Conditional Debug and Radioactive Tracing

- Feature history for conditional debug and radioactive tracing, on page 1
- Conditional debugging, on page 2
- Radioactive tracing, on page 2
- Conditional debugging and radioactive tracing, on page 3
- Location of tracefiles, on page 3
- Configure conditional debugging (GUI), on page 3
- Configure conditional debugging (CLI), on page 4
- Radioactive Tracing for L2 Multicast, on page 5
- Recommended workflow for trace files, on page 6
- Copy tracefiles off the box, on page 6
- Configuration examples for conditional debugging, on page 7
- Verify conditional debugging, on page 7
- Example: verify radioactive tracing log for SISF, on page 8

Feature history for conditional debug and radioactive tracing

This table provides release and related information about the feature explained in this section.

This feature is also available in all the releases subsequent to the one in which they are introduced in, unless noted otherwise.

Table 1: Feature history for conditional debug and radioactive tracing

Feature Name	Release Information	Feature Description
Conditional debug and radioactive tracing	Cisco IOS XE 17.11.1	Conditional debugging is a diagnostic feature that selectively activates debugging and logging based on defined conditions to optimize resource usage and performance in complex networks, while radioactive tracing is a diagnostic mechanism that stitches together execution chains with increased verbosity, enabling conditional detailed debug information across system components for troubleshooting specific features.

Conditional debugging

A conditional debugging is a diagnostic feature that

- enables selective activation of debugging and logging for specific features based on conditions you define
- allows you to define custom conditions such as interface, IP address, or MAC address to narrow debug scope, and
- helps reduce system resource usage and performance impact in large, complex networks.

Use this feature in systems with many supported features.

Conditional debug allows granular debugging in networks operating at large scale with many features. It allows you to observe detailed debug output for specific instances within the system.

In contrast, the general debug command produces output without filtering feature objects being processed. The general debug command uses significant system resources and affects performance.

Radioactive tracing

A radioactive tracing is a diagnostic tracing mechanism that

- stitches together a chain of execution for operations of interest across the system, at an increased verbosity level
- enables conditional printing of detailed debug information across threads, processes and function calls, and
- troubleshoot system features, especially First-Hop Security (FHS) and mesh networking functions.

Note

- For more information on First Hop Security features, see **System Management** > **Wireless Multicast** > **Information About Wireless Multicast** > **Information About IPv6 Snooping**.
- The radioactive tracing filter does not work, if the certificate is not valid.
- To debug mesh features effectively, add both Ethernet and Radio MAC addresses as conditional MACs for RA tracing when you collect logs.
- To enable debugging for wireless IPs, use the **debug platform condition feature wireless ip** *ip-address* command.

Table 2: Components Supporting Radio Active Tracing

Components	Details
SISF or FHS	The first-hop security features, includes IPv6 Address Glean and IPv6 Device Tracking. For more information, see *Information About IPv6 Snooping*.

Components	Details
LISP	Locator or ID Separation Protocol.

Conditional debugging and radioactive tracing

A conditional debugging and radioactive tracing feature is a diagnostic mechanism that

- combines the precision of conditional debugging to target specific execution contexts
- leverages radioactive tracing to follow code paths across multiple control flows, and
- enables use of a single debug CLI to debug all execution contexts related to the condition.

Note Use the **clear platform condition all** command to remove the debug conditions applied to the platform.

Location of tracefiles

Tracefiles are generated for each process and saved in either the `/tmp/rp/trace` or `/tmp/fp/trace` directory. Each trace log is written to a file of 1 MB. The directory can hold up to a maximum of 25 such files for a given process. When a tracefile in the `/tmp` directory reaches its 1MB limit or the configured size, it is rotated out to an archive location in the `/crashinfo` partition under the `tracelogs` directory.

- Process-name_Process-ID_running-counter.timestamp.gz

  Example: IOSRP_R0-0.bin_0.14239.20151101234827.gz

- Process-name_pmanlog_Process-ID_running-counter.timestamp.bin.gz

  Example: wncmgrd_R0-0.27958_1.20180902081532.bin.gz

The `/tmp` directory holds only a single tracefile for a given process. Once a tracefile reaches its size limit, the system moves it to `/crashinfo/tracelogs`. The archive directory stores up to 25 files and replaces the oldest file with each new file from `/tmp`. File size varies by process. Some processes use larger files that can reach 10 MB.

- The number of files stored in the `tracelogs` directory also depends on the process. For example, the WNCD process may use a limit of up to 400 files per instance, depending on the platform.
- You can verify these logs (per-process) using the **show platform software trace message** *process_name* **chassis active R0** command.

Configure conditional debugging (GUI)

Enable targeted troubleshooting by configuring conditional debugging through the GUI, allowing you to capture and analyze logs for specific MAC or IP addresses.

Procedure

Step 1 Choose **Troubleshooting** > **Radioactive Trace**. Click **Add**.

Step 2 Enter the **MAC/IP Address**.

Step 3 Click **Apply to Device**.

Step 4 Click **Start** to start or **Stop** to stop the conditional debug.

Step 5 Click **Generate** to create a radioactive trace log.

Step 6 Click the radio button to set the time interval.

Step 7 Click the **Download Logs** icon that is displayed next to the trace file name, to download the logs to your local folder.

Step 8 Click the **View Logs** icon that is displayed next to the trace file name, to view the log files on the GUI page.

Step 9 Click **Load More** to view more lines of the log file. Click **Apply to Device**.

Configure conditional debugging (CLI)

Enable targeted troubleshooting for wireless features by filtering debug output to specific APs, clients, or IP addresses using commands.

Procedure

Step 1 Configure conditional debugging for a feature using the specified MAC address.

Example:

```
Device# debug platform condition feature wireless mac b838.61a1.5433
```

Note
This is supported with AP or client MAC/IP and also on CMX IP address and mobility peer IP.

Step 2 Start conditional debugging.

Example:

```
Device# debug platform condition start
```

This starts radioactive tracing if there is a match on one of the conditions above.

Note
This is supported with AP or client MAC/IP and also on CMX IP address and mobility peer IP.

Step 3 Display the current conditions set.

Example:

```
Device# show platform condition
```

```
Device# show debugging
```

Step 4 Stop conditional debugging.

Example:

```
Device# debug platform condition stop
```

This will stop radioactive tracing.

Note
This is supported with AP or client MAC/IP and also on CMX IP address and mobility peer IP.

Step 5 Display the logs from the latest wireless profile.

Example:

```
Device# show logging profile wireless [ counter start last {20 minutes} | filter mac
{mac-address}  to-file bootflash:logs.txt]
```

Note
You can use either the **show logging profile wireless** command or **show logging process** command to collect the logs.

Step 6 Display the logs collection specific to the process.

Example:

```
Device# show logging process wncd to-file flash:wncd.txt
```

Step 7 Clear all conditions.

Example:

```
Device# clear platform condition all
```

What to do next

The command **request platform software trace filter-binary wireless** *mac-address* generates 3 flash files:

- collated_log_<.date..>
- mac_log <..date..>
- mac_database .. file

The *mac_log <..date..>* file is the most important because it contains messages related to the MAC address under investigation. The **show platform software trace filter-binary** command generates the same flash files and displays the mac_log on the screen.

Radioactive Tracing for L2 Multicast

To identify a specific multicast receiver, specify the MAC address of the joiner or the receiver client, Group Multicast IP address and Snooping VLAN. Additionally, enable the trace level for the debug. The debug level will provide detailed traces and better visibility into the system.

debug platform condition feature multicast controlplane *igmp-mld-debug igmp-snooping level mld-snooping pim*

Recommended workflow for trace files

The recommended workflow for trace files is listed below:

- To request the tracelogs for a specific time period, such as 1 day, use the command: Device# **show logging process wncd to-file flash:wncd.txt**.
- The system generates a text file of the tracelogs in the location /flash:
- Copy the file off the device so the tracelogs can be used to work offline. For more details on copying files, see section below.
- Delete the tracelog file (.txt) from the /flash: location to ensure enough space on the device for other operations.

Copy tracefiles off the box

This is an example of the tracefile:

```
Device# dir crashinfo:/tracelogs
Directory of crashinfo:/tracelogs/
50664 -rwx 760 Sep 22 2015 11:12:21 +00:00 plogd_F0-0.bin_0.gz
50603 -rwx 991 Sep 22 2015 11:12:08 +00:00 fed_pmanlog_F0-0.bin_0.9558.20150922111208.gz
50610 -rw- 11 Nov 2 2015 00:15:59 +00:00 timestamp
50611 -rwx 1443 Sep 22 2015 11:11:31 +00:00
auto_upgrade_client_sh_pmanlog_R0-.bin_0.3817.20150922111130.gz
50669 -rwx 589 Sep 30 2015 03:59:04 +00:00 cfgwr-8021_R0-0.bin_0.gz
50612 -rwx 1136 Sep 22 2015 11:11:46 +00:00 reflector_803_R0-0.bin_0.1312.20150922111116.gz

               50794 -rwx 4239 Nov 2 2015 00:04:32 +00:00
IOSRP_R0-0.bin_0.14239.20151101234827.gz
50615 -rwx 131072 Nov 2 2015 00:19:59 +00:00 linux_iosd_image_pmanlog_R0-0.bin_0
```

The trace files can be copied using one of the various options shown below:

```
Device# copy crashinfo:/tracelogs ?
  crashinfo: Copy to crashinfo: file system
  flash: Copy to flash: file system
  ftp: Copy to ftp: file system
  http: Copy to http: file system
  https: Copy to https: file system
  null: Copy to null: file system
  nvram: Copy to nvram: file system
  rcp: Copy to rcp: file system
  running-config Update (merge with) current system configuration
  scp: Copy to scp: file system
  startup-config Copy to startup configuration
  syslog: Copy to syslog: file system
  system: Copy to system: file system
  tftp: Copy to tftp: file system
  tmpsys: Copy to tmpsys: file system
```

The general syntax for copying onto a TFTP server is as follows:

```
Device# copy source: tftp:
Device# copy crashinfo:/tracelogs/IOSRP_R0-0.bin_0.14239.20151101234827.gz tftp:
Address or name of remote host []? 2.2.2.2
Destination filename [IOSRP_R0-0.bin_0.14239.20151101234827.gz]?
```

Note Clear the generated report and archive files from the switch to ensure that sufficient flash space remains available for tracelog and other uses.

Configuration examples for conditional debugging

This is an output example of the **show platform condition** command.

```
Device# show platform condition
Conditional Debug Global State: Stop
Conditions Direction
------------------------------------------------------------------------------------------|---------
MAC Address 0024.D7C7.0054 N/A
Feature Condition Type Value
----------------------|-----------------------|--------------------------------
```

This is an output example of the **show debug** command.

```
Device# show debug
IOSXE Conditional Debug Configs:
Conditional Debug Global State: Start
Conditions Direction
------------------------------------------------------------------------------------------|---------
MAC Address 0024.D7C7.0054 N/A
Feature Condition Type Value
----------------------|-----------------------|--------------------------------
Packet Infra debugs:
Ip Address Port
------------------------------------------------------|----------
```

Verify conditional debugging

The table shown below lists the various commands that can be used to verify conditional debugging:

Command	Purpose
show platform condition	Displays the current conditions set.
show debug	Displays the current debug conditions set.
show platform software trace filter-binary	Displays logs merged from the latest tracefile.
request platform software trace filter-binary	Displays historical logs of merged tracefiles on the system.

Example: verify radioactive tracing log for SISF

This is an output example of the **show platform software trace message ios chassis active R0 | inc sisf** command.

```
Device# show platform software trace message ios chassis active R0 | inc sisf
2017/10/26 13:46:22.104 {IOSRP_R0-0}{1}: [parser]: [5437]:  UUID: 0, ra: 0 (note):  CMD:
'show platform software trace message ios switch active R0 | inc sisf' 13:46:22 UTC Thu Oct
 26 2017
2017/10/26 13:46:10.667 {IOSRP_R0-0}{1}: [sisf]: [5437]:  UUID: 4800000000060, ra: 7 (debug):
   FF8E802918 semaphore system unlocked
2017/10/26 13:46:10.667 {IOSRP_R0-0}{1}: [sisf]: [5437]:  UUID: 4800000000060, ra: 7 (debug):
  Unlocking, count is now 0
2017/10/26 13:46:10.667 {IOSRP_R0-0}{1}: [sisf]: [5437]:  UUID: 4800000000060, ra: 7 (debug):
   FF8E802918 semaphore system unlocked
2017/10/26 13:46:10.667 {IOSRP_R0-0}{1}: [sisf]: [5437]:  UUID: 4800000000060, ra: 7 (debug):
  Unlocking, count is now 1
2017/10/26 13:46:10.667 {IOSRP_R0-0}{1}: [sisf]: [5437]:  UUID: 4800000000060, ra: 7 (debug):
  Gi1/0/5 vlan 10 aaaa.bbbb.cccc Setting State to 2
2017/10/26 13:46:10.667 {IOSRP_R0-0}{1}: [sisf]: [5437]:  UUID: 4800000000060, ra: 7 (debug):
  Gi1/0/5 vlan 10 aaaa.bbbb.cccc Start timer 0
2017/10/26 13:46:10.667 {IOSRP_R0-0}{1}: [sisf]: [5437]:  UUID: 4800000000060, ra: 7 (debug):
  Gi1/0/5 vlan 10 aaaa.bbbb.cccc  Timer value/granularity for 0 :299998/1000
2017/10/26 13:46:10.667 {IOSRP_R0-0}{1}: [sisf]: [5437]:  UUID: 4800000000060, ra: 7 (debug):
  Gi1/0/5 vlan 10 aaaa.bbbb.cccc Updated Mac Timer : 299998
2017/10/26 13:46:10.667 {IOSRP_R0-0}{1}: [sisf]: [5437]:  UUID: 4800000000060, ra: 7 (debug):
  Gi1/0/5 vlan 10 aaaa.bbbb.cccc Before Timer :  350000
2017/10/26 13:46:10.667 {IOSRP_R0-0}{1}: [sisf]: [5437]:  UUID: 4800000000060, ra: 7 (debug):
  Gi1/0/5 vlan 10 aaaa.bbbb.cccc Timer 0, default value is 350000
2017/10/26 13:46:10.667 {IOSRP_R0-0}{1}: [sisf]: [5437]:  UUID: 4800000000060, ra: 7 (debug):
   Allocating timer wheel for 0
2017/10/26 13:46:10.667 {IOSRP_R0-0}{1}: [sisf]: [5437]:  UUID: 4800000000060, ra: 7 (debug):
  Gi1/0/5 vlan 10 aaaa.bbbb.cccc No timer running
2017/10/26 13:46:10.667 {IOSRP_R0-0}{1}: [sisf]: [5437]:  UUID: 4800000000060, ra: 7 (debug):
  Granularity for timer MAC_T1  is 1000
2017/10/26 13:46:10.667 {IOSRP_R0-0}{1}: [sisf]: [5437]:  UUID: 4800000000060, ra: 7 (debug):
  Gi1/0/5 vlan 10 aaaa.bbbb.cccc Current State :MAC-STALE, Req Timer : MAC_T1  Current Timer
 MAC_T1
```