Conditional Debug and Radioactive Tracing

- Introduction to Conditional Debugging, on page 1
- Introduction to Radioactive Tracing, on page 2
- How to Configure Conditional Debug and Radioactive Tracing, on page 2
- Monitoring Conditional Debugging, on page 6
- Configuration Examples for Conditional Debugging, on page 6
- Additional References for Conditional Debugging and Radioactive Tracing, on page 7
- Feature History for Conditional Debugging and Radioactive Tracing, on page 7

Introduction to Conditional Debugging

The Conditional Debugging feature allows you to selectively enable debugging and logging for specific features based on the set of conditions you define. This feature is useful in systems where a large number of features are supported.

Note Only Control Plane Tracing is supported.

The Conditional debug allows granular debugging in a network that is operating at a large scale with a large number of features. It allows you to observe detailed debugs for granular instances within the system. This is very useful when we need to debug only a particular session among thousands of sessions. It is also possible to specify multiple conditions.

A condition refers to a feature or identity, where identity could be an interface, IP Address, or a MAC address and so on.

Note MAC address is the only supported condition.

This is in contrast to the general debug command, that produces its output without discriminating on the feature objects that are being processed. General debug command consumes a lot of system resources and impacts the system performance.

Introduction to Radioactive Tracing

Radioactive tracing provides the ability to stitch together a chain of execution for operations of interest across the system, at an increased verbosity level. This provides a way to conditionally print debug information (up to DEBUG Level or a specified level) across threads, processes and function calls.

Note The default level is **DEBUG**. The users cannot change this to another level.

The following features are enabled for Radioactive Tracing:

- IGMP Snooping
- Layer 2 Multicast

How to Configure Conditional Debug and Radioactive Tracing

Conditional Debugging and Radioactive Tracing

Radioactive Tracing when coupled with Conditional Debugging, enable us to have a single debug CLI to debug all execution contexts related to the condition. This can be done without being aware of the various control flow processes of the feature within the box and without having to issue debugs at these processes individually.

Location of Tracefiles

By default the tracefile logs will be generated for each process and saved into either the **/tmp/rp/trace** or **/tmp/fp/trace** directory. In this temp directory, the trace logs are written to files, which are of 1 MB size each. 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 whatever size was configured for it during the boot time, it is rotated out to an archive location in the **/crashinfo** partition under **tracelogs** directory.

The **/tmp** directory holds only a single tracefile for a given process. Once the file reaches its file size limit it is rotated out to **/crashinfo/tracelogs**. In the archive directory, up to 25 files are accumulated, after which the oldest one is replaced by the newly rotated file from **/tmp**.

The tracefiles in the crashinfo directory are located in the following formats:

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

   Example: IOSRP_R0-0.bin_0.14239.20151101234827.gz

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

   Example: wcm_pmanlog_R0-0.30360_0.20151028233007.bin.gz

Configuring Conditional Debugging

To configure conditional debugging, follow the steps given below:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**debug platform condition mac** {*mac-address*} **Example:** `Device# debug platform condition mac bc16.6509.3314`	Configures conditional debugging for the MAC Address specified.
Step 3	**debug platform condition start** **Example:** `Device# debug platform condition start`	Starts conditional debugging (this will start radioactive tracing if there is a match on one of the conditions above).

Step 4	**show platform condition** OR **show debug**
	Example:
	`Device# show platform condition`
	`Device# show debug`
		Displays the current conditions set.

Step 5	**debug platform condition stop** **Example:** `Device# debug platform condition stop`	Stops conditional debugging (this will stop radioactive tracing).
Step 6	**request platform software trace archive** [**last** {*number*} **days**] [**target** {*crashinfo:* | *flashinfo:*}] **Example:** `# request platform software trace archive last 2 days`	(Optional) Displays historical logs of merged tracefiles on the system. Filter on any combination of number of days or location.
Step 7	**show platform software trace** [**filter-binary** | **level** | **message**] **Example:** `Device# show platform software trace message`	(Optional) Displays logs merged from the latest tracefile. Filter on any combination of application condition, trace module name, and trace level. • **filter-binary** - Filter the modules to be collated • **level** - Show trace levels

	Command or Action	Purpose
		• **message** - Show trace message ring contents **Note** On the device: • Available from IOS console in addition to linux shell. • Generates a file with merged logs. • Displays merged logs only from staging area
Step 8	**clear platform condition all** **Example:** `Device# clear platform condition all`	Clears all conditions.

What to do next

Note The commands **request platform software trace filter-binary** and **show platform software trace filter-binary** work in a similar way. The only difference is:

- **request platform software trace filter-binary** - Sources the data from historical logs.
- **show platform software trace filter-binary** – Sources the data from the flash Temp directory.

Of these, *mac_log <..date..>* is the most important file, as it gives the messages for the MAC we are debugging. The command **show platform software trace filter-binary** also generates the same flash files, and also prints 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 mac *client MAC address* **ip** *Group IP address* **vlan** *id* **level** *debug level*

Recommended Workflow for Trace files

The Recommended Workflow for Trace files is listed below:

1. To request the tracelogs for a specific time period.

   EXAMPLE 1 day.

Use the command:

Device#**request platform software trace archive last 1 day**

2. The system generates a tar ball (.gz file) of the tracelogs in the location /flash:
3. Copy the file off the switch. By copying the file, the tracelogs can be used to work offline. For more details on copying files, see section below.
4. Delete the tracelog file (.gz) file from /flash: location. This will ensure enough space on the switch for other operations.

Copying tracefiles off the box

An example of the tracefile is shown below:

```
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
--More—
```

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 It is important to clear the generated report or archive files off the switch in order to have flash space available for tracelog and other purposes.

Monitoring Conditional Debugging

The table shown below lists the various commands that can be used to monitor 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.

Configuration Examples for Conditional Debugging

The following 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
----------------------|----------------------|--------------------------------
Device#
```

The following 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
------------------------------------------------------|----------
Device#
```

The following is a sample of the *debug platform condition stop* command.

```
Device# debug platform condition stop
Conditional Debug Global State: Stop
```

Additional References for Conditional Debugging and Radioactive Tracing

Related Documents

Related Topic	Document Title
For complete syntax and usage information for the commands used in this chapter.	*Command Reference (Catalyst 9600 Series Switches)*

Feature History for Conditional Debugging and Radioactive Tracing

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	Conditional Debugging and Radioactive Tracing	The Conditional Debugging feature allows you to selectively enable debugging and logging for specific features based on the set of conditions you define.
Cisco IOS XE Cupertino 17.7.1	Conditional Debugging and Radioactive Tracing	Support for this feature was introduced on the Cisco Catalyst 9600 Series Supervisor 2 Module.

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.