HSI Data Collection for Technical Support Service Requests

Document ID: 50921

Contents

Introduction

When you open a Service Request with Cisco Technical Support, some preliminary information is required to better identify and qualify the issue. Some of this information is always required; other information requirements depend on the nature of the issue. If you wait to collect this information until after you have opened a Service Request and an engineer asks for it, then it is inevitable that there will be a delay in resolution.

Thus, the primary goal of this document is to identify the required preliminary information, based on the type of issue, so that it can be provided to the engineer immediately. The secondary goal of this document is to provide you with general guidelines to follow when you collect information for Cisco Technical Support, to avoid repetitive testing and recollection of identical data.

This document is intended for Cisco customers that support Voice Signaling Solutions based on the Cisco H.323 Signaling Interface (HSI) system and the Cisco PGW 2200 (formerly called SC 2200 and VSC 3000, or Cisco Telephony Controller, or Media Gateway Controller).

Prerequisites

Requirements

Readers of this document should have knowledge of these topics:

- H.323 Signaling Interface Guide
- Cisco Media Gateway Controllers

Components Used

The information in this document is based on the H.323 Signaling Interface (version 2.x or higher) and the PGW Media Gateway Controller (version 9).

Conventions

Commands that are shown in this document might appear prefixed by one of these prompts, which give an indication of the application environment in which the command must be executed:

Prompt	Environment
`%`	UNIX csh−shell prompt. This is the default command−line interface (CLI) prompt for the mgcusr UNIX account after login.
`#`	UNIX root−level shell prompt. This is the default CLI prompt for the root user. Issue the **su** UNIX command to get there.
`mml>`	Man−Machine Language (MML) application prompt. Issue the **mml** command from the csh−shell prompt to get there.

For more information on document conventions, refer to the Cisco Technical Tips Conventions.

Standard Required Information

For *all* HSI−related issues, this standard information should *always* be provided immediately to Cisco Technical Support:

1. Problem description
2. General information
3. HSI system information
4. HSI current configuration

Problem Specific Information

Depending on the nature of the problem, additional information might be needed. This document considers these problem types:

- HSI call−related issues
- HSI application core dump

Call−Related Issues

This information should be provided, if the problem involves a call through the HSI:

1. Standard required information
2. HSI UNIX snoop trace
3. HSI application trace

Because the HSI interworks closely with the PGW, this information must be collected on the PGW as well:

1. PGW Cisco Snooper (PTC−MT) trace
2. PGW system information
3. PGW current configuration
4. PGW (MDL) call trace

If the HSI is talking to a Cisco IOS® software H.323 endpoint, this information should be collected:

1. H.323 endpoint system information
2. H.323 endpoint debug command ouput

Note: Requested traces and debugs should be gathered *simultaneously* and for *one and the same call*! This simplifies the correlation of events between different components. Failure to do so might result in a new, duplicate request for information from the Technical Support engineer and will inevitably lead to a delay in resolution.

HSI Application Core Dump

This information must be provided, if the HSI application has suffered a core dump:

1. Standard required information
2. HSI core file
3. **pstack** and **pmap** command output

Preparing File Attachments

When you prepare files to be submitted to a Service Request or to a Technical Support engineer, you should try to compress (and bundle) the files beforehand.

To simplify this task, create a temporary subdirectory into which to copy the relevant files. For example, issue the **mkdir /var/tmp/ciscotac** command to make a temporary subdirectory called **ciscotac**. Then, use one of these methods to compress the files:

- All files compressed together:

```
% cd /var/tmp/ciscotac

% tar cf − . | compress > ../files4tac.tar.Z

!−−− This method creates one archive file in the parent directory
!−−− (so that tar fdoes not archive its own archive) that contains
!−−− all of the files from /var/tmp/ciscotac.
!−−− If you have gzip installed, you may replace compress with gzip.
```

- Each file compressed individually:

```
% cd /var/tmp/ciscotac

% compress *

!--- If you issue compress (or gzip) by itself, it compresses and
!--- replace each file individually, instead of creating a single
!--- archive file. This is useful if the previous method would result
!--- in an archive file that is too large to upload.
!--- For core dump files, always use this method.
```

Once the archive file or files have been provided to the Technical Support engineer, remove them (and the temporary directory) from your file system.

Information Details

This section provides specifics and detailed procedures about the information to be gathered.

Problem Description

Provide step−by−step details of that actions that the user performs when the issue occurs. The detailed information should also include these items:

- Expected behavior
- Detailed observed behavior
- If the problem involves a call:
    - ♦ Calling and called party number, and any other numbers that might be involved in the call scenario
    - ♦ Call direction (Identify the originating and terminating call signalling protocols. For example: SS7 to H.323.)
- Copied and pasted contents of any error messages seen
- Is the issue reproducible?
- What is the frequency of the problem?
- Does the behavior change based on call direction, software release, components used, or anything else? In other words, is the same functionality known to work correctly when you use a different set of variables?

General Information

Provide this general information:

- Product hardware and software release identification of *all* components involved.
- Topology. This can be graphical or written and should at least include any components involved in the call path and their IP addresses.
- Network deployment status:
    - ♦ Is this a new installation?
    - ♦ Is this a lab (test) environment?
    - ♦ Is this a production network? If so:
        - ◊ When was the first occurrence of the problem?
        - ◊ What recent changes have been made to the components involved?

HSI System Information

Collect the output from these MML commands:

```
mml> rtrv-ne

mml> rtrv-ne-health
```

Issue this UNIX command to see the HSI patch level:

```
% ls /opt/GoldWing/currentPM/bin/*main*
```

HSI Current Configuration

This text file contains the HSI s current configuration:

```
/opt/GoldWing/currentPM/var/prov/active_config/session.dat
```

Alternatively, you can issue the **rtrv−config** MML command to capture the current configuration.

HSI UNIX Snoop Trace

Snoop is a packet sniffer tool that comes standard bundled with Solaris.

As root, issue this command before you make a test call:

```
snoop -d interface -o fail.snoop

!--- interface is the relevant interface name and fail.snoop is
!--- the file name of the trace file that you want to write.
```

Now, make the test call. You should see the packet count go up. Press **Ctrl+C** after you end the call.

Submit the fail.snoop file to the Service Request or to the Technical Support engineer (see Preparing File Attachments).

Tip: Issue the **/sbin/ifconfig −a** command if you are unsure about the *interface* name.

HSI Application Trace

Follow this procedure to capture an application trace to a file.

1. Enable logging via MML:

```
mml> set-log:eisup:level=0xffff

mml> set-log:callcontrol:level=0xffff

mml> set-log:h323:level=0xffff

mml> radlog::start

mml> quit
```

2. Empty the platform.log file prior to the test:

```
% cd /opt/GoldWing/currentPM/var/log
```

```
% log_erase

!--- This command purges the platform.log file. Source the
!--- /opt/GoldWing/currentPM/local/setup.gw.csh file, if this
!--- command is not recognized.
```

3. Make the test call.
4. When finished, save a copy of the platform.log file and disable logging:

```
% cp platform.log fail.log

mml> set-log:all:level=0x0000

mml> radlog::stop

mml> quit
```

5. Submit the fail.log file to the Service Request or to the Technical Support engineer (see Preparing File Attachments).

PGW Cisco Snooper (PTC−MT) Trace

Cisco Snooper is a Cisco internal packet sniffer tool. PTC−MT is the commercialized version of Snooper.

To get a trace in ASCII format, issue these commands on the PGW:

```
# cd snooper_directory


# ./snooper int interface_x ss7 nosnts mgcp noauep eisup detail hex >
  fail_interface_x.snooper &

!--- This command must be issued on one line.
!--- Issue this command for every redundant interface (interface_x)
!--- in the PGW. fail_interface_x.snooper is the file name of the
!--- ASCII trace file that you want to write.
```

Alternatively, to get the trace in binary format, issue these commands:

```
# cd snooper_directory


# ./snooper int interface_x file fail_interface_x.snooper &

!--- Issue this command for every redundant interface (interface_x)
!--- in the PGW. fail_interface_x.snooper is the file name of the
!--- binary trace file that you want to write.
```

Once you have captured the test call, do not forget to kill the Snooper processes.

Submit the fail_*interface_x*.snooper file or files to the Service Request or to the Technical Support engineer (see Preparing File Attachments).

Notes:

- Make sure that the seedfile.txt file is correctly configured!
- If you have chosen to collect a binary trace, do not forget to forward the seedfile.txt file as well.

- If you are using PTC−MT instead of Snooper, replace **snooper** in the previous commands with **ptcmt** and replace **nosnts** with **nomtm**.
- If you do not have Snooper or PTC−MT installed, use the UNIX snoop tool instead.

Cisco PGW System Information

Collect the output from these MML commands:

```
mml> rtrv−ne

mml> rtrv−ne−health
```

This UNIX shell−command displays the installed MGC patches:

```
% pkginfo −i | grep CSC
```

PGW Current Configuration

This MML command exports the PGW s current configuration:

```
mml> prov−exp:all:dirname="directory−name"
```

All of the files that result from that command are stored in the /opt/CiscoMGC/etc/cust_specific/*directory−name* directory.

Submit the files to the Service Request or to the Technical Support engineer (see Preparing File Attachments).

PGW (MDL) Call Trace

The MDL call trace should be run as briefly as possible. This keeps the possible impact on system performance as low as possible; and it also limits the number of calls in the trace, as much as possible, to the relevant call only. Multiple calls in the trace is not desirable, as it makes it more complicated to locate the relevant call.

1. Start the call trace:

   ```
   mml> sta−sc−trc:sig−path:confirm

   !−−− sig−path is the call s incoming signaling path.
   ```

   Note: Issue the **rtrv−dest:all** command to determine *sig−path*.
2. Make the test call.
3. Stop the call trace:

   ```
   mml> stp−sc−trc:all

   !−−− Note the BTR file name that is displayed at this point.

   mml> quit
   ```
4. Start the get_trc.sh script:

   ```
   % cd /opt/CiscoMGC/var/trace

   % get_trc.sh filename.btr

   !−−− filename.btr is the file name that was displayed when you
   ```

```
!--- stopped the trace.
```

5. If multiple calls are present in the trace, navigate first to the relevant call ID by issuing the **N**, **P**, or **id** commands.
6. Issue the **C** command to write out a trace of this call, in ASCII format, to a TRC file.
7. Submit the TRC file to the Service Request or to the Technical Support engineer (see Preparing File Attachments).

For more details about this procedure, refer to the Tracing section of the PGW documentation.

H.323 Endpoint System Information

From within enable mode, collect the output of these commands:

```
show version

show running-config
```

If the device is non−Cisco IOS, try to get similar information.

H.323 Endpoint debug Command Output

If system load permits it, collect the output of the next list of **debug** commands for a faulty call.

Note: Before you issue these **debug** commands, make sure that you have millisecond timestamps and sequence−numbers enabled in the configuration:

```
service timestamps debug datetime msec

service timestamps log datetime msec

service sequence-numbers

debug cch323 session

debug cch323 h225

debug cch323 h245

debug h225 asn1

debug h225 asn1 errors

debug h225 events

debug h225 q931

debug h245 asn1

debug h245 asn1 errors

debug h245 events
```

If the device is non−Cisco IOS, try to get similar debug information.

HSI Core File

If the HSI application crashes, a core dump is written to a file with this file name format:

```
/opt/GoldWing/currentPM/bin/core_timestamp

!--- timestamp is in the form YYYYMMDDhhmmss.
```

Core file or files must be compressed separately before you submit them to the Service Request or to the Technical Support engineer (see Preparing File Attachments).

pstack and pmap Command Output

Issue these UNIX commands for the HSI core file:

```
# cd /opt/GoldWing/currentPM/bin

# ls -l core_*

# pstack core_file > core_file.proc

# pmap core_file >> core_file.proc

!--- core_file is the core dump file name that you retrieved
!--- with the ls -l core_* command.
```

Submit the *core_file*.proc file to the Service Request or to the Technical Support engineer (see Preparing File Attachments).

Related Information

- **Cisco PGW 2200 Softswitch Tech Notes**
- **Voice Technology Support**
- **Voice and Unified Communications Product Support**
- **Troubleshooting Cisco IP Telephony**
- **Technical Support − Cisco Systems**

Updated: Feb 02, 2006 Document ID: 50921