

debug mpoa client

To display MPC debug information, use the **debug mpoa client** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug mpoa client { all | data | egress | general | ingress | keep-alives | platform-specific }
                    [ name mpc-name ]
```

```
no debug mpoa client { all | data | egress | general | ingress | keep-alives | platform-specific }
                    [ name mpc-name ]
```

Syntax Description

all	Displays debugging information for all MPC activity.
data	Displays debugging information for data plane activity only. This option applies only to routers.
egress	Displays debugging information for egress functionality only.
general	Displays general debugging information only.
ingress	Displays debugging information for ingress functionality only.
keep-alives	Displays debugging information for keep-alive activity only.
platform-specific	Displays debugging information for specific platforms only. This option applies only to the Catalyst 5000 series ATM module.
name <i>mpc-name</i>	Specifies the name of the MPC with the specified name.

Defaults

The default is debugging turned on for all MPCs.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3	This command was introduced.

Examples

The following shows how to turn on debugging for the MPC ip_mpc:

```
ATM# debug mpoa client all name ip_mpc
```

Related Commands

Command	Description
debug mpoa server	Displays information about the MPOA server.

debug mpoa server

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

```
debug mpoa server [name mps-name]
```

```
no debug mpoa server [name mps-name]
```

Syntax Description	<i>name mps-name</i>	(Optional) Specifies the name of a MPOA server.
--------------------	----------------------	---

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

Command History	Release	Modification
	11.3	This command was introduced.

Usage Guidelines	The debug mpo server command optionally limits the output only to the specified MPS.
------------------	---

Examples	The following turns on debugging only for the MPS named ip_mps:
----------	---

```
Router# debug mpoa server name ip_mps
```

Related Commands	Command	Description
	debug modem traffic	Displays MPC debug information.

debug mrcp

To display debug messages for Media Resource Control Protocol (MRCP) operations, use the **debug mrcp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug mrcp {all | api | error | pmh | session | state}
```

```
no debug mrcp {all | api | error | pmh | session | state}
```

Syntax Description

all	Displays all MRCP debug messages.
api	Displays messages between the application and the MRCP stack.
error	Displays MRCP error messages.
pmh	Displays protocol message handler (PMH) messages.
session	Displays messages about active MRCP sessions.
state	Displays Finite State Machine (FSM) messages.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(11)T	This command was introduced on the Cisco 3640, Cisco 3660, Cisco AS5300, Cisco AS5350, and Cisco AS5400.

Examples

The following examples show output from the **debug mrcp** commands. Comments are imbedded in the examples to describe significant sections of the output.

The following example shows output from the **debug mrcp api** command:

```
Router# debug mrcp api
```

The first four lines show Real Time Streaming Protocol (RTSP) socket commands for Text-To-Speech (TTS) operations:

```
*Apr 17 16:31:16.323:mrcp_add_param:param:Kill-On-Barge-In:
*Apr 17 16:31:16.323:mrcp_add_param:param:Speech-Language:
*Apr 17 16:31:16.323:mrcp_add_param:param:Logging-Tag:
*Apr 17 16:31:16.323:mrcp_add_param:param:Content-Base:

*Apr 17 16:31:16.323:mrcp_create_session:same host/port
*Apr 17 16:31:16.323:mrcp_associate_call 5 10
*Apr 17 16:31:16.323:mrcp_associate_call 5 10
*Apr 17 16:31:16.323:mrcp_synth_speak 5
*Apr 17 16:31:16.323:mrcp_add_param:param:Content-Base:
*Apr 17 16:31:16.323:mrcp_recognizer_define_grammar 5
```

The following lines show RTSP socket commands for Automatic Speech Recognition (ASR) operations:

```
*Apr 17 16:31:16.323:mrcp_add_param:param:Confidence-Threshold:
*Apr 17 16:31:16.323:mrcp_add_param:param:Sensitivity-Level:
*Apr 17 16:31:16.323:mrcp_add_param:param:Speed-Vs-Accuracy:
*Apr 17 16:31:16.323:mrcp_add_param:param:Dtmf-Interdigit-Timeout:
*Apr 17 16:31:16.323:mrcp_add_param:param:Dtmf-Term-Timeout:
*Apr 17 16:31:16.323:mrcp_add_param:param:Dtmf-Term-Char:
*Apr 17 16:31:16.323:mrcp_add_param:param:No-Input-Timeout:
*Apr 17 16:31:16.323:mrcp_add_param:param:Logging-Tag:
*Apr 17 16:31:16.327:mrcp_add_param:param:Content-Base:
*Apr 17 16:31:16.327:mrcp_add_param:param:Recognizer-Start-Timers:
*Apr 17 16:31:16.327:mrcp_recognizer_start 5
*Apr 17 16:31:26.715:mrcp_add_param:param:Kill-On-Barge-In:
*Apr 17 16:31:26.715:mrcp_add_param:param:Speech-Language:
*Apr 17 16:31:26.715:mrcp_add_param:param:Logging-Tag:
*Apr 17 16:31:26.715:mrcp_add_param:param:Content-Base:
*Apr 17 16:31:26.715:mrcp_synth_speak 5
*Apr 17 16:31:30.451:mrcp_destroy_session 5 type:SYNTHESIZER
*Apr 17 16:31:30.451:mrcp_destroy_session 5 type:RECOGNIZER
```

The following examples show output from the **debug mrcp error** command:

```
Router# debug mrcp error
```

This output shows an error when the response from the server is incorrect:

```
*May 9 20:29:09.936:Response from 10.1.2.58:554 failed
*May 9 20:29:09.936:MRCP/1.0 71 422 COMPLETE
```

This output shows an error when the RTSP connection to the server fails:

```
*May 9 20:29:09.936:Connecting to 10.1.2.58:554 failed
```

This output shows an error when the recognize request comes out of sequence:

```
*May 9 20:29:09.936:act_idle_recognize:ignoring old recognize request
```

The following example shows output from the **debug mrcp pmh** command:

```
Router# debug mrcp pmh
```

```
*Apr 17 16:32:51.777:param:Kill-On-Barge-In: true
*Apr 17 16:32:51.777:param:Speech-Language: en-US
*Apr 17 16:32:51.777:param:Logging-Tag: 14:14
*Apr 17 16:32:51.777:param:Content-Base: http://server-asr/
*Apr 17 16:32:51.777:param:Content-Base: http://server-asr/
*Apr 17 16:32:51.777:param:Confidence-Threshold: 50
*Apr 17 16:32:51.781:param:Sensitivity-Level: 50
*Apr 17 16:32:51.781:param:Speed-Vs-Accuracy: 50
*Apr 17 16:32:51.781:param:Dtmf-Interdigit-Timeout: 10000
*Apr 17 16:32:51.781:param:Dtmf-Term-Timeout: 10000
*Apr 17 16:32:51.781:param:Dtmf-Term-Char: #
*Apr 17 16:32:51.781:param:No-Input-Timeout: 10000
*Apr 17 16:32:51.781:param:Logging-Tag: 14:14
*Apr 17 16:32:51.781:param:Content-Base: http://server-asr/
*Apr 17 16:32:51.781:param:Recognizer-Start-Timers: false
*Apr 17 16:32:51.877:GRAMMAR-CONTENT-HEADER
*Apr 17 16:32:51.877:Content-Type:application/grammar+xml
Content-Id:field2@field.grammar
Content-Length:356
```

```

*Apr 17 16:32:51.885:GRAMMAR-CONTENT-HEADER
*Apr 17 16:32:51.885:Content-Type:text/uri-list
Content-Length:30

*Apr 17 16:32:51.885:Total-Length=365
*Apr 17 16:32:51.885:@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
*Apr 17 16:32:51.885:RECOGNIZE 20 MRCP/1.0
Confidence-Threshold:50
Sensitivity-Level:50
Speed-Vs-Accuracy:50
Dtmf-Interdigit-Timeout:10000
Dtmf-Term-Timeout:10000
Dtmf-Term-Char:#
No-Input-Timeout:10000
Logging-Tag:14:14
Content-Base:http://server-asr/
Recognizer-Start-Timers:false

*Apr 17 16:32:51.885:Content-Type:text/uri-list
Content-Length:30

*Apr 17 16:32:51.885:session:field2@field.grammar

*Apr 17 16:32:51.885:@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

*Apr 17 16:32:51.889:SPEECH-MARKUP-TYPE-HEADER
*Apr 17 16:32:51.889:Content-Type:application/synthesis+ssml
Content-Length:126

*Apr 17 16:32:51.889:Total-Length=313
*Apr 17 16:32:51.889:@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
*Apr 17 16:32:51.889:SPEAK 18 MRCP/1.0
Kill-On-Barge-In:true
Speech-Language:en-US
Logging-Tag:14:14
Content-Base:http://server-asr/

*Apr 17 16:32:51.889:Content-Type:application/synthesis+ssml
Content-Length:126

*Apr 17 16:32:51.889:<?xml version="1.0"?><speak> Who do you want speak to?? Joe, Carl,
Alex?. And I am extending the length of the text</speak>
*Apr 17 16:32:51.889:@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

*Apr 17 16:32:51.925:mrcp_pmh_parse_response:Length:28
Apr 17 16:32:51.925:mrcp_pmh_get_request_line:Line:MRCP/1.0 19 200 COMPLETE
*Apr 17 16:32:51.925:Request-tag:19 resp-code:200 Status:COMPLETE
*Apr 17 16:32:51.925:No Of Properties:0
*Apr 17 16:32:51.925:mrcp_process_recog_response:
*Apr 17 16:32:51.933:mrcp_pmh_parse_response:Length:31
Apr 17 16:32:51.933:mrcp_pmh_get_request_line:Line:MRCP/1.0 20 200 IN-PROGRESS
*Apr 17 16:32:51.933:Request-tag:20 resp-code:200 Status:IN-PROGRESS
*Apr 17 16:32:51.933:No Of Properties:0
*Apr 17 16:32:51.933:mrcp_process_recog_response:
*Apr 17 16:32:53.413:mrcp_pmh_parse_response:Length:31
Apr 17 16:32:53.413:mrcp_pmh_get_request_line:Line:MRCP/1.0 18 200 IN-PROGRESS
*Apr 17 16:32:53.413:Request-tag:18 resp-code:200 Status:IN-PROGRESS
*Apr 17 16:32:53.413:No Of Properties:0
*Apr 17 16:32:53.413:mrcp_process_synth_response:
*Apr 17 16:33:01.685:mrcp_pmh_parse_response:Length:100

```

```

Apr 17 16:33:01.689:mrcp_pmh_get_event_line:Line:SPEAK-COMplete 18 COMPLETE MRCP/1.0
*Apr 17 16:33:01.689:Request-tag:18 resp-code:200 Status:COMPLETE
*Apr 17 16:33:01.689:No Of Properties:2
*Apr 17 16:33:01.689:mrcp_process_synth_events:
*Apr 17 16:33:01.689: COMPLETION-CAUSE:1
*Apr 17 16:33:01.689:mrcp_send_synth_app_response:
*Apr 17 16:33:01.689:mrcp_pmh_parse_response:Length:61
Apr 17 16:33:01.689:mrcp_pmh_get_event_line:Line:START-OF-SPEECH 20 IN-PROGRESS MRCP/1.0
*Apr 17 16:33:01.689:Request-tag:20 resp-code:200 Status:IN-PROGRESS
*Apr 17 16:33:01.689:No Of Properties:1
*Apr 17 16:33:01.689:mrcp_process_recog_events:
*Apr 17 16:33:02.653:mrcp_pmh_parse_response:Length:815
Apr 17 16:33:02.653:mrcp_pmh_get_event_line:Line:RECOGNITION-COMplete 20 COMPLETE
MRCP/1.0
*Apr 17 16:33:02.653:Request-tag:20 resp-code:200 Status:COMPLETE
*Apr 17 16:33:02.653:No Of Properties:2
*Apr 17 16:33:02.653:mrcp_process_recog_events:
*Apr 17 16:33:02.653: COMPLETION-CAUSE:0
*Apr 17 16:33:02.653:mrcp_send_recog_app_response:
*Apr 17 16:33:02.661:param:Kill-On-Barge-In: true
*Apr 17 16:33:02.661:param:Speech-Language: en-US
*Apr 17 16:33:02.661:param:Logging-Tag: 14:14
*Apr 17 16:33:02.661:param:Content-Base: http://server-asr/
*Apr 17 16:33:02.665:SPEECH-MARKUP-TYPE-HEADER
*Apr 17 16:33:02.665:Content-Type:application/synthesis+ssml
Content-Length:57

*Apr 17 16:33:02.665:Total-Length=243
*Apr 17 16:33:02.665:@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
*Apr 17 16:33:02.665:SPEAK 22 MRCP/1.0
Kill-On-Barge-In:true
Speech-Language:en-US
Logging-Tag:14:14
Content-Base:http://server-asr/

*Apr 17 16:33:02.665:Content-Type:application/synthesis+ssml
Content-Length:57

*Apr 17 16:33:02.665:<?xml version="1.0"?><speak> You have joe mails</speak>
*Apr 17 16:33:02.665:@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

*Apr 17 16:33:02.833:mrcp_pmh_parse_response:Length:31
Apr 17 16:33:02.833:mrcp_pmh_get_request_line:Line:MRCP/1.0 22 200 IN-PROGRESS
*Apr 17 16:33:02.833:Request-tag:22 resp-code:200 Status:IN-PROGRESS
*Apr 17 16:33:02.833:No Of Properties:0
*Apr 17 16:33:02.833:mrcp_process_synth_response:
*Apr 17 16:33:06.382:mrcp_pmh_parse_response:Length:98
Apr 17 16:33:06.382:mrcp_pmh_get_event_line:Line:SPEAK-COMplete 22 COMPLETE MRCP/1.0
*Apr 17 16:33:06.382:Request-tag:22 resp-code:200 Status:COMPLETE
*Apr 17 16:33:06.382:No Of Properties:2
*Apr 17 16:33:06.382:mrcp_process_synth_events:
*Apr 17 16:33:06.382: COMPLETION-CAUSE:0
*Apr 17 16:33:06.382:mrcp_send_synth_app_response:

```

The following example shows output from the **debug mrcp session** command:

Router# **debug mrcp session**

```
*Apr 17 16:34:07.851:mrcp_create_session:
*Apr 17 16:34:07.851:mrcp_create_session:New SCB creation
*Apr 17 16:34:07.851:mrcp_create_svr_session_url:
*Apr 17 16:34:07.851:mrcp_create_session:
*Apr 17 16:34:07.851:mrcp_create_session:Already an SCB is created for this call
*Apr 17 16:34:07.851:mrcp_process_events:event:LIB_CONNECT SYNTHESIZERCONN-STATUS=0
*Apr 17 16:34:07.855:mrcp_process_events:event:SPEAK SYNTHESIZER
*Apr 17 16:34:07.855:mrcp_process_events:event:SPEAK defered
*Apr 17 16:34:07.855:mrcp_process_events:event:LIB_CONNECT RECOGNIZERCONN-STATUS=0
*Apr 17 16:34:07.855:mrcp_process_events:event:DEFINE_GRAMMAR RECOGNIZER
*Apr 17 16:34:07.855:mrcp_process_events:event:DEFINE_GRAMMAR defered
*Apr 17 16:34:07.855:mrcp_process_events:event:LIB_CONNECT RECOGNIZERCONN-STATUS=0
*Apr 17 16:34:07.855:mrcp_process_events:event:RECOGNIZE RECOGNIZER
*Apr 17 16:34:07.855:mrcp_process_events:event:RECOGNIZE defered
*Apr 17 16:34:07.855:mrcp_response_handler:status=RTSPLIB_STATUS_SERVER_CONNECTED
*Apr 17 16:34:07.855:mrcp_process_events:event:LIB_CONNECTED SYNTHESIZERCONN-STATUS=4
*Apr 17 16:34:07.947:mrcp_response_handler:status=RTSPLIB_STATUS_RTP_RECORD_SETUP
*Apr 17 16:34:07.947:mrcp_process_events:event:RECOG_RTP_SETUP RECOGNIZER
*Apr 17 16:34:07.947:mrcp_process_defered_events:event:DEFINE_GRAMMAR
*Apr 17 16:34:07.947:mrcp_process_defered_events:event:RECOGNIZECONN-STATUS=2
*Apr 17 16:34:07.971:mrcp_response_handler:status=RTSPLIB_STATUS_RECORD_ASSOCIATED
*Apr 17 16:34:07.971:mrcp_response_handler:status=RTSPLIB_STATUS_RTP_PLAY_SETUP
*Apr 17 16:34:07.975:mrcp_process_events:event:RECOGNIZER_ASSOCIATED RECOGNIZER
*Apr 17 16:34:07.975:mrcp_process_events:event:SYNTH_RTP_SETUP SYNTHESIZER
*Apr 17 16:34:07.975:mrcp_process_defered_events:event:SPEAKCONN-STATUS=1
*Apr 17 16:34:07.975:mrcp_response_handler:status=RTSPLIB_STATUS_PLAY_ASSOCIATED
*Apr 17 16:34:07.975:mrcp_process_events:event:SYNTHESIZER_ASSOCIATED SYNTHESIZER
*Apr 17 16:34:08.007:mrcp_response_handler:status=RTSPLIB_STATUS_RESP_OK
*Apr 17 16:34:08.019:mrcp_response_handler:status=RTSPLIB_STATUS_RESP_OK
*Apr 17 16:34:08.059:mrcp_response_handler:status=RTSPLIB_STATUS_RESP_OK
*Apr 17 16:34:17.611:mrcp_response_handler:status=RTSPLIB_STATUS_RESP_OK
*Apr 17 16:34:17.611:mrcp_response_handler:status=RTSPLIB_STATUS_RESP_OK
*Apr 17 16:34:17.611:mrcp_process_events:event:SPEECH_COMPLETE SYNTHESIZER
*Apr 17 16:34:17.611:mrcp_process_events:event:START_OF_SPEECH RECOGNIZER
*Apr 17 16:34:18.575:mrcp_response_handler:status=RTSPLIB_STATUS_RESP_OK
*Apr 17 16:34:18.575:mrcp_process_events:event:RECOGNITION_COMPLETE RECOGNIZER
*Apr 17 16:34:18.583:mrcp_process_events:event:SPEAK SYNTHESIZER
*Apr 17 16:34:18.587:mrcp_response_handler:status=RTSPLIB_STATUS_PLAY_ASSOCIATED
*Apr 17 16:34:18.587:mrcp_process_events:event:SYNTHESIZER_ASSOCIATED SYNTHESIZER
*Apr 17 16:34:18.763:mrcp_response_handler:status=RTSPLIB_STATUS_RESP_OK
*Apr 17 16:34:22.279:mrcp_response_handler:status=RTSPLIB_STATUS_RESP_OK
*Apr 17 16:34:22.283:mrcp_process_events:event:SPEECH_COMPLETE SYNTHESIZER
*Apr 17 16:34:22.307:mrcp_process_events:event:LIB_DESTROY SYNTHESIZERCONN-STATUS=12
*Apr 17 16:34:22.311:mrcp_process_events:event:LIB_DESTROY RECOGNIZERCONN-STATUS=12
*Apr 17 16:34:22.311:mrcp_response_handler:status=RTSPLIB_STATUS_CLEANUP
*Apr 17 16:34:22.315:mrcp_free_fsm:
*Apr 17 16:34:22.315:mrcp_free_scb:
*Apr 17 16:34:22.315:mrcp_create_session_history:scb=0x62C712F4
*Apr 17 16:34:22.315:mrcp_insert_session_history_record:current=0x62999544, callID=0x12
*Apr 17 16:34:22.315:mrcp_insert_session_history_record:count = 3
*Apr 17 16:34:22.315:mrcp_insert_session_history_record:starting history record
deletion_timer of 10 minutes
```

The following example shows output from the **debug mrcp state** command:

Router# **debug mrcp state**

```
*Apr 17 16:35:25.141:mr_cp_add_synthesizer_fsm:adding synthesizer fsm
*Apr 17 16:35:25.141:mr_cp_add_connection_fsm:adding connection fsm
*Apr 17 16:35:25.141:mr_cp_add_rtpsetup_fsm:adding rtpsetup fsm
*Apr 17 16:35:25.145:hash_get: key=7
*Apr 17 16:35:25.145:mr_cp_add_recognizer_fsm:adding recognizer fsm
*Apr 17 16:35:25.145:mr_cp_add_connection_fsm:adding connection fsm
*Apr 17 16:35:25.145:mr_cp_add_rtpsetup_fsm:adding rtpsetup fsm
*Apr 17 16:35:25.145:mr_cp_fsm_execute:type=SYNTHESIZER
```

The following lines show the gateway connecting to the TTS server:

```
*Apr 17 16:35:25.145: curr[CONNECT_IDLE] ev-id[LIB_CONNECT]
next[CONNECTING] action=610B8FD00
*Apr 17 16:35:25.145:act_idle_libconnect
*Apr 17 16:35:25.145:mr_cp_shortcut_connection_fsm
*Apr 17 16:35:25.149:mr_cp_fsm_execute:type=SYNTHESIZER
*Apr 17 16:35:25.149: curr[CONNECTING] ev-id[LIB_CONNECT_PENDING]
next[CONNECTING] action=610B90F80
*Apr 17 16:35:25.149:act_connecting_libpending
*Apr 17 16:35:25.149:mr_cp_fsm_execute:type=SYNTHESIZER
*Apr 17 16:35:25.149: curr[CONNECTING] ev-id[LIB_CONNECT]
next[CONNECTING] action=610B8D480
*Apr 17 16:35:25.149:act_connectfsm_error
*Apr 17 16:35:25.149:mr_cp_fsm_execute:type=SYNTHESIZER
*Apr 17 16:35:25.149: curr[CONNECTING] ev-id[LIB_CONNECT]
```

The following lines show the gateway successfully connected to the TTS server:

```
next[CONNECTING] action=610B8D480
*Apr 17 16:35:25.149:act_connectfsm_error
*Apr 17 16:35:25.149:mr_cp_fsm_execute:type=SYNTHESIZER
*Apr 17 16:35:25.149: curr[CONNECTING] ev-id[LIB_CONNECTED]
next[CONNECTED] action=610B913C0
*Apr 17 16:35:25.149:act_connecting_libconnected
*Apr 17 16:35:25.149:act_rtpsetupfsm_libdescribed
*Apr 17 16:35:25.237:mr_cp_fsm_execute:type=RESOURCE_NONE
*Apr 17 16:35:25.237: curr[RTP_IDLE] ev-id[RECOG_RTP_SETUP]
next[RTP_RECOG_SETUP_DONE] action=610B94F40
*Apr 17 16:35:25.237:act_idle_reco_rtpsetup
*Apr 17 16:35:25.237:mr_cp_fsm_execute:type=RECOGNIZER
*Apr 17 16:35:25.237: curr[RECOG_IDLE] ev-id[DEFINE_GRAMMAR]
next[RECOG_IDLE] action=610B99340
*Apr 17 16:35:25.237:act_idle_define_grammar:
*Apr 17 16:35:25.237:hash_add: key=31
*Apr 17 16:35:25.237:mr_cp_fsm_execute:type=RECOGNIZER
*Apr 17 16:35:25.237: curr[RECOG_IDLE] ev-id[RECOGNIZE]
next[RECOG_ASSOCIATING] action=610B98400
*Apr 17 16:35:25.237:act_idle_recognize:
*Apr 17 16:35:25.245:mr_cp_fsm_execute:type=RECOGNIZER
*Apr 17 16:35:25.245: curr[RECOG_ASSOCIATING] ev-id[RECOGNIZER_ASSOCIATED]
next[RECOGNIZING] action=610B9AB40
*Apr 17 16:35:25.245:act_associating_recognizer_associated:
*Apr 17 16:35:25.249:hash_add: key=32
*Apr 17 16:35:25.249:mr_cp_fsm_execute:type=RESOURCE_NONE
*Apr 17 16:35:25.249: curr[RTP_IDLE] ev-id[SYNTH_RTP_SETUP]
next[RTP_SYNTH_SETUP_DONE] action=610B93D40
*Apr 17 16:35:25.249:act_idle_synth_rtpsetup
*Apr 17 16:35:25.249:mr_cp_fsm_execute:type=SYNTHESIZER
*Apr 17 16:35:25.249: curr[SYNTH_IDLE] ev-id[SPEAK]
next[SYNTH_ASSOCIATING] action=610BA5540
*Apr 17 16:35:25.249:act_idle_speak
```

```
*Apr 17 16:35:25.249:mrcp_fsm_execute:type=SYNTHESIZER
*Apr 17 16:35:25.249:    curr[SYNTH_ASSOCIATING]  ev-id[SYNTHESIZER_ASSOCIATED]
```

The following lines show the TTS server performing speech synthesis:

```
    next[SPEAKING] action=610BA7B40
*Apr 17 16:35:25.249:act_associating_speak_associated
*Apr 17 16:35:25.249:hash_add:  key=30
*Apr 17 16:35:25.285:hash_get:  key=31
*Apr 17 16:35:25.285:hash_delete:  key=31
*Apr 17 16:35:25.293:hash_get:  key=32
*Apr 17 16:35:25.293:hash_get:  key=30
*Apr 17 16:35:32.805:hash_get:  key=30
*Apr 17 16:35:32.805:hash_delete:  key=30
*Apr 17 16:35:32.805:mrcp_fsm_execute:type=SYNTHESIZER
*Apr 17 16:35:32.805:    curr[SPEAKING]  ev-id[SPEECH_COMPLETE]
    next[SYNTH_IDLE] action=610BAA680
*Apr 17 16:35:32.805:act_speaking_speech_complete
*Apr 17 16:35:32.809:hash_get:  key=32
*Apr 17 16:35:32.809:mrcp_fsm_execute:type=RECOGNIZER
*Apr 17 16:35:32.809:    curr[RECOGNIZING]  ev-id[START_OF_SPEECH]
    next[RECOGNIZING] action=610B9F3C0
*Apr 17 16:35:32.809:act_recognizing_start_of_speech
*Apr 17 16:35:33.781:hash_get:  key=32
*Apr 17 16:35:33.781:hash_delete:  key=32
*Apr 17 16:35:33.781:mrcp_fsm_execute:type=RECOGNIZER
*Apr 17 16:35:33.781:    curr[RECOGNIZING]  ev-id[RECOGNITION_COMPLETE]
    next[RECOGNIZED] action=610B9D240
*Apr 17 16:35:33.781:act_recognizing_recognition_complete:
*Apr 17 16:35:33.789:mrcp_fsm_execute:type=SYNTHESIZER
*Apr 17 16:35:33.789:    curr[SYNTH_IDLE]  ev-id[SPEAK]
    next[SYNTH_ASSOCIATING] action=610BA5540
*Apr 17 16:35:33.789:act_idle_speak
*Apr 17 16:35:33.793:mrcp_fsm_execute:type=SYNTHESIZER
*Apr 17 16:35:33.793:    curr[SYNTH_ASSOCIATING]  ev-id[SYNTHESIZER_ASSOCIATED]
    next[SPEAKING] action=610BA7B40
*Apr 17 16:35:33.793:act_associating_speak_associated
*Apr 17 16:35:33.793:hash_add:  key=34
*Apr 17 16:35:33.949:hash_get:  key=34
*Apr 17 16:35:37.221:hash_get:  key=34
*Apr 17 16:35:37.221:hash_delete:  key=34
*Apr 17 16:35:37.221:mrcp_fsm_execute:type=SYNTHESIZER
*Apr 17 16:35:37.221:    curr[SPEAKING]  ev-id[SPEECH_COMPLETE]
    next[SYNTH_IDLE] action=610BAA680
*Apr 17 16:35:37.221:act_speaking_speech_complete
*Apr 17 16:35:37.245:mrcp_fsm_execute:type=SYNTHESIZER
*Apr 17 16:35:37.249:    curr[CONNECTED]  ev-id[LIB_DESTROY]
    next[CONNECTED] action=610B8DD00
*Apr 17 16:35:37.249:act_connected_libdestroy
*Apr 17 16:35:37.249:mrcp_fsm_execute:type=SYNTHESIZER
*Apr 17 16:35:37.249:    curr[CONNECTED]  ev-id[LIB_DESTROY]
    next[CONNECTED] action=610B8DD00
*Apr 17 16:35:37.249:act_connected_libdestroy
```

Related Commands

Command	Description
show mrcp client session active	Displays information about active MRCP sessions.
show mrcp client session history	Displays information about past MRCP sessions.
show mrcp client statistics hostname	Displays statistics about MRCP sessions.

debug mspi receive

To display debug messages for the mail Service Provider Interface (MSPI), use the **debug mspi receive** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug mspi receive

no debug mspi receive

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(3)XI	This command was introduced on the Cisco AS5300 universal access server.
12.1(5)T	This command was integrated into Cisco IOS Release 12.1(5)T.
12.2(4)T	This command was implemented on the Cisco 1750 access router.

Examples

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

```
Router# debug mspi receive

Jan  1 05:09:33.890: mspi_tel_num_trans: from: Radhika,
ph#in: fax=5271714 ph#dial: 5271714
Jan  1 05:09:33.890:  incoming destPat(5271714), matched(7), tag(22)
Jan  1 05:09:33.890:  out destPat(5.....), tag(20), dgt strip enabled
Jan  1 05:09:33.890: mspi_off_new_rcpt: envlp_to [fax=5271714@smith.abccompany.com], 30
Jan  1 05:09:33.890:  tel_numb_dial: 5271714, subaddr:[], cover page
Jan  1 05:09:39.122: mspi_offramp_rfc822_header: msgType=0
Jan  1 05:09:39.122:  envlp_from: [Radhika], 8
Jan  1 05:09:39.122: mspi_off_put_buff: ignore mime type=1, st=CONNECTING, len=0
Jan  1 05:09:39.122: moff_save_buffer: cid=0x1F, mime=9, len=4
Jan  1 05:09:39.122:  offramp disabled receiving!
Dec 31 21:09:44.078: %ISDN-6-CONNECT: Interface Serial0:22 is now connected to 5271714
Jan  1 05:09:52.154: mspi_bridge: cid=0x1F, dst cid=0x22, data dir=OFFRAMP, conf dir=DEST
Jan  1 05:09:52.154: mspi_offramp_send_buffer: cid=0x1F, mime=9
Jan  1 05:09:52.154:  buffer with only CR/LF - set buff_len=0
Jan  1 05:09:52.154: mspi_offramp_send_buffer: cid=0x1F, mime=9 rx BUFF_END_OF_PART,
offramp rcpt enabled
Jan  1 05:09:54.126: mspi_offramp_send_buffer: cid=0x1F, mime=11
Jan  1 05:09:54.134: mspi_offramp_send_buffer: cid=0x1F, mime=11
```

Related Commands

Command	Description
debug mspi send	Displays debug messages for MSPI send.

debug mspi send

To display debug messages for the sending mail Service Provider Interface (MSPI), use the **debug mspi send** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug mspi send

no debug mspi send

Syntax Description This command has no arguments or keywords.

Defaults No default behavior or values.

Command Modes Privileged EXEC

Command History

Release	Modification
12.1(3)XI	This command was introduced on the Cisco AS5300 universal access server.
12.1(5)T	This command was integrated into Cisco IOS Release 12.1(5)T.
12.2(4)T	This command was introduced on the Cisco 1750 access router.

Examples

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

```
Router# debug mspi send

*Oct 16 08:40:27.515: mspi_bridge: cid=0x21, dst cid=0x26, data dir=OFFRAMP, conf
dir=DEST
*Oct 16 08:40:29.143: mspi_setup_req: for cid=0x27
*Oct 16 08:40:29.147:   envelope_from=5??????@fax.cisco.com
*Oct 16 08:40:29.147:   envelope_to=ilyau@cisco.com
*Oct 16 08:40:30.147: mspi_chk_connect: cid=0x27, cnt=0,
*Oct 16 08:40:30.147: SMTP connected to the server !
*Oct 16 08:40:30.147: mspi_bridge: cid=0x27, dst cid=0x28, data dir=ONRAMP, conf dir=SRC
*Oct 16 08:40:38.995: mspi_xmit: cid=0x27, st=CONFERENCED, src_cid=0x28, buf cnt=0
```

Related Commands

Command	Description
debug mspi receive	Displays debug messages for mail SPI receive.

debug mta receive all

To show output relating to the activity on the Simple Mail Transfer Protocol (SMTP) server, use the **debug mta receive all** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug mta receive all

no debug mta receive all

Syntax Description

This command has no arguments or keywords.

Defaults

This command has no default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(4)T	This command was introduced.
12.1(5)T	This command was integrated into Cisco IOS Release 12.1(5)T.
12.2(4)T	This command was implemented on the Cisco 1750 access router.
12.2(8)T	This command was implemented on the Cisco 1751 access routers, Cisco 3725 access routers, and Cisco 3745 access routers.
12.2(13)T	This feature was introduced on the Cisco 7200 series routers.

Examples

The following example shows the messages exchanged (for example, the handshake) between the e-mail server and the off-ramp gateway:

```
Router# debug mta receive all

Jan  1 05:07:41.314: esmtp_server_work: calling helo
Jan  1 05:07:43.354: esmtp_server_work: calling mail
Jan  1 05:07:45.386: esmtp_server_work: calling rcpt
Jan  1 05:07:47.426: esmtp_server_work: calling data
Jan  1 05:07:49.514: (S)R: 'Content-Type: multipart/mixed;
boundary="-----11F7CD9D2EB3E8B8D5627C62"'
Jan  1 05:07:49.514: (S)R: ''
Jan  1 05:07:49.514: esmtp_server_engine_new_part:
Jan  1 05:07:49.514: (S)R: 'Content-Type: text/plain; charset=us-ascii'
Jan  1 05:07:49.514: (S)R: 'Content-Transfer-Encoding: 7bit'
Jan  1 05:07:49.514: (S)R: ''
Jan  1 05:07:49.514: esmtp_server_engine_new_part:
Jan  1 05:07:49.514: esmtp_server_work: freeing temp header
Jan  1 05:07:49.514: (S)R: 'Content-Type: image/tiff; name="DevTest.8.1610.tif"'
Jan  1 05:07:49.514: (S)R: 'Content-Transfer-Encoding: base64'
Jan  1 05:07:49.514: (S)R: 'Content-Disposition: inline; filename="DevTest.8.1610.tif"'
Jan  1 05:07:49.514: (S)R: ''
Jan  1 05:07:49.514: esmtp_server_engine_update_recipient_status: status=6
Jan  1 05:07:49.514: esmtp_server_engine_new_part:
```

debug mta receive all

```
Jan 1 05:07:49.518: esmtp_server_work: freeing temp header
Jan 1 05:08:03.014: esmtp_server_engine_update_recipient_status: status=7
Jan 1 05:08:04.822: esmtp_server_engine_update_recipient_status: status=6
Jan 1 05:08:33.042: esmtp_server_engine_update_recipient_status: status=7
Jan 1 05:08:34.906: esmtp_server_engine_getline: Unexpected end of file on socket 1
Jan 1 05:08:34.906: esmtp_server_work: error occurred with ctx=0x61FFF710, socket=1
```

Related Commands

Command	Description
debug mta send all	Displays output for all the on-ramp client connections.

debug mta send all

To display output for all of the on-ramp client connections, use the **debug mta send all** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug mta send all

no debug mta send all

Syntax Description

This command has no arguments or keywords.

Defaults

This command has no default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(4)T	This command was introduced.
12.1(5)T	This command was integrated into Cisco IOS Release 12.1(5)T.
12.2(4)T	This command was implemented on the Cisco 1750 access router.
12.2(8)T	This command was implemented on the Cisco 1751 access routers, Cisco 3725 access routers, and Cisco 3745 access routers.
12.2(13)T	This feature was introduced on the Cisco 7200 series routers.

Examples

The following example shows the messages exchanged (for example, the handshake) between the e-mail server and the on-ramp gateway:

```
Router# debug mta send all

*Oct 16 09:04:13.055: esmtp_client_engine_open: from=5551212@fax.cisco.com,
to=madeup@abccompany.com
*Oct 16 09:04:13.055: esmtp_client_engine_add_headers: from_comment=
*Oct 16 09:04:13.111: esmtp_client_work: socket 0 attempting to connect to IP address
171.71.154.56
*Oct 16 09:04:13.111: esmtp_client_work: socket 0 readable for first time
*Oct 16 09:04:13.135: esmtp_client_work: socket 0 readable for first time
*Oct 16 09:04:13.135: (C)R: 220 madeup.abccompany.com ESMTP Sendmail 8.8.4-Cisco.1/8.6.5
ready at Wed, 27 Sep 2000 11:45:46 -0700 (PDT)
*Oct 16 09:04:13.135: (C)S: EHLO mmoip-c.cisco.com
*Oct 16 09:04:13.183: (C)R: 250-madeup.abccompany.com Hello [172.22.95.16], pleased to
meet you
*Oct 16 09:04:13.183: (C)R: 250-EXPN
*Oct 16 09:04:13.183: (C)R: 250-VERB
```

Related Commands

Command	Description
debug mta receive all	Displays output for all the off-ramp client connections.
debug mta send rcpt-to	Displays output for a specific on-ramp SMTP client connection during an e-mail transmission.

debug mta send rcpt-to

To display output for a specific on-ramp Simple Mail Transfer Protocol (SMTP) client connection during an e-mail transmission, use the **debug mta send rcpt-to** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug mta send rcpt-to *string*

no debug mta send rcpt-to *string*

Syntax Description

string E-mail address.

Defaults

This command has no default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(4)T	This command was introduced.
12.1(5)T	This command was integrated into Cisco IOS Release 12.1(5)T.
12.2(4)T	This command was implemented on the Cisco 1750 access router.
12.2(8)T	This command was implemented on the Cisco 1751 access routers, Cisco 3725 access routers, and Cisco 3745 access routers.
12.2(13)T	This feature was introduced on the Cisco 7200 series routers.

Examples

The following example shows debugging information displayed when the **debug mta send rcpt-to** command has been enabled and the SMTP client is sending an e-mail message.

```
Router# debug mta send rcpt-to 5551212

Router# socket 0 attempting to connect to IP address 100.00.00.00
socket 0 readable for first time - let's try to read it
R:220 madeup.abc.com ESMTP Sendmail 8.8.4-abc.1/8.6.5 ready at Tue, 6
Apr 1999 13:35:39 -0700 (PDT)
S:EHLO mmoip-c.abc.com
R:250-quisp.cisco.com Hello [100.00.00.00], pleased to meet you
R:250-EXPN
R:250-VERB
R:250-8BITMIME
R:250-SIZE
R:250-DSN
R:250-ETRN
R:250-XUSR
R:250 HELP
S:MAIL FROM:<testing@> RET=HDRS
R:250 <testing@>... Sender ok
S:RCPT TO:<madeup@abc.com> NOTIFY=SUCCESS ORCPT=rfc822;testing@
R:250 <madeup@abc.com>... Recipient ok
R:354 Enter mail, end with "." on a line by itself
```

```

S:Received:(Cisco Powered Fax System) by mmoip-c.cisco.com for
<madeup@abc.com> (with Cisco NetWorks); Fri, 17 Oct 1997 14:54:27 +0800
S:To: <madeup@abc.com>
S:Message-ID:<000F1997145427146@mmoip-c.cisco.com>
S>Date:Fri, 17 Oct 1997 14:54:27 +0800
S:Subject:mmoip-c subject here
S:X-Mailer:IOS (tm) 5300 Software (C5300-IS-M)
S:MIME-Version:1.0
S:Content-Type:multipart/mixed;
S: boundary="yradnuoB=_000E1997145426826.mmoip-ccisco.com"
S:From:"Test User" <testing@>
S:--yradnuoB=_000E1997145426826.mmoip-ccisco.com
S:Content-ID:<00101997145427150@mmoip-c.cisco.com>
S:--yradnuoB=_000E1997145426826.mmoip-ccisco.com--
Sending terminating dot ...(socket=0)
S:.
R:250 NAA09092 Message accepted for delivery
S:QUIT
R:221 madeup@abc.com closing connection
Freeing SMTP ctx at 0x6121D454
returned from work_routine, context freed

```

Related Commands

Command	Description
debug mta send all	Displays output for all the on-ramp client connections.

debug mwi relay errors

To debug message waiting indication (MWI) relay errors, use the **debug mwi relay errors** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug mwi relay errors

no debug mwi relay errors

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(2)XT	This command was introduced on the following platforms: Cisco 1750, Cisco 1751, Cisco 2600 series and Cisco 3600 series multiservice routers; and Cisco IAD2420 series Integrated Access Devices (IADs).
12.2(8)T	This command was integrated into Cisco IOS Release 12.2(8)T and implemented on the Cisco 3725 and Cisco 3745 routers.
12.2(8)T1	This command was implemented on the Cisco 2600-XM and Cisco 2691 routers.
12.2(11)T	This command was integrated into Cisco IOS Release 12.2(11)T and implemented on the Cisco 1760 routers.

Usage Guidelines

The **debug mwi relay errors** command provides a debug monitor display of any error messages, when MWI Relay Server (Cisco IOS Telephony Server) is trying to do MWI Relay to extensions on remote ITS.

Examples

The following examples show errors when MWI Relay Server tries to do an MWI Relay to extension 7004, but location of 7004 is not known to the MWI Relay Server:

```
Router# debug mwi relay errors

mwi-relay error info debugging is on
01:46:48: MWI-APP: mwi_notify_status: No ClientID (7004) registered
```

Related Commands

Command	Description
debug mwi relay events	Sets MWI relay events debugging for the Cisco IOS Telephony Service router.
debug ephone mwi	Sets MWI debugging for the Cisco IOS Telephony Service router.

debug mwi relay events

To set message waiting indication (MWI) relay events debugging, use the **debug mwi relay events** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug mwi relay events

no debug mwi relay events

Syntax Description This command has no arguments or keywords.

Defaults No default behavior or values.

Command Modes Privileged EXEC

Command History	Release	Modification
	12.2(2)XT	This command was introduced on the following platforms: Cisco 1750, Cisco 1751, Cisco 2600 series and Cisco 3600 series multiservice routers; and Cisco IAD2420 series Integrated Access Devices (IADs).
	12.2(8)T	This command was integrated into Cisco IOS Release 12.2(8)T and implemented on the Cisco 3725 and Cisco 3745 routers.
	12.2(8)T1	This command was implemented on the Cisco 2600-XM and Cisco 2691 routers.
	12.2(11)T	This command was integrated into Cisco IOS Release 12.2(11)T and implemented on the Cisco 1760 routers.

Usage Guidelines The **debug mwi relay events** command provides a debug monitor display of events, when MWI Relay Server (Cisco IOS Telephony Server) is trying to do MWI Relay to extensions on remote Cisco IOS Telephony Services (ITS).

Examples The following debug messages are shown when the MWI Relay server tries to send MWI Information to remote client 7001 and the location of 7001 is known by the MWI Relay Server:

```
Router# debug mwi relay events
mwi-relay events info debugging is on

01:45:34: mwi_notify_status: Queued event for mwi_app_queue
01:45:34: MWI-APP: mwi_app_process_event:
01:45:34: MWI-APP: mwi_app_process_event: MWI Event for ClientID(7001)@(1.8.17.22)
```

Related Commands	Command	Description
	debug mwi relay errors	Sets MWI relay errors debugging for the Cisco IOS Telephony Service router.
	debug ephone mwi	Sets MWI debugging for the Cisco IOS Telephony Service router.

debug ncia circuit

To display circuit-related information between the native client interface architecture (NCIA) server and client, use the **debug ncia circuit** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ncia circuit [**error** | **event** | **flow-control** | **state**]

no debug ncia circuit [**error** | **event** | **flow-control** | **state**]

Syntax Description

error	(Optional) Displays the error situation for each circuit.
event	(Optional) Displays the packets received and sent for each circuit.
flow-control	(Optional) Displays the flow control information for each circuit.
state	(Optional) Displays the state changes for each circuit.

Usage Guidelines

NCIA is an architecture developed by Cisco for accessing SNA applications. This architecture allows native SNA interfaces on hosts and clients to access TCP/IP backbones.

You cannot enable debugging output for a particular client or particular circuit.



Caution

Do not enable the **debug ncia circuit** command during normal operation because this command generates a substantial amount of output messages and could slow down the router.

Examples

The following is sample output from the **debug ncia circuit error** command. In this example, the possible errors are displayed. The first error message indicates that the router is out of memory. The second message indicates that the router has an invalid circuit control block. The third message indicates that the router is out of memory. The remaining messages identify errors related to the finite state machine.

```
Router# debug ncia circuit error

NCIA: ncia_circuit_create memory allocation fail
NCIA: ncia_send_ndlc: invalid circuit control block
NCIA: send_ndlc: fail to get buffer for ndlc primitive xxx
NCIA: ncia circuit fsm: Invalid input
NCIA: ncia circuit fsm: Illegal state
NCIA: ncia circuit fsm: Illegal input
NCIA: ncia circuit fsm: Unexpected input
NCIA: ncia circuit fsm: Unknown error rtn code
```

The following is sample output from the **debug ncia circuit event** command. In this example, a session start-up sequence is displayed.

```
Router# debug ncia circuit event

NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_START_DL, Len: 24, tmac: 4000.1060.1000,
         tsap: 4, csap 8, oid: 8A91E8, tid 0, lfs 16, ws 1
NCIA: create circuit: saddr 4000.1060.1000, ssap 4, daddr 4000.3000.0003, dsap 8 sid:
         8B09A8
NCIA: send NDLC_DL_STARTED to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_DL_STARTED, Len: 2,4 tmac: 4000.1060.1000,
```

```

      tsap: 4, csap 8, oid: 8A91E8, tid 8B09A8, lfs 16, ws 1
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 12, sid: 8B09A8, FC 0xC1
NCIA: send NDLC_XID_FRAME to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 12, sid: 8A91E8, FC 0xC1
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 18, sid: 8B09A8, FC 0xC1
NCIA: send NDLC_CONTACT_STN to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_CONTACT_STN, Len: 12, sid: 8A91E8, FC 0xC1
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_STN_CONTACTED, Len: 12, sid: 8B09A8, FC 0xC1
NCIA: send NDLC_INFO_FRAME to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_INFO_FRAME, Len: 30, sid: 8A91E8, FC 0xC1

```

Table 184 describes the significant fields in the output.

Table 184 debug ncia circuit event Field Descriptions

Field	Description
IN	Incoming message from client.
OUT	Outgoing message to client.
Ver_Id	NDLC version ID.
MsgType	NDLC message type.
Len	NDLC message length.
tmac	Target MAC.
tsap	Target SAP.
csap	Client SAP.
oid	Origin ID.
tid	Target ID.
lfs	Largest frame size flag.
ws	Window size.
saddr	Source MAC address.
ssap	Source SAP.
daddr	Destination MAC address.
dsap	Destination SAP.
sid	Session ID.
FC	Flow control flag.

In the following messages, an NDLC_START_DL messages is received from a client. to start a data-link session:

```

NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_START_DL, Len: 24, tmac: 4000.1060.1000,
      tsap: 4, csap 8, oid: 8A91E8, tid 0, lfs 16, ws 1
NCIA: create circuit: saddr 4000.1060.1000, ssap 4, daddr 4000.3000.0003, dsap 8 sid:
      8B09A8

```

The next two messages indicate that an NDLC_DL_STARTED message is sent to a client. The server informs the client that a data-the link session is started.

```
NCIA: send NDLC_DL_STARTED to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_DL_STARTED, Len: 2,4 tmac: 4000.1060.1000,
          tsap: 4, csap 8, oid: 8A91E8, tid 8B09A8, lfs 16, ws 1
```

In the following two messages, an NDLC_XID_FRAME message is received from a client, and the client starts an XID exchange:

```
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 12, sid: 8B09A8, FC 0x81
NCIA: send NDLC_XID_FRAME to client 10.2.20.3 for ckt: 8B09A8
```

In the following two messages, an NDLC_XID_FRAME message is sent from a client, and an NDLC_XID_FRAME message is received from a client:

```
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 12, sid: 8A91E8, FC 0xC1
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 18, sid: 8B09A8, FC 0xC1
```

The next two messages show that an NDLC_CONTACT_STN message is sent to a client:

```
NCIA: send NDLC_CONTACT_STN to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_CONTACT_STN, Len: 12, sid: 8A91E8, FC 0xC1
```

In the following message, an NDLC_STN_CONTACTED message is received from a client. The client informs the server that the station has been contacted.

```
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_STN_CONTACTED, Len: 12, sid: 8B09A8, FC 0xC1
```

In the last two messages, an NDLC_INFO_FRAME is sent to a client, and the server sends data to the client:

```
NCIA: send NDLC_INFO_FRAME to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_INFO_FRAME, Len: 30, sid: 8A91E8, FC 0xC1
```

The following is sample output from the **debug ncia circuit flow-control** command. In this example, the flow control in a session startup sequence is displayed:

```
Router# debug ncia circuit flow-control

NCIA: no flow control in NDLC_DL_STARTED frame
NCIA: receive Increment Window Op for circuit 8ADE00
NCIA: ncia_flow_control_in FC 0x81, IW 1 GP 2 CW 2, Client IW 1 GP 0 CW 1
NCIA: grant client more packet by sending Repeat Window Op
NCIA: ncia_flow_control_out FC: 0xC1, IW 1 GP 2 CW 2, Client IW 1 GP 2 CW 2
NCIA: receive FCA for circuit 8ADE00
NCIA: receive Increment Window Op for circuit 8ADE00
NCIA: ncia_flow_control_in FC 0xC1, IW 1 GP 5 CW 3, Client IW 1 GP 2 CW 2
NCIA: grant client more packet by sending Repeat Window Op
NCIA: ncia_flow_control_out FC: 0xC1, IW 1 GP 5 CW 3, Client IW 1 GP 5 CW 3
NCIA: receive FCA for circuit 8ADE00
NCIA: receive Increment Window Op for circuit 8ADE00
NCIA: ncia_flow_control_in FC 0xC1, IW 1 GP 9 CW 4, Client IW 1 GP 5 CW 3
NCIA: grant client more packet by sending Repeat Window Op
NCIA: ncia_flow_control_out FC: 0xC1, IW 1 GP 8 CW 4, Client IW 1 GP 9 CW 4
NCIA: reduce ClientGrantPacket by 1 (Granted: 8)
NCIA: receive FCA for circuit 8ADE00
NCIA: receive Increment Window Op for circuit 8ADE00
```

Table 185 describes the significant fields shown in the display.

Table 185 *debug ncia circuit flow-control Field Descriptions*

Field	Description
IW	Initial window size.
GP	Granted packet number.
CW	Current window size.

The following is sample output from the **debug ncia circuit state** command. In this example, a session startup sequence is displayed:

```
Router# debug ncia circuit state

NCIA: pre-server fsm: event CONN_OPENED
NCIA: pre-server fsm: event NDLC_PRIMITIVES
NCIA: server event: WAN - STDL state: CLSOED
NCIA: ncia server fsm action 32
NCIA: circuit state: CLOSED -> START_DL_RCVD
NCIA: server event: DLU - TestStn.Rsp state: START_DL_RCVD
NCIA: ncia server fsm action 17
NCIA: circuit state: START_DL_RCVD -> DL_STARTED_SND
NCIA: pre-server fsm: event NDLC_PRIMITIVES
NCIA: server event: WAN - XID state: DL_STARTED_SND
NCIA: ncia server fsm action 33
NCIA: circuit state: DL_STARTED_SND -> DL_STARTED_SND
NCIA: server event: DLU - ReqOpnStn.Req state: DL_STARTED_SND
NCIA: ncia server fsm action 33
NCIA: circuit state: DL_STARTED_SND -> OPENED
NCIA: server event: DLU - Id.Rsp state: OPENED
NCIA: ncia server fsm action 11
NCIA: circuit state: OPENED -> OPENED
NCIA: pre-server fsm: event NDLC_PRIMITIVES
NCIA: server event: WAN - XID state: OPENED
NCIA: ncia server fsm action 33
NCIA: circuit state: OPENED -> OPENED
NCIA: server event: DLU - Connect.Req state: OPENED
NCIA: ncia server fsm action 6
NCIA: circuit state: OPENED -> CONNECT_PENDING
NCIA: pre-server fsm: event NDLC_PRIMITIVES
NCIA: server event: WAN - CONR state: CONNECT_PENDING
NCIA: ncia server fsm action 33 --> CLS_CONNECT_CNF sets NciaClsBusy
NCIA: circuit state: CONNECT_PENDING -> CONNECTED
NCIA: server event: DLU - Flow.Req (START) state: CONNECTED
NCIA: ncia server fsm action 25 --> unset NciaClsBusy
NCIA: circuit state: CONNECTED -> CONNECTED
NCIA: server event: DLU - Data.Rsp state: CONNECTED
NCIA: ncia server fsm action 8
NCIA: circuit state: CONNECTED -> CONNECTED
```

Table 186 describes the significant fields shown in the display.

Table 186 *debug ncia circuit state Field Descriptions*

Field	Description
WAN	Event from WAN (client).
DLU	Event from upstream module—dependent logical unit (DLU).
ADMIN	Administrative event.
TIMER	Timer event.

Related Commands

Command	Description
debug dmsp fax-to-doc	Enables debugging of DLSw+.
debug ncia client	Displays debug information for all NCIA client processing that occurs in the router.
debug ncia server	Displays debug information for the NCIA server and its upstream software modules.

debug ncia client

To display debug information for all native client interface architecture (NCIA) client processing that occurs in the router, use the **debug ncia client** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ncia client [*ip-address* | **error** [*ip-address*] | **event** [*ip-address*] | **message** [*ip-address*]]

no debug ncia client [*ip-address* | **error** [*ip-address*] | **event** [*ip-address*] | **message** [*ip-address*]]

Syntax Description

<i>ip-address</i>	(Optional) The remote client IP address.
error	(Optional) Triggers the recording of messages only when errors occur. The current state and event of an NCIA client are normally included in the message. If you do not specify an IP address, the error messages are logged for all active clients.
event	(Optional) Triggers the recording of messages that describe the current state and event—and sometimes the action that just completed—for the NCIA client. If you do not specify an IP address, the messages are logged for all active clients.
message	(Optional) Triggers the recording of messages that contain up to the first 32 bytes of data in a TCP packet sent to or received from an NCIA client. If you do not specify an IP address, the messages are logged for all active clients.

Command Modes

Privileged EXEC

Usage Guidelines

NCIA is an architecture developed by Cisco for accessing SNA applications. This architecture allows native SNA interfaces on hosts and clients to access TCP/IP backbones.

Use the **debug ncia client error** command to see only certain error conditions that occur.

Use the **debug ncia client event** command to determine the sequences of activities that occur while a NCIA client is in different processing states.

Use the **debug ncia client message** command to see only the first 32 bytes of data in a TCP packet sent to or received from an NCIA client.

The **debug ncia client** command can be used in conjunction with the **debug ncia server** and **debug ncia circuit** commands to get a complete picture of NCIA activity.

Examples

The following is sample output from the **debug ncia circuit** command. Following the example is a description of each sample output message.

```
Router# debug ncia client
```

```
NCIA: Passive open 10.2.20.123(1088) -> 1973
NCIA: index for client hash queue is 27
NCIA: number of element in client hash queue 27 is 1
NCIA: event PASSIVE_OPEN, state NCIA_CLOSED for client 10.2.20.123
NCIA: Rcvd msg type NDLC_CAP_XCHG in tcp packet for client 10.2.20.123
NCIA: First 17 byte of data rcvd: 8112001100000000000000400050104080C
```

```

NCIA: Sent msg type NDLC_CAP_XCHG in tcp packet to client 10.2.20.123
NCIA: First 17 byte of data sent: 811200111000000010000400050104080C
NCIA: event CAP_CMD_RCVD, state NCIA_CAP_WAIT, for client 10.2.20.123, cap xchg cmd sent
NCIA: Rcvd msg type NDLC_CAP_XCHG in tcp packet for client 10.2.20.123
NCIA: First 17 byte of data rcvd: 811200111000000010000000050104080C
NCIA: event CAP_RSP_RCVD, state NCIA_CAP_NEG for client 10.2.20.123

NCIA: Rcvd msg type NDLC_PEER_TEST_REQ in tcp packet for client 10.2.20.123
NCIA: First 4 byte of data rcvd: 811D0004
NCIA: event KEEPALIVE_RCVD, state NCIA_OPENED for client 10.2.20.123
NCIA: Sent msg type NDLC_PEER_TEST_RSP in tcp packet to client 10.2.20.123
NCIA: First 4 byte of data sent: 811E0004IA

NCIA: event TIME_OUT, state NCIA_OPENED, for client 10.2.20.123, keepalive_count = 0
NCIA: Sent msg type NDLC_PEER_TEST_REQ, in tcp packet to client 10.2.20.123
NCIA: First 4 byte of data sent: 811D0004
NCIA: Rcvd msg type NDLC_PEER_TEST_RSP in tcp packet for client 10.2.20.123
NCIA: First 4 byte of data rcvd: 811E0004
NCIA: event KEEPALIVE_RSP_RCVD, state NCIA_OPENED for client 10.2.20.123

NCIA: Error, event PASIVE_OPEN, state NCIA_OPENED, for client 10.2.20.123, should not have
occurred.
NCIA: Error, active_open for pre_client_fsm while client 10.2.20.123 is active or not
configured, registered.

```

Messages in lines 1 through 12 show the events that occur when a client connects to the router (the NCIA server). These messages show a passive_open process.

Messages in lines 13 to 17 show the events that occur when a TIME_OUT event is detected by a client PC workstation. The workstation sends an NDLC_PEER_TEST_REQ message to the NCIA server, and the router responds with an NDLC_PEER_TEST_RSP message.

Messages in lines 18 to 23 show the events that occur when a TIME_OUT event is detected by the router (the NCIA server). The router sends an NDLC_PEER_TEST_REQ message to the client PC workstation, and the PC responds with an NDLC_PEER_TEST_RSP message.

When you use the **debug ncia client message** command, the messages shown on lines 6, 8, 11, 14, 17, 20, and 22 are output in addition to other messages not shown in this example.

When you use the **debug ncia client error** command, the messages shown on lines 24 and 25 are output in addition to other messages not shown in this example.

Related Commands

Command	Description
debug ncia circuit	Displays debug information for all NCIA client processing that occurs in the router.
debug ncia server	Displays debug information for the NCIA server and its upstream software modules.

debug ncia server

To display debug information for the native client interface architecture (NCIA) server and its upstream software modules, use the **debug ncia server** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ncia server

no debug ncia server

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

Usage Guidelines NCIA is an architecture developed by Cisco for accessing SNA applications. This architecture allows native SNA interfaces on hosts and clients to access TCP/IP backbones.

The **debug ncia server** command displays all Cisco Link Services (CLS) messages between the NCIA server and its upstream modules, such as data-link switching (DLSw) and downstream physical units (DSPUs). Use this command when a problem exists between the NCIA server and other software modules within the router.

You cannot enable debugging output for a particular client or particular circuit.

Examples The following is sample output from the **debug ncia server** command. In this example, a session startup sequence is displayed. Following the example is a description of each group of sample output messages.

```
Router# debug ncia server

NCIA: send CLS_TEST_STN_IND to DLU
NCIA: Receive TestStn.Rsp
NCIA: send CLS_ID_STN_IND to DLU
NCIA: Receive ReqOpnStn.Req
NCIA: send CLS_REQ_OPNSTN_CNF to DLU
NCIA: Receive Id.Rsp
NCIA: send CLS_ID_IND to DLU
NCIA: Receive Connect.Req
NCIA: send CLS_CONNECT_CNF to DLU
NCIA: Receive Flow.Req
NCIA: Receive Data.Req
NCIA: send CLS_DATA_IND to DLU
NCIA: send CLS_DISC_IND to DLU
NCIA: Receive Disconnect.Rsp
```

In the following messages, the client is sending a test message to the host and the test message is received by the host:

```
NCIA: send CLS_TEST_STN_IND to DLU
NCIA: Receive TestStn.Rsp
```

In the next message, the server is sending an XID message to the host:

```
NCIA: send CLS_ID_STN_IND to DLU
```

In the next two messages, the host opens the station and the server responds:

```
NCIA: Receive ReqOpnStn.Reg
NCIA: send CLS_REQ_OPNSTN_CNF to DLU
```

In the following two messages, the client is performing an XID exchange with the host:

```
NCIA: Receive Id.Rsp
NCIA: send CLS_ID_IND to DLU
```

In the next group of messages, the host attempts to establish a session with the client:

```
NCIA: Receive Connect.Reg
NCIA: send CLS_CONNECT_CNF to DLU
NCIA: Receive Flow.Reg
```

In the next two messages, the host sends data to the client:

```
NCIA: Receive Data.Reg
NCIA: send CLS_DATA_IND to DLU
```

In the last two messages, the client closes the session:

```
NCIA: send CLS_DISC_IND to DLU
NCIA: Receive Disconnect.Rsp
```

Related Commands

Command	Description
<code>debug dmosp fax-to-doc</code>	Enables debugging of DLSw+.
<code>debug ncia circuit</code>	Displays circuit-related information between the NCIA server and client.
<code>debug ncia client</code>	Displays debug information for all NCIA client processing that occurs in the router.

debug netbios error

To display information about Network Basic Input/Output System (NetBIOS) protocol errors, use the **debug netbios error** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug netbios error

no debug netbios error

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

Usage Guidelines For complete information on the NetBIOS process, use the **debug netbios packet** command along with the **debug netbios error** command.

Examples The following is sample output from the **debug netbios error** command. This example shows that an illegal packet has been received on the asynchronous interface.

```
Router# debug netbios error
```

```
Asyncl nbf Bad packet
```

Related Commands	Command	Description
	debug netbios-name-cache	Displays name caching activities on a router.
	debug netbios packet	Displays general information about NetBIOS packets.

debug netbios-name-cache

To display name caching activities on a router, use the **debug netbios-name-cache** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command. **debug netbios-name-cache**

no debug netbios-name-cache

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

Examine the display to diagnose problems in NetBIOS name caching.

Examples

The following is sample output from the **debug netbios-name-cache** command:

```
Router# debug netbios-name-cache

NETBIOS: L checking name ORINDA, vrn=0
NETBIOS name cache table corrupted at offset 13
NETBIOS name cache table corrupted at later offset, at location 13
NETBIOS: U chk name=ORINDA, addr=1000.4444.5555, idb=TR1, vrn=0, type=1
NETBIOS: U upd name=ORINDA, addr=1000.4444.5555, idb=TR1, vrn=0, type=1
NETBIOS: U add name=ORINDA, addr=1000.4444.5555, idb=TR1, vrn=0, type=1
NETBIOS: U no memory to add cache entry. name=ORINDA, addr=1000.4444.5555
NETBIOS: Invalid structure detected in netbios_name_cache_ager
NETBIOS: flushed name=ORINDA, addr=1000.4444.5555
NETBIOS: expired name=ORINDA, addr=1000.4444.5555
NETBIOS: removing entry. name=ORINDA, addr=1000.4444.5555, idb=TR1, vrn=0
NETBIOS: Tossing ADD_NAME/STATUS/NAME/ADD_GROUP frame
NETBIOS: Lookup Failed -- not in cache
NETBIOS: Lookup Worked, but split horizon failed
NETBIOS: Could not find RIF entry
NETBIOS: Cannot duplicate packet in netbios_name_cache_proxy
```



Note

The sample display is a composite output. Debugging output that you actually see would not necessarily occur in this sequence.

[Table 187](#) describes the significant fields shown in the display.

Table 187 *debug netbios-name-cache Field Descriptions*

Field	Description
NETBIOS	NetBIOS name caching debugging output.
L, U	L means lookup; U means update.
addr=1000.4444.5555	MAC address of machine being looked up in NetBIOS name cache.
idb=TR1	Indicates that the name of machine was learned from Token Ring interface number 1; idb is into interface data block.

Table 187 *debug netbios-name-cache Field Descriptions (continued)*

Field	Description
vrn=0	Packet comes from virtual ring number 0. This packet actually comes from a real Token Ring interface, because virtual ring number 0 is not valid.
type=1	Indicates the way that the router learned about the specified machine. The possible values are as follows: <ul style="list-style-type: none"> • 1 - Learned from traffic • 2 - Learned from a remote peer • 4, 8 - Statically entered via the configuration of the router

With the first line of output, the router declares that it has examined the NetBIOS name cache table for the machine name ORINDA and that the packet that prompted the lookup came from virtual ring 0. In this case, this packet comes from a real interface—virtual ring number 0 is not valid.

```
NETBIOS: L checking name ORINDA, vrn=0
```

The following two lines indicate that an invalid NetBIOS entry exists and that the corrupted memory was detected. The invalid memory will be removed from the table; no action is needed.

```
NetBIOS name cache table corrupted at offset 13
NetBIOS name cache table corrupted at later offset, at location 13
```

The following line indicates that the router attempted to check the NetBIOS cache table for the name ORINDA with MAC address 1000.4444.5555. This name was obtained from Token Ring interface 1. The type field indicates that the name was learned from traffic.

```
NETBIOS: U chk name=ORINDA, addr=1000.4444.5555, idb=TR1, vrn=0, type=1
```

The following line indicates that the NetBIOS name ORINDA is in the name cache table and was updated to the current value:

```
NETBIOS: U upd name=ORINDA, addr=1000.4444.5555, idb=TR1, vrn=0, type=1
```

The following line indicates that the NetBIOS name ORINDA is not in the table and must be added to the table:

```
NETBIOS: U add name=ORINDA, addr=1000.4444.5555, idb=TR1, vrn=0, type=1
```

The following line indicates that there was insufficient cache buffer space when the router tried to add this name:

```
NETBIOS: U no memory to add cache entry. name=ORINDA, addr=1000.4444.5555
```

The following line indicates that the NetBIOS ager detects an invalid memory in the cache. The router clears the entry; no action is needed.

```
NETBIOS: Invalid structure detected in netbios_name_cache_ager
```

The following line indicates that the entry for ORINDA was flushed from the cache table:

```
NETBIOS: flushed name=ORINDA, addr=1000.4444.5555
```

The following line indicates that the entry for ORINDA timed out and was flushed from the cache table:

```
NETBIOS: expired name=ORINDA, addr=1000.4444.5555
```

The following line indicates that the router removed the ORINDA entry from its cache table:

```
NETBIOS: removing entry. name=ORINDA,addr=1000.4444.5555,idb=TR1,vrn=0
```

The following line indicates that the router discarded a NetBIOS packet of type ADD_NAME, STATUS, NAME_QUERY, or ADD_GROUP. These packets are discarded when multiple copies of one of these packet types are detected during a certain period of time.

```
NETBIOS: Tossing ADD_NAME/STATUS/NAME/ADD_GROUP frame
```

The following line indicates that the system could not find a NetBIOS name in the cache:

```
NETBIOS: Lookup Failed -- not in cache
```

The following line indicates that the system found the destination NetBIOS name in the cache, but located on the same ring from which the packet came. The router will drop this packet because the packet should not leave this ring.

```
NETBIOS: Lookup Worked, but split horizon failed
```

The following line indicates that the system found the NetBIOS name in the cache, but the router could not find the corresponding RIF. The packet will be sent as a broadcast frame.

```
NETBIOS: Could not find RIF entry
```

The following line indicates that no buffer was available to create a NetBIOS name cache proxy. A proxy will not be created for the packet, which will be forwarded as a broadcast frame.

```
NETBIOS: Cannot duplicate packet in netbios_name_cache_proxy
```

Related Commands

Command	Description
debug netbios error	Displays information about NetBIOS protocol errors.
debug netbios packet	Displays general information about NetBIOS packets.

debug netbios packet

To display general information about NetBIOS packets, use the **debug netbios packet** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug netbios packet

no debug netbios packet

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

Usage Guidelines For complete information on the NetBIOS process, use the **debug netbios error** command along with the **debug netbios packet** command.

Examples The following is sample output from the **debug netbios packet** and **debug netbios error** commands. This example shows the LLC header for an asynchronous interface followed by the NetBIOS information. For additional information on the NetBIOS fields, refer to *IBM LAN Technical Reference IEEE 802.2*.

```
Router# debug netbios packet

Asyncl (i) U-format UI C_R=0x0
(i) NETBIOS_ADD_NAME_QUERY
  Resp_correlator= 0x6F 0x0
  Src name=CS-NT-1

Asyncl (i) U-format UI C_R=0x0
(i) NETBIOS_ADD_GROUP_QUERY
  Resp_correlator= 0x6F 0x0
  Src name=COMMSERVER-WG

Asyncl (i) U-format UI C_R=0x0
(i) NETBIOS_ADD_NAME_QUERY
  Resp_correlator= 0x6F 0x0
  Src name=CS-NT-1

Ethernet0 (i) U-format UI C_R=0x0
(i) NETBIOS_DATAGRAM
  Length= 0x2C 0x0
  Dest name=COMMSERVER-WG
  Src name=CS-NT-3
```

Related Commands	Command	Description
	debug netbios error	Displays information about NetBIOS protocol errors.
	debug netbios-name-cache	Displays name caching activities on a router.

debug ntp

To display debug messages for Network Time Protocol (NTP) features, use the **debug ntp** command. To stop the output of ntp debugging messages, use the **no** form of this command.

```
debug ntp {adjust | authentication | events | loopfilter | packets | params | refclock | select | sync
| validity}
```

```
no debug ntp {adjust | authentication | events | loopfilter | packets | params | refclock | select |
sync | validity}
```

Syntax Description

adjust	Displays debugging information on NTP clock adjustments.
authentication	Displays debugging information on NTP authentication.
events	Displays debugging information on NTP events.
loopfilter	Displays debugging information on NTP loop filters.
packets	Displays debugging information on NTP packets.
params	Displays debugging information on NTP clock parameters.
refclock	Displays debugging information on NTP reference clocks.
select	Displays debugging information on NTP clock selection.
sync	Displays debugging information on NTP clock synchronization.
validity	Displays debugging information on NTP peer clock validity.

Defaults

Debug commands are disabled by default.

Command History

Release	Modification
12.0 T	This command was introduced in a release prior to Cisco IOS Release 12.1.

Related Commands

Command	Description
ntp refclock	Configures an external clock source for use with NTP services.

debug oam

To display operation and maintenance (OAM) events, use the **debug oam** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug oam

no debug oam

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Examples

The following is sample output from the **debug oam** command:

```
Router# debug oam

4/0(O) : VCD:0x0 DM:0x300 *OAM Cell* Length:0x39
0000 0300 0070 007A 0018 0100 0000 05FF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
FFFF FFFF FFFF FFFF FF6A 6A6A 6A6A 6A6A 6A6A 6A6A 6A6A 6A6A 6A00 0000
```

[Table 188](#) describes the significant fields in the display.

Table 188 *debug oam Field Descriptions*

Field	Description
0000	VCD Special OAM indicator.
0300	Descriptor MODE bits for the AIP.
0	GFC (4 bits).
07	VPI (8 bits).
0007	VCI (16 bits).
A	Payload type field (PTI) (4 bits).
00	Header Error Correction (8 bits).
1	OAM Fault mangement cell (4 bits).
8	OAM LOOPBACK indicator (4 bits).
01	Loopback indicator value, always 1 (8 bits).
00000005	Loopback unique ID, sequence number (32 bits).
FF6A	Fs and 6A required in the remaining cell, per UNI3.0.

debug packet

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

debug packet [**interface** *number* [**vcd** *vcd-number*] | **vc** *vpi/vci* | *vc-name*]

no debug packet [**interface** *number* [**vcd** *vcd-number*] | **vc** *vpi/vci* | *vc-name*]

Syntax Description

interface <i>number</i>	(Optional) interface or subinterface number.
vcd <i>vcd-number</i>	(Optional) Number of the virtual circuit designator (VCD).
vc <i>vpi/vci</i>	(Optional) VPI and VCI numbers of the VC.
<i>vc-name</i>	(Optional) Name of the PVC or SVC.

Defaults

Debugging for packets is disabled by default.

Command Modes

Privileged EXEC

Command Modes

Release	Modification
9.21	This command was introduced.
12.2(13)T	Support for Apollo Domain and Banyan VINES was removed.

Usage Guidelines

The **debug packet** command displays all process-level packets for both outbound and inbound packets. This command is useful for determining whether packets are being received and sent correctly. The output reports information online when a packet is received or a transmission is attempted.

For sent packets, the information is displayed only after the protocol data unit (PDU) is entirely encapsulated and a next hop VC is found. If information is not displayed, the address translation probably failed during encapsulation. When a next hop VC is found, the packet is displayed exactly as it will be presented on the wire. Having a display indicates that the packets are properly encapsulated for transmission.

For received packets, information is displayed for all incoming frames. The display can show whether the sending station properly encapsulates the frames. Because all incoming frames are displayed, this information is useful when performing back-to-back testing and corrupted frames cannot be dropped by an intermediary switch.

The **debug packet** command also displays the initial bytes of the actual PDU in hexadecimal. This information can be decoded only by qualified support or engineering personnel.



Caution

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

Examples

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

```
Router# debug packet
```

```
2/0.5(I): VCD:0x9 VCI:0x23 Type:0x0 SAP:AAAA CTL:03 OUI:000000 TYPE:0800 Length0x70
4500 002E 0000 0000 0209 92ED 836C A26E FFFF FFFF 1108 006D 0001 0000 0000
A5CC 6CA2 0000 000A 0000 6411 76FF 0100 6C08 00FF FFFF 0003 E805 DCFE 0105
```

[Table 189](#) describes the significant fields in the display.

Table 189 debug packet Field Descriptions

Field	Description
2/0.5	Indicates the subinterface that generated this packet.
(I)	Indicates a receive packet. (O) indicates an output packet.
VCD: 0xn	Indicates the virtual circuit associated with this packet, where <i>n</i> is some value.
DM: 0xnnnn	Indicates the descriptor mode bits on output only, where <i>nnnn</i> is a hexadecimal value.
TYPE:n	Displays the encapsulation type for this packet.
Length:n	Displays the total length of the packet including the headers.

The following two lines of output are the binary data, which are the contents of the protocol PDU before encapsulation:

```
4500 002E 0000 0000 0209 92ED 836C A26E FFFF FFFF 1108 006D 0001 0000 0000
A5CC 6CA2 0000 000A 0000 6411 76FF 0100 6C08 00FF FFFF 0003 E805 DCFE 0105
```

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

```
Router# debug packet
```

```
Ethernet0: Unknown ARPA, src 0000.0c00.6fa4, dst ffff.ffff.ffff, type 0x0a0
data 00000c00f23a00000c00ab45, len 60
Serial3: Unknown HDLC, size 64, type 0xaaaa, flags 0x0F00
Serial2: Unknown PPP, size 128
Serial7: Unknown FRAME-RELAY, size 174, type 0x5865, DLCI 7a
Serial0: compressed TCP/IP packet dropped
```

[Table 190](#) describes the significant fields shown in the display.

Table 190 debug packet Field Descriptions

Field	Description
Ethernet0	Name of the Ethernet interface that received the packet.
Unknown	Network could not classify this packet. Examples include packets with unknown link types.

Table 190 *debug packet Field Descriptions (continued)*

Field	Description
ARPA	<p>Packet uses ARPA-style encapsulation. Possible encapsulation styles vary depending on the media command mode (MCM) and encapsulation style.</p> <p>Ethernet (MCM)—Encapsulation Style:</p> <ul style="list-style-type: none">• ARP• ETHERTALK• ISO1• ISO3• LLC2• NOVELL-ETHER• SNAP
	<p>FDDI (MCM)—Encapsulation Style:</p> <ul style="list-style-type: none">• ISO1• ISO3• LLC2• SNAP <p>Frame Relay—Encapsulation Style:</p> <ul style="list-style-type: none">• BRIDGE• FRAME-RELAY

Table 190 debug packet Field Descriptions (continued)

Field	Description
	Serial (MCM)—Encapsulation Style: <ul style="list-style-type: none"> • BFEX25 • BRIDGE • DDN-X25 • DDNX25-DCE • ETHERTALK • FRAME-RELAY • HDLC • HDH • LAPB • LAPBDCE • MULTI-LAPB • PPP • SDLC-PRIMARY • SDLC-SECONDARY • SLIP • SMDS • STUN • X25 • X25-DCE
	Token Ring (MCM)—Encapsulation Style: <ul style="list-style-type: none"> • 3COM-TR • ISO1 • ISO3 • MAC • LLC2 • NOVELL-TR • SNAP • VINES-TR
src 0000.0c00.6fa4	MAC address of the node generating the packet.
dst.ffff.ffff.ffff	MAC address of the destination node for the packet.
type 0x0a0	Packet type.
data...	First 12 bytes of the datagram following the MAC header.
len 60	Length of the message (in bytes) that the interface received from the wire.
size 64	Length of the message (in bytes) that the interface received from the wire. Equivalent to the len field.

Table 190 *debug packet Field Descriptions (continued)*

Field	Description
flags 0x0F00	HDLC or PP flags field.
DLCI 7a	The DLCI number on Frame Relay.
compressed TCP/IP packet dropped	TCP header compression is enabled on an interface and the packet is not HDLC or X25.

debug pad

To display debug messages for all packet assembler/disassembler (PAD) connections, use the **debug pad** command in privileged EXEC mode. To disable PAD debugging, use the **no** form of this command.

debug pad

no debug pad

Syntax Description This command has no arguments or keywords.

Defaults No default behavior or values.

Command Modes Privileged EXEC

Command History	Release	Modification
	12.0	This command was introduced in a release prior to Cisco IOS Release 12.0.

Examples Use the **debug pad** command to gather information to forward to the Cisco Technical Assistance Center (TAC) to assist in troubleshooting a problem that involves packet assembler/disassembler (PAD) connections.

The following example shows output of the **debug pad** and **debug x25 event** commands for an incoming PAD call destined for a terminal line. The incoming PAD call is rejected by the terminal line because the selected network closed user group (CUG) has not been subscribed to by the caller:

```
Router# debug pad
Router# debug x25 event

Serial1/1:X.25 I R1 Call (16) 8 lci 8
  From (7):2001534 To (9):200261150
  Facilities:(2)
    Closed User Group (basic):99
  Call User Data (4):0x01000000 (pad)
pad_svc_announce:destination matched 1
PAD:incoming call to 200261150 on line 130 CUD length 4
!PAD130:Incoming Call packet, Closed User Group (CUG) service protection, selected network
CUG not subscribed
PAD:CUG service protection Cause:11 Diag:65
Serial1/1:X.25 O R1 Clear (5) 8 lci 8
  Cause 0, Diag 65 (DTE originated/Facility code not allowed)
Serial1/1:X.25 I R1 Clear Confirm (3) 8 lci 8
```

The following example shows the output of the **debug pad** command for an outgoing PAD call initiated from a terminal line with a subscribed CUG that bars outgoing access:

```
!PAD130:Outgoing Call packet, Closed User Group - CUG service validation, selected CUG
!bars outgoing access
PAD130:Closing connection to . In 0/0, out 0/0
```