

debug crypto pki transactions

To display debug messages for the trace of interaction (message type) between the certification authority (CA) and the router, use the **debug crypto pki transactions** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug crypto pki transactions

no debug crypto pki transactions

Syntax Description This command has no arguments or keywords.

Defaults Disabled

Command History	Release	Modification
	12.0	This command was introduced.

Usage Guidelines Use the **debug crypto pki transactions** command to display debug messages pertaining to public key infrastructure (PKI) certificates. The messages will show status information during certificate enrollment and verification.

You can also use the **show crypto ca certificates** command to display information about your certificate.

Examples The following example, which authenticates and enrolls a CA, contains sample output for the **debug crypto pki transactions** command:

```
Router(config)# crypto ca authenticate msca
Certificate has the following attributes:
Fingerprint:A5DE3C51 AD8B0207 B60BED6D 9356FB00
% Do you accept this certificate? [yes/no]:y
```

```
Router# debug crypto pki transactions
```

```
00:44:00:CRYPTO_PKI:Sending CA Certificate Request:
GET /certsrv/mscep/mscep.dll/pkiclient.exe?operation=GetCACert&message=msca HTTP/1.0
```

```
00:44:00:CRYPTO_PKI:http connection opened
00:44:01:CRYPTO_PKI:HTTP response header:
HTTP/1.1 200 OK
Server:Microsoft-IIS/5.0
Date:Fri, 17 Nov 2000 18:50:59 GMT
Content-Length:2693
Content-Type:application/x-x509-ca-ra-cert
```

Content-Type indicates we have received CA and RA certificates.

```
00:44:01:CRYPTO_PKI:WARNING:A certificate chain could not be constructed while selecting
certificate status
```

```
00:44:01:CRYPTO_PKI:WARNING:A certificate chain could not be constructed while selecting
certificate status

00:44:01:CRYPTO_PKI:Name:CN = msca-rootRA, O = Cisco System, C = US
00:44:01:CRYPTO_PKI:Name:CN = msca-rootRA, O = Cisco System, C = US
00:44:01:CRYPTO_PKI:transaction GetCACert completed
00:44:01:CRYPTO_PKI:CA certificate received.
00:44:01:CRYPTO_PKI:CA certificate received.
Router(config)# crypto ca enroll msca
%
% Start certificate enrollment ..
% Create a challenge password. You will need to verbally provide this
password to the CA Administrator in order to revoke your certificate.
For security reasons your password will not be saved in the configuration.
Please make a note of it.

Password:
Re-enter password:

% The subject name in the certificate will be:Router.cisco.com
% Include the router serial number in the subject name? [yes/no]:n
% Include an IP address in the subject name? [yes/no]:n
Request certificate from CA? [yes/no]:y
% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the fingerprint.

Router(config)#      Fingerprint: 2CFC6265 77BA6496 3AEFCB50 29BC2BF2

00:44:29:CRYPTO_PKI:transaction PKCSReq completed
00:44:29:CRYPTO_PKI:status:
00:44:29:CRYPTO_PKI:http connection opened
00:44:29:CRYPTO_PKI: received msg of 1924 bytes
00:44:29:CRYPTO_PKI:HTTP response header:
  HTTP/1.1 200 OK
Server:Microsoft-IIS/5.0
Date:Fri, 17 Nov 2000 18:51:28 GMT
Content-Length:1778
Content-Type:application/x-pki-message

00:44:29:CRYPTO_PKI:signed attr:pki-message-type:
00:44:29:13 01 33
00:44:29:CRYPTO_PKI:signed attr:pki-status:
00:44:29:13 01 30
00:44:29:CRYPTO_PKI:signed attr:pki-recipient-nonce:
00:44:29:04 10 B4 C8 2A 12 9C 8A 2A 4A E1 E5 15 DE 22 C2 B4 FD
00:44:29:CRYPTO_PKI:signed attr:pki-transaction-id:
00:44:29:13 20 34 45 45 41 44 42 36 33 38 43 33 42 42 45 44 45 39 46
00:44:29:34 38 44 33 45 36 39 33 45 33 43 37 45 39
00:44:29:CRYPTO_PKI:status = 100:certificate is granted
00:44:29:CRYPTO_PKI:All enrollment requests completed.
00:44:29:%CRYPTO-6-CERTRET:Certificate received from Certificate Authority
```

Related Commands

Command	Description
crypto ca authenticate	Authenticates the CA (by getting the certificate of the CA).
crypto ca enroll	Obtains the certificate of your router from the CA.
debug crypto pki messages	Displays debug messages for details of the interaction (message dump) between the CA and the router.
show crypto ca certificates	Displays information about your certificate, the certificate of the CA, and any RA certificates.

debug crypto sesmgmt

To show connection setup messages and their flow through the router, use the **debug crypto sesmgmt** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug crypto sesmgmt

no debug crypto sesmgmt

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Encryption and authentication are provided by a software service on the router called a *crypto engine*. The crypto engine performs authentication through DSS public and private keys when a connection is set up. DSS is a means of sending a “signature” at the end of a message that positively identifies the author of the message. The signature cannot be forged or duplicated by others, so whoever receives a message with a DSS signature knows exactly who sent the message.

When connections are not completing, use the **debug crypto sesmgmt** command to follow the progress of connection messages as a first step in diagnosing the problem. You see a record of each connection message as the router discovers it, and can track its progress through the necessary signing, verifying, and encryption session setup operations. Other significant connection setup events, such as the pregeneration of Diffie-Hellman public numbers, are also shown. For information on Diffie-Hellman public numbers, refer to the *Security Configuration Guide*.

Also use the **show crypto connections** command to display additional information on connections.

Examples

The following is sample output from the **debug crypto sesmgmt** command. The first shows messages from a router that initiates a successful connection. The second shows messages from a router that receives a connection.

```
Router# debug crypto sesmgmt

CRYPTO: Dequeued a message: Inititate_Connection
CRYPTO: DH gen phase 1 status for conn_id 2 slot 0:OK
CRYPTO: Signing done. Status:OK
CRYPTO: ICMP message sent: s=172.21.114.163, d=172.21.114.162
CRYPTO-SDU: send_nnc_req: NNC Echo Request sent
CRYPTO: Dequeued a message: CRM
CRYPTO: DH gen phase 2 status for conn_id 2 slot 0:OK
CRYPTO: Verify done. Status=OK
CRYPTO: Signing done. Status:OK
CRYPTO: ICMP message sent: s=172.21.114.163, d=172.21.114.162
CRYPTO-SDU: recv_nnc_rpy: NNC Echo Confirm sent
CRYPTO: Create encryption key for conn_id 2 slot 0:OK
CRYPTO: Replacing -2 in crypto maps with 2 (slot 0)
```

```
Router# debug crypto sesmgmt

CRYPTO: Dequeued a message: CIM
CRYPTO: Verify done. Status=OK
CRYPTO: DH gen phase 1 status for conn_id 1 slot 0:OK
CRYPTO: DH gen phase 2 status for conn_id 1 slot 0:OK
CRYPTO: Signing done. Status:OK
CRYPTO: ICMP message sent: s=172.21.114.162, d=172.21.114.163
```

■ debug crypto sesgmt

```
CRYPTO-SDU: act_on_nnc_req: NNC Echo Reply sent
CRYPTO: Create encryption key for conn_id 1 slot 0:OK
CRYPTO: Replacing -2 in crypto maps with 1 (slot 0)
CRYPTO: Dequeued a message: CCM
CRYPTO: Verify done. Status=OK
```

Related Commands

Command	Description
debug crypto key-exchange	Displays DSS public key exchange messages.

debug csm tgrm

To view Call Switching Module (CSM) trunk group resource manager information, use the **debug csm tgrm** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug csm tgrm

no debug csm tgrm

Syntax Description This command has no arguments or keywords.

Defaults Disabled

Command Modes Privileged EXEC

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

Usage Guidelines Disable console logging and use buffered logging before using the **debug csm tgrm** command. Using the **debug csm tgrm** command generates a large volume of debugs, which can affect router performance.

Examples A sample output of the **debug csm tgrm** command is shown below.

The output shows that the call type is voice, the direction is incoming, and the call is accepted by the CSM.

```
Router#
00:02:25:CSM-TGRM:csm_rx_cas_event_from_neat(EVENT_DIAL_IN) - c(T1 7/1:1:3)
call_type=VOICE, dir=INCOMING
Router#
00:02:30:CSM-TGRM:csm_proc_ic3_wait_for_res_resp() c(T1 7/1:1:3) VOICE <ACCEPTED !!>
```

[Table 47](#) provides an alphabetical listing of the **debug csm tgrm** command fields and a description of each field.

Table 47 *debug csm tgrm* Field Descriptions

Field	Description
call_type	Type of call: VOICE or MODEM.
dir	Direction of the call: INCOMING or OUTGOING.

debug csm voice

To turn on debugging for all CSM VoIP calls, use the **debug csm voice** command in privileged EXEC mode. Use the **no** form of this command to disable debugging output.

```
debug csm voice [slot | dspm | dsp | dsp-channel]
```

```
no debug csm voice [slot | dspm | dsp | dsp-channel]
```

Syntax Description

slot | *dspm* | *dsp* | *dsp-channel* (Optional) Identifies the location of a particular DSP channel.

Usage Guidelines

The **debug csm voice** command turns on debugging for all CSM Voice-over-IP calls. If this command has no keyword specified, then debugging is enabled for all voice calls. The **no debug csm voice** command turns off debugging information for all voice calls.

If the keyword *slot* | *dspm* | *dsp* | *dsp-channel* argument is specified, then (if the specified DSP channel is engaged in a CSM call) CSM call-related debugging information will be turned on for this channel. The **no** form of this command turns off debugging for that particular channel.

Examples

The following examples show sample output from the **debug csm voice** command. The following shows that CSM has received an event from ISDN.

```
Oct 18 04:05:07.052: EVENT_FROM_ISDN::dchan_idb=0x60D7B6B8, call_id=0xCF, ces=0x1
bchan=0x0, event=0x1, cause=0x0
```

In this example, terms are explained as follows:

- *dchan_idb*—Indicates the address of the hardware IDB for the D channel
- *call_id*—Indicates the call ID assigned by ISDN
- *bchan*—Indicates the number of the B channel assigned for this call
- *cause*—Indicates the ISDN event cause

The following shows that CSM has allocated the CSM voice control block for the DSP device on slot 1 port 10 for this call.

```
Oct 18 04:05:07.052: VDEV_ALLOCATE: slot 1 and port 10 is allocated.
```

This AS5300 access server might not be actually used to handle this call. CSM must first allocate the CSM voice control block to initiate the state machine. After the voice control block has been allocated, CSM obtains from the DSP Resource Manager the actual DSP channel that will be used for the call. At that point, CSM will switch to the actual logical port number. The slot number refers to the physical slot on the AS5300 access server. The port number is the logical DSP number interpreted as listed in [Table 48](#).

Table 48 Logical DSP Numbers

Logical Port Number	Physical DSP Channel
Port 0	DSPRM 1, DSP 1, DSP channel 1
Port 1	DSPRM 1, DSP 1, DSP channel 2
Port 2	DSPRM 1, DSP 2, DSP channel 1
Port 3	DSPRM 1, DSP 2, DSP channel 2
Port 4	DSPRM 1, DSP 3, DSP channel 1
Port 5	DSPRM 1, DSP 3, DSP channel 2
Port 6	DSPRM 1, DSP 4, DSP channel 1
Port 7	DSPRM 1, DSP 4, DSP channel 2
Port 8	DSPRM 1, DSP 5, DSP channel 1
Port 9	DSPRM 1, DSP 5, DSP channel 2
Port 10	DSPRM 1, DSP 6, DSP channel 1
Port 11	DSPRM 1, DSP 6, DSP channel 2
Port 12	DSPRM 2, DSP 1, DSP channel 1
Port 13	DSPRM 2, DSP 1, DSP channel 2
Port 14	DSPRM 2, DSP 2, DSP channel 1
Port 15	DSPRM 2, DSP 2, DSP channel 2
Port 16	DSPRM 2, DSP 3, DSP channel 1
Port 17	DSPRM 2, DSP 3, DSP channel 2
Port 18	DSPRM 2, DSP 4, DSP channel 1
Port 19	DSPRM 2, DSP 4, DSP channel 2
Port 20	DSPRM 2, DSP 5, DSP channel 1
Port 21	DSPRM 2, DSP 5, DSP channel 2
Port 22	DSPRM 2, DSP 6, DSP channel 1
Port 23	DSPRM 2, DSP 6, DSP channel 2
Port 48	DSPRM 5, DSP 1, DSP channel 1
Port 49	DSPRM 5, DSP 1, DSP channel 2
Port 50	DSPRM 5, DSP 2, DSP channel 1
Port 51	DSPRM 5, DSP 2, DSP channel 2
Port 52	DSPRM 5, DSP 3, DSP channel 1
Port 53	DSPRM 5, DSP 3, DSP channel 2
Port 54	DSPRM 5, DSP 4, DSP channel 1
Port 55	DSPRM 5, DSP 4, DSP channel 2
Port 56	DSPRM 5, DSP 5, DSP channel 1
Port 57	DSPRM 5, DSP 5, DSP channel 2
Port 58	DSPRM 5, DSP 6, DSP channel 1
Port 59	DSPRM 5, DSP 6, DSP channel 2

The following shows that the function `csm_vtsp_init_tdm()` has been called with a voice control block of address `0x60B8562C`. This function will be called only when the call is treated as a voice call.

```
Oct 18 04:05:07.052: csm_vtsp_init_tdm (voice_vdev=0x60B8562C)
```

The following shows that CSM has obtained a DSP channel from the DSP Resource Manager:

```
Oct 18 04:05:07.052: csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm slot 1, dspm 2, dsp 5,
dsp_channel 1csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm stream 5, channel 9, bank 0,
bp_channel 10
```

The DSP channel has the following initialized TDM channel information:

- TDM slot 1, dspm 2, dsp 5, dsp_channel 1—Indicates the physical DSP channel that will be used for this call.
- TDM stream 5, channel 9, bank 0, bp_channel 10—Indicates the on-chip and backplane TDM channel assigned to this DSP channel. Stream 5, channel 9 gives the on-chip TDM channel mapped to the DSP; bank 0, bp_channel 10 means that the backplane stream 0 and backplane channel #1 are assigned to this DSP.

The following shows that CSM has received an incoming call event from ISDN:

```
Oct 18 04:05:07.052: EVENT_FROM_ISDN:(00CF): DEV_INCALL at slot 1 and port 20
```

Slot 1, port 20 means the logical DSP channel 20 (mapped to DSPRM 2, DSP 5, DSP channel 1).

The following shows that the `DEV_INCALL` message has been translated into a `CSM_EVENT_ISDN_CALL` message:

```
Oct 18 04:05:07.052: CSM_PROC_IDLE: CSM_EVENT_ISDN_CALL at slot 1, port 20
```

This message is passed to the CSM central state machine while it is in the `CSM_IDLE` state and is in the `CSM_PROC_IDLE` procedure. The logical DSP channel port 20 on slot 1 is used to handle this call.

The following shows that CSM has invoked the `vtsp_ic_notify()` function with a CSM voice call control block `0x60B8562C`.

```
Oct 18 04:05:07.052: vtsp_ic_notify : (voice_vdev= 0x60B8562C)
```

Inside this function, CSM will send a `SETUP INDICATION` message to the VTSP. This function will be invoked only if the call is a voice call.

The following shows that CSM has received a `SETUP INDICATION RESPONSE` message from the VTSP as an acknowledgement.

```
Oct 18 04:05:07.056: csm_vtsp_call_setup_resp (vdev_info=0x60B8562C, vtsp_cdb=0x60FCA114)
```

This means that the VTSP has received the `CALL SETUP INDICATION` message previously sent and has proceeded to process the call.

- `vdev_info`—Contains the address of the CSM voice data block.
- `vtsp_cdb`—Contains the address of the VTSP call control block.

The following shows that CSM has received a `CALL CONNECT` message from the VTSP:

```
Oct 18 04:05:07.596: csm_vtsp_call_connect (vtsp_cdb=0x60FCA114, voice_vdev=0x60B8562C)
```

This indicates that the VTSP has received a CONNECT message for the call leg initiated to the Internet side.

- vtsp_cdb—Contains the address of the VTSP call control block.
- voice_vdev—Contains the address of the CSM voice data block.

The following shows that while CSM is in the CSM_IC2_RING state, it receives a SETUP INDICATION RESPONSE from the VTSP. This message is translated into CSM_EVENT_MODEM_OFFHOOK and passed to the CSM central state machine.

```
Oct 18 04:05:07.596: CSM_PROC_IC2_RING: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 20
```

The following shows that CSM has received a CONNECT message from ISDN for the call using the logical DSP channel on slot 1 and port 20:

```
Oct 18 04:05:07.616: EVENT_FROM_ISDN:(00CF): DEV_CONNECTED at slot 1 and port 20
```

The following shows that CSM has translated the CONNECT event from ISDN into the CSM_EVENT_ISDN_CONNECTED message, which is then passed to the CSM central state machine:

```
Oct 18 04:05:07.616: CSM_PROC_IC4_WAIT_FOR_CARRIER: CSM_EVENT_ISDN_CONNECTED at slot 1, port 20
```

The following shows that CSM has received a CALL SETUP REQUEST from the VTSP:

```
May 16 12:22:27.580: csm_vtsp_call_setup_request (vtsp_cdb=0x60FCFA20, vtsp_sdb=0x60DFB608)
```

This represents a request to make an outgoing call to the PSTN.

- vtsp_cdb—Contains the address of the VTSP call control block.
- vtsp_sdb—Contains the address of the signalling data block for the signalling interface to be used to send the outgoing call.

The following shows that the physical DSP channel has been allocated for this outgoing call:

```
May 16 12:22:27.580: csm_vtsp_call_setup_request: tdm slot 1, dspm 5, dsp 4, dsp_channel 1
```

The following shows the on-chip and backplane TDM channel assigned to this DSP channel:

```
May 16 12:22:27.580: csm_vtsp_call_setup_request: tdm stream 5, channel 25, bank 0, bp_channel 27
```

In this sample output, tdm stream 5, channel 25, bank 0, bp_channel 27 indicates the on-chip and backplane TDM channel assigned to this DSP channel. Stream 5, channel 25 gives the on-chip TDM channel mapped to the DSP; bank 0, bp_channel 27 means that the backplane stream 0 and backplane channel 1 are assigned to this DSP.

The following shows the calling number and the called number for this call.

```
May 16 12:22:27.580: csm_vtsp_call_setup_request: calling number: 10001, called number: 30001
```

The following shows that the CALL SETUP REQUEST from the VTSP has been translated into the ' CSM_EVENT_MODEM_OFFHOOK message and is passed to the CSM central state machine:

```
May 16 12:22:27.580: CSM_PROC_IDLE: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 54
```

The logical DSP channel number for the DSP (slot 1, port 54) is now displayed, which maps to the physical DSP channel slot 1, dspm 5, dsp 4, dsp_channel 1.

The following shows that CSM has collected all the digits for dialing out:

```
May 16 12:22:27.580: CSM_PROC_OC3_COLLECT_ALL_DIGIT: CSM_EVENT_GET_ALL_DIGITS at slot 1, port 54
```

For PRI and for applications that do not require digit collection of outdialing digits (for example, voice calls), the intermediate digit collection states are omitted and the CSM state machine moves to this state directly, pretending that the digit collection has been done.

The following shows an information message:

```
May 16 12:22:27.580: CSM_PROC_OC3_COLLECT_ALL_DIGIT: called party num: (30001) at slot 1, port 54
```

The following shows that CSM attempts to find a free signalling D channel to direct the outgoing call:

```
May 16 12:22:27.580: csm_vtsp_check_dchan (voice_vdev=0x60B8562C)
May 16 12:22:27.580: csm_vtsp_check_dchan (vtsp requested dchan=0x60D7ACB0,
dchan_idb=0x60E8ACF0)
May 16 12:22:27.580: csm_vtsp_check_dchan (voice_vdev=0x60B8562C)
May 16 12:22:27.580: csm_vtsp_check_dchan (vtsp requested dchan=0x60D7ACB0,
dchan_idb=0x60D7ACB0)
```

In the case of voice calls, the free signalling D channel must match the voice interface specified inside the signalling data block (vtsp_sdb) passed from the VTSP.

The following shows that CSM has received an event from ISDN:

```
May 16 12:22:27.624: EVENT_FROM_ISDN::dchan_idb=0x60D7ACB0, call_id=0xA121, ces=0x1
bchan=0x1E, event=0x3, cause=0x0
```

In this sample output:

- dchan_idb—indicates the address of the hardware IDB for the D channel
- call_id—Indicates the call id assigned by ISDN
- bchan—Indicates the number of the B channel assigned for this call
- cause—Indicates the ISDN event cause

The following shows that CSM has received a CALL PROCEEDING message from ISDN.

```
May 16 12:22:27.624: EVENT_FROM_ISDN:(A121): DEV_CALL_PROC at slot 1 and port 54
```

The following shows that the CALL PROCEEDING event received from ISDN has been interpreted as a CSM_EVENT_ISDN_BCHAN_ASSIGNED message:

```
*May 16 12:22:27.624: CSM_PROC_OC4_DIALING: CSM_EVENT_ISDN_BCHAN_ASSIGNED at slot 1, port 54
```

ISDN has assigned a B channel for this outgoing call. This B channel must be on the same PRI span as the signalling D channel allocated previously.

The following shows that the `csm_vtsp_setup_for_oc` function is called:

```
May 16 12:22:27.624: csm_vtsp_setup_for_oc (voice_vdev=0x60B8562C)
```

This is invoked when an outgoing call initiated by the VTSP receives a response from the ISDN stack.

The following shows that ISDN has sent a `CONNECT` message to CSM indicating that the call leg to the PSTN side has been established:

```
May 16 12:22:28.084: EVENT_FROM_ISDN::dchan_idb=0x60D7ACB0, call_id=0xA121, ces=0x1
    bchan=0x1E, event=0x4, cause=0x0
```

```
May 16 12:22:28.084: EVENT_FROM_ISDN:(A121): DEV_CONNECTED at slot 1 and port 54
```

The following shows that while CSM is in the `OC5_WAIT_FOR_CARRIER` state, it has received the 'CONNECT' message from ISDN and has translated it into the `CSM_EVENT_ISDN_CONNECTED` message, which is passed to the CSM central state machine:

```
May 16 12:22:28.084: CSM_PROC_OC5_WAIT_FOR_CARRIER: CSM_EVENT_ISDN_CONNECTED at slot 1,
port 54
```

The following shows that the function `vtsp_confirm_oc()` has been called:

```
May 16 12:22:28.084: vtsp_confirm_oc : (voice_vdev= 0x60B8562C)
```

This is invoked after CSM received the `CONNECT` message from ISDN. CSM sends a confirmation of the `CONNECT` to the VTSP.

debug ctunnel

To display debug messages for the IP over a CLNS Tunnel feature, use the **debug ctunnel** command in privileged EXEC mode. To disable the debug messages, use the **no** form of this command.

debug ctunnel

no debug ctunnel

Syntax Description This command has no arguments or keywords.

Defaults No default behavior or values.

Command History	Release	Modification
	12.1(5)	This command was introduced.

Examples As packets are sent over the virtual interface, the following type of output will appear on the console when the **debug ctunnel** command is used:

```
4d21h: CTunnel1: IPCLNP encapsulated 49.0001.1111.1111.1111.00->49.0001.2222.2222.2222.00
(linktype=7, len=89)
```

debug custom-queue

To enable custom queueing output, use the **debug custom-queue EXEC** command. Use the **no** form of this command to disable custom queueing output.

debug custom-queue

no debug custom-queue

Syntax Description

This command has no arguments or keywords.

Examples

The following is an example of enabling custom queueing output:

```
Router# debug custom-queue

Custom output queueing debugging is on
```

The following is sample output from the **debug custom-queue** command:

```
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 2
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 2 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 2
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 2 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 2
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 2 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
```

Related Commands

Command	Description
debug priority	Enables priority queueing output.

debug dampening

To display debug trace information messages for interface dampening, use the **debug dampening** command in privileged EXEC mode. To disable debugging messages, use the **no** form of this command.

debug dampening [all | interface]

no debug dampening [all | interface]

Syntax Description	all	(Optional) Enables trace debugging for all dampening features.
	interface	(Optional) Enables trace debugging for IP event dampening.

Defaults No default behavior or values.

Command Modes Privileged EXEC

Command History	Release	Modification
		12.0(22)S
	12.2(13)T	This command was integrated into Cisco IOS Release 12.2(13)T.

Examples The following sample output is similar to the output that will be displayed when the **debug dampening** command is entered with the **interface** keyword. The sample output shows the following information:

- Ethernet interface 1/1 is configured with the IP Event Dampening feature. The half-life period is set to 30 seconds, the reuse threshold to 500, the suppress threshold to 1000, and the restart penalty to 90.
- The **shutdown** command and then the **no shutdown** command was entered on Ethernet interface 1/1. The interface was suppressed and then reused by the IP Event Dampening feature.

```
Router# debug dampening interface
```

```
00:07:17:IF-EvD(Ethernet1/1):CLNS Routing reports state transition from UP to DOWN
00:07:17:EvD(Ethernet1/1):charge penalty 1000, new accum. penalty 1000, flap count 1
00:07:17:EvD(Ethernet1/1):accum. penalty 1000, now suppressed with a reuse intervals of 30
00:07:17:IF-EvD(Ethernet1/1):update CLNS Routing state to DOWN, interface is suppressed
00:07:17:IF-EvD(Ethernet1/1):IP Routing reports state transition from DOWN to DOWN
00:07:17:IF-EvD(Ethernet1/1):update IP Routing state to DOWN, interface is suppressed
00:07:17:IF-EvD(Ethernet1/1):CLNS Routing reports state transition from DOWN to UP
00:07:17:IF-EvD(Ethernet1/1):CLNS Routing reports state transition from UP to DOWN
00:07:17:EvD(Ethernet1/1):accum. penalty decayed to 1000 after 0 second(s)
00:07:17:EvD(Ethernet1/1):charge penalty 1000, new accum. penalty 2000, flap count 2
00:07:17:EvD(Ethernet1/1):accum. penalty 2000, now suppressed with a reuse intervals of 60
00:07:17:IF-EvD(Ethernet1/1):CLNS Routing reports state transition from DOWN to UP
00:07:17:IF-EvD(Ethernet1/1):CLNS Routing reports state transition from UP to DOWN
00:07:17:EvD(Ethernet1/1):accum. penalty decayed to 2000 after 0 second(s)
00:07:17:EvD(Ethernet1/1):charge penalty 1000, new accum. penalty 3000, flap count 3
00:07:17:EvD(Ethernet1/1):accum. penalty 3000, now suppressed with a reuse intervals of 78
00:07:17:IF-EvD(Ethernet1/1):CLNS Routing reports state transition from DOWN to UP
```

```

00:07:17:IF-EvD(Ethernet1/1):IP Routing reports state transition from UP to UP
00:07:17:IF-EvD(Ethernet1/1):IP Routing reports state transition from UP to UP
00:07:17:IF-EvD(Ethernet1/1):IP Routing reports state transition from UP to UP
00:07:17:IF-EvD(Ethernet1/1):IP Routing reports state transition from UP to UP
00:07:17:IF-EvD(Ethernet1/1):IP Routing reports state transition from UP to UP
00:07:20:IF-EvD(Ethernet1/1):CLNS Routing reports state transition from UP to UP
00:07:20:IF-EvD(Ethernet1/1):IP Routing reports state transition from UP to UP
00:07:47:IF-EvD:unsuppress interfaces
00:08:36:IF-EvD:unsuppress interfaces
00:08:36:EvD(Ethernet1/1):accum. penalty decayed to 483 after 79 second(s)
00:08:36:EvD(Ethernet1/1):accum. penalty 483, now unsuppressed
00:08:36:IF-EvD(Ethernet1/1):update IP Routing state to UP, interface is not suppressed
00:08:36:IF-EvD(Ethernet1/1):update CLNS Routing state to UP, interface is not suppressed
00:08:36:IF-EvD(Ethernet1/1):CLNS Routing reports state transition from UP to UP

```

Table 49 describes the significant fields shown in the sample output of the **debug dampening** command.

Table 49 *debug dampening Field Descriptions*

Field	Description
... Routing reports state transition from UP to DOWN	Displays the status of the specified interface from the perspective of the specified protocol. Interface state changes are displayed. The interface is specified within parentheses. The protocol is specified at the beginning of the message.
charge penalty 1000, new accum. penalty 1000, flap count 1	Displays the penalty assigned to the flapping interface and amount of penalty that is added to the accumulated penalty. The interface flap count is also displayed.
accum. penalty 1000, now suppressed with a reuse intervals of 30	Displays the status of the interface, accumulated penalty, and configured reuse threshold.
update CLNS Routing state to DOWN, interface is suppressed	Displays the status of the specified interface. Interface state changes and suppression status are displayed.
accum. penalty decayed to 1000 after 0 second(s)	Displays the decay rate of the accumulated penalty.
unsuppress interfaces	Indicates that dampened interfaces have been unsuppressed.

debug dbconn all

To turn on all debug flags for Database Connection, use the **debug dbconn all** command in privileged EXEC mode. The Database Connection debug flags are **appc**, **config**, **drda**, **event**, and **tcp**. Use the **no** form of this command to disable all debugging output.

debug dbconn all

no debug dbconn all

Syntax Description This command has no arguments or keywords.

Defaults Debugging is not enabled for Database Connection.

Usage Guidelines The **debug dbconn all** command displays debug output for APPC, Database Connection configuration, DRDA, error messages, event traces, and TCP.

Examples See the sample output provided for the [debug dbconn appc](#), [debug dbconn config](#), [debug dbconn drda](#), [debug dbconn event](#), and [debug dbconn tcp](#) commands.

Related Commands	Command	Description
	debug dbconn appc	Displays APPC-related trace or error messages.
	debug dbconn config	Displays trace or error messages for Database Connection configuration and control blocks.
	debug dbconn drda	Displays error messages and stream traces for DRDA.
	debug dbconn event	Displays trace or error messages for Database Connection events.
	debug dbconn tcp	Displays error messages and traces for TCP.

debug dbconn appc

To display APPC-related trace or error messages, use the **debug dbconn appc** command in privileged EXEC mode. Use the **no** form of this command to disable debugging output.

debug dbconn appc

no debug dbconn appc

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

In a router with stable Database Connection, the `alias_cp_name` field in the trace message should not be blank. There should be no other APPC error message. You can use APPN debug commands with this debug command to track APPN-related errors.

Examples

The following is sample output from the **debug dbconn appc** command. In a normal situation, only the following message is displayed:

```
DBCONN-APPC: alias_cp_name is "ASH"
```

The following error messages are displayed if there is a network configuration error or other APPN-related problem:

```
DBCONN-APPC-612C2B28: APPC error: opcode 0x1, primary_rc 0x0003,
secondary_rc 0x00000004
DBCONN-APPC-612C2B28: Verb block =
DBCONN-APPC-612C2B28: 0001 0200 0003 0000 0000 0004 0020 100C
DBCONN-APPC-612C2B28: 610A 828B 0000 0000 0000 0000 0000 0000
DBCONN-APPC-612C2B28: 0000 0000 8014 0003 0000 0000 0000 0000
DBCONN-APPC-612C2B28: D3E4 F6F2 E2E3 C1D9 C4C2 F240 4040 4040
DBCONN-APPC-612C2B28: 4040 4040 4040 4040 4040 4040 4040 4040
DBCONN-APPC-612C2B28: 4040 4040 4040 4040 4040 4040 4040 4040
DBCONN-APPC-612C2B28: 4040 4040 4040 4040 4040 4040 4040 4040
DBCONN-APPC-612C2B28: 4040 4040 4040 4040 0200 0000 0000 0000
DBCONN-APPC-612C2B28: 0000 0000 D4C5 D9D9 C9C5 4040 4040 D7C5
DBCONN-APPC-612C2B28: E3C5 D940 4040 4040 0000 0000 0000 0000
DBCONN-APPC-612C2B28: 00E2 E3C1 D9E6 4BE3 D6D9 C3C8 4040 4040
DBCONN-APPC-612C2B28: 4040 0000 0000 0000 0000 0000
DBCONN-APPC-612C2B28: ALLOCATE verb block =
DBCONN-APPC-612C2B28: 0001 0200 0003 0000 0000 0004 0020 100C
DBCONN-APPC-612C2B28: 610A 828B 0000 0000 0000 0000 0000 0000
DBCONN-APPC-612C2B28: 0000 0000 8014 0003 0000 0000 0000 0000
DBCONN-APPC-612C2B28: D3E4 F6F2 E2E3 C1D9 C4C2 F240 4040 4040
DBCONN-APPC-612C2B28: 4040 4040 4040 4040 4040 4040 4040 4040
DBCONN-APPC-612C2B28: 4040 4040 4040 4040 4040 4040 4040 4040
DBCONN-APPC-612C2B28: 4040 4040 4040 4040 4040 4040 4040 4040
DBCONN-APPC-612C2B28: 4040 4040 4040 4040 0200 0000 0000 0000
```

You can use the **debug appn** command to obtain more information.

The following message is displayed if a database connection is manually cleared and an outstanding APPC verb is pending:

```
DBCONN-APPC-%612C2B28: Canceling pending APPC verb 0x1
```

Related Commands	Command	Description
	debug dbconn all	Turns on all debug flags for Database Connection.
	debug dbconn config	Displays trace or error messages for Database Connection configuration and control blocks.
	debug dbconn drda	Displays error messages and stream traces for DRDA.
	debug dbconn event	Displays trace or error messages for Database Connection events.
	debug dbconn tcp	Displays error messages and traces for TCP.

debug dbconn config

To display trace or error messages for Database Connection configuration and control blocks, use the **debug dbconn config** command in privileged EXEC mode. Use the **no** form of this command to disable debugging output.

debug dbconn config

no debug dbconn config

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Most of the messages for Database Connection and control blocks do not report any errors. If a connection is inactive and cannot be cleared, use this command with the **debug dbconn appc**, **debug dbconn tcp**, and **debug appn** commands to locate the problem. The `alias_cp_name` field must match the configured APPN cpname.

Examples

The following is sample output from the **debug dbconn config** command:

```
DBCONN-CONFIG: alias_cp_name is "ASH      "
DBCONN-CONFIG: connection 612BDAAC matching server on 198.147.235.5:0 with
rdbname=STELLA
DBCONN-CONFIG: APPN shutdown; clearing connection 1234abcd
DBCONN-CONFIG: created server 612C2720
DBCONN-CONFIG: server 612C2720 (listen 60F72E94) is active
DBCONN-CONFIG: server 612C2720 (listen 60F72E94) is active
DBCONN-CONFIG: new connection 612BDAAC
DBCONN-CONFIG: listen 60F72E94 accepts connection 612BDAAC
DBCONN-CONFIG: server 60F74614 takes connection 612BDAAC
DBCONN-CONFIG: listen 60F72E94 releases connection 612BDAAC
DBCONN-CONFIG: server 60F74614 releases connection 612BDAAC
DBCONN-CONFIG: deleting connection 612BDAAC
DBCONN-CONFIG: listen 60F72E94 abandons connection 612BDAAC
DBCONN-CONFIG: server 612C2720 abandons connection 612BDAAC
DBCONN-CONFIG: deleting server 612C2720
DBCONN-CONFIG: daemon 60381738 takes zombie connection 612BDAAC
DBCONN-CONFIG: daemon 60381738 releases zombie connection 612BDAAC
```

Related Commands

Command	Description
debug dbconn all	Turns on all debug flags for Database Connection.
debug dbconn appc	Displays APPC-related trace or error messages.
debug dbconn drda	Displays error messages and stream traces for DRDA.
debug dbconn event	Displays trace or error messages for Database Connection events.
debug dbconn tcp	Displays error messages and traces for TCP.

debug dbconn drda

To display error messages and stream traces for DRDA, use the **debug dbconn drda** command in privileged EXEC mode. Use the **no** form of this command to disable debugging output.

debug dbconn drda

no debug dbconn drda

Syntax Description This command has no arguments or keywords.

Defaults By default, debugging is not enabled for the dbconn subsystem.

Command History	Release	Modification
	11.3(2)T	This command was introduced.
	12.0(5)XN	This command was moved from the CDBC feature to the CTRC feature.

Examples The following example displays output from the **debug dbconn drda** command:

```
Router# debug dbconn drda

*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: DSS X'006CD0410001', length 108, in chain,
REQDSS, correlator 1
*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: OBJECT X'00661041', length 98, code point
X'1041'
*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: OBJECT X'0020115E' in COLLECTION X'1041',
length 28, code point X'115E'
*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: OBJECT X'000C116D' in COLLECTION X'1041',
length 8, code point X'116D'
*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: OBJECT X'0013115A' in COLLECTION X'1041',
length 15, code point X'115A' (skipping...)
```

Related Commands	Command	Description
	debug dbconn all	Displays all CTRC debugging information related to communications with DB2.
	debug dbconn appc	Displays APPC-related trace or error messages for communications with DB2.
	debug dbconn config	Displays trace or error messages for CTRC configuration and control blocks for DB2 communications.
	debug dbconn event	Displays trace or error messages for CTRC events related to DB2 communications.
	debug dbconn tcp	Displays error messages or traces for TCP/IP communications with DB2.
	debug snasw	Displays debugging information related to SNA Switching Services.

debug dbconn event

To display trace or error messages for CTRC events related to DB2 communications, use the **debug dbconn event** command in privileged EXEC mode. Use the **no** form of this command to disable debugging output.

debug dbconn event

no debug dbconn event

Syntax Description

This command has no arguments or keywords.

Defaults

By default, debugging is not enabled for the dbconn subsystem.

Command History

Release	Modification
11.3(2)T	This command was introduced.
12.0(5)XN	This command was moved from the CDBC feature to the CTRC feature.

Examples

The following examples display output from the **debug dbconn event** command in a variety of situations. A normal trace for the **debug dbconn event** displays as follows:

```
Router# debug dbconn event

DBCONN-EVENT: Dispatch to 60FD6C00, from 0, msg 60F754CC, msgid 6468 'dh',
buffer 0.
DBCONN-EVENT: [*] Post to 61134240(cn), from 60EC5470(tc), msg 611419E4,
msgid 0x6372 'cr', buffer 612BF68C.
DBCONN-EVENT: Flush events called for pto 61182742, pfrom 61239837.
DBCONN-EVENT: Event discarded: to 61182742 (cn), from 61239837 (ap), msg
61339273, msgid 0x6372 'cr' buffer 0.
DBCONN-EVENT: == Send to 1234abcd, from 22938acd, msg 72618394, msgid
0x6372 'cr', buffer 0.
```

If the following messages are displayed, contact Cisco technical support personnel:

```
DBCONN-TCPFSM-1234abcd: Cannot occur in state 2 on input 6363 ('cc')
DBCONN-APPCFSM-1234abcd: Cannot occur in state 3 on input 6363 ('cc')
```

Related Commands	Command	Description
	debug dbconn all	Displays all CTRC debugging information related to communications with DB2.
	debug dbconn appc	Displays APPC-related trace or error messages for communications with DB2.
	debug dbconn config	Displays trace or error messages for CTRC configuration and control blocks for DB2 communications.
	debug dbconn drda	Displays error messages or stream traces for DRDA communications with DB2.
	debug dbconn tcp	Displays error messages or traces for TCP/IP communications with DB2.
	debug snasw	Displays debugging information related to SNA Switching Services.
	show debugging	Displays the state of each debugging option.

debug dbconn tcp

To display error messages and traces for TCP, use the **debug dbconn tcp** command in privileged EXEC mode. Use the **no** form of this command to disable debugging output.

debug dbconn tcp

no debug dbconn tcp

Syntax Description This command has no arguments or keywords.

Defaults Debugging is not enabled for the dbconn subsystem.

Command History	Release	Modification
	11.3(2)T	This command was introduced.
	12.0(5)XN	This command was moved from the CDBC feature to the CTRC feature.

Examples The following example displays output from the **debug dbconn tcp** command:

```
Router# debug dbconn tcp

DBCONN-TCP-63528473: tcpdriver_passive_open returned NULL
DBCONN-TCP-63528473: (no memory) tcp_reset(63829482) returns 4
DBCONN-TCP: tcp_accept(74625348,&error) returns tcb 63829482, error 4
DBCONN-TCP: (no memory) tcp_reset(63829482) returns 4
DBCONN-TCP-63528473: (open) tcp_create returns 63829482, error = 4
DBCONN-TCP-63528473: tcb_connect(63829482,1.2.3.4,2010) returns 4
DBCONN-TCP-63528473: (open error) tcp_reset(63829482) returns 4
DBCONN-TCP-63528473: tcp_create returns 63829482, error = 4
DBCONN-TCP-63528473: tcb_bind(63829482,0.0.0.0,2001) returns 4
DBCONN-TCP-63528473: tcp_listen(63829482,,) returns 4
DBCONN-TCP-63528473: (errors) Calling tcp_close (63829482)
```

Related Commands	Command	Description
	debug dbconn all	Displays all CTRC debugging information related to communications with DB2.
	debug dbconn appc	Displays APPC-related trace or error messages for communications with DB2.
	debug dbconn config	Displays trace or error messages for CTRC configuration and control blocks for DB2 communications.
	debug dbconn drda	Displays error messages or stream traces for DRDA communications with DB2.

Command	Description
debug dbconn event	Displays trace or error messages for CTRC events related to DB2 communications.
debug ip tcp	Displays debugging information related to TCP/IP.
debug snasw	Displays debugging information related to SNA Switching Services.
show debugging	Displays the state of each debugging option.

debug decnet adj

To display debugging information on DECnet adjacencies, use the **debug decnet adj** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug decnet adj

no debug decnet adj

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the sample **debug decnet adj** command:

```
Router# debug decnet adj

DNET-ADJ: Level 1 hello from 1.3
DNET-ADJ: sending hellos
DNET-ADJ: Sending hellos to all routers on interface Ethernet0, blksize 1498
DNET-ADJ: Level 1 hello from 1.3
DNET-ADJ: 1.5 adjacency initializing
DNET-ADJ: sending triggered hellos
DNET-ADJ: Sending hellos to all routers on interface Ethernet0, blksize 1498
DNET-ADJ: Level 1 hello from 1.3
DNET-ADJ: 1.5 adjacency up
DNET-ADJ: Level 1 hello from 1.5
DNET-ADJ: 1.5 adjacency down, listener timeout
```

The following line indicates that the router is sending hello messages to all routers on this segment, which in this case is Ethernet 0:

```
DNET-ADJ: Sending hellos to all routers on interface Ethernet0, blksize 1498
```

The following line indicates that the router has heard a hello message from address 1.5 and is creating an adjacency entry in its table. The initial state of this adjacency will be *initializing*.

```
DNET-ADJ: 1.5 adjacency initializing
```

The following line indicates that the router is sending an unscheduled (triggered) hello message as a result of some event, such as new adjacency being heard:

```
DNET-ADJ: sending triggered hellos
```

The following line indicates that the adjacency with 1.5 is now up, or active:

```
DNET-ADJ: 1.5 adjacency up
```

The following line indicates that the adjacency with 1.5 has timed out, because no hello message has been heard from adjacency 1.5 in the time interval originally specified in the hello message from 1.5:

```
DNET-ADJ: 1.5 adjacency down, listener timeout
```

The following line indicates that the router is sending an unscheduled hello message, as a result of some event, such as the adjacency state changing:

```
DNET-ADJ: hello update triggered by state changed in dn_add_adjacency
```

debug decnet connects

To display debugging information of all connect packets that are filtered (permitted or denied) by DECnet access lists, use the **debug decnet connects** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug decnet connects

no debug decnet connects

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

When you use connect packet filtering, it may be helpful to use the **decnet access-group** configuration command to apply the following basic access list:

```
access-list 300 permit 0.0 63.1023 eq any
```

You can then log all connect packets sent on interfaces to which you applied this list, in order to determine those elements on which your connect packets must be filtered.



Note

Packet password and account information is not logged in the **debug decnet connects** message, nor is it displayed by the **show access** EXEC command. If you specify **password** or **account** information in your access list, they can be viewed by anyone with access to the configuration of the router.

Examples

The following is sample output from the **debug decnet connects** command:

```
Router# debug decnet connects

DNET-CON: list 300 item #2 matched src=19.403 dst=19.309 on Ethernet0: permitted
  srcname="RICK" srcuic=[0,017]
  dstobj=42 id="USER"
```

[Table 50](#) describes significant fields in the output.

Table 50 *debug decnet connects Field Descriptions*

Field	Description
DNET-CON:	Indicates that this is a debug decnet connects packet.
list 300 item #2 matched	Indicates that a packet matched the second item in access list 300.
src=19.403	Indicates the source DECnet address for the packet.
dst=19.309	Indicates the destination DECnet address for the packet.
on Ethernet0:	Indicates the router interface on which the access list filtering the packet was applied.
permitted	Indicates that the access list permitted the packet.
srcname = "RICK"	Indicates the originator user of the packet.
srcuic=[0,017]	Indicates the source UIC of the packet.

Table 50 debug decnet connects Field Descriptions (continued)

Field	Description
dstobj=42	Indicates that DECnet object 42 is the destination.
id="USER"	Indicates the access user.

debug decnet events

To display debugging information on DECnet events, use the **debug decnet events** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug decnet events

no debug decnet events

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug decnet events
```

```
DNET: Hello from area 50 rejected - exceeded 'max area' parameter (45)
```

```
DNET: Hello from area 50 rejected - exceeded 'max area' parameter (45)
```

The following line indicates that the router received a hello message from a router whose area was greater than the max-area parameter with which this router was configured:

```
DNET: Hello from area 50 rejected - exceeded'max area' parameter (45)
```

The following line indicates that the router received a hello message from a router whose node ID was greater than the max-node parameter with which this router was configured:

```
DNET: Hello from node 1002 rejected - exceeded'max node' parameter (1000)
```

debug decnet packet

To display debugging information on DECnet packet events, use the **debug decnet packet** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug decnet packet

no debug decnet packet

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug decnet packet
```

```
DNET-PKT: src 1.4 dst 1.5 sending to PHASEV
```

```
DNET-PKT: Packet fwded from 1.4 to 1.5, via 1.5, snpa 0000.3080.cf90, TokenRing0
```

The following line indicates that the router is sending a converted packet addressed to node 1.5 to Phase V:

```
DNET-PKT: src 1.4 dst 1.5 sending to PHASEV
```

The following line indicates that the router forwarded a packet from node 1.4 to node 1.5. The packet is being sent to the next hop of 1.5 whose subnetwork point of attachment (MAC address) on that interface is 0000.3080.cf90.

```
DNET-PKT: Packet fwded from 1.4 to 1.5, via 1.5, snpa 0000.3080.cf90, TokenRing0
```

debug decnet routing

To display all DECnet routing-related events occurring at the router, use the **debug decnet routing** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug decnet routing

no debug decnet routing

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug decnet routing** command:

```
Router# debug decnet routing

DNET-RT: Received level 1 routing from 1.3 on Ethernet0 at 1:16:34
DNET-RT: Sending routes
DNET-RT: Sending normal routing updates on Ethernet0
DNET-RT: Sending level 1 routing updates on interface Ethernet0
DNET-RT: Level1 routes from 1.5 on Ethernet0: entry for node 5 created
DNET-RT: route update triggered by after split route pointers in dn_rt_input
DNET-RT: Received level 1 routing from 1.5 on Ethernet 0 at 1:18:35
DNET-RT: Sending L1 triggered routes
DNET-RT: Sending L1 triggered routing updates on Ethernet0
DNET-RT: removing route to node 5
```

The following line indicates that the router has received a level 1 update on Ethernet interface 0:

```
DNET-RT: Received level 1 routing from 1.3 on Ethernet0 at 1:16:34
```

The following line indicates that the router is sending its scheduled updates on Ethernet interface 0:

```
DNET-RT: Sending normal routing updates on Ethernet0
```

The following line indicates that the route will send an unscheduled update on this interface as a result of some event. In this case, the unscheduled update is a result of a new entry created in the routing table of the interface.

```
DNET-RT: route update triggered by after split route pointers in dn_rt_input
```

The following line indicates that the router sent the unscheduled update on Ethernet 0:

```
DNET-RT: Sending L1 triggered routes
DNET-RT: Sending L1 triggered routing updates on Ethernet0
```

The following line indicates that the router removed the entry for node 5 because the adjacency with node 5 timed out, or the route to node 5 through a next-hop router was disconnected:

```
DNET-RT: removing route to node 5
```

debug dhcp

To display debugging information about the Dynamic Host Configuration Protocol (DHCP) client activities and to monitor the status of DHCP packets, use the **debug dhcp** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug dhcp [detail]

no debug dhcp [detail]

Syntax Description

detail (Optional) Displays additional debug information.

Usage Guidelines

You can also use the **debug dhcp** command to monitor the subnet allocation and releasing for on-demand address pools.

For debugging purposes, the **debug dhcp detail** command provides the most useful information such as the lease entry structure of the client and the state transitions of the lease entry. The debug output shows the scanned option values from received DHCP messages that are replies to a router request. The values of the op, htype, hlen, hops, server identifier option, xid, secs, flags, ciaddr, yiaddr, siaddr, and giaddr fields of the DHCP packet are shown in addition to the length of the options field.

Examples

The following examples show and explain some of the typical debug messages you might see when using the **debug dhcp detail** command.

The following example shows the debug output when a DHCP client sends out a DHCPDISCOVER broadcast message to find its local DHCP server:

```
Router# debug dhcp detail

00:07:16:DHCP:DHCP client process started:10
00:07:16:RAC:Starting DHCP discover on Ethernet2
00:07:16:DHCP:Try 1 to acquire address for Ethernet2
00:07:16:%SYS-5-CONFIG_I:Configured from console by console
00:07:19:DHCP:Shutting down from get_netinfo()
00:07:19:DHCP:Attempting to shutdown DHCP Client
00:07:21:DHCP:allocate request
00:07:21:DHCP:new entry. add to queue
00:07:21:DHCP:SDiscover attempt # 1 for entry:
```

The first seven lines of the following output show the current values stored in the lease entry structure for the client:

```
00:07:21:Temp IP addr:0.0.0.0 for peer on Interface:Ethernet2
00:07:21:Temp sub net mask:0.0.0.0
00:07:21: DHCP Lease server:0.0.0.0, state:1 Selecting
00:07:21: DHCP transaction id:582
00:07:21: Lease:0 secs, Renewal:0 secs, Rebind:0 secs
00:07:21: Next timer fires after:00:00:03
00:07:21: Retry count:1 Client-ID:cisco-0010.7b6e.afd8-Et2
00:07:21:DHCP:SDiscover:sending 308 byte length DHCP packet
00:07:21:DHCP:SDiscover 308 bytes
00:07:21: B'cast on Ethernet2 interface from 0.0.0.0
```

The following example shows the offered addresses and parameters sent to the DHCP client by the DHCP server via a DHCPOFFER message. The messages containing the field “Scan” indicate the options that were scanned from the received BOOTP packet and the corresponding values.

```
00:07:23:DHCP:Received a BOOTREP pkt
00:07:23:DHCP:Scan:Message type:DHCP Offer
00:07:23:DHCP:Scan:Server ID Option:10.1.1.1 = A010101
00:07:23:DHCP:Scan:Lease Time:180
00:07:23:DHCP:Scan:Renewal time:90
00:07:23:DHCP:Scan:Rebind time:157
00:07:23:DHCP:Scan:Subnet Address Option:255.255.255.0
```

The following debug output shows selected fields in the received BOOTP packet:

```
00:07:23:DHCP:rcvd pkt source:10.1.1.1, destination: 255.255.255.255
00:07:23: UDP sport:43, dport:44, length:308
00:07:23: DHCP op:2, htype:1, hlen:6, hops:0
00:07:23: DHCP server identifier:10.1.1.1
00:07:23: xid:582, secs:0, flags:8000
00:07:23: client:0.0.0.0, your:10.1.1.2
00:07:23: srvr: 0.0.0.0, gw:0.0.0.0
00:07:23: options block length:60

00:07:23:DHCP Offer Message Offered Address:10.1.1.2
00:07:23:DHCP:Lease Seconds:180 Renewal secs: 90 Rebind secs:157
00:07:23:DHCP:Server ID Option:10.1.1.1
00:07:23:DHCP:offer received from 10.1.1.1
```

The following example shows the debug output when the DHCP client sends out a DHCPREQUEST broadcast message to the DHCP server to accept the offered parameters:

```
00:07:23:DHCP:SRequest attempt # 1 for entry:
00:07:23:Temp IP addr:10.1.1.2 for peer on Interface:Ethernet2
00:07:23:Temp sub net mask:255.255.255.0
00:07:23: DHCP Lease server:10.1.1.1, state:2 Requesting
00:07:23: DHCP transaction id:582
00:07:23: Lease:180 secs, Renewal:0 secs, Rebind:0 secs
00:07:23: Next timer fires after:00:00:02
00:07:23: Retry count:1 Client-ID:cisco-0010.7b6e.afd8-Et2
00:07:23:DHCP:SRequest- Server ID option:10.1.1.1
00:07:23:DHCP:SRequest- Requested IP addr option:10.1.1.2
00:07:23:DHCP:SRequest placed lease len option:180
00:07:23:DHCP:SRequest:326 bytes
00:07:23:DHCP:SRequest:326 bytes
00:07:23: B'cast on Ethernet2 interface from 0.0.0.0
```

The following example shows the debug output when the DHCP server sends a DHCPACK message to the client with the full set of configuration parameters:

```
00:07:23:DHCP:Received a BOOTREP pkt
00:07:23:DHCP:Scan:Message type:DHCP Ack
00:07:23:DHCP:Scan:Server ID Option:10.1.1.1 = A010101
00:07:23:DHCP:Scan:Lease Time:180
00:07:23:DHCP:Scan:Renewal time:90
00:07:23:DHCP:Scan:Rebind time:157
00:07:23:DHCP:Scan:Subnet Address Option:255.255.255.0
00:07:23:DHCP:rcvd pkt source:10.1.1.1, destination: 255.255.255.255
00:07:23: UDP sport:43, dport:44, length:308
00:07:23: DHCP op:2, htype:1, hlen:6, hops:0
00:07:23: DHCP server identifier:10.1.1.1
00:07:23: xid:582, secs:0, flags:8000
00:07:23: client:0.0.0.0, your:10.1.1.2
00:07:23: srvr: 0.0.0.0, gw:0.0.0.0
00:07:23: options block length:60
```

```

00:07:23:DHCP Ack Messag
00:07:23:DHCP:Lease Seconds:180 Renewal secs: 90 Rebind secs:157
00:07:23:DHCP:Server ID Option:10.1.1.1Interface Ethernet2 assigned DHCP address 10.1.1.2,
mask 255.255.255.0

00:07:26:DHCP Client Pooling:***Allocated IP address:10.1.1.2
00:07:26:Allocated IP address = 10.1.1.2 255.255.255.0

```

Most fields are self-explanatory; however, fields that may need further explanation are described in [Table 51](#).

Table 51 *debug dhcp Command Field Descriptions*

Fields	Description
DHCP:Scan:Subnet Address Option:255.255.255.0	Subnet mask option (option 1).
DHCP server identifier:1.1.1.1	Value of the DHCP server id option (option 54). Note that this is not the same as the siaddr field, which is the server IP address.
srvr:0.0.0.0, gw:0.0.0.0	srvr is the value of the siaddr field. gw is the value of the giaddr field.

Related Commands

Command	Description
debug ip dhcp server	Enables DHCP server debugging.
show dhcp lease	Displays DHCP addresses leased from a server.

debug dialer events

To display debugging information about the packets received on a dialer interface, use the **debug dialer events** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug dialer events

no debug dialer events

Syntax Description

This command has no arguments or keywords.

Examples

When DDR is enabled on the interface, information concerning the cause of any call (called the *Dialing cause*) is displayed. The following line of output for an IP packet lists the name of the DDR interface and the source and destination addresses of the packet:

```
Dialing cause: Serial0: ip (s=172.16.1.111 d=172.16.2.22)
```

The following line of output for a bridged packet lists the DDR interface and the type of packet (in hexadecimal). For information on these packet types, see the “Ethernet Type Codes” appendix of the *Cisco IOS Bridging and IBM Networking Command Reference* publication.

```
Dialing cause: Serial1: Bridge (0x6005)
```

Most messages are self-explanatory; however, messages that may need some explanation are described in [Table 52](#).

Table 52 General debug dialer events Message Descriptions

Message	Description
Dialer0: Already <i>xxx</i> call(s) in progress on Dialer0, dialing not allowed	Number of calls in progress (<i>xxx</i>) exceeds the maximum number of calls set on the interface.
Dialer0: No free dialer - starting fast idle timer	All the lines in the interface or rotary group are busy, and a packet is waiting to be sent to the destination.
BRI0: rotary group to <i>xxx</i> overloaded (<i>yyy</i>)	Number dialer (<i>xxx</i>) exceeds the load set on the interface (<i>yyy</i>).
BRI0: authenticated host <i>xxx</i> with no matching dialer profile	No dialer profile matches <i>xxx</i> , the CHAP name or remote name of the remote host.
BRI0: authenticated host <i>xxx</i> with no matching dialer map	No dialer map matches <i>xxx</i> , the CHAP name or remote name of the remote host.
BRI0: Can't place call, verify configuration	Dialer string or dialer pool on an interface not set.

Table 53 describes the messages that the **debug dialer events** command can generate for a serial interface used as a V.25bis dialer for DDR.

Table 53 *debug dialer events Command Message Descriptions for DDR*

Message	Description
Serial 0: Dialer result = xxxxxxxxxx	Result returned from the V.25bis dialer. It is useful in debugging if calls are failing. On some hardware platforms, this message cannot be displayed due to hardware limitations. Possible values for the xxxxxxxxxx variable depend on the V.25bis device with which the router is communicating.
Serial 0: No dialer string defined. Dialing cannot occur.	Packet is received that should cause a call to be placed. However, no dialer string is configured, so dialing cannot occur. This message usually indicates a configuration problem.
Serial 0: Attempting to dial xxxxxxxxxx	Packet has been received that passes the dial-on-demand access lists. That packet causes phone number xxxxxxxxxx to be dialed.
Serial 0: Unable to dial xxxxxxxxxx	Phone call to xxxxxxxxxx cannot be placed. This failure might be due to a lack of memory, full output queues, or other problems.
Serial 0: disconnecting call	Router hangs up a call.
Serial 0: idle timeout Serial 0: re-enable timeout Serial 0: wait for carrier timeout	One of these three messages is displayed when a dialer timer expires. These messages are mostly informational, but are useful for debugging a disconnected call or call failure.

Related Commands

Command	Description
debug dialer packets	Displays debugging information about the packets received on a dialer interface.

debug dialpeer

To view dial peer information, use the **debug dialpeer** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug dialpeer

no debug dialpeer

Syntax Description This command has no arguments or keywords.

Defaults Disabled

Command Modes Privileged EXEC

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

Usage Guidelines Disable console logging and use buffered logging before using the **debug dialpeer** command. Using the **debug dialpeer** command generates a large volume of debugs, which can affect router performance.

Examples A sample output of the **debug dialpeer** command is shown below.

The output shows the destination pattern configured on the matched dial-peer. Expanded string is the string after applying number translation to the original number. It shows that dial-peer 1311 was an incoming dial-peer match. It also shows that routing label was att1. It shows that dial-peer 5108888 and 111399 are an outgoing dial-peer match.

```
Router#
00:22:28: Inside dpMatchCore:
00:22:28: destination pattn:5108880101 expanded string:5108880101
00:22:28:MatchNextPeer:Peer 1311 matched
00:22:28: Inside dpMatchCore:
00:22:28: destination pattn:5108880101 expanded string:5108880101
00:22:28: Inside dpMatchCore:
00:22:28: destination pattn:4088880101 expanded string:4088880101
00:22:28: Inside dpMatchCore:
00:22:28: destination pattn:4088880101 expanded string:4088880101
00:22:28: dpAssociateIncomingPeer_T:Matching route label att1
00:22:28: Inside dpMatchCore:
00:22:28: destination pattn:5108880101 expanded string:5108880101
00:22:28: dpAssociateIncomingPeer_T:Matching peer with src route label att1 failed
00:22:28: Inside dpMatchCore:
00:22:28: destination pattn:5108880101 expanded string:5108880101
00:22:28:MatchNextPeer:Peer 1311 matched
00:22:28: Inside dpMatchPeersMoreArg
00:22:28:dpMatchPeersMoreArg:Match Dest. pattern; called (5108880101)
00:22:28: Inside dpMatchCore:
```

```

00:22:28: destination pa
Router#ttn:5108880101 expanded string:5108880101
00:22:28:MatchNextPeer:Peer 5108888 matched
00:22:28:MatchNextPeer:Peer 111399 matched
00:22:28:dpMatchPeersMoreArg:Result=0 after MATCH_ORIGINATE

```

Table 54 provides an alphabetical listing of the **debug dialpeer** command fields and a description of each field.

Table 54 *debug dialpeer Field Descriptions*

Field	Description
destination pattn	Destination pattern configured on the dial peer.
expanded string	The string after applying number translation to the original number.
Match Dest. pattern; called	Indicates that dial-peer match is going to match destination pattern against the called number.
Matching route label	The trunk group label or carrier id that is used for matching a dial peer.
MatchNextPeer	Indicates the dial peer tag that matched.
Result	Indicates the result of dial peer matching algorithm: <ul style="list-style-type: none"> 0 = Successful 1 = More digits needed for a possible match -1 = No match (match failed) -2 = The digits matched, but the destination address could not be obtained

Related Commands

Command	Description
call-block (dial peer)	Enables blocking of incoming calls on the dial peer.
carrier-id (dial-peer)	Identifies the carrier handling the incoming call.
session target (ENUM)	Specifies the ENUM search table for the target session.
show dial-peer voice	Displays the configuration of the dial peer.
translation-profile (dial-peer)	Assigns a translation profile to the dial peer.
trunkgroup (dial-peer)	Assigns a trunk group to the dial peer.
trunk-group-label (dial-peer)	Identifies the trunk group handling the incoming call.

debug dlsw

To enable debugging of DLSw+, use the **debug dlsw** command in privileged EXEC mode. The **no** form of this command disables debugging output.

```
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]
```

```
no debug dlsw [border-peers [interface interface | ip address ip-address] | core [flow-control
messages | state | xid] [circuit-number] | local-circuit circuit-number | peers
[interface interface [fast-errors | fast-paks] | ip address ip-address [fast-errors | fast-paks |
fst-seq | udp]] | reachability [error | verbose] [sna | netbios]
```

Syntax Description

border-peers	(Optional) Enables debugging output for border peer events.
interface <i>interface</i>	(Optional) Specifies a remote peer to debug by a direct interface.
ip address <i>ip-address</i>	(Optional) Specifies a remote peer to debug by its IP address.
core	(Optional) Enables debugging output for DLSw core events.
flow-control	(Optional) Enables debugging output for congestion in the WAN or at the remote end station.
messages	(Optional) Enables debugging output of core messages—specific packets received by DLSw either from one of its peers or from a local medium via the Cisco link services interface.
state	(Optional) Enables debugging output for state changes on the circuit.
xid	(Optional) Enables debugging output for the exchange identification state machine.
<i>circuit-number</i>	(Optional) Specifies the circuit for which you want core debugging output to reduce the of output.
local-circuit <i>circuit-number</i>	(Optional) Enables debugging output for circuits performing local conversion. Local conversion occurs when both the input and output data-link connections are on the same local peer and no remote peer exists.
peers	(Optional) Enables debugging output for peer events.
fast-errors	(Optional) Debugs errors for fast-switched packets.
fast-paks	(Optional) Debugs fast-switched packets.
fst-seq	(Optional) Debugs FST sequence numbers on fast switched packets.
udp	(Optional) Debugs UDP packets.
reachability	(Optional) Enables debugging output for reachability events (explorer traffic). If no options are specified, event-level information is displayed for all protocols.

error verbose	(Optional) Specifies how much reachability information you want displayed. The verbose keyword displays everything, including errors and events. The error keyword displays error information only. If no option is specified, event-level information is displayed.
sna netbios	(Optional) Specifies that reachability information be displayed for only SNA or NetBIOS protocols. If no option is specified, information for all protocols is displayed.

Usage Guidelines

When you specify no optional keywords, the **debug dlsw** command enables all available DLSw debugging output.

Normally you need to use only the **error** or **verbose** option of the **debug dlsw reachability** command to help identify problems. The **error** option is recommended for use by customers and provides a subset of the messages from the normal event-level debugging. The **verbose** option provides a very detailed view of events, and is typically used only by service personnel.

To reduce the amount of debug information displayed, use the **sna** or **netbios** option with the **debug dlsw reachability** command if you know that you have an SNA or NetBIOS problem.

The DLSw core is the engine that is responsible for the establishment and maintenance of remote circuits. If possible, specifying the index of the specific circuit you want to debug reduces the amount of output displayed. However, if you want to watch a circuit initially come up, do not use the *circuit-number* option with the **core** keyword.

The **core flow-control** option provides information about congestion in the WAN or at the remote end station. In these cases, DLSw sends Receiver Not Ready (RNR) frames on its local circuits, slowing data traffic on established sessions and giving the congestion an opportunity to clear.

The **core state** option allows you to see when the circuit changes state. This capability is especially useful for determining why a session cannot be established or why a session is being disconnected.

The **core XID** option allows you to track the XID-state machine. The router tracks XID commands and responses used in negotiations between end stations before establishing a session.

Examples

The following examples show and explain some of the typical DLSw debug messages you might see when using the **debug dlsw** command.

The following example enables UDP packet debugging for a specific remote peer:

```
Router# debug dlsw peers ip-address 1.1.1.6 udp
```

The following message is sample output from the **debug dlsw border-peers** command:

```
*Mar 10 17:39:56: CSM: delete group mac cache for group 0
*Mar 10 17:39:56: CSM: delete group name cache for group 0
*Mar 10 17:40:19: CSM: update group cache for mac 0000.3072.1070, group 10
*Mar 10 17:40:22: DLSw: send_to_group_members(): copy to peer 10.19.32.5
```

The following message is from a router that initiated a TCP connection:

```
DLSw: START-TPFSM (peer 10.3.8.7(2065)): event:ADMIN-OPEN CONNECTION state:DISCONN
DLSw: dtp_action_a() attempting to connect peer 10.3.8.7(2065)
DLSw: END-TPFSM (peer 10.3.8.7(2065)): state:DISCONN->WAIT_WR
DLSw: Async Open Callback 10.3.8.7(2065) -> 11002
DLSw: START-TPFSM (peer 10.3.8.7(2065)): event:TCP-WR PIPE OPENED state:WAIT_WR
DLSw: dtp_action_f() start read open timer for peer 10.3.8.7(2065)
DLSw: END-TPFSM (peer 10.3.8.7(2065)): state:WAIT_WR->WAIT_RD
DLSw: passive open 10.3.8.7(11004) -> 2065
```

```

DLSw: START-TPFSM (peer 10.3.8.7(2065)): event:TCP-RD PIPE OPENED state:WAIT_RD
DLSw: dtp_action_g() read pipe opened for peer 10.3.8.7(2065)
DLSw: CapExId Msg sent to peer 10.3.8.7(2065)
DLSw: END-TPFSM (peer 10.3.8.7(2065)): state:WAIT_RD->WAIT_CAP
DLSw: START-TPFSM (peer 10.3.8.7(2065)): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dtp_action_j() cap msg rcvd from peer 10.3.8.7(2065)
DLSw: Recv CapExId Msg from peer 10.3.8.7(2065)
DLSw: Pos CapExResp sent to peer 10.3.8.7(2065)
DLSw: END-TPFSM (peer 10.3.8.7(2065)): state:WAIT_CAP->WAIT_CAP
DLSw: START-TPFSM (peer 10.3.8.7(2065)): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dtp_action_j() cap msg rcvd from peer 10.3.8.7(2065)
DLSw: Recv CapExPosRsp Msg from peer 10.3.8.7(2065)
DLSw: END-TPFSM (peer 10.3.8.7(2065)): state:WAIT_CAP->WAIT_CAP
DLSw: Processing delayed event:SSP-CAP EXCHANGED - prev state:WAIT_CAP
DLSw: START-TPFSM (peer 10.3.8.7(2065)): event:SSP-CAP EXCHANGED state:WAIT_CAP
DLSw: dtp_action_k() cap xchged for peer 10.3.8.7(2065)
DLSw: closing read pipe tcp connection for peer 10.3.8.7(2065)
DLSw: END-TPFSM (peer 10.3.8.7(2065)): state:WAIT_CAP->PCONN_WT
DLSw: Processing delayed event:TCP-PEER CONNECTED - prev state:PCONN_WT
DLSw: START-TPFSM (peer 10.3.8.7(2065)): event:TCP-PEER CONNECTED state:PCONN_WT
DLSw: dtp_action_m() peer connected for peer 10.3.8.7(2065)
DLSw: END-TPFSM (peer 10.3.8.7(2065)): state:PCONN_WT->CONNECT
DLSw: START-TPFSM (peer 10.3.8.7(2065)): event:CORE-ADD CIRCUIT state:CONNECT
DLSw: dtp_action_u(), peer add circuit for peer 10.3.8.7(2065)
DLSw: END-TPFSM (peer 10.3.8.7(2065)): state:CONNECT->CONNECT

```

The following message is from a router that received a TCP connection:

```

DLSw: passive open 10.10.10.4(11002) -> 2065
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:TCP-RD PIPE OPENED state:DISCONN
DLSw: dtp_action_c() opening write pipe for peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:DISCONN->WWR_RDOP
DLSw: Async Open Callback 10.10.10.4(2065) -> 11004
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:TCP-WR PIPE OPENED state:WWR_RDOP
DLSw: dtp_action_i() write pipe opened for peer 10.10.10.4(2065)
DLSw: CapExId Msg sent to peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:WWR_RDOP->WAIT_CAP
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dtp_action_j() cap msg rcvd from peer 10.10.10.4(2065)
DLSw: Recv CapExId Msg from peer 10.10.10.4(2065)
DLSw: Pos CapExResp sent to peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:WAIT_CAP->WAIT_CAP
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dtp_action_j() cap msg rcvd from peer 10.10.10.4(2065)
DLSw: Recv CapExPosRsp Msg from peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:WAIT_CAP->WAIT_CAP
DLSw: Processing delayed event:SSP-CAP EXCHANGED - prev state:WAIT_CAP
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:SSP-CAP EXCHANGED state:WAIT_CAP
DLSw: dtp_action_k() cap xchged for peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:WAIT_CAP->PCONN_WT
DLSw: dlsw_tcpd_fini() for peer 10.10.10.4(2065)
DLSw: dlsw_tcpd_fini() closing write pipe for peer 10.10.10.4
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:TCP-CLOSE WR PIPE state:PCONN_WT
DLSw: dtp_action_l() close write pipe for peer 10.10.10.4(2065)
DLSw: closing write pipe tcp connection for peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:PCONN_WT->PCONN_WT
DLSw: Processing delayed event:TCP-PEER CONNECTED - prev state:PCONN_WT
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:TCP-PEER CONNECTED state:PCONN_WT
DLSw: dtp_action_m() peer connected for peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:PCONN_WT->CONNECT
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:CORE-ADD CIRCUIT state:CONNECT
DLSw: dtp_action_u(), peer add circuit for peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:CONNECT->CONNECT

```

The following message is from a router that initiated an FST connection:

```

DLsw: START-FSTPFMSM (peer 10.10.10.4(0)): event:ADMIN-OPEN CONNECTION state:DISCONN
DLsw: dfstp_action_a() attempting to connect peer 10.10.10.4(0)
DLsw: Connection opened for peer 10.10.10.4(0)
DLsw: CapExId Msg sent to peer 10.10.10.4(0)
DLsw: END-FSTPFMSM (peer 10.10.10.4(0)): state:DISCONN->WAIT_CAP
DLsw: START-FSTPFMSM (peer 10.10.10.4(0)): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLsw: dfstp_action_e() cap msg rcvd for peer 10.10.10.4(0)
DLsw: Recv CapExPosRsp Msg from peer 10.10.10.4(0)
DLsw: END-FSTPFMSM (peer 10.10.10.4(0)): state:WAIT_CAP->WAIT_CAP
DLsw: START-FSTPFMSM (peer 10.10.10.4(0)): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLsw: dfstp_action_e() cap msg rcvd for peer 10.10.10.4(0)
DLsw: Recv CapExId Msg from peer 10.10.10.4(0)
DLsw: Pos CapExResp sent to peer 10.10.10.4(0)
DLsw: END-FSTPFMSM (peer 10.10.10.4(0)): state:WAIT_CAP->WAIT_CAP
DLsw: Processing delayed event:SSP-CAP EXCHANGED - prev state:WAIT_CAP
DLsw: START-FSTPFMSM (peer 10.10.10.4(0)): event:SSP-CAP EXCHANGED state:WAIT_CAP
DLsw: dfstp_action_f() cap xchged for peer 10.10.10.4(0)
DLsw: END-FSTPFMSM (peer 10.10.10.4(0)): state:WAIT_CAP->CONNECT

```

The following message is from a router that received an FST connection:

```

DLsw: START-FSTPFMSM (peer 10.3.8.7(0)): event:SSP-CAP MSG RCVD state:DISCONN
DLsw: dfstp_action_c() cap msg rcvd for peer 10.3.8.7(0)
DLsw: Recv CapExId Msg from peer 10.3.8.7(0)
DLsw: Pos CapExResp sent to peer 10.3.8.7(0)
DLsw: CapExId Msg sent to peer 10.3.8.7(0)
DLsw: END-FSTPFMSM (peer 10.3.8.7(0)): state:DISCONN->WAIT_CAP
DLsw: START-FSTPFMSM (peer 10.3.8.7(0)): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLsw: dfstp_action_e() cap msg rcvd for peer 10.3.8.7(0)
DLsw: Recv CapExPosRsp Msg from peer 10.3.8.7(0)
DLsw: END-FSTPFMSM (peer 10.3.8.7(0)): state:WAIT_CAP->WAIT_CAP
DLsw: Processing delayed event:SSP-CAP EXCHANGED - prev state:WAIT_CAP
DLsw: START-FSTPFMSM (peer 10.3.8.7(0)): event:SSP-CAP EXCHANGED state:WAIT_CAP
DLsw: dfstp_action_f() cap xchged for peer 10.3.8.7(0)
DLsw: END-FSTPFMSM (peer 10.3.8.7(0)): state:WAIT_CAP->CONNECT

```

The following message is from a router that initiated an LLC2 connection:

```

DLsw-LLC2: Sending enable port ; port no : 0
          PEER-DISP Sent : CLSI Msg : ENABLE.Reg  dlen: 20
DLsw: Peer Received : CLSI Msg : ENABLE.Cfm CLS_OK dlen: 20
DLsw-LLC2 : Sending activate sap for Serial1 - port_id = 887C3C
          port_type = 7 dgra(UsapID) = 952458
          PEER-DISP Sent : CLSI Msg : ACTIVATE_SAP.Reg  dlen: 60
DLsw: Peer Received : CLSI Msg : ACTIVATE_SAP.Cfm CLS_OK dlen: 60
DLsw Got ActSapcnf back for Serial1 - port_id = 8978204, port_type = 7, psap_id = 0

DLsw: START-LLC2PFMSM (peer on interface Serial1): event:ADMIN-OPEN CONNECTION
state:DISCONN
DLsw: dllc2p_action_a() attempting to connect peer on interface Serial1
          PEER-DISP Sent : CLSI Msg : REQ_OPNSTN.Reg  dlen: 106
DLsw: END-LLC2PFMSM (peer on interface Serial1): state:DISCONN->ROS_SENT

DLsw: Peer Received : CLSI Msg : REQ_OPNSTN.Cfm CLS_OK dlen: 106
DLsw: START-LLC2PFMSM (peer on interface Serial1): event:CLS-REQOPNSTN.CNF state:ROS_SENT
DLsw: dllc2p_action_c()
          PEER-DISP Sent : CLSI Msg : CONNECT.Reg  dlen: 16
DLsw: END-LLC2PFMSM (peer on interface Serial1): state:ROS_SENT->CON_PEND

DLsw: Peer Received : CLSI Msg : CONNECT.Cfm CLS_OK dlen: 28
DLsw: START-LLC2PFMSM (peer on interface Serial1): event:CLS-CONNECT.CNF state:CON_PEND
DLsw: dllc2p_action_e() send capabilities to peer on interface Serial1
          PEER-DISP Sent : CLSI Msg : SIGNAL_STN.Reg  dlen: 8

```

```

PEER-DISP Sent : CLSI Msg : DATA.Req  dlen: 418
DLSw: CapExId Msg sent to peer on interface Serial1
DLSw: END-LLC2PFISM (peer on interface Serial1): state:CON_PEND->WAIT_CAP

DLSw: Peer Received : CLSI Msg : DATA.Ind  dlen: 418
DLSw: START-LLC2PFISM (peer on interface Serial1): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dllc2p_action_k() cap msg rcvd for peer on interface Serial1
DLSw: Recv CapExId Msg from peer on interface Serial1
    PEER-DISP Sent : CLSI Msg : DATA.Req  dlen: 96
DLSw: Pos CapExResp sent to peer on interface Serial1
DLSw: END-LLC2PFISM (peer on interface Serial1): state:WAIT_CAP->WAIT_CAP

DLSw: Peer Received : CLSI Msg : DATA.Ind  dlen: 96
DLSw: START-LLC2PFISM (peer on interface Serial1): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dllc2p_action_k() cap msg rcvd for peer on interface Serial1
DLSw: Recv CapExPosRsp Msg from peer on interface Serial1
DLSw: END-LLC2PFISM (peer on interface Serial1): state:WAIT_CAP->WAIT_CAP

DLSw: Processing delayed event:SSP-CAP EXCHANGED - prev state:WAIT_CAP
DLSw: START-LLC2PFISM (peer on interface Serial1): event:SSP-CAP EXCHANGED state:WAIT_CAP
DLSw: dllc2p_action_l() cap xchgd for peer on interface Serial1
DLSw: END-LLC2PFISM (peer on interface Serial1): state:WAIT_CAP->CONNECT

```

The following message is from a router that received an LLC2 connection:

```

DLSw-LLC2: Sending enable port ; port no : 0
    PEER-DISP Sent : CLSI Msg : ENABLE.Req  dlen: 20
DLSw: Peer Received : CLSI Msg : ENABLE.Cfm CLS_OK dlen: 20
DLSw-LLC2 : Sending activate sap for Serial0 - port_id = 887C3C
    port_type = 7 dgra(UsapID) = 93AB34
    PEER-DISP Sent : CLSI Msg : ACTIVATE_SAP.Req  dlen: 60
DLSw: Peer Received : CLSI Msg : ACTIVATE_SAP.Cfm CLS_OK dlen: 60
DLSw Got ActSapcnf back for Serial0 - port_id = 8944700, port_type = 7, psap_id = 0

DLSw: Peer Received : CLSI Msg : CONECT_STN.Ind  dlen: 39
DLSw: START-LLC2PFISM (peer on interface Serial0): event:CLS-CONNECT_STN.IND state:DISCONN
DLSw: dllc2p_action_s() conn_stn for peer on interface Serial0
    PEER-DISP Sent : CLSI Msg : REQ_OPNSTN.Req  dlen: 106
DLSw: END-LLC2PFISM (peer on interface Serial0): state:DISCONN->CONS_PEND

DLSw: Peer Received : CLSI Msg : REQ_OPNSTN.Cfm CLS_OK dlen: 106
DLSw: START-LLC2PFISM (peer on interface Serial0): event:CLS-REQOPNSTN.CNF state:CONS_PEND
DLSw: dllc2p_action_h() send capabilities to peer on interface Serial0
    PEER-DISP Sent : CLSI Msg : CONNECT.Rsp  dlen: 20
    PEER-DISP Sent : CLSI Msg : DATA.Req  dlen: 418
DLSw: CapExId Msg sent to peer on interface Serial0
DLSw: END-LLC2PFISM (peer on interface Serial0): state:CONS_PEND->WAIT_CAP

DLSw: Peer Received : CLSI Msg : CONNECTED.Ind  dlen: 8
DLSw: START-LLC2PFISM (peer on interface Serial0): event:CLS-CONNECTED.IND state:WAIT_CAP
DLSw: END-LLC2PFISM (peer on interface Serial0): state:WAIT_CAP->WAIT_CAP

DLSw: Peer Received : CLSI Msg : DATA.Ind  dlen: 418
DLSw: START-LLC2PFISM (peer on interface Serial0): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dllc2p_action_k() cap msg rcvd for peer on interface Serial0
DLSw: Recv CapExId Msg from peer on interface Serial0
    PEER-DISP Sent : CLSI Msg : DATA.Req  dlen: 96
DLSw: Pos CapExResp sent to peer on interface Serial0
DLSw: END-LLC2PFISM (peer on interface Serial0): state:WAIT_CAP->WAIT_CAP

DLSw: Peer Received : CLSI Msg : DATA.Ind  dlen: 96
DLSw: START-LLC2PFISM (peer on interface Serial0): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dllc2p_action_k() cap msg rcvd for peer on interface Serial0
DLSw: Recv CapExPosRsp Msg from peer on interface Serial0

```

```

DLSw: END-LLC2PFSM (peer on interface Serial0): state:WAIT_CAP->WAIT_CAP

DLSw: Processing delayed event:SSP-CAP EXCHANGED - prev state:WAIT_CAP
DLSw: START-LLC2PFSM (peer on interface Serial0): event:SSP-CAP EXCHANGED state:WAIT_CAP
DLSw: dllc2p_action_1() cap xchged for peer on interface Serial0
DLSw: END-LLC2PFSM (peer on interface Serial0): state:WAIT_CAP->CONNECT

```

The following messages occur when a CUR_ex (CANUREACH explorer) frame is received from other peers, and the peer statements or the **promiscuous** keyword have not been enabled so that the router is not configured correctly:

```

22:42:44: DLSw: Not promiscuous - Rej conn from 172.20.96.1(2065)
22:42:51: DLSw: Not promiscuous - Rej conn from 172.20.99.1(2065)

```

In the following messages, the router sends a keepalive message every 30 seconds to keep the peer connected. If three keepalive messages are missed, the peer is torn down. These messages are displayed only if keepalives are enabled (by default, keepalives are disabled):

```

22:44:03: DLSw: Keepalive Request sent to peer 172.20.98.1(2065) (168243148)
22:44:03: DLSw: Keepalive Response from peer 172.20.98.1(2065) (168243176)
22:44:34: DLSw: Keepalive Request sent to peer 172.20.98.1(2065) (168274148)
22:44:34: DLSw: Keepalive Response from peer 172.20.98.1(2065) (168274172)

```

The following peer debug messages indicate that the local peer is disconnecting from the specified remote peer because of missed peer keepalives:

```

0:03:24: DLSw: keepalive failure for peer on interface Serial0
0:03:24: DLSw: action_d(): for peer on interface Serial0
0:03:24: DLSW: DIRECT aborting connection for peer on interface Serial0
0:03:24: DLSw: peer on interface Serial0, old state CONNECT, new state DISCONN

```

The following peer debug messages result from an attempt to connect to an IP address that does not have DLSw enabled. The local router attempts to connect in 30-second intervals:

```

23:13:22: action_a() attempting to connect peer 172.20.100.1(2065)
23:13:22: DLSw: CONN: peer 172.20.100.1 open failed, rejected [9]
23:13:22: action_a() retries: 8 next conn time: 861232504
23:13:52: action_a() attempting to connect peer 172.20.100.1(2065)
23:13:52: DLSw: CONN: peer 172.20.100.1 open failed, rejected [9]
23:13:52: action_a() retries: 9 next conn time: 861292536

```

The following peer debug messages that indicates a remote peer statement is missing on the router (address 172.20.100.1) to which the connection attempt is sent:

```

23:14:52: action_a() attempting to connect peer 172.20.100.1(2065)
23:14:52: DLSw: action_a(): Write pipe opened for peer 172.20.100.1(2065)
23:14:52: DLSw: peer 172.20.100.1(2065), old state DISCONN, new state WAIT_RD
23:14:52: DLSw: dlsw_tcpd_fini() closing connection for peer 172.20.100.1
23:14:52: DLSw: action_d(): for peer 172.20.100.1(2065)
23:14:52: DLSw: aborting tcp connection for peer 172.20.100.1(2065)
23:14:52: DLSw: peer 172.20.100.1(2065), old state WAIT_RD, new state DISCONN

```

The following messages show a peer connection opening with no errors or abnormal events:

```

23:16:37: action_a() attempting to connect peer 172.20.100.1(2065)
23:16:37: DLSw: action_a(): Write pipe opened for peer 172.20.100.1(2065)
23:16:37: DLSw: peer 172.20.100.1(2065), old state DISCONN, new state WAIT_RD
23:16:37: DLSW: passive open 172.20.100.1(17762) -> 2065
23:16:37: DLSw: action_c(): for peer 172.20.100.1(2065)
23:16:37: DLSw: peer 172.20.100.1(2065), old state WAIT_RD, new state CAP_EXG
23:16:37: DLSw: peer 172.20.100.1(2065) conn_start_time set to 861397784
23:16:37: DLSw: CapExId Msg sent to peer 172.20.100.1(2065)
23:16:37: DLSw: Recv CapExId Msg from peer 172.20.100.1(2065)
23:16:37: DLSw: Pos CapExResp sent to peer 172.20.100.1(2065)

```

```

23:16:37: DLSw: action_e(): for peer 172.20.100.1(2065)
23:16:37: DLSw: Recv CapExPosRsp Msg from peer 172.20.100.1(2065)
23:16:37: DLSw: action_e(): for peer 172.20.100.1(2065)
23:16:37: DLSw: peer 172.20.100.1(2065), old state CAP_EXG, new state CONNECT
23:16:37: DLSw: dlsw_tcpd_fini() closing write pipe for peer 172.20.100.1
23:16:37: DLSw: action_g(): for peer 172.20.100.1(2065)
23:16:37: DLSw: closing write pipe tcp connection for peer 172.20.100.1(2065)
23:16:38: DLSw: peer_act_on_capabilities() for peer 172.20.100.1(2065)

```

The following two messages show that an information frame is passing through the router:

```

DLSw: dlsw_tr2fct() lmac:c000.a400.0000 rmac:0800.5a29.75fe ls:5 rs:4 i:34
DLSw: dlsw_tr2fct() lmac:c000.a400.0000 rmac:0800.5a29.75fe ls:4 rs:4 i:34

```

Sample Debug DLSw Reachability Messages

The messages in this section are based on the following criteria:

- Reachability is stored in cache. DLSw+ maintains two reachability caches: one for MAC addresses and one for NetBIOS names. Depending on how long entries have been in the cache, they are either fresh or stale.
- If a router has a fresh entry in the cache for a certain resource, it answers a locate request for that resource without verifying that it is still available. A locate request is typically a TEST frame for MAC addresses or a FIND_NAME_QUERY for NetBIOS.
- If a router has a stale entry in the cache for a certain resource, it verifies that the entry is still valid before answering a locate request for the resource by sending a frame to the last known location of the resource and waits for a resource. If the entry is a REMOTE entry, the router sends a CUR_ex frame to the remote peer to verify. If the entry is a LOCAL entry, it sends either a TEST frame or a NetBIOS FIND_NAME_QUERY on the appropriate local port.
- By default, all reachability cache entries remain fresh for 4 minutes after they are learned. For MAC addresses, you can change this time with the **dlsw timer sna-verify-interval** command. For NetBIOS names, you can change this time with the **dlsw timer netbios-verify-interval** command.
- By default, all reachability cache entries age out of the cache 16 minutes after they are learned. For MAC addresses, you can change this time with the **dlsw timer sna-cache-timeout** command. For NetBIOS names, you can change the time with the **dlsw timer netbios-cache-timeout** command.

Table 55 describes the debug output indicating that the DLSW router received an SSP message that is flow controlled and should be counted against the window of the sender.

```

Dec 6 11:26:49: CSM: Received SSP CUR csex flags = 80, mac 4000.90b1.26cf,
The csex flags = 80 means that this is an CUR_ex (explorer).
Dec 5 10:48:33: DLSw: 1620175180 decr r - s:27 so:0 r:27 ro:0

```

Table 55 debug output Command Descriptions

Field	Description
decr r	Decrement received count.
s	This DLSW router's granted units for the circuit.
so	0=This DLSW router does not owe a flow control acknowledgment. 1=This router owes a flow control acknowledgment.
r	Partner's number of granted units for the circuit.
ro	Indicates whether the partner owes flow control acknowledgment.

The following message shows that DLSw is sending an I frame to a LAN:

```
Dec 5 10:48:33: DISP Sent : CLSI Msg : DATA.Reg dlen: 1086
```

The following message shows that DLSw received the I frame from the LAN:

```
Dec 5 10:48:35: DLSW Received-disp : CLSI Msg : DATA.Ind dlen: 4
```

The following messages show that the reachability cache is cleared:

```
Router# clear dlsw rea
```

```
23:44:11: CSM: Clearing CSM cache
23:44:11: CSM: delete local mac cache for port 0
23:44:11: CSM: delete local name cache for port 0
23:44:11: CSM: delete remote mac cache for peer 0
23:44:11: CSM: delete remote name cash dlsw rea
```

The next group of messages show that the DLSw reachability cache is added, and that a name query is perform from the router Marian:

```
23:45:11: CSM: core_to_csm CLSI_MSG_PROC - port_id 5EFBB4
23:45:11: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:11: CSM: update local cache for mac 0800.5a30.7a9b, port 5EFBB4
23:45:11: CSM: update local cache for name MARIAN , port 5EFBB4
23:45:11: CSM: Received CLS_UDATA_STN from Core
23:45:11: CSM: Received netbios frame type A
23:45:11: CSM: Processing Name Query
23:45:11: CSM: Netbios Name Query: ws_status = 6
23:45:11: CSM: Write to peer 0 ok.
23:45:11: CSM: Freeing clsi message
23:45:11: CSM: core_to_csm CLSI_MSG_PROC - port_id 658AB4
23:45:11: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:11: CSM: update local cache for mac 0800.5a30.7a9b, port 658AB4
23:45:11: CSM: update local cache for name MARIAN , port 658AB4
23:45:11: CSM: Received CLS_UDATA_STN from Core
23:45:11: CSM: Received netbios frame type A
23:45:11: CSM: Processing Name Query
23:45:11: CSM: Netbios Name Query: ws_status = 5
23:45:11: CSM: DLXNR_PEND match found.... drop name query
23:45:11: CSM: Freeing clsi message
23:45:12: CSM: core_to_csm CLSI_MSG_PROC - port_id 5EFBB4
23:45:12: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:12: CSM: update local cache for mac 0800.5a30.7a9b, port 5EFBB4
23:45:12: CSM: update local cache for name MARIAN , port 5EFBB4
23:45:12: CSM: Received CLS_UDATA_STN from Core
23:45:12: CSM: Received netbios frame type A
23:45:12: CSM: Processing Name Query
23:45:12: CSM: Netbios Name Query: ws_status = 5
23:45:12: CSM: DLXNR_PEND match found.... drop name query
23:45:12: CSM: Freeing clsi message
23:45:12: CSM: core_to_csm CLSI_MSG_PROC - port_id 658AB4
23:45:12: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:12: CSM: update local cache for mac 0800.5a30.7a9b, port 658AB4
23:45:12: CSM: update local cache for name MARIAN , port 658AB4
23:45:12: CSM: Received CLS_UDATA_STN from Core
23:45:12: CSM: Received netbios frame type A
23:45:12: CSM: Processing Name Query
23:45:12: CSM: Netbios Name Query: ws_status = 5
23:45:12: CSM: DLXNR_PEND match found.... drop name query
23:45:12: CSM: Freeing clsi message
23:45:12: CSM: core_to_csm CLSI_MSG_PROC - port_id 5EFBB4
23:45:12: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:12: CSM: update local cache for mac 0800.5a30.7a9b, port 5EFBB4
23:45:12: CSM: update local cache for name MARIAN , port 5EFBB4
```

```

23:45:12: CSM: Received CLS_UDATA_STN from Core
23:45:12: CSM: Received netbios frame type A
23:45:12: CSM: Processing Name Query
23:45:12: CSM: Netbios Name Query: ws_status = 5
23:45:12: CSM: DLXNR_PEND match found.... drop name query
23:45:12: CSM: Freeing clsi message
23:45:12: CSM: core_to_csm CLSI_MSG_PROC - port_id 658AB4
23:45:12: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:12: CSM: update local cache for mac 0800.5a30.7a9b, port 658AB4
23:45:12: CSM: update local cache for name MARIAN , port 658AB4
23:45:12: CSM: Received CLS_UDATA_STN from Core
23:45:12: CSM: Received netbios frame type A
23:45:12: CSM: Processing Name Query
23:45:12: CSM: Netbios Name Query: ws_status = 5
23:45:12: CSM: DLXNR_PEND match found.... drop name query
23:45:12: CSM: Freeing clsi message
23:45:18: CSM: Deleting Reachability cache
23:45:18: CSM: Deleting DLX NR pending record...
23:45:38: CSM: core_to_csm CLSI_MSG_PROC - port_id 5EFBB4
23:45:38: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:38: CSM: update local cache for mac 0800.5a30.7a9b, port 5EFBB4
23:45:38: CSM: update local cache for name MARIAN , port 5EFBB4
23:45:38: CSM: Received CLS_UDATA_STN from Core
23:45:38: CSM: Received netbios frame type 8
23:45:38: CSM: Write to peer 0 ok.
23:45:38: CSM: Freeing clsi message
23:45:38: CSM: core_to_csm CLSI_MSG_PROC - port_id 658AB4
23:45:38: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:38: CSM: update local cache for mac 0800.5a30.7a9b, port 658AB4
23:45:38: CSM: update local cache for name MARIAN , port 658AB4
23:45:38: CSM: Received CLS_UDATA_STN from Core
23:45:38: CSM: Received netbios frame type 8
23:45:38: CSM: Write to peer 0 ok.
23:45:38: CSM: Freeing clsi message

```

The following messages show that the router named Marian is added to the network:

```

23:45:38: CSM: core_to_csm CLSI_MSG_PROC - port_id 5EFBB4
23:45:38: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:38: CSM: update local cache for mac 0800.5a30.7a9b, port 5EFBB4
23:45:38: CSM: update local cache for name MARIAN , port 5EFBB4
23:45:38: CSM: Received CLS_UDATA_STN from Core
23:45:38: CSM: Received netbios frame type 8
23:45:38: CSM: Write to peer 0 ok.
23:45:38: CSM: Freeing clsi message
23:45:38: CSM: core_to_csm CLSI_MSG_PROC - port_id 658AB4
23:45:38: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:38: CSM: update local cache for mac 0800.5a30.7a9b, port 658AB4
23:45:38: CSM: update local cache for name MARIAN , port 658AB4
23:45:38: CSM: Received CLS_UDATA_STN from Core
23:45:38: CSM: Received netbios frame type 8
23:45:38: CSM: Write to peer 0 ok.
23:45:38: CSM: Freeing clsi message

```

In the next group of messages, an attempt is made to add the router named Ginger on the Ethernet interface:

```

0:07:44: CSM: core_to_csm CLSI_MSG_PROC - port_id 658AB4
0:07:44: CSM: 0004.f545.24e6 passes local mac excl. filter
0:07:44: CSM: update local cache for mac 0004.f545.24e6, port 658AB4
0:07:44: CSM: update local cache for name GINGER , port 658AB4
0:07:44: CSM: Received CLS_UDATA_STN from Core
0:07:44: CSM: Received netbios frame type 8
0:07:44: CSM: Write to peer 0 ok.

```

In the following example, the output from the **show dlsw reachability** command indicates that Ginger is on the Ethernet interface and Marian is on the Token Ring interface:

```
Router# show dlsw reachability
```

```
DLSw MAC address reachability cache list
```

Mac Addr	status	Loc.	peer/port	rif
0004.f545.24e6	FOUND	LOCAL	P007-S000	--no rif--
0800.5a30.7a9b	FOUND	LOCAL	P000-S000	06C0.0621.7D00
			P007-S000	F0F8.0006.A6FC.005F.F100.0000.0000.0000

```
DLSw NetBIOS Name reachability cache list
```

NetBIOS Name	status	Loc.	peer/port	rif
GINGER	FOUND	LOCAL	P007-S000	--no rif--
MARIAN	FOUND	LOCAL	P000-S000	06C0.0621.7D00
			P007-S000	--no rif--

debug dmsp doc-to-fax

To display debug messages for the doc Media Service Provider TIFF or text2Fax engine, use the **debug dmsp doc-to-fax** EXEC command. To disable the debug messages, use the **no** form of this command.

debug dmsp doc-to-fax [**text-to-fax** | **tiff-reader**]

no debug dmsp doc-to-fax [**text-to-fax** | **tiff-reader**]

Syntax Description		
	text-to-fax	(Optional) Displays debug messages that occur while the DocMSP Component is receiving text packets and producing T4 fax data.
	tiff-reader	(Optional) Displays debug messages that occur while the DocMSP Component is receiving TIFF packets and producing T4 fax data.

Defaults No default behavior or values.

Command History	Release	Modification
	12.1(3)XI	This command was introduced on the Cisco AS5300 access server.

Examples

The following example displays output from the **debug dmsp doc-to-fax** command.

```
Router# debug dmsp doc-to-fax
```

```
Jan 1 04:58:39.898: docmsp_call_setup_request: callid=18
Jan 1 04:58:39.902: docmsp_call_setup_request(): ramp data dir=OFFRAMP, conf dir=SRC
Jan 1 04:58:39.902: docmsp_caps_ind: call id=18, src=17
Jan 1 04:58:39.902: docmsp_bridge cfid=5, srccid=18, dstcid=17

Jan 1 04:58:39.902: docmsp_bridge(): ramp data dir=OFFRAMP, conf dir=SRC, encode out=2
Jan 1 04:58:39.902: docmsp_rcv_msp_ev: call id =18, evID = 42
Jan 1 04:58:39.902: docmsp_bridge cfid=6, srccid=18, dstcid=15

Jan 1 04:58:39.902: docmsp_bridge(): ramp data dir=OFFRAMP, conf dir=DEST, encode out=2
Jan 1 04:58:39.902: docmsp_process_rcv_data: call id src=0, dst=18
Jan 1 04:58:39.902: docmsp_generate_page:
Jan 1 04:58:39.902: docmsp_generate_page: new context for Call 18
Jan 1 04:58:39.922: docmsp_get_msp_event_buffer:
Jan 1 04:58:42.082: docmsp_xmit: call id src=15, dst=18
Jan 1 04:58:42.082: docmsp_process_rcv_data: call id src=15, dst=18
Jan 1 04:58:42.082: offramp_data_process:
Jan 1 04:58:42.102: docmsp_xmit: call id src=15, dst=18
Jan 1 04:58:42.106: docmsp_process_rcv_data: call id src=15, dst=18
Jan 1 04:58:42.106: offramp_data_process:
Jan 1 04:58:42.122: docmsp_xmit: call id src=15, dst=18
Jan 1 04:58:42.126: docmsp_process_rcv_data: call id src=15, dst=18
Jan 1 04:58:42.126: offramp_data_process:
Jan 1 04:58:42.142: docmsp_xmit: call id src=15, dst=18
Jan 1 04:58:42.146: docmsp_xmit: call id src=15, dst=18
```

Related Commands

Command	Description
debug dmsp fax-to-doc	Displays debug messages for the doc Media Service Provider fax-to-doc TIFF engine.
