



## debug Commands

---

- [debug arp, on page 2](#)
- [debug ble, on page 2](#)
- [debug capwap client, on page 3](#)
- [debug capwap client avc, on page 4](#)
- [debug cdp, on page 5](#)
- [debug cleanair, on page 5](#)
- [debug dhcp, on page 6](#)
- [debug dot11 driver level, on page 7](#)
- [debug dot11 client data-path, on page 7](#)
- [debug dot11 client management, on page 8](#)
- [debug dot11 client probe, on page 9](#)
- [debug dot11 driver slot, on page 9](#)
- [debug dot11 firmware , on page 10](#)
- [debug dot11 sensor, on page 11](#)
- [debug dtls client, on page 12](#)
- [debug ethernet, on page 12](#)
- [debug flexconnect, on page 13](#)
- [debug lldp, on page 14](#)
- [debug memory, on page 14](#)
- [debug memory pool, on page 15](#)
- [debug memory pool alloc, on page 15](#)
- [debug memory pool free, on page 16](#)
- [debug mesh, on page 17](#)
- [debug mesh adjacency, on page 17](#)
- [debug mesh path-control, on page 18](#)
- [debug rrm neighbor, on page 19](#)
- [debug rrm reports, on page 19](#)
- [debug sip, on page 20](#)
- [debug wips, on page 20](#)
- [debug process memory, on page 21](#)
- [debug traffic, on page 21](#)
- [debug tunnel, on page 22](#)
- [debug client trace, on page 22](#)

- [no](#), on page 23
- [traceroute](#), on page 24
- [undebug](#), on page 24

## debug arp

To enable debugging of ARP, use the **debug arp** command.

**debug arp** {**errors** | **events** | **packets**}

### Syntax Description

<b>errors</b>	Enable debugging of ARP errors
<b>events</b>	Enable debugging of ARP events
<b>packets</b>	Enable debugging of ARP Tx and Rx packets

### Command Modes

Privileged EXEC (#)

### Command History

Release	Modification
8.1.111.0	This command was introduced.

### Examples

The following example shows how to enable debugging of ARP errors:

```
cisco-ap# debug arp errors
```

## debug ble

To enable debugging of Bluetooth Low Energy (BLE), use the **debug ble** command.

**debug ble** {**critical** | **error** | **events** | **fastpath** {**rss** | **scan** | **sync**} | **receive** | **transmit**}

### Syntax Description

<b>critical</b>	Enables debugging of BLE critical events
<b>error</b>	Enables debugging of BLE error events
<b>events</b>	Enables debugging of BLE events
<b>fastpath</b> { <b>rss</b>   <b>scan</b>   <b>sync</b> }	Shows data exported to CMX. The following options are available: <ul style="list-style-type: none"> <li>• RSSI data</li> <li>• Scan data</li> <li>• Sync data</li> </ul>

<b>receive</b>	Enables debugging of BLE packet received from BLE radio
<b>transmit</b>	Enables debugging of BLE packet transmitted to BLE radio

**Command Modes**

Privileged EXEC (#)

**Command History**

Release	Modification
8.7	This command was introduced.

**Examples**

The following example shows how to enable debugging of BLE critical events:

```
cisco-ap# debug ble critical
```

## debug capwap client

To enable debugging of CAPWAP clients, use the **debug capwap client** command.

**debug capwap client** { **ble** | **detail** | **efficient-upgrade** | **error** | **events** | **flexconnect** | **info** | **keepalive** | **payload** | **pmtu** | **qos** | **reassemble** | **security** }

**Syntax Description**

<b>ble</b>	Enables debugging of CAPWAP BLE detail
<b>detail</b>	Enables debugging of CAPWAP detail
<b>efficient-upgrade</b>	Enables debugging of image predownload
<b>error</b>	Enables debugging of CAPWAP error
<b>events</b>	Enables debugging of CAPWAP events
<b>flexconnect</b>	Enables debugging of CAPWAP FlexConnect mode event
<b>info</b>	Enables debugging of CAPWAP information
<b>keepalive</b>	Enables debugging of CAPWAP keepalive
<b>payload</b>	Enables debugging of CAPWAP payload
<b>pmtu</b>	Enables debugging of CAPWAP path MTU
<b>qos</b>	Enables debugging of CAPWAP QoS
<b>reassemble</b>	Enables debugging of CAPWAP reassembly
<b>security</b>	Enables debugging of CAPWAP security

---

**Command Modes** Privileged EXEC (#)

---

Command History	Release	Modification
	8.1.111.0	This command was introduced.

---

### Examples

The following example shows how to enable debugging of CAPWAP client detail:

```
cisco-ap# debug capwap client detail
```

## debug capwap client avc

To enable debugging of CAPWAP client AVC, use the **debug capwap client avc** command.

**debug capwap client avc** {**all** | **detail** | **error** | **event** | **info** | **netflow** {**all** | **detail** | **error** | **event** | **packet**} | **numflows**}

Syntax Description		
<b>all</b>	Enables debugging of all CAPWAP client AVC	
<b>detail</b>	Enables debugging of CAPWAP AVC detail	
<b>error</b>	Enables debugging of CAPWAP AVC error	
<b>event</b>	Enables debugging of CAPWAP AVC event	
<b>info</b>	Enables debugging of CAPWAP AVC information	
<b>netflow</b>	Enables debugging of CAPWAP client AVC NetFlow	
<b>netflow all</b>	Enables debugging of all CAPWAP client AVC NetFlow	
<b>netflow detail</b>	Enables debugging of CAPWAP client AVC NetFlow detail	
<b>netflow error</b>	Enables debugging of CAPWAP client AVC NetFlow error	
<b>netflow event</b>	Enables debugging of CAPWAP client AVC NetFlow event	
<b>netflow packet</b>	Enables debugging of CAPWAP client AVC NetFlow packet	
<b>numflows</b>	Enables debugging of CAPWAP client AVC numflows	

---

**Command Modes** Privileged EXEC (#)

---

Command History	Release	Modification
	8.1.111.0	This command was introduced.

---

**Examples**

The following example shows how to enable debugging of all CAPWAP client AVC:

```
cisco-ap# debug capwap client avc all
```

## debug cdp

To enable debugging of controller discovery protocol (CDP), use the **debug cdp** command.

```
debug cdp {adjacency | events | ilp | packets}
```

Syntax Description	
<b>adjacency</b>	Enables debugging of CDP neighbors
<b>events</b>	Enables debugging of CDP events
<b>ilp</b>	Enables debugging of inline power
<b>packets</b>	Enables debugging of CDP packets

**Command Modes** Privileged EXEC (#)

**Command History** **Release** **Modification**

8.1.111.0 This command was introduced.

**Examples**

The following example shows how to enable debugging of CDP events:

```
cisco-ap# debug cdp events
```

## debug cleanair

To configure debugging of CleanAir, use the **debug cleanair** command.

```
debug cleanair {bringup | event | logdebuglow | major | nsi | offchan {0 | 1}}
```

Syntax Description	
<b>bringup</b>	Enables debugging of CleanAir port or bringups
<b>events</b>	Enables debugging of normal CleanAir events
<b>logdebug</b>	Logs CleanAir debug output to a logfile
<b>low</b>	Enables debugging of hex dump of some messages

<b>major</b>	Enables debugging of major CleanAir events
<b>nsi</b>	Enables debugging of NSI messages
<b>offchan 0   1</b>	Enables debugging of CleanAir MSMT requests. You have to specify the radio slot as either 0 or 1

**Command Modes** Privileged EXEC (#)

**Command History** **Release Modification**

8.1.111.0 This command was introduced.

### Examples

The following example shows how to enable debugging of major CleanAir events:

```
cisco-ap# debug cleanair major
```

## debug dhcp

To configure debugging of DHCP, use the **debug dhcp** command.

**debug dhcp {errors | events | packets}**

<b>Syntax Description</b>	<b>errors</b> Enables debugging of DHCP errors
	<b>events</b> Enables debugging of DHCP events
	<b>packets</b> Enables debugging of DHCP packets

**Command Modes** Privileged EXEC (#)

**Command History** **Release Modification**

8.1.111.0 This command was introduced.

### Examples

The following example shows how to enable debugging of DHCP errors:

```
cisco-ap# debug dhcp errors
```

## debug dot11 driver level

To enable debugging of 802.11, use the **debug dot11 driver level** command.

```
debug dot11 driver level { critical | errors | events | info }
```

Syntax Description	
<b>critical</b>	Enables 802.11 critical level debugging
<b>errors</b>	Enables 802.11 error level debugging
<b>events</b>	Enables 802.11 event level debugging
<b>info</b>	Enables 802.11 information level debugging

**Command Modes** Privileged EXEC (#)

Command History	Release	Modification
	8.1.111.0	This command was introduced.

### Examples

The following example shows how to enable debugging of 802.11 error level:

```
cisco-ap# debug dot11 driver level errors
```

## debug dot11 client data-path

To enable debugging of 802.11 client data-path, use the **debug dot11 client data-path** command.

```
debug dot11 client data-path {{ all-types | arp | dhcp | eapol | ipv6-ra |.opendns | dns-acl }} { addr { mac-addr1 | mac-addr2 | mac-addr3 | mac-addr4 }}
```

Syntax Description	
<b>arp</b>	Enables client datapath ARP debugging
<b>dhcp</b>	Enables client datapath DHCP debugging
<b>eapol</b>	Enables client datapath EAPOL debugging
<b>dns-acl</b>	Enables client datapath DNS-ACL debugging
<b>ipv6-ra</b>	Enables client data-path IPv6 RA-MC2UC debugging
<b>opendns</b>	Enables client data-path openDNS debugging
<b>{addr   all-types}</b>	Option to specify MAC address of specific clients or all clients

---

{*mac-addr1* | *mac-addr2* | *mac-addr3* | *mac-addr4*} MAC addresses of clients that you have to enter  
| *mac-addr4*}

---

**Command Modes**

Privileged EXEC (#)

**Command History****Release Modification**


---

8.1.111.0 This command was introduced.

---

**Examples**

The following example shows how to enable debugging of client data-path ARP:

```
cisco-ap# debug dot11 client data-path arp
```

## debug dot11 client management

To enable 802.11 client debugging level, use the **debug dot11 client management** command.

```
debug dot11 client management { critical | errors | events | info } { addr { mac-addr1 | mac-addr2 | mac-addr3 | mac-addr4 } }
```

**Syntax Description**

<b>critical</b>	Enables client critical level debugging
<b>errors</b>	Enables client error level debugging
<b>events</b>	Enables client event level debugging
<b>info</b>	Enables client information level debugging
{ <i>mac-addr1</i>   <i>mac-addr2</i>   <i>mac-addr3</i>   <i>mac-addr4</i> }	MAC addresses of clients that you have to enter

---

**Command Modes**

Privileged EXEC (#)

**Command History****Release Modification**


---

8.1.111.0 This command was introduced.

---

**Examples**

The following example shows how to enable debugging of a client at the event level:

```
cisco-ap# debug dot11 client management events e1:90:6f:7e:e6:29
```



## debug dot11 client probe

To enable 802.11 client debugging probe, use the **debug dot11 client probe** command.

```
debug dot11 client probe { { address mac-addr1 | mac-addr2 | mac-addr3 | mac-addr4 } | all }
```

### Syntax Description

<b>address</b>	Probe specific clients using their MAC addresses.
<i>mac-addr</i>	MAC addresses of the clients. You can enter upto four MAC addresses.
<b>all</b>	Probe all the clients associated with the AP.

### Command Modes

Privileged EXEC (#)

### Command History

Release	Modification
8.10	This command was introduced.

### Example

The following example shows how to enable debugging of all clients:

```
cisco-wave2-ap# debug dot11 client probe all
```

## debug dot11 driver slot

To enable debugging of 802.11 drivers, use the **debug dot11 driver slot** command.

```
debug dot11 driver slot { 0 | 1 } { all-types | { cac { info | metrics } } | chd | save-accounting-data | save-on-failure [ extended ] | stop-on-failure | metrics traffic | metrics video | type { all | association | authentication | dhcp | eap | icmp | probe } mac-addr1 | mac-addr2 | mac-addr3 | mac-addr4
```

### Syntax Description

<b>slot</b> { <i>0</i>   <i>1</i> }	Enables 802.11 driver debugs per radio
<b>all-types</b>	Enables all 802.11 driver debugs
<b>cac</b>	Enables 802.11 CAC debugs
<b>cac info</b>	Enables 802.11 CAC info level debugs
<b>cac metrics</b>	Enables debugging of 802.11 CAC metrics
<b>chd</b>	Enables 802.11 CHD debugs
<b>save-accounting-data</b>	Saves the radio accounting data

<b>save-on-failure</b>	Saves the radio crash information upon radio failure
<b>save-on-failure extended</b>	Saves extended information on radio failure
<b>stop-on-failure</b>	Stops the AP from reboot on radio failure
<b>metrics traffic</b>	Enables 802.11 traffic stream metric debugs
<b>metrics video</b>	Enables 802.11 video metric debugs
<b>type</b>	Enables the debug types.
<b>all</b>	Enables the all type debugging.
<b>association</b>	Enables the association debugging.
<b>authentication</b>	Enables the authentication debugging.
<b>dhcp</b>	Enables the dhcp debugging.
<b>eap</b>	Enables the eap debugging.
<b>icmp</b>	Enables the icmp debugging.
<b>probe</b>	Enables the probe debugging.
<i>mac-addr</i>	MAC addresses of the clients. You can enter upto four MAC addresses.

**Command Modes**

Privileged EXEC (#)

**Command History**

Release	Modification
8.1.111.0	This command was introduced.
8.5.140.0 and 8.8	This command was enhanced by adding the <b>type</b> parameter.

**Examples**

The following example shows how to enable debugging of CAC at the information level:

```
cisco-ap# debug dot11 driver slot cac info
```

## debug dot11 firmware

To debug the 802.11 firmware, use the **debug dot11 firmware** command.

```
debug dot11 firmware slot slot_ID level { all-level | critical | emergency | error | info }
address { mac-addr1 | mac-addr2 | mac-addr3 | mac-addr4 }
```

**Syntax Description**

<i>slot_ID</i>	Enables 802.11 driver debugs per radio
----------------	--

<b>all-level</b>	Enables all the debug levels.
<b>critical</b>	Enables critical level debugs.
<b>emergency</b>	Enables emergency level debugs.
<b>error</b>	Enables error level debugs.
<b>info</b>	Enables info level debugs.
<b>address</b>	To add client address for driver/firmware debugging.
<b>mac-addr</b>	MAC addresses of the clients. You can enter upto four MAC addresses.

**Command Modes** Privileged EXEC (#)

<b>Command History</b>	<b>Release</b>	<b>Modification</b>
	8.5.140.0 and 8.8	This command was introduced.

### Example

The following example shows how to enable debugging of 802.11 emergency level:

```
cisco-wave2-ap# debug dot11 firmware slot 1 emergency address 92:FB:D6:B3:7A:6C
```

## debug dot11 sensor

To enable debugging of 802.11 sensors, use the **debug dot11 sensor** command.

```
debug dot11 sensor {dns | file-transfer | mail-server | ping | radius | ssh | telnet | web-server}
```

<b>Syntax Description</b>	
<b>dns</b>	Enables debugging of 802.11 sensor DNS
<b>file-transfer</b>	Enables debugging of 802.11 sensor file transfer
<b>mail-server</b>	Enables debugging of 802.11 sensor mail server
<b>ping</b>	Enables debugging of 802.11 sensor ping
<b>radius</b>	Enables debugging of 802.11 sensor radius
<b>ssh</b>	Enables debugging of 802.11 sensor SSH
<b>telnet</b>	Enables debugging of 802.11 sensor Telnet.
<b>web-server</b>	Enables debugging of 802.11 sensor web server

---

**Command Modes** Privileged EXEC (#)

---

Command History	Release	Modification
	8.1.111.0	This command was introduced.

---

### Examples

The following example shows how to enable debugging of 802.11 sensor file transfer:

```
cisco-ap# debug dot11 sensor file-transfer
```

## debug dtls client

To configure DTLS client error and event debugging, use the **debug dtls client** command.

**debug dtls client** { **error** | **event** [**detail**] }

Syntax Description	error	event [detail]
	Configures debugging of DTLS client errors	Configures debugging of DTLS client events

---



---

**Command Modes** Privileged EXEC (#)

---

Command History	Release	Modification
	8.1.111.0	This command was introduced.

---

### Examples

The following example shows how to enable debugging of DTLS client events:

```
cisco-ap# debug dtls client event
```

## debug ethernet

To configure Ethernet debugging, use the **debug ethernet** command.

**debug ethernet** *interface-number* { **both** | **rcv** | **xmt** }

Syntax Description	<i>interface-number</i>	both
	Interface number that you have to enter as either 0 or 1	Enables debugging of both transmission and reception

---

<b>rcv</b>	Enables debugging of reception
<b>xmt</b>	Enables debugging of transmission

**Command Modes**

Privileged EXEC (#)

**Command History****Release Modification**

8.1.111.0 This command was introduced.

**Examples**

The following example shows how to enable debugging of transmission for interface 0:

```
cisco-ap# debug ethernet 0 xmt
```

## debug flexconnect

To debug FlexConnect features, use the **debug flexconnect** command.

```
debug flexconnect {acl | cckm | dot11r | event | multicast {igmp | traffic} | pmk | proxy-arp | vsa | wlan-vlan | wsastats}
```

**Syntax Description**

<b>acl</b>	Configures debugging of FlexConnect ACL
<b>cckm</b>	Configures debugging of CCKM
<b>dot11r</b>	Configures debugging of 802.11r
<b>event</b>	Configures debugging of wireless control protocol (WCP) events
<b>multicast igmp</b>	Configures debugging of Multicast IGMP
<b>multicast traffic</b>	Configures debugging of Multicast traffic
<b>pmk</b>	Configures debugging of opportunistic key caching (OKC) or pairwise master key caching
<b>vsa</b>	Configures debugging of AAA vendor specific attributes (VSA)
<b>wlan-vlan</b>	Configures debugging of WLAN-VLAN mapping
<b>wsastats</b>	Configures debugging of RADIUS or DHCP wireless service assurance statistics

**Command Modes**

Privileged EXEC (#)

**Command History****Release Modification**

8.1.111.0 This command was introduced.

**Examples**

The following example shows how to enable debugging of FlexConnect ACL:

```
cisco-ap# debug flexconnect acl
```

## debug lldp

To debug LLDP, use the **debug lldp** command.

```
debug lldp {errors | events | packet}
```

Syntax Description	
<b>errors</b>	Debugs LLDP errors
<b>events</b>	Debugs LLDP events
<b>packet</b>	Debugs LLDP packets

Command Modes	Privileged EXEC (#)
---------------	---------------------

Command History	Release	Modification
	8.1.111.0	This command was introduced.

**Examples**

The following example shows how to enable debugging of LLDP errors:

```
cisco-ap# debug lldp errors
```

## debug memory

To debug memory, use the **debug memory** command.

```
debug memory {clear | save}
```

Syntax Description	
<b>clear</b>	Removes memory debug upon boot-up
<b>save</b>	Saves current debug level and applies it upon following boots

Command Modes	Privileged EXEC (#)
---------------	---------------------

Command History	Release	Modification
	8.1.111.0	This command was introduced.

### Examples

The following example shows how to remove memory debug upon boot-up:

```
cisco-ap# debug memory clear
```

## debug memory pool

To debug memory pool, use the **debug memory pool** command.

```
debug memory pool {diff | realtime interval 1-1000000-seconds | start}
```

Syntax Description	diff	Shows memory pool debug difference in detail
	<b>realtime interval</b> <i>1-1000000-seconds</i>	Configures realtime interval for the memory pool
	<b>start</b>	Starts the debug for the memory pool

**Command Modes** Privileged EXEC (#)

Command History	Release	Modification
	8.1.111.0	This command was introduced.

### Examples

The following example shows how to configure realtime interval of 180 seconds for the memory pool:

```
cisco-ap# debug memory pool realtime interval 180
```

## debug memory pool alloc

To debug memory pool allocation calls, use the **debug memory pool alloc** command.

```
debug memory pool alloc {all | name pool-name} {diff | realtime interval 1-1000000-seconds | start}
```

Syntax Description	all	Configures debug for all memory pool allocation calls
	<b>name</b> <i>pool-name</i>	Configures debug for a specific memory pool's allocation call

<b>diff</b>	Shows memory pool debug allocation call difference in detail
<b>realtime interval</b> <i>1-1000000-seconds</i>	Configures realtime interval for the memory pool allocation calls
<b>start</b>	Starts the debug for the memory pool allocation calls

**Command Modes**

Privileged EXEC (#)

**Command History****Release Modification**

8.1.111.0 This command was introduced.

**Examples**

The following example shows how to configure the start of the debug for all memory pool allocation calls:

```
cisco-ap# debug memory pool alloc all start
```

## debug memory pool free

To debug memory pool free calls, use the **debug memory pool free** command.

```
debug memory pool free {all | name pool-name} {diff | realtime interval 1-1000000-seconds | start}
```

**Syntax Description**

<b>all</b>	Configures debug for all memory pool free calls
<b>name</b> <i>pool-name</i>	Configures debug for a specific memory pool's free call
<b>diff</b>	Shows memory pool debug free call difference in detail
<b>realtime interval</b> <i>1-1000000-seconds</i>	Configures realtime interval for the memory pool free calls
<b>start</b>	Starts the debug for the memory pool free calls

**Command Modes**

Privileged EXEC (#)

**Command History****Release Modification**

8.1.111.0 This command was introduced.

**Examples**

The following example shows how to configure the start of the debugging of all memory pool free calls:



```
cisco-ap# debug memory pool free all start
```

## debug mesh

To configure debugging of mesh networks, use the **debug mesh** command.

```
debug mesh {channel | clear | convergence | events | forward-mcast | forward-packet | forward-table | linktest | path-control | port-control | security | trace}
```

### Syntax Description

<b>channel</b>	Configures debugging of mesh channel
<b>clear</b>	Resets all mesh debugs
<b>convergence</b>	Configures debugging of mesh convergence
<b>events</b>	Configures debugging of mesh events
<b>forward-mcast</b>	Configures debugging of mesh forwarding Multicast
<b>forward-packet</b>	Configures debugging of mesh forwarding packets
<b>forward-table</b>	Configures debugging of mesh forwarding table
<b>linktest</b>	Configures debugging of mesh linktest
<b>port-control</b>	Configures debugging of mesh port control
<b>security</b>	Configures debugging of mesh security
<b>trace</b>	Configures debugging of mesh trace

### Command Modes

Privileged EXEC (#)

### Command History

Release	Modification
8.1.111.0	This command was introduced.

### Examples

The following example shows how to enable debugging of mesh channel:

```
cisco-ap# debug mesh channel
```

## debug mesh adjacency

To debug mesh adjacency, use the **debug mesh adjacency** command.

```
debug mesh adjacency {child | clear | dfs | message | packet | parent}
```

Syntax Description	
<b>adjacency</b>	Debug mesh adjacency
<b>child</b>	Debug mesh adjacency child
<b>clear</b>	Debug clear mesh adjacency
<b>dfs</b>	Debug mesh DFS
<b>message</b>	Debug mesh adjacency messages
<b>packet</b>	Debug mesh adjacency packet
<b>parent</b>	Debug mesh adjacency parent

**Command Modes** Privileged EXEC (#)

Command History	Release	Modification
	8.1.111.0	This command was introduced.

### Examples

The following example shows how to enable debugging of mesh adjacency parent:

```
cisco-ap# debug mesh adjacency parent
```

## debug mesh path-control

To configure debugging of mesh path control, use the **debug mesh path-control** command.

```
debug mesh path-control {error | events | packets }
```

Syntax Description	
<b>error</b>	Configures debugging of mesh path control errors
<b>events</b>	Configures debugging of mesh path control events
<b>packets</b>	Configures debugging of mesh path control packets

**Command Modes** Privileged EXEC (#)

Command History	Release	Modification
	8.1.111.0	This command was introduced.

### Examples

The following example shows how to enable debugging of mesh path control errors:

```
cisco-ap# debug mesh path-control error
```

## debug rrm neighbor

To enable RRM neighbor debugging, use the **debug rrm neighbor** command.

```
debug rrm neighbor {tx | rx | detail }
```

Syntax Description	tx	Enable RRM neighbor Tx debugging
	<b>rx</b>	Enable RRM neighbor Rx debugging
	<b>detail</b>	Enable RRM neighbor detail debugging

Command Modes	Privileged EXEC (#)
---------------	---------------------

Command History	Release	Modification
	8.1.111.0	This command was introduced.

### Examples

The following example shows how to enable debugging of RRM neighbor transmissions:

```
cisco-ap# debug rrm neighbor tx
```

## debug rrm reports

To enable RRM reports debugging, use the **debug rrm reports** command.

```
debug rrm reports
```

Syntax Description	reports	Enables RRM report debugging
--------------------	---------	------------------------------

Command Modes	Privileged EXEC (#)
---------------	---------------------

Command History	Release	Modification
	8.1.111.0	This command was introduced.

### Examples

The following example shows how to enable debugging of RRM reports:

```
cisco-ap# debug rrm reports
```

## debug sip

To enable session initiation protocol (SIP) debugging, use the **debug sip** command.

```
debug sip {all | tx | rx}
```

Syntax Description	
<b>all</b>	Enabling SIP transmission and reception debugging
<b>tx</b>	Enabling SIP transmission debugging
<b>rx</b>	Enabling SIP reception debugging

Command Modes	
	Privileged EXEC (#)

Command History	Release Modification
	8.1.111.0 This command was introduced.

### Examples

The following example shows how to enable debugging of SIP transmissions and reception:

```
cisco-ap# debug sip all
```

## debug wips

To enable wIPS debugging, use the **debug wips** command.

```
debug wips {errors | events | critical}
```

Syntax Description	
<b>errors</b>	Enable wIPS error level debugging
<b>events</b>	Enable wIPS event level debugging
<b>critical</b>	Enable wIPS critical level debugging

Command Modes	
	Privileged EXEC (#)

Command History	Release Modification
	8.1.111.0 This command was introduced.

### Examples

The following example shows how to enable wIPS error level debugging:

```
cisco-ap# debug wips errors
```

## debug process memory

To process memory debugging, use the **debug process memory** command.

```
debug process memory {diff | realtime [interval interval-in-seconds ] | start}
```

Syntax Description	diff	Process memory debug show diff
	<b>realtime</b>	Process memory real time debug
	<i>interval</i>	Update interval; valid range 1 to 1000000 seconds
	<b>start</b>	Process memory debug start

**Command Modes** Privileged EXEC (#)

Command History	Release	Modification
	8.1.111.0	This command was introduced.

### Examples

The following example shows how to enable the start of debugging of process memory:

```
cisco-ap# debug process memory start
```

## debug traffic

To enable traffic debugging, use the **debug traffic** command.

```
debug traffic {host {icmpv6 | ip | ipv6 | tcp | udp { verbose}}} | wired {ip | tcp | udp { verbose}}
```

Syntax Description	host	Enabling host traffic debugging
	<b>wired</b>	Enabling wired traffic debugging
	<b>verbose</b>	Display verbose output
	<b>icmpv6</b>	Enabling host ICMPv6 traffic dump

<b>ip</b>	Enabling host IP traffic dump
<b>ipv6</b>	Enabling host IPv6 traffic dump
<b>tcp</b>	Enabling TCP traffic dump
<b>udp</b>	Enabling UDP traffic dump

**Command Modes** Privileged EXEC (#)

Command History	Release	Modification
	8.1.111.0	This command was introduced.

### Examples

The following example shows how to enable debugging of host IP traffic dump:

```
cisco-ap# debug traffic host ip
```

## debug tunnel

To configure debugging of tunnel, use the **debug tunnel** command.

### debug tunnel eogre

Syntax Description	eogre	Configures debugging of EoGRE tunnel
--------------------	-------	--------------------------------------

**Command Modes** Privileged EXEC (#)

Command History	Release	Modification
	8.1.111.0	This command was introduced.

### Examples

The following example shows how to enable debugging of EoGRE tunnel:

```
cisco-ap# debug tunnel eogre
```

## debug client trace

To enable client trace debugging, use the **debug client trace** command.

**debug client trace** {**all** | **address** *mac-address* | **enable** | **filter** {**assoc** | **auth** | **dhcp** | **eap** | **icmp** | **mgmt** | **probe** | **proto**}}

Syntax Description	all	Configure all clients tracing
	<b>address</b>	Configure address(es) to trace
	<i>mac-address</i>	MAC address to trace
	<b>enable</b>	Enable tracing
	<b>filter</b>	Configure trace filter
	<b>assoc</b>	Trace Association packets
	<b>auth</b>	Trace Authentication packets
	<b>dhcp</b>	Trace DHCP packets
	<b>eap</b>	Trace EAP packets
	<b>icmp</b>	Trace ICMP packets
	<b>mgmt</b>	Trace probe, assoc, auth, EAP packets
	<b>probe</b>	Trace probe packets
	<b>proto</b>	Trace DHCP, ICMP packets

**Command Modes** Privileged EXEC (#)

**Command History**

Release	Modification
8.1.111.0	This command was introduced.

### Examples

The following example shows how to enable tracing of all clients:

```
cisco-ap# debug client trace all
```

## no

To negate a command or set to its defaults, use the **no** command.

**no**

**Command Modes** Privileged EXEC (#)

**Command History****Release Modification**

8.1.111.0 This command was introduced.

To negate a command or set to its defaults, use this command:

```
cisco-ap# no debug
```

## tracert

To view the routes followed by packets traveling in the network, use the **tracert** command.

**tracert** *destination-address*

**Syntax Description**

*destination-address* IP address of the destination of the packets

**Command Modes**

Privileged EXEC (#)

**Command History****Release Modification**

8.1.111.0 This command was introduced.

**Examples**

The following example shows how to view the routes followed by packets traveling in the network, with a destination IP address specified:

```
cisco-ap# tracert 209.165.200.224
```

## undeb

To disable debugging on the access point, use the **undeb** command.

**undeb** [**all**]

**Syntax Description**

**a** Disables all debugging messages.

**Command Modes**

Privileged EXEC (#)

**Command History****Release Modification**

8.1.111.0 This command was introduced.



### Examples

The following example shows how to disable all debugging messages:

```
cisco-ap# undebug all
```

