lv 0x60EBBA60
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.1: timer for lv 0x60EBBA60 stopped
LANE SIG ATM1/0.1: received ATM_RELEASE_COMPLETE callid 0x60EBE24C cause 0 lv 0x60EBBB80
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.1: timer for lv 0x60EBBB80 stopped
LANE SIG ATM1/0.1: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.1: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.2: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.2: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.3: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.3: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.1: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.1: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.2: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.2: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.3: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.3: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
```

debug lapb

To display all traffic for interfaces using Link Access Procedure, Balanced (LAPB) encapsulation, use the **debug lapb** privileged EXEC command. The **no** form of this command disables debugging output.

debug lapb

no debug lapb

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command displays information on the X.25 Layer 2 protocol. It is useful to users familiar with the LAPB protocol.

You can use the **debug lapb** command to determine why X.25 interfaces or LAPB connections are going up and down. It is also useful for identifying link problems, as evidenced when the **show interfaces** EXEC command displays a high number of rejects or frame errors over the X.25 link.

Caution Because the **debug lapb** command generates a substantial amount of output, use it when the aggregate of all LAPB traffic on X.25 and LAPB interfaces is fewer than five frames per second.

Examples

The following is sample output from the **debug lapb** command (the numbers 1 through 7 at the top of the display have been added in order to aid documentation):

```
1         2   3    4     5    6    7
Serial0: LAPB I CONNECT (5) IFRAME P 2 1
Serial0: LAPB O REJSENT (2) REJ F 3
Serial0: LAPB O REJSENT (5) IFRAME 0 3
Serial0: LAPB I REJSENT (2) REJ (C) 7
Serial0: LAPB I DISCONNECT (2) SABM P
Serial0: LAPB O CONNECT (2) UA F
Serial0: LAPB O CONNECT (5) IFRAME 0 0
Serial0: LAPB T1 CONNECT 357964 0
```

Each line of output describes a LAPB event. There are two types of LAPB events: frame events (when a frame enters or exits the LAPB) and timer events. In the sample output, the last line describes a timer event; all of the other lines describe frame events. Table 117 describes the first seven fields.

Table 117 debug lapb Field Descriptions

Field	Description
First field (1)	Interface type and unit number reporting the frame event.
Second field (2)	Protocol providing the information.
Third field (3)	Frame event type. Possible values are as follows: • I—Frame input • O—Frame output • T1—T1 timer expired • T3—Interface outage timer expired • T4—Idle link timer expired
Fourth field (4)	State of the protocol when the frame event occurred. Possible values are as follows: • BUSY (RNR frame received) • CONNECT • DISCONNECT • DISCSENT (disconnect sent) • ERROR (FRMR frame sent) • REJSENT (reject frame sent) • SABMSENT (SABM frame sent)
Fifth field (5)	In a frame event, this value is the size of the frame (in bytes). In a timer event, this value is the current timer value (in milliseconds).

Table 117 debug lapb Field Descriptions (continued)

Field	Description
Sixth field (6)	In a frame event, this value is the frame type name. Possible values for frame type names are as follows: • DISC—Disconnect • DM—Disconnect mode • FRMR—Frame reject • IFRAME—Information frame • ILLEGAL—Illegal LAPB frame • REJ—Reject • RNR—Receiver not ready • RR—Receiver ready • SABM—Set asynchronous balanced mode • SABME—Set asynchronous balanced mode, extended • UA—Unnumbered acknowledgment In a T1 timer event, this value is the number of retransmissions already attempted.
Seventh field (7) (This field will not print if the frame control field is required to appear as either a command or a response, and that frame type is correct.)	This field is only present in frame events. It describes the frame type identified by the LAPB address and Poll/Final bit. Possible values are as follows: • (C)—Command frame • (R)—Response frame • P—Command/Poll frame • F—Response/Final frame • /ERR—Command/Response type is invalid for the control field. An ?ERR generally means that the DTE/DCE assignments are not correct for this link. • BAD-ADDR—Address field is neither Command nor Response

A timer event only displays the first six fields of **debug lapb** command output. For frame events, however, the fields that follow the sixth field document the LAPB control information present in the frame. Depending on the value of the frame type name shown in the sixth field, these fields may or may not appear. Descriptions of the fields following the first six fields follow.

After the Poll/Final indicator, depending on the frame type, three different types of LAPB control information can be printed.

For information frames, the value of the N(S) field and the N(R) field will be printed. The N(S) field of an information frame is the sequence number of that frame, so this field will rotate between 0 and 7 for (modulo 8 operation) or 0 and 127 (for modulo 128 operation) for successive outgoing information frames and (under normal circumstances) also will rotate for incoming information frame streams. The N(R) field is a “piggybacked” acknowledgment for the incoming information frame stream; it informs the other end of the link which sequence number is expected next.

RR, RNR, and REJ frames have an N(R) field, so the value of that field is printed. This field has exactly the same significance that it does in an information frame.

For the FRMR frame, the error information is decoded to display the rejected control field, V(R) and V(S) values, the Response/Command flag, and the error flags WXYZ.

In the following example, the output shows an idle link timer action (T4) where the timer expires twice on an idle link, with the value of T4 set to five seconds:

```
Serial2: LAPB T4 CONNECT 255748
Serial2: LAPB O CONNECT (2) RR P 5
Serial2: LAPB I CONNECT (2) RR F 5
Serial2: LAPB T4 CONNECT 260748
Serial2: LAPB O CONNECT (2) RR P 5
Serial2: LAPB I CONNECT (2) RR F 5
```

The next example shows an interface outage timer expiration (T3):

```
Serial2: LAPB T3 DISCONNECT 273284
```

The following example output shows an error condition when no DCE to DTE connection exists. Note that if a frame has only one valid type (for example, a SABM can only be a command frame), a received frame that has the wrong frame type will be flagged as a receive error (R/ERR in the following output). This feature makes misconfigured links (DTE-DTE or DCE-DCE) easy to spot. Other, less common errors will be highlighed too, such as a too-short or too-long frame, or an invalid address (neither command nor response).

```
Serial2: LAPB T1 SABMSENT 1026508 1
Serial2: LAPB O SABMSENT (2) SABM P
Serial2: LAPB I SABMSENT (2) SABM (R/ERR)
Serial2: LAPB T1 SABMSENT 1029508 2
Serial2: LAPB O SABMSENT (2) SABM P
Serial2: LAPB I SABMSENT (2) SABM (R/ERR)
```

The output in the next example shows the router is misconfigured and has a standard (modulo 8) interface connected to an extended (modulo 128) interface. This condition is indicated by the SABM balanced mode and SABME balanced mode extended messages appearing on the same interface.

```
Serial2: LAPB T1 SABMSENT 1428720 0
Serial2: LAPB O SABMSENT (2) SABME P
Serial2: LAPB I SABMSENT (2) SABM P
Serial2: LAPB T1 SABMSENT 1431720 1
Serial2: LAPB O SABMSENT (2) SABME P
Serial2: LAPB I SABMSENT (2) SABM P
```

debug lapb-ta

To display debug messages for LAPB-TA, use the **debug lapb-ta** privileged EXEC command. Use the **no** form of the command to disable debugging output.

debug lapb-ta [**error** | **event** | **traffic**]

no debug lapb-ta [**error** | **event** | **traffic**]

Syntax Description

error	(Optional) Displays LAPB-TA errors.
event	(Optional) Displays LAPB-TA normal events.
traffic	(Optional) Displays LAPB-TA in/out traffic data.

Defaults

Debugging for LAPB-TA is not enabled.

Command History

Release	Modification
12.0(4)T	This command was introduced.

Examples

The following is sample output from the **debug lapb-ta** command with the **error**, **event**, and **traffic** keywords activated:

```
Router# debug lapb-ta error

LAPB-TA error debugging is on
Router# debug lapb-ta event

LAPB-TA event debugging is on
Router# debug lapb-ta traffic

LAPB-TA traffic debugging is on

Mar  9 12:11:36.464:LAPB-TA:Autodetect trying to detect LAPB on
BR3/0:1
Mar  9 12:11:36.464:  sampled pkt: 2 bytes: 1 3F.. match
Mar  9 12:11:36.468:LAPBTA:get_ll_config:BRI3/0:1
Mar  9 12:11:36.468:LAPBTA:line 130 allocated for BR3/0:1
Mar  9 12:11:36.468:LAPBTA:process 79
Mar  9 12:11:36.468:BR3/0:1:LAPB-TA started
Mar  9 12:11:36.468:LAPBTA:service change:LAPB physical layer up,
context 6183E144 interface up, protocol down
Mar  9 12:11:36.468:LAPBTA:service change:, context 6183E144 up
Mar  9 12:11:36.468:LAPB-TA:BR3/0:1, 44 sent
2d14h:%LINEPROTO-5-UPDOWN:Line protocol on Interface BRI3/0:1, changed state to up
2d14h:%ISDN-6-CONNECT:Interface BRI3/0:1 is now connected to 60213
Mar  9 12:11:44.508:LAPB-TA:BR3/0:1, 1 rcvd
Mar  9 12:11:44.508:LAPB-TA:BR3/0:1, 3 sent
Mar  9 12:11:44.700:LAPB-TA:BR3/0:1, 1 rcvd
Mar  9 12:11:44.700:LAPB-TA:BR3/0:1, 3 sent
Mar  9 12:11:44.840:LAPB-TA:BR3/0:1, 1 rcvd
Mar  9 12:11:44.840:LAPB-TA:BR3/0:1, 14 sent
Mar  9 12:11:45.852:LAPB-TA:BR3/0:1, 1 rcvd
```

```
Mar  9 12:11:46.160:LAPB-TA:BR3/0:1, 2 rcvd
Mar  9 12:11:47.016:LAPB-TA:BR3/0:1, 1 rcvd
Mar  9 12:11:47.016:LAPB-TA:BR3/0:1, 10 sent
```

debug lat packet

To display information on all LAT events, use the **debug lat packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug lat packet

no debug lat packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

For each datagram (packet) received or sent, a message is logged to the console.

Caution This command severely impacts LAT performance and is intended for troubleshooting use only.

Examples

The following is sample output from the **debug lat packet** command:

```
Router# debug lat packet

LAT: I int=Ethernet0, src=0000.0c01.0509, dst=0900.2b00.000f, type=0, M=0, R=0
LAT: I int=Ethernet0, src=0800.2b11.2d13, dst=0000.0c01.7876, type=A, M=0, R=0
LAT: O dst=0800.2b11.2d13, int=Ethernet0, type= A, M=0, R=0, len= 20, next 0 ref 1
```

The second line of output describes a packet that is input to the router. Table 118 describes the fields in this line.

Table 118 debug lat packet Field Descriptions

Field	Description
LAT:	Indicates that this display shows LAT debugging output.
I	Indicates that this line of output describes a packet that is input to the router (I) or output from the router (O).
int = Ethernet0	Indicates the interface on which the packet event took place.
src = 0800.2b11.2d13	Indicates the source address of the packet.

Table 118 debug lat packet Field Descriptions (continued)

Field	Description
dst=0000.0c01.7876	Indicates the destination address of the packet.
type=A	Indicates the message type (in hexadecimal notation). Possible values are as follows: • 0 = Run Circuit • 1 = Start Circuit • 2 = Stop Circuit • A = Service Announcement • C = Command • D = Status • E = Solicit Information • F = Response Information

The third line of output describes a packet that is output from the router. Table 119 describes the last three fields in this line.

Table 119 debug lat packet Field Descriptions

Field	Description
len= 20	Indicates the length (in hexadecimal notation) of the packet (in bytes).
next 0	Indicates the link on the transmit queue.
ref 1	Indicates the count of packet users.

debug lex rcmd

To debug LAN Extender remote commands, use the **debug lex rcmd** privileged EXEC command. The **no** form of this command disables debugging output.

debug lex rcmd

no debug lex rcmd

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug lex rcmd** command:

```
Router# debug lex rcmd

LEX-RCMD: "shutdown" command received on unbound serial interface- Serial0
LEX-RCMD: Lex0: "inventory" command received
Rcvd rcmd: FF 03 80 41 41 13 00 1A 8A 00 00 16 01 FF 00 00
Rcvd rcmd: 00 02 00 00 07 5B CD 15 00 00 0C 01 15 26
LEX-RCMD: ACK or response received on Serial0 without a corresponding ID
LEX-RCMD: REJ received
LEX-RCMD: illegal CODE field received in header: <number>
LEX-RCMD: illegal length for Lex0: "lex input-type-list"
LEX-RCMD: Lex0 is not bound to a serial interface
LEX-RCMD: encapsulation failure
LEX-RCMD: timeout for Lex0: "lex priority-group" command
LEX-RCMD: re-transmitting Lex0: "lex priority-group" command
LEX-RCMD: lex_setup_and_send called with invalid parameter
LEX-RCMD: bind occurred on shutdown LEX interface
LEX-RCMD: Serial0- No free Lex interface found with negotiated MAC address 0000.0c00.d8db
LEX-RCMD: No active Lex interface found for unbind
```

The following output indicates that a LAN Extender remote command packet was received on a serial interface that is not bound to a LAN Extender interface:

```
LEX-RCMD: "shutdown" command received on unbound serial interface- Serial0
```

This message can occur for any of the LAN Extender remote commands. Possible causes of this message are as follows:

- FLEX state machine software error
- Serial line momentarily goes down, which is detected by the host but not by FLEX

The following output indicates that a LAN Extender remote command response has been received. The hexadecimal values are for internal use only.

```
LEX-RCMD: Lex0: "inventory" command received
Rcvd rcmd: FF 03 80 41 41 13 00 1A 8A 00 00 16 01 FF 00 00
Rcvd rcmd: 00 02 00 00 07 5B CD 15 00 00 0C 01 15 26
```

The following output indicates that when the host router originates a LAN Extender remote command to FLEX, it generates an 8-bit identifier that is used to associate a command with its corresponding response:

```
LEX-RCMD: ACK or response received on Serial0 without a corresponding ID
```

This message could be displayed for any of the following reasons:

- FLEX was busy at the time that the command arrived and could not send an immediate response. The command timed out on the host router and then FLEX finally sent the response.
- Transmission error.
- Software error.

Possible responses to Config-Request are Config-ACK, Config-NAK, and Config-Rej. The following output shows that some of the options in the Config-Request are not recognizable or are not acceptable to FLEX due to transmission errors or software errors:

```
LEX-RCMD: REJ received
```

The following output shows that a LAN Extender remote command response was received but that the CODE field in the header was incorrect:

```
LEX-RCMD: illegal CODE field received in header: <number>
```

The following output indicates that a LAN Extender remote command response was received but that it had an incorrect length field. This message can occur for any of the LAN Extender remote commands.

```
LEX-RCMD: illegal length for Lex0: "lex input-type-list"
```

The following output shows that a host router was about to send a remote command when the serial link went down:

```
LEX-RCMD: Lex0 is not bound to a serial interface
```

The following output shows that the serial encapsulation routine of the interface failed to encapsulate the remote command datagram because the LEX-NCP was not in the OPEN state. Due to the way the PPP state machine is implemented, it is normal to see a single encapsulation failure for each remote command that gets sent at bind time.

```
LEX-RCMD: encapsulation failure
```

The following output shows that the timer expired for the given remote command without having received a response from the FLEX device. This message can occur for any of the LAN Extender remote commands.

```
LEX-RCMD: timeout for Lex0: "lex priority-group" command
```

This message could be displayed for any of the following reasons:

- FLEX too busy to respond
- Transmission failure
- Software error

The following output indicates that the host is resending the remote command after a timeout:

```
LEX-RCMD: re-transmitting Lex0: “lex priority-group” command
```

The following output indicates that an illegal parameter was passed to the lex_setup_and_send routine. This message could be displayed for due to a host software error.

```
LEX-RCMD: lex_setup_and_send called with invalid parameter
```

The following output is informational and shows when a bind occurs on a shutdown interface:

```
LEX-RCMD: bind occurred on shutdown LEX interface
```

The following output shows that the LEX-NCP reached the open state and a bind operation was attempted with the FLEX's MAC address, but no free LAN Extender interfaces were found that were configured with that MAC address. This output can occur when the network administrator does not configure a LAN Extender interface with the correct MAC address.

```
LEX-RCMD: Serial0- No free Lex interface found with negotiated MAC address 0000.0c00.d8db
```

The following output shows that the serial line that was bound to the LAN Extender interface went down and the unbind routine was called, but when the list of active LAN Extender interfaces was searched, the LAN Extender interface corresponding to the serial interface was not found. This output usually occurs because of a host software error.

```
LEX-RCMD: No active Lex interface found for unbind
```

debug list

To filter debugging information on a per-interface or per-access list basis, use the **debug list** privileged EXEC command. The **no** form of this command turns off the list filter.

debug list [*list*] [*interface*]

no debug list [*list*] [*interface*]

Syntax Description

list	(Optional) An access list number in the range from 1100 to 1199.
interface	(Optional) The nterface type. Allowed values are the following: • **channel**—IBM Channel interface • **ethernet**—IEEE 802.3 • **fddi**—ANSI X3T9.5 • **null**—Null interface • **serial**—Serial • **tokenring**—IEEE 802.5 • **tunnel**—Tunnel interface

Usage Guidelines

The **debug list** command is used with other **debug** commands for specific protocols and interfaces to filter the amount of debug information that is displayed. In particular, this command is designed to filter specific physical unit (PU) output from bridging protocols. The **debug list** command is supported with the following commands:

- **debug llc2 errors**
- **debug llc2 packets**
- **debug llc2 state**
- **debug rif**
- **debug sdlc**
- **debug token ring**

Note All **debug** commands that support access list filtering use access lists in the range from 1100 to 1199. The access list numbers shown in the examples are merely samples of valid numbers.

Examples

To use the **debug list** command on only the first of several LLC2 connections, use the **show llc2** command to display the active connections:

```
Router# show llc2

SdllcVirtualRing2008 DTE: 4000.2222.22c7 4000.1111.111c 04 04 state NORMAL
SdllcVirtualRing2008 DTE: 4000.2222.22c8 4000.1111.1120 04 04 state NORMAL
SdllcVirtualRing2008 DTE: 4000.2222.22c1 4000.1111.1104 04 04 state NORMAL
```

Next, configure an extended bridging access list, numbered 1103, for the connection you want to filter:

```
access-list 1103 permit 4000.1111.111c 0000.0000.0000 4000.2222.22c7 0000.0000.0000 0xC 2
eq 0x404
```

The convention for the LLC **debug list** command filtering is to use dmac = 6 bytes, smac = 6 bytes, dsap_offset = 12, and ssap_offset = 13.

Finally, you invoke the following **debug** commands:

```
Router# debug list 1103

Router# debug llc2 packet

LLC2 Packets debugging is on
for access list: 1103
```

To use the **debug list** command for SDLC connections, with the exception of address 04, create access list 1102 to deny the specific address and permit all others:

```
access-list 1102 deny 0000.0000.0000 0000.0000.0000 0000.0000.0000 0000.0000.0000 0xC 1 eq
0x4
access-list 1102 permit 0000.0000.0000 0000.0000.0000 0000.0000.0000 0000.0000.0000
```

The convention is to use dmac = 0.0.0, smac = 0.0.0, and sdlc_frame_offset = 12.

Invoke the following **debug** commands:

```
Router# debug list 1102

Router# debug sdlc

SDLC link debugging is on
for access list: 1102
```

To enable SDLC debugging (or debugging for any of the other supported protocols) for a specific interface rather than for all interfaces on a router, use the following commands:

```
Router# debug list serial 0

Router# debug sdlc

SDLC link debugging is on
for interface: Serial0
```

To enable Token Ring debugging between two MAC address, 0000.3018.4acd and 0000.30e0.8250, configure an extended bridging access list 1106:

```
access-list 1106 permit 0000.3018.4acd 8000.0000.0000 0000.30e0.8250 8000.0000.0000
access-list 1106 permit 0000.30e0.8250 8000.0000.0000 0000.3018.4acd 8000.0000.0000
```

Invoke the following **debug** commands:

```
Router# debug list 1106

Router# debug token ring
Token Ring Interface debugging is on
for access list: 1106
```

To enable RIF debugging for a single MAC address, configure an access list 1109:

```
access-list 1109 permit permit 0000.0000.0000 ffff.ffff.ffff 4000.2222.22c6 0000.0000.0000
```

Invoke the following debug commands:

```
Router# debug list 1109
Router# debug rif
RIF update debugging is on
```

```
for access list: 1109
```

Related Commands

Command	Description
debug llc2 errors	Displays LLC2 protocol error conditions or unexpected input.
debug llc2 packet	Displays all input and output from the LLC2 protocol stack.
debug llc2 state	Displays state transitions of the LLC2 protocol.
debug rif	Displays information on entries entering and leaving the RIF cache.
debug rtsp	Displays information on SDLC frames received and sent by any router serial interface involved in supporting SDLC end station functions.
debug token ring	Displays messages about Token Ring interface activity.

debug llc2 dynwind

To display changes to the dynamic window over Frame Relay, use the **debug llc2 dynwind** privileged EXEC command. The **no** form of this command disables debugging output.

debug llc2 dynwind

no debug llc2 dynwind

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug llc2 dynwind** command:

```
Router# debug llc2 dynwind

LLC2/DW: BECN received! event REC_I_CMD, Window size reduced to 4
LLC2/DW: 1 consecutive I-frame(s) received without BECN
LLC2/DW: 2 consecutive I-frame(s) received without BECN
LLC2/DW: 3 consecutive I-frame(s) received without BECN
LLC2/DW: 4 consecutive I-frame(s) received without BECN
LLC2/DW: 5 consecutive I-frame(s) received without BECN
LLC2/DW: Current working window size is 5
```

In this example, the router receives a backward explicit congestion notification (BECN) and reduces the window size to four. After receiving five consecutive I frames without a BECN, the router increases the window size to five.

Related Commands

Command	Description
debug llc2 errors	Displays LLC2 protocol error conditions or unexpected input.
debug llc2 packet	Displays all input and output from the LLC2 protocol stack.
debug llc2 state	Displays state transitions of the LLC2 protocol.

debug llc2 errors

To display Logical Link Control, type 2 (LLC2) protocol error conditions or unexpected input, use the **debug llc2 errors** privileged EXEC command. The **no** form of this command disables debugging output.

debug llc2 errors

no debug llc2 errors

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug llc2 errors** command from a router ignoring an incorrectly configured device:

```
Router# debug llc2 errors

LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
```

Each line of output contains the remote MAC address, the local MAC address, the remote service access point (SAP), and the local SAP. In this example, the router receives unsolicited RR frames marked as responses.

Related Commands

Command	Description
debug list	Filters debugging information on a per-interface or per-access list basis.
debug llc2 dynwind	Displays changes to the dynamic window over Frame Relay.
debug llc2 packet	Displays all input and output from the LLC2 protocol stack.
debug llc2 state	Displays state transitions of the LLC2 protocol.

debug llc2 packet

To display all input and output from the Logical Link Control, type 2 (LLC2) protocol stack, use the **debug llc2 packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug llc2 packet

no debug llc2 packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command also displays information about some error conditions as well as internal interactions between the Common Link Services (CLS) layer and the LLC2 layer.

Examples

The following is sample output from the **debug llc2 packet** command from the router sending ping data back and forth to another router:

```
Router# debug llc2 packet

LLC: llc2_input
401E54F0:                            10400000              .@..
401E5500: 303A90CF 0006F4E1 2A200404 012B5E    0:.O..ta* ...+
LLC: i REC_RR_CMD N(R)=21 p/f=1
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 NORMAL REC_RR_CMD (3)
LLC (rs): 0006.f4e1.2a20 0000.303a.90cf 04 04 REC_RR_CMD N(R)=42
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 txmt RR_RSP N(R)=20 p/f=1
LLC: llc_sendframe
401E5610:              0040 0006F4E1 2A200000        .@..ta* ..
401E5620: 303A90CF 04050129 00               N  0:.O...).       2012
LLC: llc_sendframe
4022E3A0:                        0040 0006F4E1            .@..ta
4022E3B0: 2A200000 303A90CF 04042A28 2C000202  * ..0:.O..*(,...
4022E3C0: 00050B90 A02E0502 FF0003D1 004006C1  .... ......Q.@.A
4022E3D0: D7C9D5C    0.128
 C400130A C1D7D7D5 4BD5F2F0  WIUGD...AWWUKUrp
4022E3E0: F1F30000 011A6071 00010860 D7027000  qs....`q...`W.p.
4022E3F0: 00003B00 1112FF01 03000243 6973636F  ..;........Cisco
4022E400: 20494F53 69                           IOSi
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 txmt I N(S)=21 N(R)=20 p/f=0 size=90
LLC: llc2_input
401E5620:                   10400000 303A90CF          .@..0:.O
401E5630: 0006F4E1 2A200404 282C2C00 02020004  ..ta* ..(,,.....
401E5640: 03902000 1112FF01 03000243 6973636F  .. ........Cisco
401E5650: 20494F53 A0                           IOS
LLC: i REC_I_CMD N(R)=22 N(S)=20 V(R)=20 p/f=0
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 NORMAL REC_I_CMD (1)
LLC (rs): 0006.f4e1.2a20 0000.303a.90cf 04 04 REC_I_CMD N(S)=20 V(R)=20
LLC (rs): 0006.f4e1.2a20 0000.303a.90cf 04 04 REC_I_CMD N(R)=44
LLC: INFO: 0006.f4e1.2a20 0000.303a.90cf 04 04 v(r) 20
```

The first three lines indicate that the router has received some input from the link:

```
LLC: llc2_input
401E54F0:                            10400000              .@..
```

```
401E5500: 303A90CF 0006F4E1 2A200404 012B5E    0:.O..ta* ...+
```

The next line indicates that this input was an RR command with the poll bit set. The other router has received sequence number 21 and is waiting for the final bit.

```
LLC: i REC_RR_CMD N(R)=21 p/f=1
```

The next two lines contain the MAC addresses of the sender and receiver, and the state of the router when it received this frame:

```
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 NORMAL REC_RR_CMD (3)
LLC (rs): 0006.f4e1.2a20 0000.303a.90cf 04 04 REC_RR_CMD N(R)=42
```

The next four lines indicate that the router is sending a response with the final bit set:

```
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 txmt RR_RSP N(R)=20 p/f=1
LLC: llc_sendframe
401E5610:              0040 0006F4E1 2A200000        .@..ta* ..
401E5620: 303A90CF 04050129 00               N  0:.O...).       2012
```

Related Commands

Command	Description
debug list	Filters debugging information on a per-interface or per-access list basis.
debug llc2 dynwind	Displays changes to the dynamic window over Frame Relay.
debug llc2 errors	Displays LLC2 protocol error conditions or unexpected input.
debug llc2 state	Displays state transitions of the LLC2 protocol.

debug llc2 state

To display state transitions of the Logical Link Control, type 2 (LLC2) protocol, use the **debug llc2 state** privileged EXEC command. The **no** form of this command disables debugging output.

debug llc2 state

no debug llc2 state

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Refer to the ISO/IEC standard 8802-2 for definitions and explanations of **debug llc2 state** command output.

Examples

The following is sample output from the **debug llc2 state** command when a router disables and enables an interface:

```
Router# debug llc2 state

LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, NORMAL -> AWAIT (P_TIMER_EXP)
LLC(rs): 0006.f4e1.2a20 0000.303a.90cf 04 04, AWAIT -> D_CONN (P_TIMER_EXP)
LLC: cleanup 0006.f4e1.2a20 0000.303a.90cf 04 04, UNKNOWN (17)
LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, ADM -> SETUP (CONN_REQ)
LLC: normalstate: set_local_busy 0006.f4e1.2a20 0000.303a.90cf 04 04
LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, NORMAL -> BUSY (SET_LOCAL_BUSY)
LLC: Connection established: 0006.f4e1.2a20 0000.303a.90cf 04 04, success
LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, SETUP -> BUSY (SET_LOCAL_BUSY)
LLC: busystate: 0006.f4e1.2a20 0000.303a.90cf 04 04 local busy cleared
LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, BUSY -> NORMAL (CLEAR_LOCAL_BUSY)
```

Related Commands

Command	Description
debug list	Filters debugging information on a per-interface or per-access list basis.
debug llc2 dynwind	Displays changes to the dynamic window over Frame Relay.
debug llc2 errors	Displays LLC2 protocol error conditions or unexpected input.
debug llc2 packet	Displays all input and output from the LLC2 protocol stack.

debug lnm events

To display any unusual events that occur on a Token Ring network, use the **debug lnm events** privileged EXEC command. The **no** form of this command disables debugging output.

debug lnm events

no debug lnm events

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Unusual events include stations reporting errors or error thresholds being exceeded.

Examples

The following is sample output from the **debug lnm events** command:

```
Router# debug lnm events

IBMNM3: Adding 0000.3001.1166 to error list
IBMNM3: Station 0000.3001.1166 going into preweight condition
IBMNM3: Station 0000.3001.1166 going into weight condition
IBMNM3: Removing 0000.3001.1166 from error list
LANMGR0: Beaconing is present on the ring
LANMGR0: Ring is no longer beaconing
IBMNM3: Beaconing, Postmortem Started
IBMNM3: Beaconing, heard from 0000.3000.1234
IBMNM3: Beaconing, Postmortem Next Stage
IBMNM3: Beaconing, Postmortem Finished
```

The following message indicates that station 0000.3001.1166 reported errors and has been added to the list of stations reporting errors. This station is located on Ring 3.

```
IBMNM3: Adding 0000.3001.1166 to error list
```

The following message indicates that station 0000.3001.1166 has passed the “early warning” threshold for error counts:

```
IBMNM3: Station 0000.3001.1166 going into preweight condition
```

The following message indicates that station 0000.3001.1166 is experiencing a severe number of errors:

```
IBMNM3: Station 0000.3001.1166 going into weight condition
```

The following message indicates that the error counts for station 0000.3001.1166 have all decayed to zero, so this station is being removed from the list of stations that have reported errors:

```
IBMNM3: Removing 0000.3001.1166 from error list
```

The following message indicates that Ring 0 has entered failure mode. This ring number is assigned internally.

```
LANMGR0: Beaconing is present on the ring
```

The following message indicates that Ring 0 is no longer in failure mode. This ring number is assigned internally.

```
LANMGR0: Ring is no longer beaconing
```

The following message indicates that the router is beginning its attempt to determine whether any stations left the ring during the automatic recovery process for the last beaconing failure. The router attempts to contact stations that were part of the fault domain to detect whether they are still operating on the ring.

```
IBMNM3: Beaconing, Postmortem Started
```

The following message indicates that the router is attempting to determine whether any stations left the ring during the automatic recovery process for the last beaconing failure. It received a response from station 0000.3000.1234, one of the two stations in the fault domain.

```
IBMNM3: Beaconing, heard from 0000.3000.1234
```

The following message indicates that the router is attempting to determine whether any stations left the ring during the automatic recovery process for the last beaconing failure. It is initiating another attempt to contact the two stations in the fault domain.

```
IBMNM3: Beaconing, Postmortem Next Stage
```

The following message indicates that the router has attempted to determine whether any stations left the ring during the automatic recovery process for the last beaconing failure. It has successfully heard back from both stations that were part of the fault domain.

```
IBMNM3: Beaconing, Postmortem Finished
```

Explanations follow for other messages that the **debug lnm events** command can generate.

The following message indicates that the router is out of memory:

```
LANMGR: memory request failed, find_or_build_station()
```

The following message indicates that Ring 3 is experiencing a large number of errors that cannot be attributed to any individual station:

```
IBMNM3: Non-isolating error threshold exceeded
```

The following message indicates that a station (or stations) on Ring 3 is receiving frames faster than they can be processed:

```
IBMNM3: Adapters experiencing congestion
```

The following message indicates that the beaconing has lasted for over 1 minute and is considered a “permanent” error:

```
IBMNM3: Beaconing, permanent
```

The following message indicates that the beaconing lasted for less than 1 minute. The router is attempting to determine whether either station in the fault domain left the ring.

```
IBMNM: Beaconing, Destination Started
```

In the preceding line of output, the following can replace “Started”: “Next State,” “Finished,” “Timed out,” and “Cannot find station *n*.”

debug lnm llc

To display all communication between the router/bridge and the LAN Network Managers (LNMs) that have connections to it, use the **debug lnm llc** privileged EXEC command. The **no** form of this command disables debugging output.

debug lnm llc

no debug lnm llc

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

One line is displayed for each message sent or received.

Examples

The following is sample output from the **debug lnm llc** command:

```
Router# debug lnm llc

IBMNM: Received LRM Set Reporting Point frame from 1000.5ade.0d8a.
IBMNM: found bridge: 001-2-00A, addresses: 0000.3040.a630 4000.3040.a630
IBMNM: Opening connection to 1000.5ade.0d8a on TokenRing0
IBMNM: Sending LRM LAN Manager Accepted to 1000.5ade.0d8a on link 0.
IBMNM: sending LRM New Reporting Link Established to 1000.5a79.dbf8 on link 1.
IBMNM: Determining new controlling LNM
IBMNM: Sending Report LAN Manager Control Shift to 1000.5ade.0d8a on link 0.
IBMNM: Sending Report LAN Manager Control Shift to 1000.5a79.dbf8 on link 1.

IBMNM: Bridge 001-2-00A received Request Bridge Status from 1000.5ade.0d8a.
IBMNM: Sending Report Bridge Status to 1000.5ade.0d8a on link 0.
IBMNM: Bridge 001-2-00A received Request REM Status from 1000.5ade.0d8a.
IBMNM: Sending Report REM Status to 1000.5ade.0d8a on link 0.
IBMNM: Bridge 001-2-00A received Set Bridge Parameters from 1000.5ade.0d8a.
IBMNM: Sending Bridge Parameters Set to 1000.5ade.0d8a on link 0.
IBMNM: sending Bridge Params Changed Notification to 1000.5a79.dbf8 on link 1.
IBMNM: Bridge 001-2-00A received Set REM Parameters from 1000.5ade.0d8a.
IBMNM: Sending REM Parameters Set to 1000.5ade.0d8a on link 0.
IBMNM: sending REM Parameters Changed Notification to 1000.5a79.dbf8 on link 1.
IBMNM: Bridge 001-2-00A received Set REM Parameters from 1000.5ade.0d8a.
IBMNM: Sending REM Parameters Set to 1000.5ade.0d8a on link 0.
IBMNM: sending REM Parameters Changed Notification to 1000.5a79.dbf8 on link 1.
IBMNM: Received LRM Set Reporting Point frame from 1000.5ade.0d8a.
IBMNM: found bridge: 001-1-00A, addresses: 0000.3080.2d79 4000.3080.2d7
```

As the output indicates, the **debug lnm llc** command output can vary somewhat in format.

Table 120 describes the significant fields shown in the display.

Table 120 debug lnm llc Field Descriptions

Field	Description
IBMNM:	Displays LLC-level debugging information.
Received	Router received a frame. The other possible value is Sending, to indicate that the router is sending a frame.

Table 120 debug lnm llc Field Descriptions (continued)

Field	Description
LRM	The function of the LLC-level software that is communicating as follows: • CRS—Configuration Report Server • LBS—LAN Bridge Server • LRM—LAN Reporting Manager • REM—Ring Error Monitor • RPS—Ring Parameter Server • RS—Ring Station
Set Reporting Point	Name of the specific frame that the router sent or received. Possible values include the following: • Bridge Counter Report • Bridge Parameters Changed Notification • Bridge Parameters Set • CRS Remove Ring Station • CRS Report NAUN Change • CRS Report Station Information • CRS Request Station Information • CRS Ring Station Removed • LRM LAN Manager Accepted • LRM Set Reporting Point • New Reporting Link Established • REM Forward MAC Frame • REM Parameters Changed Notification • REM Parameters Set • Report Bridge Status • Report LAN Manager Control Shift • Report REM Status • Request Bridge Status • Request REM Status • Set Bridge Parameters • Set REM Parameters
from 1000.5ade.0d8a	If the router has received the frame, this address is the source address of the frame. If the router is sending the frame, this address is the destination address of the frame.

The following message indicates that the lookup for the bridge with which the LAN Manager was requesting to communicate was successful:

```
IBMNM: found bridge: 001-2-00A, addresses: 0000.3040.a630 4000.3040.a630
```

The following message indicates that the connection is being opened:

```
IBMNM: Opening connection to 1000.5ade.0d8a on TokenRing0
```

The following message indicates that a LAN Manager has connected or disconnected from an internal bridge and that the router computes which LAN Manager is allowed to change parameters:

```
IBMNM: Determining new controlling LNM
```

The following line of output indicates which bridge in the router is the destination for the frame:

```
IBMNM: Bridge 001-2-00A received Request Bridge Status from 1000.5ade.0d8a.
```

debug lnm mac

To display all management communication between the router/bridge and all stations on the local Token Rings, use the **debug lnm mac** privileged EXEC command. The **no** form of this command disables debugging output.

debug lnm mac

no debug lnm mac

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

One line is displayed for each message sent or received.

Examples

The following is sample output from the **debug lnm mac** command:

```
Router# debug lnm mac

LANMGR0: RS received request address from 4000.3040.a670.
LANMGR0: RS sending report address to 4000.3040.a670.
LANMGR0: RS received request state from 4000.3040.a670.
LANMGR0: RS sending report state to 4000.3040.a670.
LANMGR0: RS received request attachments from 4000.3040.a670.
LANMGR0: RS sending report attachments to 4000.3040.a670.
LANMGR2: RS received ring purge from 0000.3040.a630.
LANMGR2: CRS received report NAUN change from 0000.3040.a630.
LANMGR2: RS start watching ring poll.
LANMGR0: CRS received report NAUN change from 0000.3040.a630.
LANMGR0: RS start watching ring poll.
LANMGR2: REM received report soft error from 0000.3040.a630.
LANMGR0: REM received report soft error from 0000.3040.a630.
LANMGR2: RS received ring purge from 0000.3040.a630.
LANMGR2: RS received AMP from 0000.3040.a630.
LANMGR2: RS received SMP from 0000.3080.2d79.
LANMGR2: CRS received report NAUN change from 1000.5ade.0d8a.
LANMGR2: RS start watching ring poll.
LANMGR0: RS received ring purge from 0000.3040.a630.
LANMGR0: RS received AMP from 0000.3040.a630.
LANMGR0: RS received SMP from 0000.3080.2d79.
LANMGR0: CRS received report NAUN change from 1000.5ade.0d8a.
LANMGR0: RS start watching ring poll.
LANMGR2: RS received SMP from 1000.5ade.0d8a.
LANMGR2: RPS received request initialization from 1000.5ade.0d8a.
LANMGR2: RPS sending initialize station to 1000.5ade.0d8a.
```

Table 121 describes the significant fields shown in the display.

Table 121 debug lnm mac Field Descriptions

Field	Description
LANMGR0:	Indicates that this line of output displays MAC-level debugging information. 0 indicates the number of the Token Ring interface associated with this line of debugging output.
RS	Indicates which function of the MAC-level software is communicating as follows: • CRS—Configuration Report Server • REM—Ring Error Monitor • RPS—Ring Parameter Server • RS—Ring Station
received	Indicates that the router received a frame. The other possible value is sending, to indicate that the router is sending a frame.
request address	Indicates the name of the specific frame that the router sent or received. Possible values include the following: • AMP • initialize station • report address • report attachments • report nearest active upstream neighbor (NAUN) change • report soft error • report state • request address • request attachments • request initialization • request state • ring purge • SMP
from 4000.3040.a670	Indicates the source address of the frame, if the router has received the frame. If the router is sending the frame, this address is the destination address of the frame.

As the output indicates, all **debug lnm mac** command messages follow the format described in Table 121 except the following:

```
LANMGR2: RS start watching ring poll
LANMGR2: RS stop watching ring poll
```

These messages indicate that the router starts and stops receiving AMP and SMP frames. These frames are used to build a current picture of which stations are on the ring.

debug local-ack state

To display the new and the old state conditions whenever there is a state change in the local acknowledgment state machine, use the **debug local-ack state** privileged EXEC command. The **no** form of this command disables debugging output.

debug local-ack state

no debug local-ack state

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug local-ack state** command:

```
Router# debug local-ack state

LACK_STATE: 2370300, hashp 2AE628, old state = disconn, new state = awaiting
LLC2 open to finish
LACK_STATE: 2370304, hashp 2AE628, old state = awaiting LLC2 open to finish,
new state = connected
LACK_STATE: 2373816, hashp 2AE628, old state = connected, new state = disconnected
LACK_STATE: 2489548, hashp 2AE628, old state = disconn, new state = awaiting
LLC2 open to finish
LACK_STATE: 2489548, hashp 2AE628, old state = awaiting LLC2 open to finish,
new state = connected
LACK_STATE: 2490132, hashp 2AE628, old state = connected, new state = awaiting
linkdown response
LACK_STATE: 2490140, hashp 2AE628, old state = awaiting linkdown response,
new state = disconnected
LACK_STATE: 2497640, hashp 2AE628, old state = disconn, new state = awaiting
LLC2 open to finish
LACK_STATE: 2497644, hashp 2AE628, old state = awaiting LLC2 open to finish,
new state = connected
```

Table 122 describes the significant fields in the display.

Table 122 debug local-ack state Field Descriptions

Field	Description
LACK_STATE:	Indicates that this packet describes a state change in the local acknowledgment state machine.
2370300	System clock.
hashp 2AE628	Internal control block pointer used by technical support staff for debugging purposes.

Table 122 debug local-ack state Field Descriptions (continued)

Field	Description
old state = disconn	Old state condition in the local acknowledgment state machine. Possible values include the following: • Disconn (disconnected) • awaiting LLC2 open to finish • connected • awaiting linkdown response
new state = awaiting LLC2 open to finish	New state condition in the local acknowledgment state machine. Possible values include the following: • Disconn (disconnected) • awaiting LLC2 open to finish • connected • awaiting linkdown response

debug management event

To monitor the activities of the Event MIB in real time on your routing device, use the **debug management event** command in privileged EXEC mode. To stop output of debug messages to your screen, use the **no** form of this command.

debug management event

no debug management event

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging output is disabled by default.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(3)T	This command was introduced.

Usage Guidelines

The **debug management event** command prints messages to the screen whenever the Event MIB evaluates a specified trigger. These messages are are given in real-time, and are intendended to be used by technical support engineers for troubleshooting purposes. Definitions for the OID (object identifier) fields can be found in the EVENT-MIB.my file, available for download from the Cisco MIB website on Cisco.com at http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml.

Examples

The following example shows sample output for this command:

```
Router# debug management event

Event Process Bool: Owner aseem, Trigger 01
 Event Bool process: invoke event
 Event Bool process: no wildcarding
Event: OID ifEntry.10.3
Event getValue abs: 69847284
 Event Bool process: Trigger Fired !
 mteSetNotifyObjects:
 Event execOnFiring: sending notification
Event: OID ifEntry.10.1
Event add_objects: Owner , Trigger
Event add_objects: Owner aseem, Trigger sethi
Event Found Owner: aseem
Event Found Name: sethi
Event: OID ifEntry.10.1
 Event: sending trap with 7 OIDs
Event: OID mteHotTrigger.0
Event: OID mteHotTargetName.0
Event: OID mteHotContextName.0
```

```
Event: OID ifEntry.10.3
Event: OID mteHotValue.0
Event: OID ifEntry.10.1
Event: OID ifEntry.10.1
Event mteDoSets: setting oid
 Event mteDoSets: non-wildcarded oid
Event: OID ciscoSyslogMIB.1.2.1.0
Event Thresh Process: Owner aseem, Trigger 01
 Event Thresh process: invoke rising event
 Event Thresh process: invoke falling event
 Event Thresh process: no wildcarding
Event: OID ifEntry.10.3
Event getValue abs: 69847284
Event Existence Process: Owner aseem, Trigger 01
 Event Exist process: invoke event
 Event Exist process: no wildcarding
Event: OID ifEntry.10.3
Event getValue abs: 69847284
 Event Check ExistTrigger for Absent
 Event Check ExistTrigger for Changed
Router# no debug management event
```

Related Commands

Command	Description
show management event	Displays the SNMP Event values that have been configured on your routing device through the use of the Event MIB.

debug mdss

To display the run-time errors and sequence of events for the multicast distributed switching services (MDSS), use the **debug mdss** privileged EXEC command. Use the **no** form of the command to disable debugging output.

debug mdss {**all** | **error** | **event**}

no debug mdss {**all** | **error** | **event**}

Syntax Description

all	Displays both errors and sequence of events for MDSS.
error	Displays the run-time errors for MDSS.
event	Displays the run-time sequence of events for MDSS.

Defaults

Debugging is not enabled.

Command History

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

Examples

The following example shows output using the **debug mdss** command with the **all** keyword:

```
Router# debug mdss all

mdss all debugging is on
Router# clear ip mroute *
Router#
01:31:03: MDSS: got MDFS_CLEARALL
01:31:03: MDSS: --> mdss_flush_all_sc
01:31:03: MDSS: enqueue a FE_GLOBAL_DELETE
01:31:03: MDSS: got MDFS_MROUTE_ADD for (0.0.0.0, 224.0.1.40)
01:31:03: MDSS: --> mdss_free_scmdb_cache
01:31:03: MDSS: got MDFS_MROUTE_ADD for (0.0.0.0, 239.255.158.197)
01:31:03: MDSS: got MDFS_MROUTE_ADD for (192.1.21.6, 239.255.158.197)
01:31:03: MDSS: got a MDFS_MIDB_ADD for (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan22
01:31:03: MDSS: -- mdss_add_oif
01:31:03: MDSS: enqueue a FE_OIF_ADD (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan22
01:31:03: MDSS: mdb (192.1.21.6, 239.255.158.197) fast_flags |
MCACHE_MTU
01:31:03: MDSS: got a MDFS_MIDB_ADD for (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan23
01:31:03: MDSS: -- mdss_add_oif
01:31:03: MDSS: enqueue a FE_OIF_ADD (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan
23
01:31:03: MDSS: mdb (192.1.21.6, 239.255.158.197) fast_flags |
MCACHE_MTU
01:31:03: MDSS: got a MDFS_MIDB_ADD for (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan24
01:31:03: MDSS: -- mdss_add_oif
```

```
01:31:03: MDSS: enqueue a FE_OIF_ADD (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan24
01:31:03: MDSS: mdb (192.1.21.6, 239.255.158.197) fast_flags |
MCACHE_MTU
01:31:03: MDSS: got a MDFS_MIDB_ADD for (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan25
01:31:03: MDSS: -- mdss_add_oif
01:31:03: MDSS: enqueue a FE_OIF_ADD (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan25
01:31:03: MDSS: mdb (192.1.21.6, 239.255.158.197) fast_flags |
MCACHE_MTU
01:31:03: MDSS: got a MDFS_MIDB_ADD for (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan26
01:31:03: MDSS: -- mdss_add_oif

01:31:03: MDSS: enqueue a FE_OIF_ADD (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan26
01:31:03: MDSS: mdb (192.1.21.6, 239.255.158.197) fast_flags |
MCACHE_MTU
01:31:03: MDSS: got a MDFS_MIDB_ADD for (192.1.21.6, 239.255.158.197,u
Vlan21) +Vlan27
```

Related Commands

Command	Description
debug mls rp ip multicast	Displays information relating to MLSP.

debug mgcp

To enable debug traces for errors, events, packets, and the parser, use the **debug mgcp** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mgcp [**all** | **errors** | **events** | **packets** | **parser**]

no debug mgcp [**all** | **errors** | **events** | **packets** | **parser**]

Syntax Description

all	(Optional) Debugs errors, events, packets, and the parser for MGCP modules. **Warning** **Using debug mgcp all may severly impact network performance**
errors	(Optional) Debugs errors for MGCP modules.
events	(Optional) Debugs events for MGCP modules.
packets	(Optional) Debugs packets for MGCP modules.
parser	(Optional) Debugs the parser for MGCP modules.

Defaults

Debugging for DRiP packets is not enabled.

Command Modes

EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced for the Cisco AS5300 access server.
12.1(3)T	The command was modified to display additional information for the gateways.

Examples

The following example illustrates the output for the **debug mgcp all** command with the **all** keyword:

```
Router# debug mgcp all

Router#
20:54:13: MGC stat - 192.168.10.10, total=37, succ=28, failed=8
20:54:13: MGCP Packet received -
CRCX 55560 s0/ds1-0/1 SGCP 1.1
C: 78980
M: sendrecv
L: a:G.726-16


20:54:13: -- mgcp_parse_packet() - call mgcp_parse_header
- mgcp_parse_header()- Request Verb FOUND CRCX
- mgcp_parse_packet() - out mgcp_parse_header
```

```
- SUCCESS: mgcp_parse_packet()-MGCP Header parsing was OK
- mgcp_parse_parameter_lines(), code_str:: 78980, code_len:2, str:1640150312
- mgcp_parse_parameter_lines(str:C: 78980) -num_toks: 19
- mgcp_parse_parameter_lines() check NULL str(78980), in_ptr(C: 78980)
- mgcp_parse_parameter_lines() return Parse function in
mgcp_parm_rules_array[1]
- mgcp_parse_call_id(in_ptr: 78980)
- SUCCESS: mgcp_parse_call_id()-Call ID string(78980) parsing is OK
- mgcp_parse_parameter_lines(), code_str:: sendrecv, code_len:2, str:1640150312
- mgcp_parse_parameter_lines(str:M: sendrecv) -num_toks: 19
- mgcp_parse_parameter_lines() check NULL str(sendrecv), in_ptr(M: sendrecv)
- mgcp_parse_parameter_lines() return Parse function in
mgcp_parm_rules_array[6]
- mgcp_parse_conn_mode(in_ptr: sendrecv)
- mgcp_parse_conn_mode()- tmp_ptr:(sendrecv)
- mgcp_parse_conn_mode(match sendrecv sendrecv
- mgcp_parse_conn_mode(case MODE_SENDRECV)
- SUCCESS: Connection Mode parsing is OK
- mgcp_parse_parameter_lines(), code_str:: a:G.726-16, code_len:2,
str:1640150312
- mgcp_parse_parameter_lines(str:L: a:G.726-16) -num_toks: 19
- mgcp_parse_parameter_lines() check NULL str(a:G.726-16), in_ptr(L:
a:G.726-16)
- mgcp_parse_parameter_lines() return Parse function in
mgcp_parm_rules_array[5]
- mgcp_parse_con_opts()
- mgcp_parse_codecs()
- SUCCESS: CODEC strings parsing is OK- SUCCESS: Local Connection option
parsing is OK- mgcp_val_mandatory_parms()

20:54:13: - SUCCESS: mgcp_parse_packet()- END of Parsing
20:54:13: MGCP msg 1

20:54:13: mgcp_search_call_by_endpt: endpt = s0/ds1-0/1, new_call = 1
20:54:13: slot=0,ds1=0,ds0=1

20:54:13: search endpoint - New call=1, callp 61C28130
20:54:13: callp: 61C28130, vdbptr: 0, state: 0
20:54:13: mgcp_remove_old_ack:
20:54:13: mgcp_idle_crcx: get capability
passthru is 3

20:54:13: process_request_ev- callp 61C28130, voice_if 61C281A4

20:54:13: process_detect_ev- callp 61C28130, voice_if 61C281A4
process_signal_ev- callp 61C28130, voice_ifp 61C281A4

20:54:13: mgcp_process_quarantine_mode- callp 61C28130, voice_if 61C281A4

20:54:13: mgcp_process_quarantine_mode- new q mode: process=0, loop=0

20:54:13: mgcp_xlat_ccapi_error_code - ack_code_tab_index = 0,
20:54:13: No SDP connection info
20:54:13: mgcp_select_codec - LC option, num codec=1, 1st codec=5
20:54:13: mgcp_select_codec - num supprt codec=11
20:54:13: mgcp_select_codec - LC codec list only
20:54:13: codec index=0, bw=16000, codec=5
20:54:13: selected codec=5mgcp_get_pkt_period: voip_codec=2, pkt_period=0, call
adjust_packetization_period
mgcp_get_pkt_period: voip_codec=2, pkt_period=10, after calling
adjust_packetization_period

20:54:13: selected codec 5
20:54:13: IP Precedence=60
```

```
20:54:13: MGCP msg qos value=0mgcp_get_pkt_period: voip_codec=2, pkt_period=0,
call adjust_packetization_period
mgcp_get_pkt_period: voip_codec=2, pkt_period=10, after calling
adjust_packetization_period
mgcp_new_codec_bytes: voip_codec=2, pkt_period=10, codec_bytes=20

20:54:13: callp : 61C28AE8, state : 2, call ID : 40, event : 5, minor evt:
1640137008

20:54:13: MGCPAPP state machine: state = 2, event = 5
20:54:13: mgcp_call_connect: call_id=40, ack will be sent later.
20:54:13: callp : 61C28AE8, new state : 3, call ID : 40

20:54:14: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:54:14: MGCP Session Appl: ignore CCAPI event 22, callp 61C28130

20:54:14: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:54:14: callp : 61C28130, state : 2, call ID : 39, event : 5, minor evt: 20

20:54:14: MGCPAPP state machine: state = 2, event = 5
20:54:14: callp : 61C28130, new state : 3, call ID : 39

20:54:14: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:54:14: callp : 61C28130, state : 3, call ID : 39, event : 6, minor evt: 20

20:54:14: MGCPAPP state machine: state = 3, event = 6
20:54:14: call_id=39, mgcp_ignore_ccapi_ev: ignore 6 for state 3

20:54:14: callp : 61C28130, new state : 3, call ID : 39

20:54:14: MGCP voice mode event

20:54:14: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:54:14: callp : 61C28130, state : 3, call ID : 39, event : 17, minor evt: 0

20:54:14: MGCPAPP state machine: state = 3, event = 17
20:54:14: mgcp_voice_mode_done(): callp 61C28130, major ev 17,
                   minor ev 0mgcp_start_ld_timer: timer already initialized

20:54:14: send_mgcp_create_ack
20:54:14: map_mgcp_error_code_to_string error_tab_index  = 0, protocol version:
2
20:54:14: MGC stat - 1.13.89.3, total=37, succ=29, failed=8
20:54:14: Codec Cnt, 1, first codec 5
20:54:14: First Audio codec, 5, local encoding, 96
20:54:14: -- mgcp_build_packet()-

20:54:14: - mgcp_estimate_msg_buf_length() - 87 bytes needed for header
- mgcp_estimate_msg_buf_length() - 125 bytes needed after checking parameter
lines
- mgcp_estimate_msg_buf_length() - 505 bytes needed  after cheking SDP lines

20:54:14: --- mgcp_build_parameter_lines() ---
- mgcp_build_conn_id()
- SUCCESS: Conn ID string building is OK
- SUCCESS: Building MGCP Parameter lines is OK
- SUCCESS: building sdp owner id (o=) line
- SUCCESS: building sdp session name (s=) line
- SUCCESS: MGCP message building OK
- SUCCESS: END of building
updating lport with 2427

20:54:14: send_mgcp_msg, MGCP Packet sent --->
200 55560
```

```
I: 10

v=0
o=- 78980 0 IN IP4 192.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
t=0 0
m=audio 16444 RTP/AVP 96
a=rtpmap:96 G.726-16/8000/1

<---

20:54:14: enqueue_ack: voice_if=61C281A4, ackqhead=0, ackqtail=0,
ackp=61D753E8, msg=61D00010
20:54:14:
mgcp_process_quarantine_after_ack:ack_code=200mgcp_delete_qb_evt_q:cleanup QB
evt q

20:54:14: callp : 61C28130, new state : 4, call ID : 39
```

The following example illustrates the output for the **debug mgcp** command with the **events** keyword:

```
Router# debug mgcp events

Router#
20:51:40: MGC stat - 192.168.10.10, total=27, succ=20, failed=6
20:51:40: MGCP Packet received -
CRCX 55550 s0/ds1-0/1 SGCP 1.1
C: 100
M: sendonly
L: a:G.726-32, s:on


20:51:40: MGCP msg 1

20:51:40: mgcp_search_call_by_endpt: endpt = s0/ds1-0/1, new_call = 1
20:51:40: slot=0,ds1=0,ds0=1

20:51:40: search endpoint - New call=1, callp 61C28130
20:51:40: callp: 61C28130, vdbptr: 0, state: 0
20:51:40: mgcp_remove_old_ack:
20:51:40: mgcp_idle_crcx: get capability
passthru is 3

20:51:40: process_request_ev- callp 61C28130, voice_if 61C281A4

20:51:40: process_detect_ev- callp 61C28130, voice_if 61C281A4
process_signal_ev- callp 61C28130, voice_ifp 61C281A4

20:51:40: mgcp_process_quarantine_mode- callp 61C28130, voice_if 61C281A4

20:51:40: mgcp_process_quarantine_mode- new q mode: process=0, loop=0

20:51:40: mgcp_xlat_ccapi_error_code - ack_code_tab_index = 0,
20:51:40: No SDP connection info
20:51:40: mgcp_select_codec - LC option, num codec=1, 1st codec=3
20:51:40: mgcp_select_codec - num supprt codec=11
20:51:40: mgcp_select_codec - LC codec list only
20:51:40: codec index=0, bw=32000, codec=3
20:51:40: selected codec=3mgcp_get_pkt_period: voip_codec=4, pkt_period=0, call
adjust_packetization_period
mgcp_get_pkt_period: voip_codec=4, pkt_period=10, after calling
adjust_packetization_period
```

```
20:51:40: selected codec 3
20:51:40: IP Precedence=60
20:51:40: MGCP msg qos value=0mgcp_get_pkt_period: voip_codec=4, pkt_period=0,
call adjust_packetization_period
mgcp_get_pkt_period: voip_codec=4, pkt_period=10, after calling
adjust_packetization_period
mgcp_new_codec_bytes: voip_codec=4, pkt_period=10, codec_bytes=40

20:51:40: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:51:40: MGCP Session Appl: ignore CCAPI event 22, callp 61C28130

20:51:40: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:51:40: callp : 61C28130, state : 2, call ID : 31, event : 5, minor evt: 20

20:51:40: MGCPAPP state machine: state = 2, event = 5
20:51:40: mgcp_call_connect: call_id=31, ack will be sent later.
20:51:40: callp : 61C28130, new state : 3, call ID : 31

20:51:40: callp : 61C28AE8, state : 2, call ID : 32, event : 5, minor evt: 0

20:51:40: MGCPAPP state machine: state = 2, event = 5
20:51:40: callp : 61C28AE8, new state : 3, call ID : 32

20:51:40: callp : 61C28AE8, state : 3, call I 32, event : 6, minor evt: 0

20:51:40: MGCPAPP state machine: state = 3, event = 6
20:51:40: call_id=32, mgcp_ignore_ccapi_ev: ignore 6 for state 3

20:51:40: callp : 61C28AE8, new state : 3, call ID : 32

20:51:41: MGCP voice mode event

20:51:41: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:51:41: callp : 61C28130, state : 3, call ID : 31, event : 17, minor evt: 0

20:51:41: MGCPAPP state machine: state = 3, event = 17
20:51:41: mgcp_voice_mode_done(): callp 61C28130, major ev 17,
                   minor ev 0mgcp_start_ld_timer: timer already initialized

20:51:41: send_mgcp_create_ack
20:51:41: map_mgcp_error_code_to_string error_tab_index  = 0, protocol version:
2
20:51:41: MGC stat - 192.168.10.10, total=27, succ=21, failed=6
20:51:41: Codec Cnt, 1, first codec 3
20:51:41: First Audio codec, 3, local encoding, 96updating lport with 2427

20:51:41: send_mgcp_msg, MGCP Packet sent --->
200 55550
I: C

v=0
o=- 100 0 IN IP4 192.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
t=0 0
m=audio 16434 RTP/AVP 96
a=rtpmap:96 G.726-32/8000/1

<---

20:51:41: enqueue_ack: voice_if=61C281A4, ackqhead=0, ackqtail=0,
ackp=61D75384, msg=61C385EC
20:51:41:
```

```
mgcp_process_quarantine_after_ack:ack_code=200mgcp_delete_qb_evt_q:cleanup QB
evt q

20:51:41: callp : 61C28130, new state : 4, call ID : 31

20:51:41: MGC stat - 192.168.10.10, total=28, succ=21, failed=6
20:51:41: MGCP Packet received -
CRCX 55551 s0/ds1-0/2 SGCP 1.1
C: 100
M: sendrecv
L: a:G.726-32, s:on

v=0
o=- 100 0 IN IP4 191.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
t=0 0
m=audio 16434 RTP/AVP 96
a=rtpmap:96 G.726-32/8000/1


20:51:41: MGCP msg 1

20:51:41: mgcp_search_call_by_endpt: endpt = s0/ds1-0/2, new_call = 1
20:51:41: slot=0,ds1=0,ds0=2

20:51:41: search endpoint - New call=1, callp 61F62380
20:51:41: callp: 61F62380, vdbptr: 0, state: 0
20:51:41: mgcp_remove_old_ack:
20:51:41: mgcp_idle_crcx: get capability
passthru is 3

20:51:41: process_request_ev- callp 61F62380, voice_if 61CDC9A8

20:51:41: process_detect_ev- callp 61F62380, voice_if 61CDC9A8
process_signal_ev- callp 61F62380, voice_ifp 61CDC9A8

20:51:41: mgcp_process_quarantine_mode- callp 61F62380, voice_if 61CDC9A8

20:51:41: mgcp_process_quarantine_mode- new q mode: process=0, loop=0

20:51:41: mgcp_xlat_ccapi_error_code - ack_code_tab_index = 0,
20:51:41: get_peer_info, type 1, proto 1, port 16434
20:51:41: mgcp_select_codec - LC option, num codec=1, 1st codec=3
20:51:41: mgcp_select_codec - SDP list, num codec=1, 1st codec=3
20:51:41: mgcp_select_codec - num supprt codec=11
20:51:41: mgcp_select_codec - peer's pref codec is ok =3
20:51:41:  codec index=100000, bw=1000000, codec=0mgcp_get_pkt_period:
voip_codec=4, pkt_period=0, call adjust_packetization_period
mgcp_get_pkt_period: voip_codec=4, pkt_period=10, after calling
adjust_packetization_period

20:51:41: selected codec 3
20:51:41: IP Precedence=60
20:51:41: MGCP msg qos value=0mgcp_get_pkt_period: voip_codec=4, pkt_period=0,
call adjust_packetization_period
mgcp_get_pkt_period: voip_codec=4, pkt_period=10, after calling
adjust_packetization_period
mgcp_new_codec_bytes: voip_codec=4, pkt_period=10, codec_bytes=40

20:51:41: callp : 61D4CC1C, state : 2, call ID : 34, event : 5, minor evt:
1643520896

20:51:41: MGCPAPP state machine: state = 2, event = 5
```

```
20:51:41: mgcp_call_connect: call_id=34, ack will be sent later.
20:51:41: callp : 61D4CC1C, new state : 3, call ID : 34

20:51:41: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:51:41: MGCP Session Appl: ignore CCAPI event 22, callp 61F62380

20:51:41: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:51:41: callp : 61F62380, state : 2, call ID : 33, event : 5, minor evt: 20

20:51:41: MGCPAPP state machine: state = 2, event = 5
20:51:41: callp : 61F62380, new state : 3, call ID : 33

20:51:41: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:51:41: callp : 61F62380, state : 3, call ID : 33, event : 6, minor evt: 20

20:51:41: MGCPAPP state machine: state = 3, event = 6
20:51:41: call_id=33, mgcp_ignore_ccapi_ev: ignore 6 for state 3

20:51:41: callp : 61F62380, new state : 3, call ID : 33

20:51:41: MGCP voice mode event

20:51:41: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:51:41: callp : 61F62380, state : 3, call ID : 33, event : 17, minor evt: 0

20:51:41: MGCPAPP state machine: state = 3, event = 17
20:51:41: mgcp_voice_mode_done(): callp 61F62380, major ev 17,
                   minor ev 0mgcp_start_ld_timer: timer already initialized

20:51:41: send_mgcp_create_ack
20:51:41: map_mgcp_error_code_to_string error_tab_index  = 0, protocol version:
2
20:51:41: MGC stat - 192.168.10.10, total=28, succ=22, failed=6
20:51:41: Codec Cnt, 1, first codec 3
20:51:41: First Audio codec, 3, local encoding, 96updating lport with 2427

20:51:41: send_mgcp_msg, MGCP Packet sent --->
200 55551
I: D

v=0
o=- 100 0 IN IP4 192.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
t=0 0
m=audio 16538 RTP/AVP 96
a=rtpmap:96 G.726-32/8000/1

<---

20:51:41: enqueue_ack: voice_if=61CDC9A8, ackqhead=0, ackqtail=0,
ackp=61D71C2C, msg=61CFF448
20:51:41:
mgcp_process_quarantine_after_ack:ack_code=200mgcp_delete_qb_evt_q:cleanup QB
evt q

20:51:41: callp : 61F62380, new state : 4, call ID : 33

20:51:41: MGC stat - 192.168.10.10, total=29, succ=22, failed=6
20:51:41: MGCP Packet received -
MDCX 55552 s0/ds1-0/1 SGCP 1.1
C: 100
I: C
M: sendrecv
```

```
L: a:G.726-32, s:on

v=0
o=- 100 0 IN IP4 192.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
t=0 0
m=audio 16538 RTP/AVP 96
a=rtpmap:96 G.726-32/8000/1


20:51:41: MGCP msg 1

20:51:41: mgcp_search_call_by_endpt: endpt = s0/ds1-0/1, new_call = 0
20:51:41: slot=0,ds1=0,ds0=1

20:51:41: search endpoint - New call=0, callp 61C28130
20:51:41: callp: 61C28130, vdbptr: 61C290AC, state: 4
20:51:41: mgcp_remove_old_ack:mgcp_modify_connection: callp 61C28130

20:51:41: process_request_ev- callp 61C28130, voice_if 61C281A4

20:51:41: process_detect_ev- callp 61C28130, voice_if 61C281A4
process_signal_ev- callp 61C28130, voice_ifp 61C281A4

20:51:41: mgcp_process_quarantine_mode- callp 61C28130, voice_if 61C281A4

20:51:41: mgcp_process_quarantine_mode- new q mode: process=0, loop=0

20:51:41: mgcp_select_codec - LC option, num codec=1, 1st codec=3
20:51:41: mgcp_select_codec - SDP list, num codec=1, 1st codec=3
20:51:41: mgcp_select_codec - num supprt codec=11
20:51:41: mgcp_select_codec - peer's pref codec is ok =3
20:51:41:  codec index=100000, bw=1000000, codec=0
20:51:41: MGCP msg qos value=0
20:51:41: get_peer_info, type 1, proto 1, port 16538
20:51:41: mgcp_modify_connection: peer_addr=10D5902, peer_port=0->16538.
20:51:41: call modify - codec change callp 61C28130, callio 31, await_ev 1
20:51:41: mgcp_modify_connection: conn_mode=3.
20:51:41: mgcp_modify_conference: conf_id=11 callid1=31 callid2=32ccapi
conference already exists

20:51:41: mgcp_modify_connection - rtp change, callp 61C28AE8, callid 32,
await_ev 2
20:51:41: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:51:41: callp : 61C28130, state : 4, call ID : 31, event : 16, minor evt:
1640137008

20:51:41: MGCPAPP state machine: state = 4, event = 16
20:51:41: mgcp_call_modified - callp 61C28130, voice_callp 61C28130 voice_if
61C281A4, await_ev 2

20:51:41: callp : 61C28130, new state : 4, call ID : 31

20:51:41: callp : 61C28AE8, state : 4, call ID : 32, event : 16, minor evt: 0

20:51:41: MGCPAPP state machine: state = 4, event = 16
20:51:41: mgcp_call_modified - callp 61C28AE8, voice_callp 61C28130 voice_if
61C281A4, await_ev 1

20:51:41: mgcp_call_modified - SUCCESS
20:51:41: map_mgcp_error_code_to_string error_tab_index  = 0, protocol version:
2
20:51:41: MGC stat - 1.13.89.3, total=29, succ=23, failed=6
```

```
20:51:41: send_mgcp_simple_ackupdating lport with 2427

20:51:41: send_mgcp_msg, MGCP Packet sent --->
200 55552 OK
```

The following example illustrates the output for the **debug mgcp** command with the **packet** keyword:

```
Router# debug mgcp pack
Media Gateway Control Protocol packets debugging is on
Router#
20:50:24: MGCP Packet received -
DLCX 55544 * SGCP 1.1


20:50:24: send_mgcp_msg, MGCP Packet sent --->
250 55544

<---

20:50:31: MGCP Packet received -
CRCX 55545 s0/ds1-0/1 SGCP 1.1
C: 100
M: sendonly
L: a:G.726-32, s:on


20:50:32: send_mgcp_msg, MGCP Packet sent --->
200 55545
I: A

v=0
o=- 100 0 IN IP4 192.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
t=0 0
m=audio 16468 RTP/AVP 96
a=rtpmap:96 G.726-32/8000/1

<---

20:50:32: MGCP Packet received -
CRCX 55546 s0/ds1-0/2 SGCP 1.1
C: 100
M: sendrecv
L: a:G.726-32, s:on

v=0
o=- 100 0 IN IP4 192.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
t=0 0
m=audio 16468 RTP/AVP 96
a=rtpmap:96 G.726-32/8000/1


20:50:32: send_mgcp_msg, MGCP Packet sent --->
200 55546
I: B

v=0
o=- 100 0 IN IP4 192.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
```

```
t=0 0
m=audio 16386 RTP/AVP 96
a=rtpmap:96 G.726-32/8000/1

<---

20:50:32: MGCP Packet received -
MDCX 55547 s0/ds1-0/1 SGCP 1.1
C: 100
I: A
M: sendrecv
L: a:G.726-32, s:on

v=0
o=- 100 0 IN IP4 192.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
t=0 0
m=audio 16386 RTP/AVP 96
a=rtpmap:96 G.726-32/8000/1


20:50:33: send_mgcp_msg, MGCP Packet sent --->
200 55547 OK
```

The following example illustrates the output for the **debug mgcp** command with the **parser** keyword:

```
Router# debug mgcp parser

Router#
20:53:21: -- mgcp_parse_packet() - call mgcp_parse_header
- mgcp_parse_header()- Request Verb FOUND CRCX
- mgcp_parse_packet() - out mgcp_parse_header
- SUCCESS: mgcp_parse_packet()-MGCP Header parsing was OK
- mgcp_parse_parameter_lines(), code_str:: 78980, code_len:2, str:1640150312
- mgcp_parse_parameter_lines(str:C: 78980) -num_toks: 19
- mgcp_parse_parameter_lines() check NULL str(78980), in_ptr(C: 78980)
- mgcp_parse_parameter_lines() return Parse function in
mgcp_parm_rules_array[1]
- mgcp_parse_call_id(in_ptr: 78980)
- SUCCESS: mgcp_parse_call_id()-Call ID string(78980) parsing is OK
- mgcp_parse_parameter_lines(), code_str:: sendrecv, code_len:2, str:1640150312
- mgcp_parse_parameter_lines(str:M: sendrecv) -num_toks: 19
- mgcp_parse_parameter_lines() check NULL str(sendrecv), in_ptr(M: sendrecv)
- mgcp_parse_parameter_lines() return Parse function in
mgcp_parm_rules_array[6]
- mgcp_parse_conn_mode(in_ptr: sendrecv)
- mgcp_parse_conn_mode()- tmp_ptr:(sendrecv)
- mgcp_parse_conn_mode(match sendrecv sendrecv
- mgcp_parse_conn_mode(case MODE_SENDRECV)
- SUCCESS: Connection Mode parsing is OK
- mgcp_parse_parameter_lines(), code_str:: a:G.726-16, code_len:2,
str:1640150312
- mgcp_parse_parameter_lines(str:L: a:G.726-16) -num_toks: 19
- mgcp_parse_parameter_lines() check NULL str(a:G.726-16), in_ptr(L:
a:G.726-16)
- mgcp_parse_parameter_lines() return Parse function in
mgcp_parm_rules_array[5]
- mgcp_parse_con_opts()
- mgcp_parse_codecs()
- SUCCESS: CODEC strings parsing is OK- SUCCESS: Local Connection option
parsing is OK- mgcp_val_mandatory_parms()
```

```
20:53:21: - SUCCESS: mgcp_parse_packet()- END of Parsing
20:53:22: -- mgcp_build_packet()-

20:53:22: - mgcp_estimate_msg_buf_length() - 87 bytes needed for header
- mgcp_estimate_msg_buf_length() - 125 bytes needed after checking parameter
lines
- mgcp_estimate_msg_buf_length() - 505 bytes needed  after cheking SDP lines

20:53:22: --- mgcp_build_parameter_lines() ---
- mgcp_build_conn_id()
- SUCCESS: Conn ID string building is OK
- SUCCESS: Building MGCP Parameter lines is OK
- SUCCESS: building sdp owner id (o=) line
- SUCCESS: building sdp session name (s=) line
- SUCCESS: MGCP message building OK
- SUCCESS: END of building
```

Related Commands

Command	Description
mgcp	Initiates the MGCP daemon.

debug mls rp

To display various IPX Multilayer Switching (MLS) debugging elements, use the **debug mls rp** privileged EXEC command. To disable debugging output, use the **no** form of the command.

debug mls rp {**error** | **events** | **ipx** | **locator** | **packets** | **all**}

no debug mls rp {**error** | **events** | **ipx** | **locator** | **packets** | **all**}

Syntax Description

error	Displays MLS error messages.
events	Displays a run-time sequence of events for the Multilayer Switching Protocol (MLSP).
ipx	Displays IPX-related events for MLS, including route purging and changes to access lists and flow masks.
locator	Identifies which switch is switching a particular flow of MLS explorer packets.
packets	Displays packet contents (in verbose and hexadecimal formats) for MLSP messages.
all	Displays all MLS debugging events.

Defaults

Debugging is not enabled.

Command History

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

Examples

The following example shows output using the **debug mls rp ipx** command:

```
Router# debug mls rp ipx

    IPX MLS debugging is on
Router# conf t
    Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# int vlan 22
Router(config-if)# no ipx access-group out
    05:44:37:FCP:flowmask changed to destination
```

Related Commands

Command	Description
debug dss ipx event	Displays debug messages for route change events that affect IPX MLS.

debug mls rp ip multicast

To display information about Multilayer Switching Protocol (MLSP), use the **debug mls rp ip multicast** privileged EXEC command. Use the **no** form of the command to disable debugging output.

debug mls rp ip multicast {**all** | **error** | **events** | **packets**}

no debug mls rp ip multicast {**all** | **error** | **events** | **packets**}

Syntax Description

all	Displays all multicast MLSP debugging information, including errors, events, and packets.
error	Displays error messages related to multicast MLSP.
events	Displays the run-time sequence of events for multicast MLSP.
packets	Displays the contents of MLSP packets.

Defaults

Debugging is not enabled.

Command History

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

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from the **debug mls rp ip multicast** command using the **error** keyword:

```
Router# debug mls rp ip multicast error

mlsm error debugging is on
chtang-7200#
06:06:45: MLSMERR: scb is INACTIVE, free INSTALL_FE
06:06:46: MLSM: --> mlsm_proc_sc_ins_req(10.0.0.1, 224.2.2.3, 10)
```

The following example shows output from the **debug mls rp ip multicast** command using the **event** keyword:

```
Router# debug mls rp ip multicast event

mlsm events debugging is on
Router#
3d23h: MSCP: incoming shortcut flow statistic  from Fa2/0.11
3d23h: MLSM: Flow_stat: (192.1.10.6, 239.255.158.197), byte :537792
packet:8403
3d23h: MLSM: byte delta:7680 packet delta:120, time delta: 10
3d23h: MSCP: incoming shortcut flow statistic  from Fa2/0.11
3d23h: MLSM: Flow_stat: (192.1.10.6, 239.255.158.197), byte :545472
packet:8523
3d23h: MLSM: byte delta:7680 packet delta:120, time delta: 10
3d23h: MSCP: Router transmits keepalive_msg on Fa2/0.11
```

```
3d23h: MSCP: incoming shortcut keepalive ACK  from Fa2/0.11
3d23h: MLSM: Include-list: (192.1.2.1  ->  0.0.0.0)
3d23h: MSCP: incoming shortcut flow statistic  from Fa2/0.11
3d23h: MLSM: Flow_stat: (192.1.10.6, 239.255.158.197), byte :553152
packet:8643
```

The following example shows output from the **debug mls rp ip multicast** command using the **packet** keyword:

```
Router# debug mls rp ip multicast packet

mlsm packets debugging is on
Router#
Router#
Router#
Router#
**23h: MSCP(I): 01 00 0c cc cc cc 00 e0 1e 7c fe 5f 00 30 aa aa
...LLL.`.|~_.0
..23h: MSCP(I): 03 00 00 0c 01 07 01 05 00 28 01 02 0a c7 00 10
.........(...G
 ..23h: MSCP(I): a6 0b b4 ff 00 00 c0 01 0a 06 ef ff 9e c5 00 00
&.4...@...o..E
3d23h: MSCP(I): 00 00 00 09 42 c0 00 00 00 00 00 00 25 0b
....B@......%.
3d23h:
**23h: MSCP(O): 01 00 0c 00 00 00 aa 00 04 00 01 04 00 00 aa aa
......*.......
LL23h: MSCP(O): 03 00 00 0c 00 16 00 00 00 00 01 00 0c cc cc cc
.............L
..23h: MSCP(O): aa 00 04 00 01 04 00 24 aa aa 03 00 00 0c 01 07
*......$**....
..23h: MSCP(O): 01 06 00 1c c0 01 02 01 aa 00 04 00 01 04 00 00
....@...*.....
3d23h: MSCP(O): 00 0b 00 00 00 00 00 00 01 01 0a 62               ...........b

3d23h:
**23h: MSCP(I): 01 00 0c cc cc cc 00 e0 1e 7c fe 5f 00 24 aa aa
...LLL.`.|~_.$
..23h: MSCP(I): 03 00 00 0c 01 07 01 86 00 1c 01 02 0a c7 00 10
.............G
..23h: MSCP(I): a6 0b b4 ff 00 00 00 0b 00 00 c0 01 02 01 00 00
..4.......@...
3d23h: MSCP(I): 00 00
3d23h:
```

Related Commands

Command	Description
debug mdss	Displays information about MDSS.

debug mmoip aaa

To display output relating to AAA services with the Store and Forward Fax feature, use the **debug mmoip aaa** EXEC command. Use the **no** form of this command to disable debugging output.

debug mmoip aaa

no debug mmoip aaa

Usage Guidelines

This command has no arguments or keywords.

Defaults

Disabled

Command History

Release	Modification
12.0(4)T	This command was introduced.

Examples

The following output shows how the **debug mmoip aaa** command provides information about AAA on-ramp or off-ramp authentication:

```
router# debug mmoip aaa

5d10h:fax_aaa_begin_authentication:User-Name = mmoip-b.cisco.com
5d10h:fax_aaa_begin_authentication:fax_account_id_origin = GATEWAY_ID
5d10h:fax_aaa_end_authentication_callback:Authentication successful
```

The following output shows how the **debug mmoip aaa** command provides information about AAA off-ramp accounting:

```
router# debug mmoip aaa

5d10h:fax_aaa_start_accounting:User-Name = mmoip-b.cisco.com
5d10h:fax_aaa_start_accounting:Calling-Station-Id = gmercuri@mail-server.cisco.com
5d10h:fax_aaa_start_accounting:Called-Station-Id = fax=571-0839@mmoip-b.cisco.com
5d10h:fax_aaa_start_accounting:fax_account_id_origin = GATEWAY_ID
mmoip-b#ax_aaa_start_accounting:fax_msg_id = <37117AF3.3D98300E@mail-server.cisco.com>
5d10h:fax_aaa_start_accounting:fax_pages = 2
5d10h:fax_aaa_start_accounting:fax_coverpage_flag = TRUE
5d10h:fax_aaa_start_accounting:fax_modem_time = 26/32
5d10h:fax_aaa_start_accounting:fax_connect_speed = 14400bps
5d10h:fax_aaa_start_accounting:fax_recipient_count = 1
5d10h:fax_aaa_start_accounting:fax_auth_status = USER SUCCESS
5d10h:fax_aaa_start_accounting:gateway_id = mmoip-b.cisco.com
5d10h:fax_aaa_start_accounting:call_type = Fax Send
5d10h:fax_aaa_start_accounting:port_used =  slot:0 modem port:0
5d10h:fax_aaa_do_offramp_accounting tty(6), Stopping accounting

5d10h:fax_aaa_stop_accounting:ftdb->cact->generic.callActiveTransmitBytes = 18038
5d10h:fax_aaa_stop_accounting:ftdb->cact->generic.callActiveTransmitPackets = 14
```

The following output shows how the **debug mmoip aaa** command provides information about AAA on-ramp accounting:

```
router# debug mmoip aaa

5d10h:fax_aaa_start_accounting:User-Name = mmoip-b.cisco.com
5d10h:fax_aaa_start_accounting:Calling-Station-Id = FAX=408@mail-from-hostname.com
5d10h:fax_aaa_start_accounting:Called-Station-Id = FAX=5710839@mail-server.cisco.com
5d10h:fax_aaa_start_accounting:fax_account_id_origin = GATEWAY_ID
5d10h:fax_aaa_start_accounting:fax_msg_id = 00391997233216263@mmoip-b.cisco.com
5d10h:fax_aaa_start_accounting:fax_pages = 2
5d10h:fax_aaa_start_accounting:fax_modem_time = 22/32
5d10h:fax_aaa_start_accounting:fax_connect_speed = 14400bps
5d10h:fax_aaa_start_accounting:fax_auth_status = USER SUCCESS
5d10h:fax_aaa_start_accounting:email_server_address = 1.14.116.1
5d10h:fax_aaa_start_accounting:email_server_ack_flag = TRUE
5d10h:fax_aaa_start_accounting:gateway_id = mmoip-b.cisco.com
5d10h:fax_aaa_start_accounting:call_type = Fax Receive
5d10h:fax_aaa_start_accounting:port_used = Cisco Powered Fax System slot:1 port:4
5d10h:fax_aaa_do_onramp_accounting tty(5), Stopping accounting

5d10h:fax_aaa_stop_accounting:endb->cact->generic.callActiveTransmitBytes = 26687
5d10h:fax_aaa_stop_accounting:ftdb->cact->generic.callActiveReceiveBytes = 18558
5d10h:fax_aaa_stop_accounting:ftdb->cact->generic.callActiveReceivePackets = 14
```