

Conditionally Triggered Debugging

Description

The Conditionally Triggered Debugging feature limits debugging messages based on their related interface or subinterface. When this feature is enabled, the router will generate debugging messages for packets entering or leaving the router on a specified interface. However, the router will not generate debugging output for packets entering or leaving through a different interface. This feature allows you to focus debugging output on the problematic interface or interfaces.

You can specify the interfaces explicitly. For example, you may only want to see debugging messages for one interface or subinterface. You can also turn on debugging for all interfaces that meet specified conditions, such as a particular username, calling party number, or called party number. If you specify multiple conditions, the interface must meet at least one of the conditions.

This feature controls the output from the following protocol-specific **debug** commands:

- **debug aaa** { **accounting** | **authorization** | **authentication** }
- **debug dialer** { **events** | **packets** }
- **debug isdn** { **q921** | **q931** }
- **debug modem** { **oob** | **trace** }
- **debug ppp** { **all** | **authentication** | **chap** | **error** | **negotiation** | **multilink events** | **packet** }

While this feature limits the output of the above commands, it does not automatically enable the generation of debugging output from these commands. Debugging messages are generated only if the protocol-specific **debug** command is enabled.

Benefits

This feature is useful on dial access servers, which have a large number of ports.

Normally, the router will generate debugging messages for every interface, resulting in a large number of messages. The large number of messages consumes system resources. For example, if the messages are displayed on the console, the router must take the time to send every message to the console. Similarly, if you are sending messages to a syslog server, the large number of generated output consumes network resources.

In addition, the large number of messages can make it difficult to find the specific information you need. Messages may scroll by on the console too quickly, or the logging buffer may wrap around before you are able to examine the contents of the buffer.

By limiting the debugging messages, you can receive messages related to only the ports you wish to troubleshoot. This limiting will decrease the number of generated messages, thus consuming less resources and making it easier for you to find the information you want.

Other Debug Output Filtering Methods

Previously, you could filter **debug** command output based on an address, an access list, or an interface. These values were specified in the actual protocol-specific **debug** command that enabled the debugging. However, this feature allows you to limit messages from multiple **debug** commands by entering one command.

The previous mechanisms for filtering debug messages are still available.

Configuration Tasks

Debug command output is controlled through two processes:

- The protocol-specific **debug** commands specify which protocols are being debugged. For example, the **debug dialer events** command generates debugging output related to dialer events.
- The **debug condition** commands limit these debugging messages to those related to a particular interface. For example, the **debug condition username bob** command generates debugging output only for interfaces with packets that specify a username of bob.

To configure Conditionally Triggered Debugging, perform the following tasks:

- Enable Protocol-Specific Debug Commands
- Enable Conditional Debugging Commands

Enable Protocol-Specific Debug Commands

In order to generate any debugging output, the protocol-specific **debug** command for the desired output must be enabled. Use the **show debugging** command to determine which types of debugging are enabled. Perform the following tasks in privileged EXEC mode to enable the desired protocol-specific **debug** commands:

Task	Command
Determine which types of debugging are enabled.	show debugging
Enable the desired debugging commands.	debug protocol
Disable the debugging commands that are not desired.	no debug protocol

If you wish to have no output, disabled all the protocol-specific **debug** commands.

Enable Conditional Debugging Commands

If no **debug condition** commands are enabled, all debugging output, regardless of the interface, will be displayed for the enabled protocol-specific **debug** commands.

The first **debug condition** command you enter enables conditional debugging. The router will only display messages for interfaces that meet one of the specified conditions. If multiple conditions are specified, the interface must meet at least one of the conditions in order for messages to be displayed.

You can enable messages for interfaces specified explicitly or for interfaces that meet certain conditions, as described in the following sections:

- Display Messages for One Interface
- Display Messages for Multiple Interfaces
- Limit Messages Based on Conditions

Display Messages for One Interface

To disable debugging messages for all interfaces except one, perform the following task in privileged EXEC mode:

Task	Command
Disable debugging messages for all interfaces except one.	debug condition interface <i>interface</i>

If you enter the **debug condition interface** command, the debugging output will be turned off for all interfaces except the specified interface. To reenabling debugging output for all interfaces, use the **no debug interface command**.

Display Messages for Multiple Interfaces

To enable debugging messages for multiple interfaces, perform the following tasks in privileged EXEC mode:

Task	Command
Disable debugging messages for all interfaces except one.	debug condition interface <i>interface</i>
Enable debugging messages for additional interfaces. Repeat this task until debugging messages are enabled for all desired interfaces.	debug condition interface <i>interface</i>

If you specify more than one interface by entering this command multiple times, debugging output will be displayed for all of the specified interfaces. To turn off debugging on a particular interface, use the **no debug interface** command. If you use the **no debug interface all** command or remove the last **debug interface** command, debugging output will be reenabling for all interfaces.

Limit Messages Based on Conditions

The router can monitor interfaces to see if any packets contain the specified value for one of the following conditions:

- username
- calling party number
- called party number

If you enter a condition, such as calling number, debug output will be stopped for all interfaces. The router will then monitor every interface to see if a packet with the specified calling party number is sent or received on any interfaces. If the condition is met on an interface or subinterface, **debug** command output will be displayed for that interface. The debugging output for an interface is

“triggered” when the condition has been met. The debugging output continues to be disabled for the other interfaces. If, at some later time, the condition is met for another interface, the debug output will become enabled for that interface as well.

Once debugging output has been triggered on an interface, the output will continue until the interface goes down. However, the session for that interface might change, resulting in a new username, called party number, or calling party number. Use the **no debug interface** command to reset the debug trigger mechanism for a particular interface. The debugging output for that interface will be disabled until the interface meets one of the specified conditions.

To limit debugging messages based on a specified condition, perform the following task in privileged EXEC mode:

Task	Command
Enable conditional debugging. The router will only display messages for interfaces that meet this condition.	debug condition { username <i>username</i> called <i>dial-string</i> caller <i>dial-string</i> }

To reenable the debugging output for all interfaces, enter the **no debug condition all** command.

Specify Multiple Conditions

To limit debugging messages based on more than one condition, perform the following tasks in privileged EXEC mode:

Task	Command
Enable conditional debugging and specify the first condition.	debug condition { username <i>username</i> called <i>dial-string</i> caller <i>dial-string</i> }
Specify the second condition. Repeat this task until all conditions are specified.	debug condition { username <i>username</i> called <i>dial-string</i> caller <i>dial-string</i> }

If you enter multiple **debug condition** commands, debugging output will be generated if an interface meets at least one of the conditions. If you remove one of the conditions, using the **no debug condition** command, interfaces that meet only that condition will no longer produce debugging output. However, interfaces that meet a condition other than the removed condition will continue to generate output. Only if no active conditions are met for an interface will the output for that interface be disabled.

Configuration Examples

In this example, four conditions have been set by the following commands:

- **debug condition interface serial 0**
- **debug condition interface serial 1**
- **debug condition interface virtual-template 1**
- **debug condition username fred**

The first three conditions have been met by one interface. The fourth condition has not yet been met.

```
Router# show debug condition

Condition 1: interface Se0 (1 flags triggered)
           Flags: Se0
Condition 2: interface Se1 (1 flags triggered)
           Flags: Se1
Condition 3: interface Vt1 (1 flags triggered)
           Flags: Vt1
Condition 4: username fred (0 flags triggered)
```

When any **debug condition** command is entered, debugging messages for conditional debugging are enabled. The following debugging messages show conditions being met on different interfaces as the serial 0 and serial 1 interfaces come up. For example, the second line of output indicates that serial interface 0 meets the username fred condition.

```
*Mar 1 00:04:41.647: %LINK-3-UPDOWN: Interface Serial0, changed state to up
*Mar 1 00:04:41.715: Se0 Debug: Condition 4, username fred triggered, count 2
*Mar 1 00:04:42.963: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0, changed
state to up
*Mar 1 00:04:43.271: Vi1 Debug: Condition 3, interface Vt1 triggered, count 1
*Mar 1 00:04:43.271: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
*Mar 1 00:04:43.279: Vi1 Debug: Condition 4, username fred triggered, count 2
*Mar 1 00:04:43.283: Vi1 Debug: Condition 1, interface Se0 triggered, count 3
*Mar 1 00:04:44.039: %IP-4-DUPADDR: Duplicate address 172.27.32.114 on Ethernet 0,
sourced by 00e0.1e3e.2d41
*Mar 1 00:04:44.283: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1,
changed state to up
*Mar 1 00:04:54.667: %LINK-3-UPDOWN: Interface Serial1, changed state to up
*Mar 1 00:04:54.731: Se1 Debug: Condition 4, username fred triggered, count 2
*Mar 1 00:04:54.735: Vi1 Debug: Condition 2, interface Se1 triggered, count 4
*Mar 1 00:04:55.735: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1, changed
state to up
```

After a period of time, the **show debug condition** command displays the revised list of conditions.

```
Router# show debug condition

Condition 1: interface Se0 (2 flags triggered)
           Flags: Se0 Vi1
Condition 2: interface Se1 (2 flags triggered)
           Flags: Se1 Vi1
Condition 3: interface Vt1 (2 flags triggered)
           Flags: Vt1 Vi1
Condition 4: username fred (3 flags triggered)
           Flags: Se0 Vi1 Se1
```

Next, the serial 1 and serial 0 interfaces go down. When an interface goes down, conditions for that interface are cleared.

```
*Mar 1 00:05:51.443: %LINK-3-UPDOWN: Interface Serial1, changed state to down
*Mar 1 00:05:51.471: Se1 Debug: Condition 4, username fred cleared, count 1
*Mar 1 00:05:51.479: Vi1 Debug: Condition 2, interface Se1 cleared, count 3
*Mar 1 00:05:52.443: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1, changed
state to down
*Mar 1 00:05:56.859: %LINK-3-UPDOWN: Interface Serial0, changed state to down
*Mar 1 00:05:56.887: Se0 Debug: Condition 4, username fred cleared, count 1
*Mar 1 00:05:56.895: Vi1 Debug: Condition 1, interface Se0 cleared, count 2
*Mar 1 00:05:56.899: Vi1 Debug: Condition 3, interface Vt1 cleared, count 1
*Mar 1 00:05:56.899: Vi1 Debug: Condition 4, username fred cleared, count 0
*Mar 1 00:05:56.903: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down
*Mar 1 00:05:57.907: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0, changed
state to down
```

```
*Mar 1 00:05:57.907: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1,  
changed state to down
```

The final **show debug condition** output is the same as the output before the interfaces came up.

```
Router# show debug condition  
  
Condition 1: interface Se0 (1 flags triggered)  
Flags: Se0  
Condition 2: interface Se1 (1 flags triggered)  
Flags: Se1  
Condition 3: interface Vt1 (1 flags triggered)  
Flags: Vt1  
Condition 4: username fred (0 flags triggered)
```

Command Reference

This section documents new and modified commands. All other command used with this feature are documented in the Cisco IOS Release 11.3 command references or *Debug Command Reference*.

debug condition

To limit output for some debugging commands based on specified conditions, use the **debug condition** EXEC command. The **no** form of this command removes the specified condition.

```
debug condition { username username | called dial-string | caller dial-string }
no debug condition { condition-id | all }
```

Syntax Description

username <i>username</i>	Generated debugging messages for interfaces with the specified username.
called <i>dial-string</i>	Generates debugging messages for interfaces with the called party number.
caller <i>dial-string</i>	Generates debugging messages for interfaces with the calling party number.
<i>condition-id</i>	Removes the condition indicated.
all	Removes all debugging conditions, conditions specified by the debug condition interface command. Use this keyword to disable conditional debugging and reenable debugging for all interfaces.

Default

All debugging messages for enabled protocol-specific **debug** commands are generated.

Usage Guidelines

Use the **debug condition** command to restrict the debug output for some commands. If any **debug condition** commands are enabled, output is only generated for interfaces associated with the specified username, called party number, or calling party number. In addition, this command enables debugging output for conditional debugging events. Messages are displayed as different interfaces meet specific conditions.

The **no** form of this command removes the debug condition specified by the condition identifier. The condition identifier is displayed after you enter a **debug condition** command or in the output of the **show debug condition** command. If the last condition is removed, debugging output resumes for all interfaces. You will be asked for confirmation before removing the last condition or all conditions.

Not all debugging output is affected by the **debug condition** command. Some commands generate output whenever they are enabled, regardless of whether or not they meet any conditions. The commands that are affected by the **debug condition** commands are generally related to dial access functions, where a large amount of output is expected. Output from the following commands is controlled by the **debug condition** command:

- **debug aaa** { **accounting** | **authorization** | **authentication** }
- **debug dialer** { **events** | **packets** }
- **debug isdn** { **q921** | **q931** }

- **debug modem** {oob | trace}
- **debug ppp** {all | authentication | chap | error | negotiation | multilink events | packet}

Example

In the following example, the router only displays debugging messages for interfaces that use a username of fred. The condition identifier displayed after the command is entered identifies this particular condition.

```
Router# debug condition username fred  
Condition 1 set
```

Related Commands

debug condition interface
show debug condition

debug condition interface

To limit output for some debugging commands based on the interface, use the **debug condition interface EXEC** command. The **no** form of this command removes the interface condition and resets the interface so that it must be triggered by a condition.

```
debug condition interface interface
no debug condition interface {interface | all}
```

Syntax Description

<i>interface</i>	Interface type and number.
all	All interfaces.

Default

All debug messages for enabled debugging commands are displayed.

Usage Guidelines

Use this command to restrict the debug output for some commands to output based on its related interface. When you enter this command, debugging output is turned off for all interfaces except the specified interface. In addition, this command enables debugging output for conditional debugging events. Messages are displayed as different interfaces meet specific conditions.

The **no** form of the command has two functions:

- It disables the **debug condition interface** command for the specified interface. Output is no longer generated for the interface, assuming that the interface meets no other conditions. If the interface meets other active conditions, as set by another **debug condition** command, debugging output will still be generated for the interface.
- The command also resets the debugging trigger on the interface. If some other **debug condition** command has been enabled, this command resets the trigger on the interface. Output is stopped for that interface until the condition is met on the interface.

You will be asked for confirmation before removing the last condition or all conditions.

Not all debugging output is affected by the **debug condition** command. Some commands generate output whenever they are enabled, regardless of whether or not they meet any conditions. The commands that are affected by the **debug condition** commands are generally related to dial access functions, where a large amount of output is expected. Output from the following commands is controlled by the **debug condition** command:

- **debug aaa** {**accounting** | **authorization** | **authentication**}
- **debug dialer** {**events** | **packets**}
- **debug isdn** {**q921** | **q931**}
- **debug modem** {**oob** | **trace**}
- **debug ppp** {**all** | **authentication** | **chap** | **error** | **negotiation** | **multilink events** | **packet**}

Example

In this example, only **debug** command output related to serial interface 1 is displayed. The condition identifier for this command is 1.

```
Router# debug condition interface serial1  
Condition 1 set
```

Related Commands

debug condition
show debug condition

show debug condition

To display the current debug conditions, use the **show debug condition EXEC** command.

```
show debug condition [condition-id]
```

Syntax Description

condition-id (Optional) Displays only the specified condition.

Command Mode

EXEC

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3 AA.

Sample Display

The following example shows sample **show debug condition** output:

```
Router# show debug condition

Condition 1: interface Se0 (2 flags triggered)
          Flags: Se0 Vi1
Condition 2: interface Se1 (2 flags triggered)
          Flags: Se1 Vi1
Condition 3: interface Vt1 (2 flags triggered)
          Flags: Vt1 Vi1
Condition 4: username fred (3 flags triggered)
          Flags: Se0 Vi1 Se1
```

Table 1 describes the fields shown in this display.

Table 1 Show Debug Condition Field Descriptions

Field	Description
Condition 1	Condition identifier.
interface Se0	Description of the condition.
2 flags triggered	Number of interfaces that meet this condition.
Flags: Se0 Vi1	Interfaces that meet this condition.

Related Commands

debug condition

debug condition interface

