Configuration Note

Multi-Service Business Routers Product Series

Mediant MSBR

Troubleshooting

Version 6.8

HD VoIP Sounds Better

AudioCodes

Table of Contents

This page is intentionally left blank.

Trademarks

WEEE EU Directive

Pursuant to the WEEE EU Directive, electronic and electrical waste must not be disposed of with unsorted waste. Please contact your local recycling authority for disposal of this product.

Customer Support

Customer technical support and services are provided by AudioCodes or by an authorized AudioCodes Service Partner. For more information on how to buy technical support for AudioCodes products and for contact information, please visit our Web site at www.audiocodes.com/support.

Abbreviations and Terminology

Each abbreviation, unless widely used, is spelled out in full when first used.

Document Revision Record

LTRT	Description
40380	Initial document release.
40381	Update to Chapters 3 and 7.
40382	Update CPU memory and utilization history.
40383	Update to Capturing Data-CPU on Physical Interfaces and Sending Information to Head Office.

Documentation Feedback

AudioCodes continually strives to produce high quality documentation. If you have any comments (suggestions or errors) regarding this document, please fill out the Documentation Feedback form on our Web site at http://www.audiocodes.com/downloads.

1 Introduction

This document describes the basic system configuration for AudioCodes Multi-Service Business Routers (MSBR), using the CLI management interface.

This page is intentionally left blank.

2 Capturing Packets

The MSBR has built-in packet capturing capabilities used for debugging the MSBR and providing an instant overview of current network traffic.

Two types of packet capturing can be performed on the MSBR:

- **Physical** – on the wire packet capturing
- **Interface** – selective interface and protocol configuration to monitor unique traffic.

The captured files are saved to a PCAP file stored in the MSBR RAM and does not survive a reset. The PCAP file size is limited to 20 MB.

The file can be exported to several destinations using FTP, TFTP, SSH or you can save the file to USB storage connected to the MSBR.

2.1 Capturing Data-CPU on Physical Interfaces

Debug capture physical provides the capability of monitoring packets on the wire. The following commands can be used to capture traffic on a physical interface.

Table 2-1: Capturing Traffic on Physical Interface Commands

Command	Description
`debug capture data physical eth-lan`	Sets the Ethernet interface as a source for capturing packets.
`debug capture data physical target tftp`	Sets the destination for the captured packet file as a TFTP server.
`debug capture data physical start`	Starts capturing files. **Note:** The captured data is collected locally, and sent to the PC later on.
`debug capture data physical stop <tftp server IP> vrf <name>`	Stops capturing files and then uploads the file to a TFTP server with IP address 192.168.0.3. It is possible to declare the VRF as the source (Optional).
`debug capture data physical insert-pad`	Inserts a PAD packet immediately. Used for marking a time stamp in the debug file to ease the file analysis.

The available sources for file captures are listed below:

Table 2-2: Sources for File Captures

Source	Description
cellular-wan	Defines Cellular WAN interfaces.
eth-lan	Defines LAN Ethernet interfaces.
eth-wan	Defines WAN Ethernet interfaces.
fiber-wan	Defines WAN fiber interfaces.
xdsl-wan	Defines any DSL interface (ADSL, VDSL) that is installed on the MSBR.
t1-e1-wan	Defines E1/T1 packet capture.
shdsl-wan	Defines WAN SHDSL interfaces.
eth-lan vlan4001	Defines internal link between Data and Voice-CPUs.
wifi-lan	Defines the Wi-Fi interface.

2.1.1 Changing the Debug File Destination Target

The target must be set prior to running the **start** command. The following commands set the debug file destination target:

Table 2-3: Debug File Destination Target Commands

Source	Description
`debug capture data physical target tftp`	Sends the capture to the TFTP server.
`debug capture data physical target usb`	Saves the capture to USB storage.
`debug capture data physical target ftp`	Uses an FTP server (specify username and password).

Note: If you are using the USB as a target destination, the USB storage must be plugged in to the MSBR.

While the target destination is the USB, the debug capture file size is limited to the USB storage capacity. The 20MB limitation is for TFTP and FTP target files. Using the USB as a debug capture target causes degradation in MSBR performance and should be used only for debugging purposes.

2.1.2 Viewing Currently Configured Capture

Use the following command to view the currently configured capture.

Table 2-4: Viewing Currently Configured Capture Command

Source	Description
`debug capture data physical show`	Displays currently configured capture.

2.1.3 Debugging Capture Physical using WinSCP Example

The following example captures data from the Ethernet interface on the LAN and WAN sides and sends it to the PC using SSH-WINSCP (Windows Secure CoPy) software.

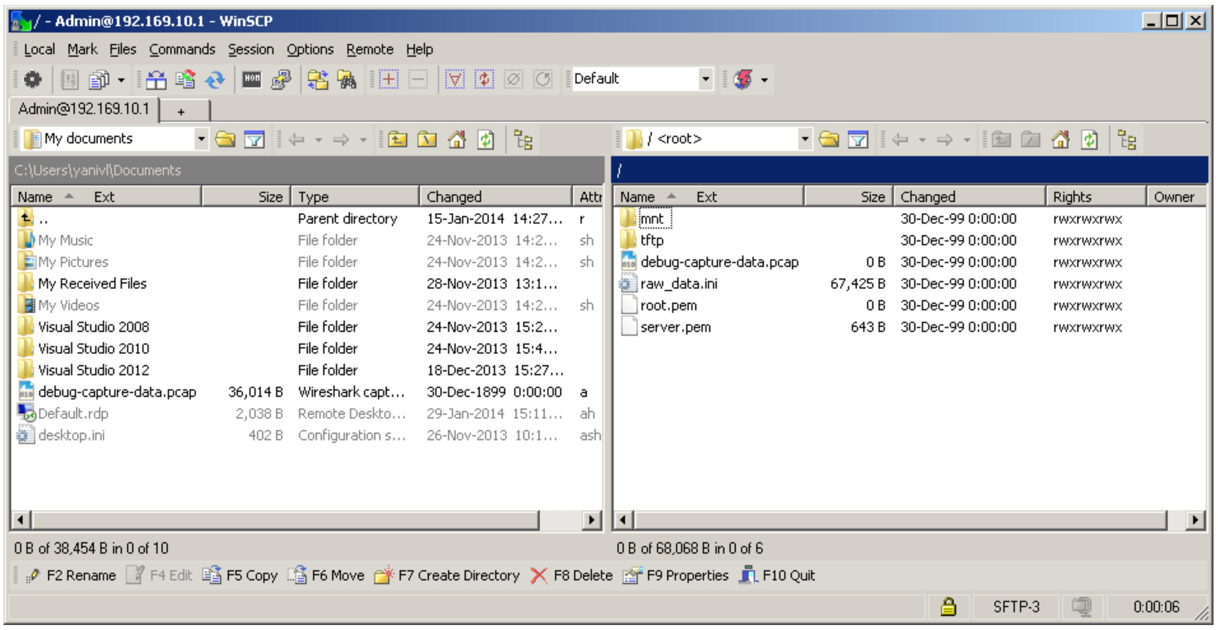
➢ **To capture physical packets using WinSCP:**

1. Activate SSH on MSBR; a reset might be required.
2. If the SSH terminal is located on the WAN side, allow SSH to be accessed from the WAN: a reset might be required.
3. Connect to the MSBR using WinSCP.
4. Copy the debug capture file to the local PC.

```
# debug capture data physical eth-lan
Interface eth-lan was added to the debug capture rules
      Use start command in order to start the debug capture
# debug capture data physical eth-wan
Interface eth-wan was added to the debug capture rules
      Use start command in order to start the debug capture
# debug capture data physical start
….. Capture packets…..
# debug capture data physical stop
```

The figure below shows the copying of the debug file from MSBR to PC using WinSCP.

Figure 2-1: Copying Debug file from MSBR to PC using WinSCP


/ - Admin@192.169.10.1 - WinSCP
Local Mark Files Commands Session Options Remote Help
Default
Admin@192.169.10.1
My documents
/ <root>

C:\Users\yanivl\Documents

Name	Ext	Size	Type	Changed	Attr
..			Parent directory	15-Jan-2014 14:27…	r
My Music			File folder	24-Nov-2013 14:2…	sh
My Pictures			File folder	24-Nov-2013 14:2…	sh
My Received Files			File folder	28-Nov-2013 13:1…	
My Videos			File folder	24-Nov-2013 14:2…	sh
Visual Studio 2008			File folder	24-Nov-2013 15:2…	
Visual Studio 2010			File folder	24-Nov-2013 15:4…	
Visual Studio 2012			File folder	18-Dec-2013 15:27…	
debug-capture-data.pcap		36,014 B	Wireshark capt…	30-Dec-1899 0:00:00	a
Default.rdp		2,038 B	Remote Deskto…	29-Jan-2014 15:11…	ah
desktop.ini		402 B	Configuration s…	26-Nov-2013 10:1…	ash

/

Name	Ext	Size	Changed	Rights	Owner
mnt			30-Dec-99 0:00:00	rwxrwxrwx	
tftp			30-Dec-99 0:00:00	rwxrwxrwx	
debug-capture-data.pcap		0 B	30-Dec-99 0:00:00	rwxrwxrwx	
raw_data.ini		67,425 B	30-Dec-99 0:00:00	rwxrwxrwx	
root.pem		0 B	30-Dec-99 0:00:00	rwxrwxrwx	
server.pem		643 B	30-Dec-99 0:00:00	rwxrwxrwx	

0 B of 38,454 B in 0 of 10 0 B of 68,068 B in 0 of 6

F2 Rename F4 Edit F5 Copy F6 Move F7 Create Directory F8 Delete F9 Properties F10 Quit
SFTP-3 0:00:06

2.1.4 Capturing Data-CPU on Physical Interface Example

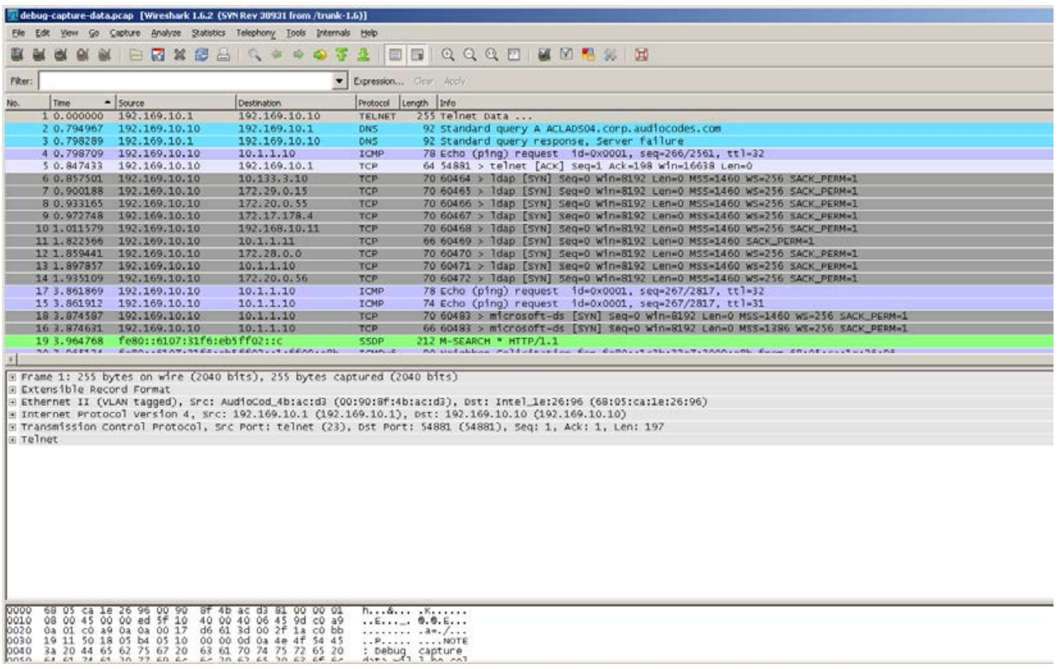
This example captures data from the Ethernet interface on the LAN side and sends it to a USB device:

```
# debug capture data physical eth-lan
Interface eth-lan was added to the debug capture rules
      Use start command in order to start the debug capture
# debug capture data physical eth-wan
Interface eth-wan was added to the debug capture rules
      Use start command in order to start the debug capture

# debug capture data physical target usb
# debug capture data physical start
Saving capture to USB storage.
File name: debug-capture-data-16032014-155634.pcap

# debug capture data physical stop
Finished. Type "usb remove" to safely remove the drive.

# usb remove
You may now remove the USB drive
```

2.1.5 Analyzing the Debug File

Since the file is aggregated from a few sources (i.e., LAN and WAN interfaces), it is important to sort the debug file according to the time the packets arrived.

Figure 2-2: Debug File Sorted by Packet Arrival Time


debug-capture-data.pcap [Wireshark 1.6.2 (SVN Rev 38931 from /trunk-1.6)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply

```
No.  Time      Source                    Destination   Protocol Length Info
   1 0.000000  192.169.10.1              192.169.10.10 TELNET    255 Telnet Data ...
   2 0.794967  192.169.10.10             192.169.10.1  DNS        92 Standard query A ACLADS04.corp.audiocodes.com
   3 0.798289  192.169.10.1              192.169.10.10 DNS        92 Standard query response, Server failure
   4 0.798709  192.169.10.10             10.1.1.10     ICMP       78 Echo (ping) request  id=0x0001, seq=266/2561, ttl=32
   5 0.847433  192.169.10.10             192.169.10.1  TCP        64 54881 > telnet [ACK] Seq=1 Ack=198 Win=16638 Len=0
   6 0.857501  192.169.10.10             10.133.3.10   TCP        70 60464 > ldap [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
   7 0.900188  192.169.10.10             172.29.0.15   TCP        70 60465 > ldap [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
   8 0.933165  192.169.10.10             172.20.0.55   TCP        70 60466 > ldap [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
   9 0.972748  192.169.10.10             172.17.178.4  TCP        70 60467 > ldap [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
  10 1.011579  192.169.10.10             192.168.10.11 TCP        70 60468 > ldap [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
  11 1.822566  192.169.10.10             10.1.1.11     TCP        66 60469 > ldap [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1
  12 1.859441  192.169.10.10             172.28.0.0    TCP        70 60470 > ldap [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
  13 1.897857  192.169.10.10             10.1.1.10     TCP        70 60471 > ldap [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
  14 1.935109  192.169.10.10             172.20.0.56   TCP        70 60472 > ldap [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
  17 3.861869  192.169.10.10             10.1.1.10     ICMP       78 Echo (ping) request  id=0x0001, seq=267/2817, ttl=32
  15 3.861912  192.169.10.10             10.1.1.10     ICMP       74 Echo (ping) request  id=0x0001, seq=267/2817, ttl=31
  18 3.874587  192.169.10.10             10.1.1.10     TCP        70 60483 > microsoft-ds [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
  16 3.874631  192.169.10.10             10.1.1.10     TCP        66 60483 > microsoft-ds [SYN] Seq=0 Win=8192 Len=0 MSS=1386 WS=256 SACK_PERM=1
  19 3.964768  fe80::6107:31f6:eb5ff02::c              SSDP      212 M-SEARCH * HTTP/1.1

⊞ Frame 1: 255 bytes on wire (2040 bits), 255 bytes captured (2040 bits)
⊞ Extensible Record Format
⊞ Ethernet II (VLAN tagged), Src: AudioCod_4b:ac:d3 (00:90:8f:4b:ac:d3), Dst: Intel_1e:26:96 (68:05:ca:1e:26:96)
⊞ Internet Protocol Version 4, Src: 192.169.10.1 (192.169.10.1), Dst: 192.169.10.10 (192.169.10.10)
⊞ Transmission Control Protocol, Src Port: telnet (23), Dst Port: 54881 (54881), Seq: 1, Ack: 1, Len: 197
⊞ Telnet

0000  68 05 ca 1e 26 96 00 90  8f 4b ac d3 81 00 00 01   h...&... .K......
0010  08 00 45 00 00 ed 5f 10  40 00 40 06 45 9d c0 a9   ..E..._. @.@.E...
0020  0a 01 c0 a9 0a 0a 00 17  d6 61 3d 00 2f 1a c0 bb   ........ .a=./...
0030  19 11 50 18 05 b4 05 10  00 00 0d 0a 4e 4f 54 45   ..P..... ....NOTE
0040  3a 20 44 65 62 75 67 20  63 61 70 74 75 72 65 20   : Debug  capture
```

2.2 Capturing Data on Logical Interfaces

The “debug capture data interface” command is used to debug traffic on a specific interface, filtering specific protocols (ICMP/IP/UDP/TCP) and a specific host. You may choose to view the packets on the terminal or upload to TFTP.

Use the following command to capture traffic on a logical interface:

Table 2-5: Capturing Traffic on Logical Interface Command

Command	Description
debug capture data interface `<interface> <ipsec> proto <all | arp | icmp | ip | ipv6 | tcp | udp> host <IP | IPv6 | all> <cr | port> <any | 1-65535 <cr | ftp | tftp> IP`	▪ <interface>: Defines the interface to capture the data on. ▪ <proto | ipsec>: If IPSec is selected, it is decrypted and captured. ▪ <all | arp | icmp | ip | ipv6 | tcp | udp>: Selects protocol for capturing. ▪ host <IP | IPv6 | all>: Selects traffic to capture using the IP or IPv6 address as a filter. ▪ <cr | port> <any | 1 – 65535>: Selects the port to capture or press Enter. If you press Enter, the packets are displayed on the console. ▪ <cr | ftp | tftp> IP: Press Enter to display the captured packets on screen, or send captured packets to TFTP or FTP server.

2.2.1 Capturing Data on an Interface Example

The following example captures data from the multilink interface on the T1-WAN side and sends it to a TFTP server. A specific host and port range can be added to make the debug capture more specific.

```
debug capture data interface multilink 1 proto tcp host any port
any

06:29:41.793398  In ethertype IPv4 (0x0800), length 1460: (tos
0x0, ttl  54, id 58110, offset 0, flags [DF], proto: TCP (6),
length: 1444) 173.194.6.231.443 > 192.168.10.20.38927: .
4141989486:4141990878(1392) ack 1232903466 win 274
<nop,nop,timestamp 300300901 26314826>
06:29:41.844319  In ethertype IPv4 (0x0800), length 1460: (tos
0x0, ttl  54, id 58111, offset 0, flags [DF], proto: TCP (6),
length: 1444) 173.194.6.231.443 > 192.168.10.20.38927: .
1392:2784(1392) ack 1 win 274 <nop,nop,timestamp 300300908
26314828>
06:29:41.793742 Out ethertype IPv4 (0x0800), length 80: (tos 0x0,
ttl  63, id 29799, offset 0, flags [DF], proto: TCP (6), length:
64) 72.1.119.102.38927 > 173.194.6.231.443: ., cksum 0x48c6
(correct), ack 4141988094 win 3451 <nop,nop,timestamp 26314849
300300824,nop,nop,sack 1 {1393:2785}>
06:29:41.793888 Out ethertype IPv4 (0x0800), length 80: (tos 0x0,
ttl  63, id 29800, offset 0, flags [DF], proto: TCP (6), length:
64) 72.1.119.102.38927 > 173.194.6.231.443: ., cksum 0x4356
(correct), ack 1 win 3451 <nop,nop,timestamp 26314849
300300824,nop,nop,sack 1 {1393:4177}>
```

The following example captures data from the Ethernet interface on the WAN side and sends it to a TFTP server:

```
# debug capture data interface gigabitethernet 0/0 proto all host
any port any tftp-server 192.168.0.50
..............................
```

2.2.2 Looking Inside the VPN IPSec Tunnel Example

The "debug capture data interface" command allows you to monitor traffic on a specific interface. When the interface is encrypting traffic, the packets are encapsulated inside ESP packets.

```
# debug capture data interface GigabitEthernet 0/0.1000 proto all
host any

05:19:08.422033 0:90:8f:3c:81:b0 a4:4c:11:e4:22:b0 0800 126:
10.42.0.1 > 10.42.0.2: ESP(spi=0xf40c9b81,seq=0x1) (ttl 64, id
61420, len 112)
05:19:08.422446 a4:4c:11:e4:22:b0 0:90:8f:3c:81:b0 0800 126:
10.42.0.2 > 10.42.0.1: ESP(spi=0xc3415d3e,seq=0x1) (DF) (ttl 255,
id 25482, len 112)
```

Adding the IPSec flag to the "debug capture data interface" command displays the encrypted packets running in to the IPSec tunnel.

```
# debug capture data interface GigabitEthernet 0/0.1000 ipsec
proto all host any

05:19:28.191142 0:90:8f:3c:81:b0 a4:4c:11:e4:22:b0 0800 74:
192.169.20.1 > 192.169.40.1: icmp: echo request (DF) (ttl 64, id
0, len 60)
05:19:28.191454 a4:4c:11:e4:22:b0 0:90:8f:3c:81:b0 0800 74:
192.169.40.1 > 192.169.20.1: icmp: echo reply (DF) (ttl 255, id 0,
len 60)
05:19:29.185888 0:90:8f:3c:81:b0 a4:4c:11:e4:22:b0 0800 74:
192.169.20.1 > 192.169.40.1: icmp: echo request (DF) (ttl 64, id
0, len 60)
05:19:29.186700 a4:4c:11:e4:22:b0 0:90:8f:3c:81:b0 0800 74:
192.169.40.1 > 192.169.20.1: icmp: echo reply (DF) (ttl 255, id 0,
len 60)
```

2.3 Capturing Voice on Physical Interfaces

Voice-CPU has two interfaces:

- **LAN switch**
- **VLAN 4001 connected to the Data-CPU:** This interface is monitored using the "debug capture data physical eth-lan vlan4001" command.

Sometimes you may need to debug or analyze traffic coming from the internal LAN switch directly to the Voice-CPU, by-passing the Data-CPU. For example, an IP Phone that is located on the same VLAN as the Voice-CPU communicates directly without passing through the Data-CPU, while the same IP Phone that is located on a different VLAN on the LAN side, passes through the Data-CPU first (Data-CPU refers to the system Router).

Debug capture voice provides the capability of monitoring the traffic that goes directly to the Voice-CPU.

Debug capture physical is applied on all wire levels. With multiple logical interfaces (VLANs), you can monitor them all using this methodology.

2.3.1 Monitoring VoIP Physical Interface Example

The same concept is applied in "debug capture voip physical" as in "debug capture data physical" command. You need to:

- Select the interface to be monitored
- Select the target to send the files (TFTP/FTP/USB)
- Trigger the **Start** flag
- Wait for relevant traffic to be captured
- Trigger the **Stop** flag, and specify the destination IP address if using the TFTP protocol

```
debug capture voip physical eth-lan
debug capture voip physical start
…..capture….
debug capture voip physical stop <tftp server ip-adders>
```

2.4 Capturing Voice on Logical Interfaces

The “debug capture voip interface” command provides the capability of monitoring specific logical interfaces (e.g. VLANs).

The same concept is applied in the “debug capture voip interface” as in the “debug capture data interface” command.

2.4.1 Monitoring VoIP Logical Interface Example

The following is an example of how to monitor a VoIP logical interface.

```
# debug capture voip interface vlan 1 proto ip host any

08:21:16.752244 00:90:8f:1e:5a:1c > 00:90:8f:1e:5a:1b, ethertype
IPv4 (0x0800), length 66: (tos 0x0, ttl  63, id 17099, offset 0,
flags [DF], proto: TCP (6), length: 52) 192.168.10.20.44498 >
192.168.100.101.80: F, cksum 0xb4e0 (correct),
275337040:275337040(0) ack 2499340571 win 229 <nop,nop,timestamp
27988585 105130724>
```

2.4.2 Monitoring VoIP Link interface from Data-CPU to VoIP CPU Example

When monitoring the traffic from VoIP-CPU to Data-CPU using an internal link (e.g., vlan-4001), the following command should be applied:

```
# debug capture voip interface vlan 4001 proto ip host any

08:24:37.590252 00:90:8f:1e:5a:1b > 00:90:8f:1e:5a:1c, ethertype
IPv4 (0x0800), length 122: (tos 0x0, ttl  64, id 37498, offset 0,
flags [DF], proto: TCP (6), length: 108) 169.254.254.254.914 >
169.254.254.253.999: P 3010687982:3010688038(56) ack 3717629414
win 13120 <nop,nop,timestamp 105336564 105308817>
```

3 Gathering Data-CPU

The MSBR uses a distributed architecture, where VoIP-CPU and Data-CPU work together. Sometimes it is necessary to further investigate the inside of one CPU in order to debug a problem at the customer's site.

The "debug rmx-serial" command provides access to the Data-CPU logs and serial terminal for advanced debugging. RMX refers to the Data-CPU.

3.1 Debugging Data-CPU

There are two methods for debugging Data-CPU:

- Reviewing serial logs of the last five MSBR runs
- Enabling direct access to the Data-CPU serial interface without using AudioCodes' proprietary serial cable

3.1.1 Reviewing Serial Logs

A cyclic buffer of 50K runs and logs of the MSBR-Data-CPU serial interface is stored.
At each reset, a new MSBR-Data-CPU serial log is created. A total of five logs are saved. On the sixth reset, the first log is overwritten. All five logs are permanently saved and are not erased on a board reset.

These commands are available on the MSBR Serial CLI port or from Telnet/SSH access. No serial cable is required.

Table 3-1: Serial Logs Commands

Command	Description
`debug rmx-serial copy-logs-usb`	Copies all saved RMX logs to USB storage.
`debug rmx-serial list-logs`	Lists saved RMX logs.
`debug rmx-serial read-log`	Reads saved RMX logs by run number.
`debug rmx-serial profile list-logs`	Lists saved profile logs.
`debug rmx-serial profile read-log`	Reads saved profile logs by run number.

When a reset reason is triggered from the Data-CPU and you want to analyze the last MSBR-Data-CPU run, search the current available Data-CPU available logs, using the "debug rmx-serial list logs" command. In this example, you want to see the log_10.txt file.

If you want to see the current Data-CPU serial log, view log_11.txt.

```
debug rmx-serial list-logs
FILE                           SIZE
------------------------------ --------
log_7.txt                      23158
log_8.txt                      24396
log_9.txt                      21485
log_10.txt                     21971
log_11.txt                     24611
```

The Data-CPU log files can be obtained by using either one of the following options:

- WinSCP
- USB Storage
- CLI Commands

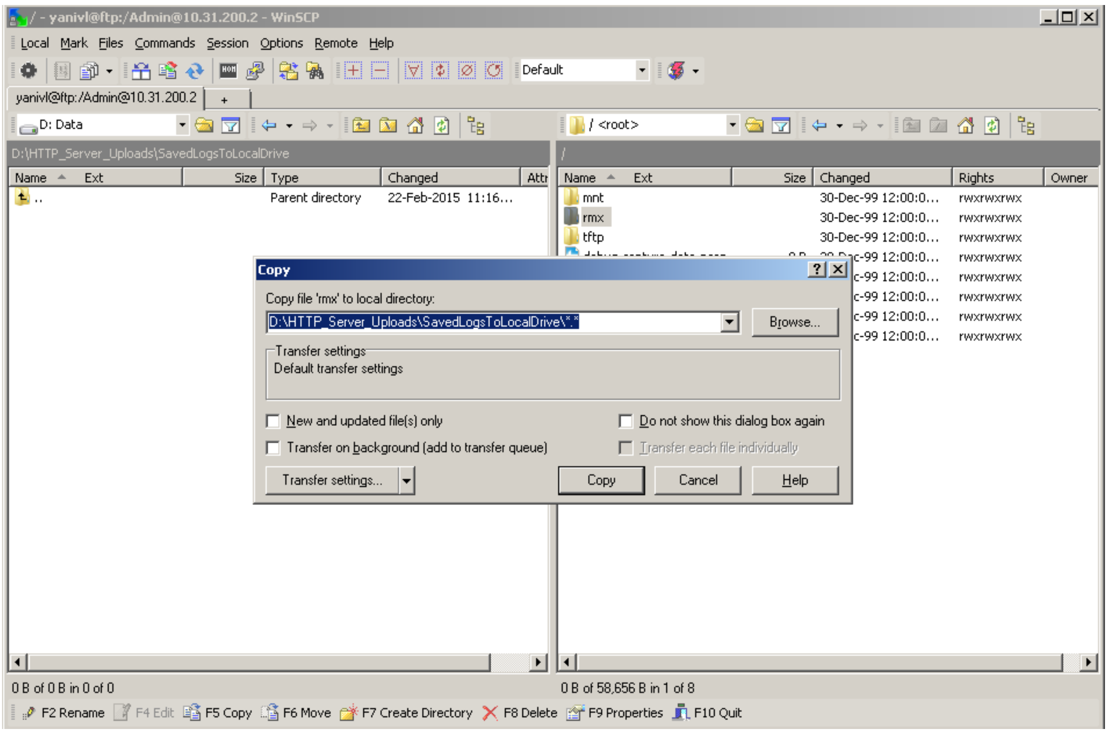
3.1.1.1 Obtaining the Data-CPU Log Files using WinSCP

The following procedure describes how to obtain the Data-CPU log files using WinSCP.

➢ **To obtain the Data-CPU log files using WinSCP:**

1. Activate SSH.
2. If you are accessing the MSBR from the WAN interface, ensure that the **wan-ssh-allow** flag is set to “on”.
3. Connect to the MSBR using WinSCP.
4. WinSCP provides copy functionality for the entire RMX directory. Right-click the **rmx** folder, and then select "copy"; the Copy window appears.
5. Click **Copy**.

Figure 3-1: WinSCP – Copy



/ - yanivl@ftp:/Admin@10.31.200.2 - WinSCP

D:\HTTP_Server_Uploads\SavedLogsToLocalDrive

Name	Ext	Size	Type	Changed	Attr
..			Parent directory	22-Feb-2015 11:16...	

/

Name	Ext	Size	Changed	Rights	Owner
mnt			30-Dec-99 12:00:0...	rwxrwxrwx	
rmx			30-Dec-99 12:00:0...	rwxrwxrwx	
tftp			30-Dec-99 12:00:0...	rwxrwxrwx	

Copy

Copy file 'rmx' to local directory:

D:\HTTP_Server_Uploads\SavedLogsToLocalDrive\*.*

Transfer settings
Default transfer settings

☐ New and updated file(s) only
☐ Transfer on background (add to transfer queue)
☐ Do not show this dialog box again
☐ Transfer each file individually

0 B of 0 B in 0 of 0

0 B of 58,656 B in 1 of 8

Local Mark Files Commands Session Options Remote Help
Default
yanivl@ftp:/Admin@10.31.200.2
D: Data
/ <root>
Browse...
Transfer settings...
Copy
Cancel
Help
F2 Rename F4 Edit F5 Copy F6 Move F7 Create Directory F8 Delete F9 Properties F10 Quit

3.1.1.2 Obtaining Data-CPU Log Files using USB Storage

The following procedure describes how to obtain the Data-CPU log files using USB Storage.

➢ **To obtain Data-CPU log files using USB storage:**

1. Copy the log files to the USB storage connected to the MSBR USB port using the "debug rmx-serial copy-logs-usb" command.
2. Run the "usb remove" command before removing the USB storage from the device.
3. Remove the USB storage.
4. View the file contents on your PC.

3.1.1.3 Obtaining Data-CPU Logs Using CLI commands

The Data-CPU log files can be obtained by running the following CLI commands:

- debug rmx-serial list-log
- debug rmx-serial read-log <log number>

```
# debug rmx-serial read-log 8
[000000.658] lo: Dropping NETIF_F_SG since no checksum feature.
[000000.667] isa bounce pool size: 16 pages
[000000.674] physmap flash device: 4000000 at 1bc00000
[000000.683] phys_mapped_flash: Found 1 x16 devices at 0x0 in 8-
bit bank
[000000.694]  Amd/Fujitsu Extended Query Table at 0x0040
[000000.702] phys_mapped_flash: CFI does not contain boot bank
location. Assuming top.
[000000.715] number of CFI chips: 1
[000000.720] cfi_cmdset_0002: Disabling erase-suspend-program due
to code brokenness.
[000000.733] cmdlinepart partition parsing not available
[000000.742] RedBoot partition parsing not available
[000000.750] u32 classifier
[000000.755]     OLD policer on
[000000.760] NET: Registered protocol family 2
[000000.784] IP route cache hash table entries: 2048 (order: 2,
16384 bytes)
[000000.795] IP route PBR cache hash table entries: 2048 (order:
2, 16384 bytes)
[000000.808] TCP established hash table entries: 8192 (order: 4,
65536 bytes)
[000000.820] TCP bind hash table entries: 8192 (order: 4, 65536
bytes)
[000000.831] TCP: Hash tables configured (established 8192 bind
8192)
[000000.841] TCP reno registered
[000000.846] IPv4 over IPv4 tunneling driver
```

3.1.2 Accessing the Data-CPU Serial Interface

Live debugging of Data-CPU sometimes requires direct access to its serial interface. AudioCodes has a proprietary serial cable for direct connectivity of both serial consoles - the **yellow** and **red** cable.

The “debug rmx-serial tap” command provides the same capability without using any cable. You can simply connect to the MSBR using SSH/Telnet or a standard serial console cable, authenticate to the CLI interface, and then start debugging the Data-CPU using the “debug rmx-serial tap” command. This command is directly connected to the Data-CPU in the same way as the user is connected to the MSBR console port.

Table 3-2: Serial Tap Logs Command

Command	Description
`debug rmx-serial tap`	Taps into RMX serial.

```
# debug rmx-serial tap
[Start RMX serial tap]

Password: *****
Username: admin
Password: *****

MSBR>
```

To exit Data-CPU tapping mode, press **Ctrl+C**.

4 Resetting History

This feature provides historical reasons and date/time of the last 20 resets that occurred on the MSBR. This information provides a complete picture of MSBR problems. By looking at the reset history, you can distinguish whether the problem is located in the VoIP-CPU, Data-CPU or there is faulty power cord.

4.1 Using Reset History

You can reset history by using either one of the following:

- **CLI:** "debug reset-history"
- **Command Shell:** "/BSP/EXCeption/ResetHistory"

```
Reset History :
[00] Reset Reason: SW Automatic Update
     Time : 15-7-2014 03:03:25
[01] Reset Reason: SW Automatic Update
     Time : 15-7-2014 02:53:51
[02] Reset Reason: Power-UP
     Time : 15-7-2014 02:43:44
[03] Reset Reason: SW WEB
     Time : 15-7-2014 02:19:54
[04] Reset Reason: SW Automatic Update
     Time : 15-7-2014 02:05:16
[05] Reset Reason: SW Automatic Update
     Time : 15-7-2014 00:02:45
[06] Reset Reason: issuing of a reset from CLI
     Time : 14-7-2014 23:53:31
[07] Reset Reason: issuing of a reset from CLI
     Time : 14-7-2014 06:40:50
[08] Reset Reason: SW Automatic Update
     Time : 14-7-2014 05:14:09
```

This page is intentionally left blank.

5 CPU and Memory Utilization History

The MSBR device has an option to view the CPU history. The MSBR device includes two CPUs: Voice CPU and Data CPU. You can view both CPU utilization and memory utilization history for both of these CPUs.

5.1 CPU Utilization History

The CPUs utilization history is collected for up to 72 hours according to the following:

- For the last 60 seconds, the CPU utilization is saved on a per second basis i.e. the MSBR retrieves the CPU utilization history of every second in the last minute.
- For the last 60 minutes (last hour), the MSBR retrieves 60 values. Every retrieved value is the average of the CPU utilization for every 60 second interval.
- For the last 72 hours (3 days) the MSBR retrieves 72 values. Every retrieved value is the average of the CPU utilization for every 60 minute interval.

The following commands show CPU utilization values:

Command	Description
`show system cpu-util history data`	Show data CPU utilization history
`show system cpu-util history voice`	Show voice CPU utilization history

The output of the command is graphical, showing time on the X axis, and CPU utilization on the Y axis. The numbers at the bottom of the Y axis represent the last time that a measurement of a utilization value was taken. The value at the top of the graph is the utilization value of the CPU.

Example:

```
Mediant 500 - MSBR# show system cpu-util history data


Data Cpu Utilization in last 72 hours



 1111111
 5495656
|
|
|
|
|
|
|
|
|
|*******
+------------------------------------------------------------------------
 123456789111111111122222222223333333333444444444455555555556666666666777
          012345678901234567890123456789012345678901234567890123456789012

Data Cpu Utilization in last 60 minutes
```

```


 111111111111111111111111111111111111111111111111111111111111
 545455455545554545454554545554555545554544555545444554555555
|
|
|
|
|
|
|
|
|
|************************************************************
+------------------------------------------------------------
 123456789111111111122222222223333333333444444444455555555556
          012345678901234567890123456789012345678901234567890

Data Cpu Utilization in last 60 seconds



 111111111111111111111111111111111111111111111111111111111111
 555444555555666555555444555445555555555555555554443335555555
|
|
|
|
|
|
|
|
|
|************************************************************
+------------------------------------------------------------
 123456789111111111122222222223333333333444444444455555555556
          012345678901234567890123456789012345678901234567890
Mediant 500 - MSBR#
```

5.2 CPU Memory Utilization History

The same method that is used for CPU utilization history is used for memory utilization history. The following commands show memory utilization values:

Command	Description
`show system utilization history data`	Show data memory utilization history.
`show system utilization history voice`	Show voice memory utilization history.

Example:

```
Mediant 500 - MSBR# sh system utilization history voice


Voice Memory Utilization in last 72 hours




 55555555
 32232222
|
|
|
|
|
|********
|********
|********
|********
|********
+------------------------------------------------------------------------
 123456789111111111122222222223333333333444444444455555555556666666666777
          012345678901234567890123456789012345678901234567890123456789012

Voice Memory Utilization in last 60 minutes




 555555555555555555555555555555555555555555555555555555555555
 223333333333333333332223222222222222222222222233222222222222
|
|
|
|
|
|************************************************************
|************************************************************
|************************************************************
|************************************************************
|************************************************************
+------------------------------------------------------------
```

```
 123456789111111111122222222223333333333444444444455555555556
          012345678901234567890123456789012345678901234567890
Mediant 500 - MSBR#
```

6 Debugging USB Devices

AudioCodes provides several USB debug utilities for debugging USB devices.
If the 3G/4G USB modem fails to connect to the MSBR, the following information needs to be provided:

- USB Hardware Information
- Software Driver Definitions
- AT Modem Command Responses

6.1 Providing USB Hardware Information

Run the “show system assembly” command to display USB hardware information.

```
show system assembly

Board Assembly Info:

|Slot No.              | Ports       |Module Type                 |
| 0/0                  | 1           | WAN-Copper                 |
| 0/1                  | 1           | WAN-Fiber                  |
| 1                    | 1-4         | LAN-GE                     |
| 2                    | 1           | E1/T1                      |

USB Port 1:  Manufacturer - HUAWEI Technology, Product - HUAWEI
Mobile, Product Id - 1001, Vendor Id - 12d1, Type - Cellular
USB Port 2:  Manufacturer - SAMSUNG, Product - Flash Disk, Product
```

6.2 Providing Software Driver Definitions

Run the "Debug usb-3g devices" command to provide a deeper look into the software driver definitions.

```
debug usb-3g devices

T:  Bus=01 Lev=00 Prnt=00 Port=00 Cnt=00 Dev#=  1 Spd=480 MxCh= 1
B:  Alloc=  7/800 us ( 1), #Int=  2, #Iso=  0
D:  Ver= 2.00 Cls=09(hub  ) Sub=00 Prot=01 MxPS=64 #Cfgs=  1
P:  Vendor=0000 ProdID=0000 Rev= 2.06
S:  Manufacturer=Linux 2.6.21.7-Cavium-Octeon dwc_otg_hcd
S:  Product=DWC OTG Controller
S:  SerialNumber=dwc_otg
C:* #Ifs= 1 Cfg#= 1 Atr=e0 MxPwr=  0mA
I:* If#= 0 Alt= 0 #EPs= 1 Cls=09(hub  ) Sub=00 Prot=00 Driver=hub
E:  Ad=81(I) Atr=03(Int.) MxPS=   4 Ivl=256ms

T:  Bus=01 Lev=01 Prnt=01 Port=00 Cnt=01 Dev#=  2 Spd=480 MxCh= 4
D:  Ver= 2.10 Cls=09(hub  ) Sub=00 Prot=02 MxPS=64 #Cfgs=  1
P:  Vendor=0451 ProdID=8043 Rev= 1.00
S:  SerialNumber=3E060851B6D1
C:* #Ifs= 1 Cfg#= 1 Atr=e0 MxPwr=  0mA
I:  If#= 0 Alt= 0 #EPs= 1 Cls=09(hub  ) Sub=00 Prot=01 Driver=hub
E:  Ad=81(I) Atr=03(Int.) MxPS=   1 Ivl=256ms
I:* If#= 0 Alt= 1 #EPs= 1 Cls=09(hub  ) Sub=00 Prot=02 Driver=hub
E:  Ad=81(I) Atr=03(Int.) MxPS=   1 Ivl=256ms

T:  Bus=01 Lev=02 Prnt=02 Port=00 Cnt=01 Dev#=  9 Spd=480 MxCh= 0
D:  Ver= 2.00 Cls=00(>ifc ) Sub=00 Prot=00 MxPS=64 #Cfgs=  1
P:  Vendor=12d1 ProdID=1001 Rev= 0.00
S:  Manufacturer=HUAWEI Technology
S:  Product=HUAWEI Mobile
C:* #Ifs= 5 Cfg#= 1 Atr=e0 MxPwr=500mA
I:* If#= 0 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=ff Prot=ff
Driver=option
E:  Ad=81(I) Atr=03(Int.) MxPS=  64 Ivl=2ms
E:  Ad=82(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=01(O) Atr=02(Bulk) MxPS= 512 Ivl=4ms
I:* If#= 1 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=ff
Driver=option
E:  Ad=83(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=02(O) Atr=02(Bulk) MxPS= 512 Ivl=4ms
I:* If#= 2 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=ff
Driver=option
E:  Ad=84(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=03(O) Atr=02(Bulk) MxPS= 512 Ivl=4ms
I:* If#= 3 Alt= 0 #EPs= 2 Cls=08(stor.) Sub=06 Prot=50
Driver=option
E:  Ad=85(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=04(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms
```

```
I:* If#= 4 Alt= 0 #EPs= 2 Cls=08(stor.) Sub=06 Prot=50
Driver=option
E:  Ad=05(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=86(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms

T:  Bus=01 Lev=02 Prnt=02 Port=01 Cnt=02 Dev#=  3 Spd=480 MxCh= 0
D:  Ver= 2.00 Cls=00(>ifc ) Sub=00 Prot=00 MxPS=64 #Cfgs=  1
P:  Vendor=0011 ProdID=7788 Rev= 1.03
S:  Manufacturer=SAMSUNG
S:  Product=Flash Disk
S:  SerialNumber=12345678
C:* #Ifs= 1 Cfg#= 1 Atr=80 MxPwr=100mA
I:* If#= 0 Alt= 0 #EPs= 2 Cls=08(stor.) Sub=06 Prot=50 Driver=usb-
storage
E:  Ad=01(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=82(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
```

6.3 Providing AT Modem Command Responses

AudioCodes provides the capability for recording AT modem commands between the USB device and the AudioCodes router. This capability allow us to analyze the 3G device in case of unsupported devices.

```
debug usb-3g serial-trace cli
Start serial trace]
TX 87
-- 0000:  7e ff 03 00 21 45 00 00 4f 00 00 40 00 40 11 80  |....!E..O..@.@..|
-- 0010:  a5 b0 0c 85 74 c3 bd c0 ba 00 a1 c8 c9 00 3b fc  |....t.........;.|
-- 0020:  56 30 31 02 01 01 04 06 70 75 62 6c 69 63 a2 24  |V01.....public.$|
-- 0030:  02 03 04 1d 0f 02 01 00 02 01 00 30 17 30 15 06  |..........0.0..|
-- 0040:  11 2b 06 01 04 01 a7 0b 09 0a 0a 02 0c 01 01 0d  |.+.............|
-- 0050:  01 00 81 00 88 d3 7e                             |.......         |
RX 119
-- 0000:  7e ff 7d 23 7d 20 21 45 7d 28 7d 20 46 7d 21 6a  |...#. !E.(. F.!j|
-- 0010:  7d 20 7d 20 74 7d 31 8b 3c c3 bd c0 ba b0 7d 2c  |. . t.1.<......,|
-- 0020:  85 74 c8 ca 7d 20 a1 7d 20 32 23 d7 30 28 7d 22  |.t... .. 2#.0(."|
-- 0030:  7d 21 7d 21 7d 24 7d 26 70 75 62 6c 69 63 a0 7d  |.!.!.$.&public..|
-- 0040:  3b 7d 22 7d 23 7d 24 7d 3d 7d 30 7d 22 7d 21 7d  |;.".#.$.=.0.".!.|
-- 0050:  20 7d 22 7d 21 7d 20 30 7d 2e 30 7d 2c 7d 26 7d  | .".!. 0..0.,.&.|
-- 0060:  28 2b 7d 26 7d 21 7d 22 7d 21 7d 21 7d 23 7d 20  |(+.&.!.".!.!.#. |
-- 0070:  7d 25 7d 20 2b b5 7e                             |.%. +..          |
TX 81
-- 0000:  7e ff 03 00 21 45 00 00 49 00 00 40 00 40 11 80  |....!E..I..@.@..|
-- 0010:  ab b0 0c 85 74 c3 bd c0 ba 00 a1 c8 ca 00 35 e0  |....t.........5.|
-- 0020:  ba 30 2b 02 01 01 04 06 70 75 62 6c 69 63 a2 1e  |.0+.....public..|
-- 0030:  02 03 04 1d 10 02 01 00 02 01 00 30 11 30 0f 06  |..........0.0..|
-- 0040:  08 2b 06 01 02 01 01 03 00 43 03 77 07 8b 9a bf  |.+.......C.w....|
-- 0050:  7e                                               |.                |
RX 136
-- 0000:  7e ff 7d 23 7d 20 21 45 7d 28 7d 20 4f 7d 21 6b  |...#. !E.(. O.!k|
-- 0010:  7d 20 7d 20 74 7d 31 8b 32 c3 bd c0 ba b0 7d 2c  |. . t.1.2......,|
-- 0020:  85 74 c8 cb 7d 20 a1 7d 20 3b 7d 20 d1 30 31 7d  |.t... .. ;. .01.|
-- 0030:  22 7d 21 7d 21 7d 24 7d 26 70 75 62 6c 69 63 a0  |".!.!.$.&public.|
-- 0040:  24 7d 22 7d 23 7d 24 7d 3d 7d 31 7d 22 7d 21 7d  |$.".#.$.=.1.".!.|
-- 0050:  20 7d 22 7d 21 7d 20 30 7d 37 30 7d 35 7d 26 7d  | .".!. 0.70.5.&.|
-- 0060:  31 2b 7d 26 7d 21 7d 24 7d 21 a7 7d 2b 7d 29 7d  |1+.&.!.$.!..+.).|
-- 0070:  2a 7d 2a 7d 22 7d 2c 7d 21 7d 21 7d 29 7d 21 7d  |*.*.".,.!.!.).!.|
-- 0080:  20 7d 25 7d 20 38 e2 7e                          | .%. 8..        |
TX 87
-- 0000:  7e ff 03 00 21 45 00 00 4f 00 00 40 00 40 11 80  |....!E..O..@.@..|
-- 0010:  a5 b0 0c 85 74 c3 bd c0 ba 00 a1 c8 cb 00 3b 00  |....t.........;.|
-- 0020:  53 30 31 02 01 01 04 06 70 75 62 6c 69 63 a2 24  |S01.....public.$|
-- 0030:  02 03 04 1d 11 02 01 00 02 01 00 30 17 30 15 06  |..........0.0..|
-- 0040:  11 2b 06 01 04 01 a7 0b 09 0a 0a 02 0c 01 01 09  |.+.............|
-- 0050:  01 00 81 00 c8 fa 7e                             |.......         |
```

7 Debugging Wi-Fi

The "show data dot11radio other-ap" command provides the ability to look up all Wi-Fi access points around the MSBR.

```
# show data dot11radio other-ap
SSID            BSSID              CHAN RATE
Audc-QA         00:0b:86:35:72:74    1   54M  6:0
AudioCodes      00:0b:86:35:72:70    1   54M  7:0
Guest-AudC      00:0b:86:35:72:71    1   54M  8:0
AudcPBX         00:0b:86:35:72:73    1   54M  7:0
AudioCodes      00:0b:86:2d:38:90    3   54M 24:0
Guest-AudC      00:0b:86:2d:38:91    3   54M 24:0
audc-ph         00:0b:86:2d:38:92    3   54M 24:0
AudcPBX         00:0b:86:2d:38:93    3   54M 25:0
Audc-QA         00:0b:86:2d:38:94    3   54M 24:0
M800A           00:90:8f:50:b1:fe    6   54M 40:0
M800D           00:90:8f:73:16:b7   10   54M 46:0
AudioCodes      00:0b:86:2d:3c:50   10   54M  7:0
M500A           00:90:8f:4a:fe:fe   12   54M 27:0
```

The "show data dot11radio hardware-stats" command provides internal Wi-Fi information regarding the hardware status of Wi-Fi counters.

```
# show data dot11radio hardware-stats
Critical Errors
===============
61 (0) recv eol interrupts
40 (0) global transmit timeout interrupts
12 (0) tx drops in wrong state

Summary Statistics
===============

       nan tx unaggregated long retry percent
      0.00 tx unaggregated excessive retry percent
      0.00 tx aggregated long retry percent
      0.00 tx aggregated excessive retry percent
      0.00 tx aggregate subframe retry percent
      0.00 tx aggregate subframe excessive retry percent
rssi of last rcv[ctl, ch0]: 10
rssi of last rcv[ctl, ch1]: 7
rssi of last rcv[ctl, ch2]: 7
rssi of last rcv[ext, ch0]: 4294967294
rssi of last rcv[ext, ch2]: 1

Detailed Statistics
===============

503 tx management frames
```

```
228 tx frames with no ack marked
9061 rx failed 'cuz of bad CRC
50229 PHY errors
    45421 phy ofdm timing
    4784 phy cck timing
    24 phy cck restart
699 periodic calibrations
1 switched default/rx antenna
Antenna profile:
[0] tx        0 rx        3
[1] tx        0 rx     1528
9 pre delimiter crc errors
124 total channel changes
118 fast channel changes
684 beacons transmitted
5 driver init calls
9 driver stop calls
1 driver resets
8 nodes allocated
6 nodes deleted
         0 rb on
         0 rb off

11n stats
        12 total tx data packets
         0 tx when h/w queue depth is low
         0 tx pkts when h/w queue is busy
       503 tx completions
         0 tx pkts filtered for requeueing
       353 txq empty occurences
       510 tx schedule pkt queue empty
         0 tx bars sent
         0 tx bar retries sent
         0 tx bar last frame failed
         0 tx bar excessive retries
         0 tx unaggregated frame completions
         0 tx unaggregated excessive retries
         0 tx unaggregated unacked frames
         0 tx unaggregated last frame failed
         0 tx aggregated completions
         0 tx block ack window advanced
         0 tx block ack window retries
         0 tx block ack window additions
         0 tx block ack window updates
         0 tx block ack window advances
         0 tx retries of sub frames
         0 tx excessive retries of aggregates
         0 tx no frame scheduled: baw limited
         0 tx frames not aggregated
         0 tx aggr good completions
         0 tx aggr excessive retries
         0 tx aggr unacked subframes
         0 tx aggr old frames requeued
         0 filtered aggr packet
         0 tx aggr: last sub-frame failed
         0 tx aggr: h/w long retries
         0 tx aggr: h/w short retries
         0 tx aggr: tx timer expired
         0 rx pkts
         0 rx aggregated packets
         0 rx bars
         0 rx non qos-data frames
         0 rx sequence resets
         0 rx old packets
         0 rx block ack window reset
         0 rx pts indicated due to baw resets
         0 rx duplicate pkts
         0 rx block ack window advanced
         0 rx pkt completions
         0 rx bar discarded
         0 rx pkts unblocked on bar reception
         0 rx pkt completions on bar reception
         0 rx pkt sequences skipped on timeout
         0 rx indications due to timeout
         0 rx descriptor status corrupted
        33 watchdog: tx is active
      1843 watchdog: tx is not active
         0 watchdog: spurious tx hang
         0 filter & requeue on 20/40 transitions
       854 draining tx queue on error
         0 draining tid buf queue on error
         0 buffers drained from pending tid queue
         0 unaggregated tx pkts filtered
         0 aggregated tx pkts filtered
         0 total sub-frames filtered
         0 tid paused
         0 tid resumed
```

This page is intentionally left blank.

8 Sending Information to Head Office

When a Service Request is issued, specific information must be provided to Head Office, so that appropriate analysis could be performed on the problem.

Please provide the output from the following commands:

- show system version
- show system assembly
- show data ip route
- show data ip interface brief
- show run (full voip/system/data must be included)
- show data crypto debug
- debug reset-history
- System INI file – for VoIP analysis
- debug-capture files describing the problem
- All RMX directory content from flash (see Section 3.1.1 on page 17)

International Headquarters
1 Hayarden Street,
Airport City
Lod 7019900, Israel
Tel: +972-3-976-4000
Fax: +972-3-976-4040

AudioCodes Inc.
27 World's Fair Drive,
Somerset, NJ 08873
Tel: +1-732-469-0880
Fax: +1-732-469-2298

Contact us: www.audiocodes.com/info
Website: www.audiocodes.com

Document #: LTRT-40383

AudioCodes