Conditional Debug and Radioactive Tracing

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Introduction to Conditional Debugging

The Conditional Debugging feature allows you to selectively enable debugging and logging for the 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 On Cisco IOS 16.1, only Control Plane Tracing is supported.

The Conditional debug allows you to debug in a very granular fashion in a network that is operating at scale even with a large number of features deployed. 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 In Cisco IOS Release 16.1 the only condition that is supported is MAC address. The support for other features will be introduced in the releases that follow.

This is in contrast to the general debug command, that produces its output without discriminating on the feature objects that are being processed, which consumes 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. It provides a way to conditionally print debug information (upto DEBUG Level or upto a specified debug level) across threads, processes and function calls.

Note In Cisco IOS Release 16.1 the default level is **DEBUG**. The users cannot change this to another level. The support for other levels will be introduced in the releases that follow.

Conditional Debugging and Radioactive Tracing

Radioactive Tracing when coupled with Conditional Debugging, enable us to have a single debug CLI to debug all executions contexts related to the condition without having to be aware of the various processes within the box that were part of a the control flow for a feature 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 upto 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 at boot time, it is rotated out to an archive location under the **/crashinfo** partition under a **tracelogs** directory.

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

The tracefiles are located in the crashinfo directory 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:

-

SUMMARY STEPS

1. **enable**
2. **debug platform condition mac** {*mac-address*}
3. **debug platform condition start**
4. **show platform condition** OR **show debug**
5. **debug platform condition stop**
6. **request platform software trace archive** [**last** {*number*} **days**] [**target** {*crashinfo:* | *flashinfo:*}]
7. **request platform software trace filter-binary** {*wire* | *wireless*} [**context** {*mac-address*} | **level** | **module**]
8. **show platform software trace** [**filter-binary** | **level** | **message**]
9. **clear platform condition all**

DETAILED STEPS

	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.

	Command or Action	Purpose
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:** `Device# 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	**request platform software trace filter-binary** {*wire* | *wireless*} [**context** {*mac-address*} | **level** | **module**] **Example:** `Device# request platform software trace filter-binary wireless context bc16.6509.3314`	(Optional) Filter the modules to collate the information (wire or wireless) and then on the context of Mac address specified. These logs can be viewed off-line. **Note** In Cisco IOS Release 16.1, from all the keywords available, the only keyword supported is wireless. This collects files from processes (ios, wcm, fman_rp, fman_fp, fed).
Step 8	**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 collated, eg. wireless • **level** - Show trace levels • **message** - Show trace message ring contents **Note** On Box: • Available from IOS console in addition to linux shell. • Generates a file with merged logs on the box. • Displays merged logs only from staging area
Step 9	**clear platform condition all** **Example:** `Device# clear platform condition all`	Clear 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.

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

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

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.

Recommended Workflow for Trace files

The Recommended Workflow for Trace files is listed below:

1. 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 box. By copying the file, the tracelogs can be used to work offline. For more details on copying files, refer section below.
4. Delete the tracelog file (.gz) file from /flash: location. This ensures that there is enough space on the box 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 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 switch of these generated report/archive files in order to have flash space available for tracelogs and other purposes.

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
```

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.