

Using Show and Debug Commands

This chapter describes how to use show and debug commands to monitor DLSw+ and to troubleshoot. Certain situations may require external equipment (such as protocol analyzers) to understand what is happening in the network environment. DLSw+ is designed to minimize such situations and to provide tools that offer sufficient information in the majority of cases.

In addition to describing DLSw+ show and debug commands, this chapter describes show and debug commands for related feature sets that can be useful in finding problems in DLSw+ environments.

Finally, this chapter includes examples that describe how to use the tools to find and correct problems in the network. The examples provide insight into the correct methodology to find and resolve DLSw+ problems.

DLSw+ Show Commands

DLSw+ provides several show commands that allow you to display relevant information about DLSw+ routers, circuits, peers, and reachability:

- show dlsw capabilities
- show dlsw circuits
- show dlsw fastcache
- show dlsw local-circuit
- show dlsw peers
- show dlsw reachability

The following sections explain each of these commands.

Show DLSw Capabilities

To display the capabilities of the local DLSw+ peer or remote peer use the show dlsw capabilities command. DLSw+ capabilities are always exchanged as part of the peer initiation process. They can also be exchanged (called a “run-time capabilities exchange”) in an active peer session if something changes. Without any keywords, the show dlsw capabilities command shows the capabilities learned from each remote peer. Keywords can be used to specify a particular peer for which capabilities should be shown or to display the capabilities the local peer will advertise to any remote peers.

This command may be useful in determining whether peers support certain features. This may be of particular importance when dealing with DLSw+ features, such as border peering.

The syntax of the show dlsw capabilities command follows:

```
show dlsw capabilities [interface type number | ip-address ip-address | local]
```

Syntax Description

interface—Interface used to access a remote peer (direct/LLC2 encapsulation)

ip-address—IP address of a remote peer (FST or TCP encapsulation)

local—Specifies the local DLSw+ peer

The following sample shows output from a show dlsw capabilities command issued for a local peer:

```
milan#show dlsw capabilities local
DLSw: Capabilities for peer 1.1.1.6(2065)
  vendor id (OUI)       : '00C' (cisco)
  version number       : 1
  release number       : 0
  init pacing window   : 20
  unsupported saps     : none
  num of tcp sessions  : 1
  loop prevent support : no
  icanreach mac-exclusive : no
  icanreach netbios-excl. : no
  reachable mac addresses : none
  reachable netbios names : none
  cisco version number : 1
  peer group number    : 0
  border peer capable  : no
  peer cost            : 3
  biu-segment configured : no
  UDP Unicast support  : yes
  local-ack configured : yes
  priority configured  : no
Cisco Internetwork Operating System Software IOS™ GS Software (GS7-K-M),
Experimental Version 11.1(10956) [sbales 139]
Copyright (c) 1986-1996 by cisco Systems, Inc.
Compiled Thu 30-May-96 09:12 by sbales8
```

Show DLSw Circuits

To display information about the end-to-end sessions using DLSw+, use the show dlsw circuits command. When using TCP encapsulation (or Frame Relay direct encapsulation with local acknowledgment), DLSw+ locally terminates the data-link connection. That is, acknowledgment and keepalive frames are exchanged locally between the end station and the data-link switch while data frames flow directly from one end station to the other. This allows the session to remain active even if the path between the DLSw+ peers suffers a short period of unavailability.

To provide this service, the two DLSw+ peers through which the session is established must keep administrative information about the session (for example, sequence numbers), and the peers must remain synchronized (for instance, if one peer receives a disconnect request from its end station, it must tell the other peer so that it can disconnect from the remote end station). This record keeping and synchronization is accomplished using DLSw+ circuits.

The show commands have been changed in Cisco IOS Releases 11.0(10.1) and 11.1(3.3) to allow users to display circuits at a particular end-station address or SAP. This simplifies and speeds up problem isolation and resolution.



In an environment where two devices are in session across a TCP DLSw+ cloud, you should see corresponding circuits on the DLSw+ peers, and the state should be CONNECTED. There is another state called CKT_ESTABLISHED, which indicates that the routers have set up the circuit successfully, but that the end stations have not yet initiated their sessions across that circuit. This message could be indicative of any number of problems, including problems with XID exchanges or devices for which a VARY ACT command has not been issued from VTAM.

Note that when using FST peers (or direct encapsulation peers not using local acknowledgment), the data-link connection is not locally terminated. The RIF (if Token Ring) is terminated, but the data-link connection is end to end, and you will not see circuits established for sessions across DLSw+ FST or direct peers not using local acknowledgment.

The syntax of the show dlsw circuits command follows:

```
show dlsw circuits [detail] [0-255] [mac-address address | sap-value value / circuit id]
```

Syntax Description

detail—Display full remote circuit details (this keyword can be specified in conjunction with any of the following keywords to minimize the volume of data returned; only one of the following keywords can be specified)

0-255—Display the circuit with this key index

mac-address—Display the remote circuits using a specific MAC

sap-value—Display all remote circuits using a specific SAP

circuit id—Display the circuit id of the circuit index

The following sample shows output from a show dlsw circuits command:

```
milan#show dlsw circuits detail
Index  local addr(lsap)      remote addr(dsap)  state uptime
194 0800.5a9b.b3b2(F0)  800.5ac1.302d(F0)  CONNECTED 00:00:13
      PCEP: 995AA4      UCEP: A52274
      Port: To0/0      peer 172.18.15.166(2065)
      Flow-Control-Tx SQ CW:20, Permitted:28; Rx CW:22, Granted:25 Op:
IWO
      Congestion: LOW(02), Flow Op: Half: 12/5 Reset 1/0
      RIF = 0680.0011.0640
```

In this example, the router had only a single circuit, so detail was specified without any qualifiers. For key routers at a central site, to minimize the output, you may chose to omit the detail if you are listing all of the active circuits.

Show DLSw Fastcache

The show dlsw fastcache command allows you to display the cache being used by DLSw+ when FST or direct (passthrough) encapsulation is used. Using DLSw+ with FST peers or direct encapsulation peers (without local acknowledgment enabled) allows you to use the router's fast-switching capabilities, improving throughput and reducing the load on the router's CPU. To do this, a fast-switching cache must be built. The first frame between two end stations will be process switched, and during this process an entry will be made in the fast-switching cache so that subsequent frames between those end stations may be fast switched.

You can view the fast-switching cache that DLSw+ has created—this information can be useful in determining what path specific data is taking, or to help determine whether traffic is flowing between two specific stations.

The following sample shows output from show dlsw fastcache command:

```
milan#show dlsw fastcache
peer                local-mac          remote-mac l/r sap rif
FST 172.18.15.166  0800.5a9b.b3b2  0800.5ac1.302d
F0/F0 0680.0011.0640
```

Show DLSw Local Circuits

Starting with Cisco IOS Release 11.1, local conversion via DLSw+ became a configurable option. Before this, to convert between diverse data-link protocols, a user had to have two routers peered to each other, each with one of the media types. With the local conversion feature, now this can be done within a single router and with no remote peers required. DLSw+ supports local conversion between SDLC or QLLC and LLC2, and between SDLC and QLLC. To do the data-link conversion, the router must keep state information similar to that described in the section “Show DLSw Circuits.” The router creates a circuit, but in this case both halves of the circuit are maintained on the same router. This information can be collected through the show dlsw local-circuit command. You can specify all local circuits or you can qualify the search in one of the arguments.

```
show dlsw local-circuit [0-63] | [mac-address address] | [sap-value value]
```

Syntax Description

0-63—Display the local circuit with this key index

mac-address—Display all local circuits using the specified MAC

sap-value—Display all local circuits using the specified SAP

The following sample shows output of this command:

```
milan#show dlsw local-circuit
key      mac-addr  sap  state      port rif
58-00   4000.1234.56c1  04  CONNECTED  Se3/7 --no rif--
        PCEP: A4BB04  UCEP: A4BA04
        4001.3745.1088  04  CONNECTED  To0/0 08B0.A041.0DE5.0640
        PCEP: 995A18  UCEP: A4BA04
59-00   4000.1234.56c2  04  CONNECTED  Se3/7 --no rif--
        PCEP: A4C290  UCEP: A4C190
        4001.3745.1088  04  CONNECTED  To0/0 08B0.A041.0DE5.0640
        PCEP: A4B7A4  UCEP: A4C190
```

Show DLSw Peers

The show dlsw peers command allows you to show the status of remote peers. With the exception of local circuits, nothing happens in DLSw+ without remote peer connections. If the peer is not in CONNECT status, no data traffic can flow between end stations that are trying to traverse the peer connection.

In addition to the state of the peer, the show dlsw peers command tells you what kind of peer this is—either configured, promiscuous, or peer-on-demand, which is created when border peers are used. To show the status of a remote peer, use the show dlsw peers command with the following syntax:

```
show dlsw peers [interface type number] | [ip-address ip-address] | [udp]
```

Syntax Description

interface—Interface used to access a remote peer (direct encapsulation)

ip-address—IP address of a remote peer (FST or TCP encapsulation)

udp—UDP frame forwarding statistics for specified peers

The following sample shows output of the show dlsw peers command:

```
milan#show dlsw peers
Peers          state    pkts_rx  pkts_tx  type drops  ckts
TCP    uptime
TCP 172.18.15.166 CONNECT  26086    8400    conf    0      1
0 00:03:42
```

Show DLSw Reachability

You can use the show dlsw reachability command to determine which SNA or NetBIOS DLSw+ end stations a router has in its cache. DLSw+ checks the reachability cache when it is trying to initiate a session to determine if it already knows the correct peer or port to use for this session. It also checks this cache when attempting to send traffic that is not session-based (that is, connectionless) across DLSw+. If DLSw+ does not know where a particular destination address is, it queries other peers that it knows about. When it does learn how to reach a destination, DLSw+ keeps that information for a specific amount of time in an effort to reduce the broadcast traffic on the network.

Reachability tables can become large. To make the table more usable, show commands have been changed in the later maintenance releases to allow you to search the reachability table for a particular MAC address or NetBIOS name. This simplifies problem isolation and diagnosis on a particular station (or a particular protocol). These changes are currently available in Cisco IOS Releases 11.0(10.1) and 11.1(3.3).

You can use the show dlsw reachability command to show the entire reachability cache, or use one of the keywords to show a portion of the reachability cache. Reachability is usually the second item to check when troubleshooting a connection that will not come up. First, check the peer to ensure that it is connected, because no traffic will flow over a disconnected peer. Then check the reachability. You should see that one of the devices is FOUND LOCAL and that the other is FOUND REMOTE (and vice versa on the other peer). If the status of one of the resources is SEARCHING, VERIFY, or not present, there may be a problem in the data path between that device and its nearest DLSw+ peer. If not found, it may imply that the cache entry has timed out.

To determine which end stations are in a router's cache use the show dlsw reachability command with the following syntax:

```
show dlsw reachability [group [value]] | local | remote] | [mac-address [address]] [netbios-names [name]]
```

Syntax Description

group—Displays contents of group reachability cache only

local—Specifies the group number for the reachability check. Only displays group cache entries for the specified group

remote—Displays contents of local reachability cache only

mac-address—Displays all addresses in the reachability cache, or the path to a specific MAC

netbios-names—Displays all NetBIOS names in the reachability cache, or the path to a specific name

The following sample shows output of the show dlsw reachability command:

```
milan#show dlsw reachability
DLSw MAC address reachability cache list
Mac Addr      status  Loc.    peer/port      rif
0800.5a9b.b3b2 FOUND   LOCAL   TokenRing0/0   06B0.0011.0640
0800.5ac1.302d FOUND   REMOTE  172.18.15.166(2065)
DLSw NetBIOS Name reachability cache list
NetBIOS Name  status  Loc.    peer/port      rif
paulo01s      FOUND   REMOTE  172.18.15.166(2065)
vito01r       FOUND   LOCAL   TokenRing0/0   06B0.0011.0640
```

Other Useful Show Commands

Show Interface

The show interface command can be useful to determine whether the data path between the end station and the DLSw+ router is active. In addition, for SDLC interfaces this command provides information about individual devices (SDLC addresses) on a single-drop or multidrop line. The following sample shows output from a show interface command:

```
Serial3/7 is up, line protocol is up
Hardware is cxBus Serial
Description: sdlc config to MVS
MTU 4400 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
Encapsulation SDLC, loopback not set
  Router link station role: PRIMARY (DCE)
  Router link station metrics:
    slow-poll 10 seconds
    T1 (reply time out) 3000 milliseconds
    N1 (max frame size) 12016 bits
    N2 (retry count) 20
    poll-pause-timer 10 milliseconds
    poll-limit-value 1
    k (window size) 7
    modulo 8
    sdlc vmac: 4000.1234.56--
sdlc addr C1 state is CONNECT
  cls_state is CLS_IN_SESSION
  VS 4, VR 4, Remote VR 4, Current retransmit count 0
  Hold queue: 0/200 IFRAMES 20/20
  TESTS 0/0 XIDs 0/0, DMs 0/0 FRMRs 0/0
  RNRs 228/0 SNRMs 1/0 DISC/RDs 0/0 REJs 0/0
  Poll: set, Poll count: 0, chain: C2/C2
sdlc addr C2 state is CONNECT
  cls_state is CLS_IN_SESSION
  VS 4, VR 6, Remote VR 4, Current retransmit count 0
  Hold queue: 0/200 IFRAMES 20/14
  TESTS 0/0 XIDs 0/0, DMs 0/0 FRMRs 0/0
  RNRs 357/0 SNRMs 1/0 DISC/RDs 0/0 REJs 0/0
  Poll: clear, Poll count: 0, ready for poll, chain: C1/C1
Last input never, output 00:00:00, output hang never
Last clearing of "show interface" counters never
Output queue 0/40, 0 drops; input queue 0/75, 0 drops
5 minute input rate 0 bits/sec, 10 packets/sec
5 minute output rate 0 bits/sec, 10 packets/sec
  1663 packets input, 8248 bytes, 0 no buffer
  Received 0 broadcasts, 0 runts, 0 giants
  0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
  1673 packets output, 6832 bytes, 0 underruns
  0 output errors, 0 collisions, 2 interface resets
  0 output buffer failures, 0 output buffers swapped out
  1 carrier transitions
  RTS down, CTS up, DTR up, DCD up, DSR up
```

Show IP Route

The show ip route command can be useful in determining why a peer is not reaching a CONNECT state. Often what appears to be a DLSw+ problem turns out to be an IP routing problem preventing one router from reaching the other. Using the show ip route command or executing an extended ping command to the remote peer address from the local peer address can help ensure that the problem is not an IP connectivity problem.

Show Source Bridge

Local SRB is used to get Token Ring frames into DLSw+. The show source-bridge command can be used to see if this local SRB has been correctly set up in the ring-group. It will also indicate if there are large numbers of SRB drops at the interface level. Large numbers of drops could indicate a problem or could simply indicate the presence of an access list.

Show Bridge

When using transparent bridging as an entry into the DLSw+ cloud (for example, end stations on Ethernet), the show bridge command allows you to determine if the router knows the MAC address of the end stations and, if so, whether they were determined from the correct interface.

Show LLC2

When locally terminating LAN sessions, DLSw+ establishes LLC2 sessions with the LAN-attached end stations. The show llc2 command is useful in monitoring the state of these LLC2 sessions. The following sample shows output of a show llc2 command:

```
milan#show llc
LLC2 Connections: total of 2 connections
TokenRing0/0 DTE: 4001.3745.1088 4000.1234.56c1 04 04 state NORMAL
  V(S)=9, V(R)=12, Last N(R)=9, Local window=7, Remote Window=127
  akmax=3, n2=8, Next timer in 2300
  xid-retry timer      0/0      ack timer          0/1000
  p timer              0/1000   idle timer        2300/10000
  rej timer            0/3200   busy timer        0/9600
  akdelay timer       0/100    txQ count         0/200
TokenRing0/0 DTE: 4001.3745.1088 4000.1234.56c2 04 04 state NORMAL
  V(S)=8, V(R)=9, Last N(R)=8, Local window=7, Remote Window=127
  akmax=3, n2=8, Next timer in 2504
  xid-retry timer      0/0      ack timer          0/1000
  p timer              0/1000   idle timer        2504/10000
  rej timer            0/3200   busy timer        0/9600
  akdelay timer       0/100    txQ count         0/200
```

Show TCP

When using TCP encapsulation, one TCP session (or more) is opened between the TCP peers. The show tcp command shows information about that session, including information about longest and average round-trip timers. This could be useful in finding WAN congestion or routing protocol issues that cause performance problems at the end station.

Other

There are many other commands that may be useful in certain environments, including:

- show dlsw transparent cache
- show dlsw transparent map
- show dlsw transparent neighbor

- show frame-relay map
- show frame-relay pvc
- show lnm station
- show interfaces accounting
- show ip rsvp request
- show ip rsvp reservation
- show ip rsvp sender

DLSw Debug Commands

Although it is possible to turn on all DLSw+ debugging, this may result in far more information than is needed in any particular situation and makes it more difficult to analyze the debug output. When possible, try to determine which debug is needed and turn on as little debug as possible. In addition, remember that it is advisable to use any router debugging only at the direction of Cisco engineers; it is possible to hang a router with too much debug, particularly if the router is running at high-CPU utilization. The following statement illustrates the syntax of the debug dlsw command:

```
debug dlsw [border-peers [interface interface | ip address ip-address] | core
           [flow-control | messages | state | xid] [circuit-number] | local-circuit circuit-number | peers
           [interface interface [fast-errors | fast-paks] | ip address ip-address [fast-errors |
           fast-paks | fst-seq | udp]] | reachability [error | verbose] [sna | netbios]]
```

Syntax Description

border-peers—Enables debugging output for border peer events.

interface—Specifies a remote peer to debug by a direct interface.

ip address—Specifies a remote peer to debug by its IP address.

core—Enables debugging output for DLSw core events.

flow-control—Enables debugging output for congestion in the WAN or at the remote end station.

messages—Enables debugging output of core messages—specific packets received by DLSw either from one of its peers or from a local medium via the Cisco link services interface.

state—Enables debugging output for state changes on the circuit.

xid—Enables debugging output for the exchange identification-state machine.

circuit-number—Specifies the circuit for which you want core debugging output to reduce the of output

local-circuit—Enables debugging output for circuits performing local conversion. Local conversion occurs when both the input and output data-link connections are on the same local peer and no remote peer exist.

peers—Enables debugging output for peer events.

fast-errors—Debugs errors for fast-switched packets.

fast-paks—Debugs fast-switched packets.

fst-seq—Debug FST sequence numbers on fast switched packets.

udp—Debug UDP packets.

reachability—Enables debugging output for reachability events (explorer traffic). If no options are specified, event-level information is displayed for all protocols.

error | verbose—Specifies how much reachability information you want displayed. The verbose keyword displays everything, including errors and events. The error keyword displays error information only. If no option is specified, event-level information is displayed.

sna | netbios—Specifies that reachability information be displayed for only SNA or NetBIOS protocols. If no option is specified, information for all protocols is displayed.

Core Debugging

The DLSw+ core is the engine responsible for establishing and maintaining remote circuits. If possible, specifying the index of the circuit you wish to debug cuts down on the amount of output you get. However, if you want to watch a circuit initially come up, this is not an option. The syntax of the debug dlsw core command is:

```
debug dlsw core [flow-control | messages | state | xid]
```

Syntax Description

flow-control—Limits DLSw+ debug to core flow control.

messages—Limits DLSw+ debug to core messages.

state—Limits DLSw+ debug to core finite state machine state transitions.

xid—Limits DLSw+ debug to core XID command/response bit tracking.

Core flow-control debugging provides information about congestion in the WAN or at the remote end station. If the WAN or remote station is congested, DLSw+ sends receiver not ready frames on its local circuits, throttling data traffic on established sessions and giving the congestion an opportunity to clear.

Core message debugging allows you to view specific packets being received by DLSw+ from one of its peers or from a local medium via Cisco Link Services Interface (CLSI).

Core state debugging allows you to see when the state of a circuit changes. This command is especially useful when attempting to determine why a session is not establishing or why it is being disconnected.

Core XID debugging allows you to track the XID state machine, which the router uses to track XID commands and responses used in negotiations between end stations prior to the establishment of a session.

Local Circuit Debugging

Local circuit debugging is comparable to core debugging for circuits that are established on a single router (Cisco IOS Release 11.1 and later). The same type of information in the complete set of debug dlsw core options is available with this command. The syntax of the debug dlsw local-circuit command is:

```
debug dlsw local-circuit
```

Peer Debugging

Peer debugging is useful in determining why a DLSw+ peer is not reaching CONNECT state or why a peer in CONNECT state is being torn down. This debug is particularly useful in debugging problems related to border peers and peer-on-demand peers. The syntax of the debug dlsw peers command is:

```
debug dlsw peers [interface interface [fast-errors | fast-paks] | ip address ip-address [fast-errors | fast-paks | fst-seq | udp]]
```

Syntax Description

interface—Interface used to reach a remote peer (direct encapsulation only)

ip-address—IP address of a remote peer (TCP or FST encapsulation only)

peers—Enables debugging output for peer events

fast-errors—Debugs errors for fast-switched packets

fast-paks—Debugs fast-switched packets

fst-seq—Debug FST sequence numbers on fast switched packets

udp—Debug UDP packets

The following sample shows output from a debug dlsw peers command during a normal peer connect sequence, displayed from the router that initiated the peer connection:

```
DLSw: action_a() attempting to connect peer 172.18.15.166(2065)
DLSw: action_a(): Write pipe opened for peer 172.18.15.166(2065)
DLSw: peer 172.18.15.166(2065), old state DISCONN, new state WAIT_RD
DLSw: passive open 172.18.15.166(11018) -> 2065
DLSw: action_c(): for peer 172.18.15.166(2065)
DLSw: peer 172.18.15.166(2065), old state WAIT_RD, new state CAP_EXG
DLSw: CapExId Msg sent to peer 172.18.15.166(2065)
DLSw: Recv CapExId Msg from peer 172.18.15.166(2065)
DLSw: Pos CapExResp sent to peer 172.18.15.166(2065)
DLSw: action_e(): for peer 172.18.15.166(2065)
DLSw: Recv CapExPosRsp Msg from peer 172.18.15.166(2065)
DLSw: action_e(): for peer 172.18.15.166(2065)
DLSw: peer 172.18.15.166(2065), old state CAP_EXG, new state CONNECT
DLSw: dlsw_tcpd_fini() for peer 172.18.15.166(2065)
DLSw: dlsw_tcpd_fini() closing write pipe for peer 172.18.15.166
DLSw: action_g(): for peer 172.18.15.166(2065)
DLSw: closing write pipe tcp connection for peer 172.18.15.166(2065)
DLSw: peer_act_on_capabilities() for peer 172.18.15.166(2065)
```

The following sample shows output from a debug dlsw peers command during a normal peer connect sequence, displayed from the router that received the peer connection request:

```
DLSw: passive open 172.18.15.166(11020) -> 2065
DLSw: action_b(): opening write pipe for peer 172.18.15.166(2065)
DLSw: peer 172.18.15.166(2065), old state DISCONN, new state CAP_EXG
DLSw: CapExId Msg sent to peer 172.18.15.166(2065)
DLSw: Recv CapExId Msg from peer 172.18.15.166(2065)
DLSw: Pos CapExResp sent to peer 172.18.15.166(2065)
DLSw: action_e(): for peer 172.18.15.166(2065)
DLSw: Recv CapExPosRsp Msg from peer 172.18.15.166(2065)
DLSw: action_e(): for peer 172.18.15.166(2065)
DLSw: peer 172.18.15.166(2065), old state CAP_EXG, new state CONNECT
DLSw: peer_act_on_capabilities() for peer 172.18.15.166(2065)
DLSw: dlsw_tcpd_fini() for peer 172.18.15.166(2065)
DLSw: dlsw_tcpd_fini() closing write pipe for peer 172.18.15.166
DLSw: action_g(): for peer 172.18.15.166(2065)
DLSw: closing write pipe tcp connection for peer 172.18.15.166(2065)
```

The following sample shows output from a debug dlsw peers command during a normal peer disconnect sequence:

```
DLSw: action_d(): for peer 172.18.15.166(2065)
DLSw: aborting tcp connection for peer 172.18.15.166(11015)
DLSw: peer 172.18.15.166(2065), old state CONNECT, new state DISCONN
```

Reachability Debugging

Reachability debugging allows you to see when entries are added to the DLSw+ reachability cache, when they are deleted from this cache, and when the core is able to find a destination MAC address or NetBIOS name in the cache (thereby avoiding a broadcast). If all this information is required, the verbose keyword should be specified.

```
debug dlsw reachability [error | verbose] [netbios | sna]
```

Syntax Description

error—Show only reachability errors
verbose—Show reachability event detail
netbios—Show only reachability events for NetBIOS
sna—Show only reachability events for SNA

The verbose keyword provides a great deal of information, so two subsets of verbose reachability debugging are available: error and event. Event debugging (default behavior if neither verbose nor error is specified) provides information only about events resulting in a state change, events that are not errors but are somewhat out of the ordinary, and errors. If only the errors are desired, the error keyword can be used. In normal operation, error should produce output only in rare situations (for example, low-memory conditions).

In a further effort to allow the user to minimize output, either the sna or netbios keywords can be specified in addition to one of the other keywords. If one is specified, only reachability debug will be produced if it was caused by that traffic protocol (or any traffic that DLSw+ cannot link to a specific protocol, such as TEST frame). If neither sna nor netbios is specified, debug does not check which protocol a message is related to before printing it.

The following example shows that DLSw+ is receiving TEST frames on the Ethernet interface:

```
CSM: Received CLSI Msg : TEST_STN.Ind dlen: 47 from TokenRing0/0
CSM:   smac c000.0000.0050, dmac 0800.5a54.ee59, ssap 4 , dsap 0
CSM: test_frame_proc: ws_status = SEARCHING
CSM: sending TEST to Serial3/7
CSM: Received CLSI Msg : TEST_STN.Ind dlen: 47 from TokenRing0/0
CSM:   smac c000.0000.0306, dmac 4000.0000.0308, ssap 4 , dsap 0
CSM: test_frame_proc: ws_status = SEARCHING
```

DLSw+ puts the source address into the reachability cache (if it is not already there). The status of SEARCHING here indicates that DLSw+ is already trying to resolve the destination MAC address. This router has already sent one CANUREACH frame to its peers, so there is no need to send another. Had the status been NOT_FOUND, this DLSw+ peer would have sent a CANUREACH frame to all of its peers. Had it been FOUND (in other words, there was already an entry in the reachability cache), the DLSw+ peer would have used that information to respond to the request or to forward the frame toward the destination (depending on whether the cache entry is fresh or stale).

Other Useful Debug Commands

Other useful debug commands include:

- debug source-bridge
- debug sdlc
- debug clsi

Debug Examples

The following examples show how the debug commands can be used to pinpoint the cause of a problem.

Problem 1

No machines from a remote site can reach the central site. The peer at the remote site has IP address 172.18.15.156.

Action 1: Checking the output from the show dlsw peers command, we see:

```
Peers:                state      pkts_rx  pkts_tx  type  drops  ckts
TCP uptime
TCP 172.18.15.156    DISCONN          0         0  conf     0     0
- -
```

Action 2: We can use debug dlsw peers command to determine the problem:

```
DLSw: action_a() attempting to connect peer 172.18.15.156(2065)
DLSw: action_a(): Write pipe opened for peer 172.18.15.156(2065)
DLSw: peer 172.18.15.156(2065), old state DISCONN, new state WAIT_RD
DLSw: dlsw_tcpd_fini() for peer 172.18.15.156(2065)
DLSw: tcp fini closing connection for peer 172.18.15.156(2065)
DLSw: action_d(): for peer 172.18.15.156(2065)
DLSw: peer 172.18.15.156(2065), old state WAIT_RD, new state DISCONN
DLSw: Not promiscuous - Rej conn from 172.18.15.166(2065)
```

Diagnosis: Attempts to open peer 172.18.15.156 are not successful. DLSw+ received an open request from 172.18.15.166, but DLSw+ rejected it because that peer was not defined. Upon investigation, we determine that the peer that we have defined was entered incorrectly and should be 172.18.15.166, which is the device attempting to peer to us. After changing this address, the peer connects:

```
Peers:                state      pkts_rx  pkts_tx  type  drops  ckts
TCP uptime
TCP 172.18.15.166    CONNECT          2         2  conf     0     0
00:24:27
```

Problem 2

SDLC-attached devices are unable to reach the host. Milan is the peer at the remote site where the SDLC devices reside.

Action 1: Issuing the show dlsw peers command tells us the peer is up:

```
milan#sh dlsw peers
Peers:                state      pkts_rx  pkts_tx  type  drops  ckts
TCP uptime
TCP 172.18.15.166    CONNECT          9        140  conf     0     0
00:02:10
```

Action 2: Issuing the show dlsw circuits tells us no circuits are up:

```
milan#show dlsw circuit
milan#
```

Action 3: Issuing a show interfaces command tells us the state of the SDLC addresses is USBUSY, which indicates that we have successfully connected to the downstream SDLC devices:

```
milan#show interfaces 3/7
Serial3/7 is up, line protocol is up
  Hardware is cxBus Serial
  Description: sdlc config to MVS
  MTU 4400 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load
1/255
  Encapsulation SDLC, loopback not set
  Router link station role: PRIMARY (DCE)
  Router link station metrics:
    slow-poll 10 seconds
    T1 (reply time out) 3000 milliseconds
    N1 (max frame size) 12016 bits
    N2 (retry count) 20
    poll-pause-timer 10 milliseconds
    poll-limit-value 1
    k (window size) 7
    modulo 8
    sdlc vmac: 4000.1234.56--

sdlc addr C1 state is USBUSY
  cls_state is CLS_STN_CLOSED
  VS 0, VR 0, Remote VR 0, Current retransmit count 0
  Hold queue: 0/200 IFRAMES 29/18
  TESTs 0/0 XIDs 0/0, DMs 0/1 FRMRs 0/0
  RNRs 620/0 SNRMs 3/0 DISC/RDs 1/0 REJs 0/0
  Poll: clear, Poll count: 0, ready for poll, chain: C2/C2
sdlc addr C2 state is USBUSY
  cls_state is CLS_STN_CLOSED
  VS 0, VR 0, Remote VR 0, Current retransmit count 0
  Hold queue: 0/200 IFRAMES 37/26
  TESTs 0/0 XIDs 0/0, DMs 0/0 FRMRs 0/0
  RNRs 730/0 SNRMs 7/0 DISC/RDs 2/0 REJs 0/0
  Poll: set, Poll count: 0, chain: C1/C1
Last input never, output 00:00:00, output hang never
Last clearing of "show interface" counters never
Output queue 0/40, 0 drops; input queue 3/75, 0 drops
5 minute input rate 0 bits/sec, 40 packets/sec
5 minute output rate 0 bits/sec, 40 packets/sec
  12740307 packets input, 25482189 bytes, 0 no buffer
  Received 0 broadcasts, 0 runts, 0 giants
  0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
  12740340 packets output, 25487483 bytes, 0 underruns
  0 output errors, 0 collisions, 5 interface resets
  0 output buffer failures, 0 output buffers swapped out
  3 carrier transitions
  RTS down, CTS up, DTR up, DCD up, DSR up
```

Action 4: By checking the configuration, we determine that these devices are defined to reach a partner at MAC address 4001.3745.1088:

```
milan#write terminal
...
!
interface Serial3/7
  description sdlc config to MVS
  mtu 4400
  no ip address
  encapsulation sdlc
  no keepalive
  clockrate 9600
  sdlc role primary
  sdlc vmac 4000.1234.5600
  sdlc N1 12016
  sdlc address C1
  sdlc xid C1 05DCCCC1
  sdlc partner 4001.3745.1088 C1
  sdlc address C2
  sdlc xid C2 05DCCCC2
  sdlc partner 4001.3745.1088 C2
  sdlc dlsw C1 C2
!
. . .
```

Action 5: Issuing the show dlsw reachability mac-address command tells us DLsw+ has not been able to find this address:

```
milan#show dlsw reachability mac-address 4001.3745.1088
DLsw MAC address reachability cache list
Mac Addr      status      Loc      peer/port      rif
4001.3745.1088  SEARCHING  REMOTE
```

Action 6: Issuing the show dlsw reachability mac-address address at the FEP-attached router (bolzano) tells us the remote peer is still searching for this resource:

```
bolzano#show dlsw reachability mac-address 4001.3745.1088
DLsw MAC address reachability cache list
Mac Addr      status      Loc.      peer/port      rif
4001.3745.1088  SEARCHING  LOCAL
```

Action 7: We know this is a Token Ring-attached FEP, yet issuing the show source-bridge command tells us that no Token Ring interfaces are set up for SRB:

```
bolzano#show source-bridge
Global RSRB Parameters:
  TCP Queue Length maximum: 100

Ring Group 100:
  No TCP peername set, TCP transport disabled
  Maximum output TCP queue length, per peer: 100
Rings:
```



Diagnosis: After adding the source-bridge statement to interface Token Ring 0, we again issue the show source-bridge command and see:

bolzano#show source-bridge

```
Local Interfaces:                receive          transmit
      srn bn trn r p s n max hops cnt:bytes
cnt:bytes      drops
To0      222 6 100 * f 7 7 7    23:6562          0:0
0
```

```
Global RSRB Parameters:
TCP Queue Length maximum: 100
```

```
Ring Group 100:
No TCP peername set, TCP transport disabled
Maximum output TCP queue length, per peer: 100
Rings:
bn: 6 rn: 222 local ma: 4000.3060.0458 TokenRing0
fwd: 0
```

```
Explorers: ----- input -----          ----- output -----
      spanning all-rings total spanning all-rings
total
To0          0          0          0          0          0
0
Local: fastswitched 19          flushed 0          max Bps 38400

      rings      inputs      bursts      throttles
output drops
To0          19          0          0
0
```

The SDLC circuits have come up:

```
bolzano#show dlsw circuits
Index local addr(lsap) remote addr(dsap) state
250-00 4001.3745.1088(04) 4000.1234.56c1(04) CONNECTED
Port:To0 peer 172.18.15.157(2065)
Flow-Control-Tx CW:20, Permitted:29; Rx CW:20, Granted:32
RIF = 08B0.A041.0DE6.0640
251-00 4001.3745.1088(04) 4000.1234.56c2(04) CONNECTED
Port:To0 peer 172.18.15.157(2065)
Flow-Control-Tx CW:20, Permitted:31; Rx CW:20, Granted:32
RIF = 08B0.A041.0DE6.0640
```

Problem 3

This case is similar to the last case, but one remote SDLC device comes up, while the other remote device does not. Milan is the router attached to the remote SDLC devices.

Action 1: Issuing the show dlsw peers command tells us the peer is up:

```
milan#show dlsw peers
Peers:                state      pkts_rx  pkts_tx  type drops ckts
TCP uptime
TCP 172.18.15.166    CONNECT    561      420    conf    0    2
0 00:26:42
```

Action 2: The show dlsw reachability mac-address command (specifying the MAC address of the FEP) tells us that reachability is all right:

```
milan#show dlsw reachability mac-address 4001.3745.1088
DLSw MAC address reachability cache list
Mac Addr      status      Loc.  peer/port      rif
4001.3745.1088 FOUND      REMOTE 172.18.15.166(2065)
```

Action 3: The show dlsw circuits mac-address command tells us that only one of the two circuits is connected:

```
milan#show dlsw circuit mac-address 4001.3745.1088
Index  local addr(lsap)  remote addr(dsap)  state
250-00 4000.1234.56c1(04) 4001.3745.1088(04) CONNECTED
251-00 4000.1234.56c2(04) 4001.3745.1088(04) CKT_ESTABLISHED
```

The state of CKT_ESTABLISHED tells us that there is a data path between the two devices over which a session could be established, but that session has not yet connected (in this case, no SABME/UA exchange has occurred).

Action 4: Issuing a show debug dlsw core command provides the following output:

```
milan#debug dlsw core state
DLSw core state debugging is on
milan#
DLSw: START-FSM (251-00): event:DLC-Id state:CKT_ESTABLISHED
DLSw: core: dlsw_action_f()
DLSw: END-FSM (251-00): state:CKT_ESTABLISHED->CKT_ESTABLISHED
DLSw: START-FSM (251-00): event:DLC-Id state:CKT_ESTABLISHED
DLSw: core: dlsw_action_f()
DLSw: END-FSM (251-00): state:CKT_ESTABLISHED->CKT_ESTABLISHED
DLSw: START-FSM (251-00): event:WAN-XID state:CKT_ESTABLISHED
DLSw: core: dlsw_action_g()
DLSw: END-FSM (251-00): state:CKT_ESTABLISHED->CKT_ESTABLISHED
DLSw: START-FSM (251-00): event:DLC-Id state:CKT_ESTABLISHED
DLSw: core: dlsw_action_f()
DLSw: END-FSM (251-00): state:CKT_ESTABLISHED->CKT_ESTABLISHED
DLSw: START-FSM (251-00): event:DLC-Id state:CKT_ESTABLISHED
DLSw: core: dlsw_action_f()
DLSw: END-FSM (251-00): state:CKT_ESTABLISHED->CKT_ESTABLISHED
DLSw: START-FSM (251-00): event:WAN-XID state:CKT_ESTABLISHED
DLSw: core: dlsw_action_g()
DLSw: END-FSM (251-00): state:CKT_ESTABLISHED->CKT_ESTABLISHED
DLSw: START-FSM (251-00): event:DLC-Id state:CKT_ESTABLISHED
DLSw: core: dlsw_action_f()
DLSw: END-FSM (251-00): state:CKT_ESTABLISHED->CKT_ESTABLISHED
```

Diagnosis: We see that DLSw+ is seeing and passing XIDs from both the SDLC-attached device and the FEP, yet the FEP is not attempting to initiate the session. This is often an issue with something in the XID (most commonly the IDBLK/IDNUM).



Action 5: Checking the configuration at milan, we see that the XID defined for use on the router is 05DCCCCC:

```
milan#write terminal
. . .
!
interface Serial3/7
  description sdlc config to MVS
  mtu 4400
no ip address
  encapsulation sdlc
  no keepalive
  clockrate 9600
  sdlc role primary
  sdlc vmac 4000.1234.5600
  sdlc N1 12016
  sdlc address C1
  sdlc xid C1 05DCCCC1
  sdlc partner 4001.3745.1088 C1
  sdlc address C2
  sdlc xid C2 05DCCCC
  sdlc partner 4001.3745.1088 C2
. . .
```

Action 6: Checking the configuration in VTAM, we see that the XID is supposed to be 05DCCCC2. There is no way to see what is defined in VTAM from the router; this must be obtained from the host. After changing this value, the session comes up:

```
milan#conf t
Enter configuration commands, one per line. End with CNTL/Z.
milan(config)#int s 3/7
milan(config-if)#sdlc xid c2 05DCCCC2
milan(config-if)#^Z
milan#show dls w circuit
Index   local addr(lsap)  remote addr(dsap)  state
250-00  4000.1234.56c1(04) 4001.3745.1088(04) CONNECTED
251-00  4000.1234.56c2(04) 4001.3745.1088(04) CONNECTED
```

These examples are not meant to be an exhaustive list of the things that can go wrong and how to detect them. However, these are fairly useful in demonstrating how to use the available tools to attack and diagnose any DLSw+ connectivity issue.

