



Log Analysis

Best Current Practices




Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Table of Contents

INTRODUCTION.....	4
INTENDED AUDIENCE.....	4
BACKGROUND	4
TROUBLESHOOTING LOG COLLECTION METHODS	4
REVIEW THE SHOW SIPD STATISTICS	6
EVALUATE THE SIP CALL FLOW	11
COLLECT THE ESBC LOG FILES	12
LOG ANALYSIS	28
TOOLS	31
ONLINE TOOLS	31
OFF-LINE TOOLS / COMMAND LINE TOOLS	31
LOG VIEWERS	34
THIRD PARTY SOFTWARE	35
JOB AIDES	35
TROUBLE-SHOOTING PROCEDURES WITH SAMPLE PROBLEM CASES	35
DNS NOT REACHABLE	36
ONE-WAY AUDIO - STEERING POOL IP OVERLAPPED	38
DOS PROTECTION PARAMETER MISCONFIGURATION	40
RECOMMENDATION TO ENGINEERING FOR ENHANCEMENT	44
ESBC OS ENHANCEMENT RECOMMENDATION	44
HELPFUL TOOLS ENHANCEMENT RECOMMENDATION	44
AUTHOR'S ADDRESS	45
DISCLAIMER	45
FULL COPYRIGHT STATEMENT	45



Introduction

This Log File Study Workshop document is the output from a study group of Oracle ESBC members from different regions understanding log files from generated from the ESBC 6300 product .Analyzing log files plays a large part in the daily life of all Oracle Systems Engineers and TAC Engineers. Nevertheless, there is no systematic document documenting how can those log files be collected, analyzed and used for troubleshooting effectively. That's why there are volunteers working as a group to discuss and analyze existing log files and communicate with Engineering so as to look forward to a systematic way to tackle our important daily life.

Intended Audience

This document is intended for use by Oracle Systems Engineers and Technical Support Engineers (collectively referred to as the Professional Services organization), third party Systems Integrators, and end users of the ESBC. It assumes that the reader is familiar with basic operations of the ESBC, and has attended the following training courses (or has equivalent experience):

EDU-COB: ESBC Configuration Basics

EDU-TS1: Troubleshooting Level 1

It also presumes that the reader is familiar with standard configuration models and archetypes; for more information, published in our Best Current Practice series of documentation.

Background

When troubleshooting problems on the ESBC (), collecting log files is a staple of the analytical process. It is self-evident that the more common the understanding of the content and pattern of the ESBC's debug log files, the earlier the problem can be isolated and fixed (as applicable). This document intends to provide a guideline for setting the log-level appropriately, collecting logs effectively, and decoding some key contents in log.sipd and log.mbcd for effective call analysis.

Troubleshooting Log Collection Methods

Before analyzing the log files, it is best to narrow down the type of problem so that the correct logs and log level can be activated to reduce the information that you have to read through to find your issue.

Collect Information about the ESBC

Knowing about the environment that the ESBC is deployed in will help in understanding the behaviors seen in the logs. This library of information will be referenced during the log analysis.

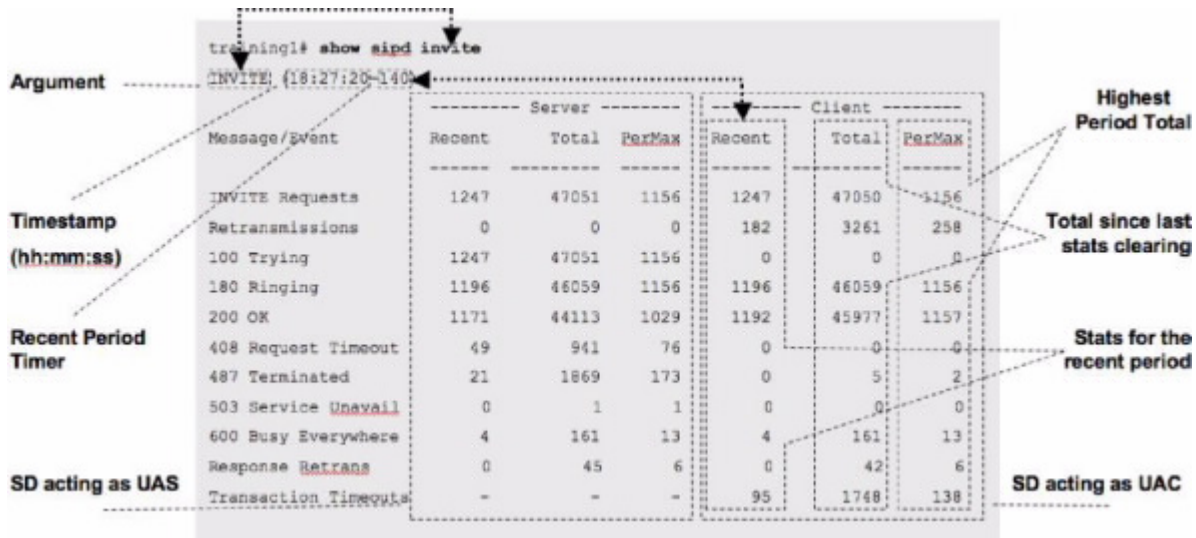
Command	Information	Comments
show version boot	ESBC chassis serial number and motherboard information	
show version image	ESBC bootparams	
show features	Active license information	
show support-info	Displays all of the stat information by issuing one command	
show running-config	ESBC's running	
show health	HA node state information	
show arp	ARP table and statistics	
show routes	IP routing information	
show interface	PHY interface details	
show virtual- interfaces	SIP/H323 stack details as well as sip-nat relationships	
verify-config	Ensure there are no simple data integrity issues with the configuration	
show about	Information for Acli	
Show accounting	Displays accounting statistics	
show directory	Displays the directories present in the SBC	
show entitlements	Displays the transcoding and session related information	

show media physical	Displays the physical statistics of the media port	
show media classify	Displays the network classification of every media port	
show ntp server	Displays NTP server related information	
show prom-info	Displays the transcoder information	

Review the show sipd statistics

The ESBC maintains statistical information about the messages it processes to help narrow down the type of problem that you look for in the log files. Analyzing the peg counters will also help determine if the problem you are investigating is caused by the ESBC or by another element in the call flow.

Figure 1 – show sipd <method> Output Format



The example above shows the functional areas within the output from a show sipd <method> command. The first output line reflects the method being reported on, followed by a timestamp (hour:minute:second format) and the recent period timer. The left-hand side (the Server columns) represents the ESBC acting as a User Agent Server (UAS), while the right-hand side (the Client columns) represents the ESBC acting as a User Agent Client (UAC). The Recent column represents statistics for the recent period (the current period plus the last period). The Total column represents the total for a particular metric since the last stats clearing. Statistics are cleared either through the issue of the reset sipd command or on reboot. The PerMax column represents the maximum for a given metric seen in any given individual (current) period. Remember that the "Recent" column represents the recent period, which includes statistics from the current and the last period, which is why that number may be higher than what is displayed in the PerMax column.

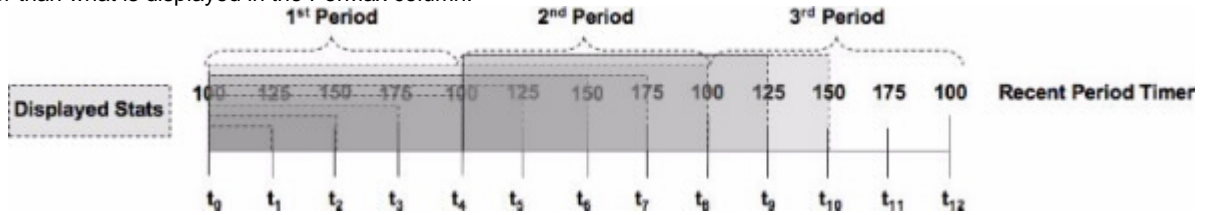


Figure 2 – Recent Period Timestamp Operation

The Current Period timestamp counts from 100 to 200 in one second increments. The statistics displayed in the Recent column for any show sipd command will reflect the appropriate behaviors for the associated value within the current period plus the last period (which constitutes a 100-200 second Recent period). This is done in order to keep the statistics from zeroing out between period transitions. That means that at time t₄, in the display above, the statistics displayed represent the last 100 seconds worth of behaviors (from the first period). The Recent Period statistics at time t₆ represent the last 150 seconds of statistics (including 100 period 1). The Recent Period statistics at time t₈ represent the last 100 seconds of statistics (including 100 from period 2). Remember that that Recent period is the sum of the Active (current) period and the previous period.

Figure 3 – Show Sisd <Non-Method> Output

The show sisd command displays a different set of statistics when not used for monitoring a Method. Specifically, show sisd status and show sisd sessions sessions share the same output format. As with the show sisd <method>

Annotations in the screenshot:

- Timestamp (hh:mm:ss):** 02:31:33-14
- Argument:** SIP STATUS
- Stats for the active period:** Sessions, Subscriptions, Dialogs, CallID Map, Rejections, ReINVITEs
- Stats for the recent period:** Media Sessions, Media Pending, Client Trans, Server Trans, ResP Contexts, Saved Contexts, Sockets, Req Dropped, DNS Trans, DNS Sockets, DNS Results
- Lifetime stats:** Total, PerMax, High (under Lifetime)

	Active	-- Period --		Lifetime		
		High	Total	Total	PerMax	High
Sessions	0	1	1	43	16	4
Subscriptions	0	0	0	0	0	0
Dialogs	0	2	2	84	32	8
CallID Map	0	2	2	86	32	8
Rejections	-	-	0	0	0	0
ReINVITEs	-	-	0	4	4	0
Media Sessions	1	1	1	43	16	16
Media Pending	0	0	0	0	0	0
Client Trans	1	3	11	1620	57	26
Server Trans	1	3	11	1621	57	26
ResP Contexts	1	3	11	1620	57	26
Saved Contexts	0	0	0	0	0	0
Sockets	25	25	0	25	13	25
Req Dropped	-	-	0	0	0	0
DNS Trans	0	0	0	0	0	0
DNS Sockets	0	0	0	0	0	0
DNS Results	0	0	0	0	0	0

Session Rate = 0.1
Load Rate = 0.5

command, the argument is reflected in the output as is the timestamp. The Active column represents the number of active (realtime) counts. The Period column represents statistics within the Recent period. The Lifetime column represents statistics for the lifetime of collection (since last reset of statistics either through reboot or reset sisd command).

The show sisd errors command can also help a lot during trouble-shooting. It can help quickly provide a brief idea and direction about whether the problem is related to any SIP based error. Here below is the sample output:

```
training1# show sisd errors
Recent
SIP Errors/Events
SDP Offer Errors          0          5          5
SDP Answer Errors         0          1          1
Drop Media Errors         0          0          0
Transaction Errors        0          0          0
Application Errors        0          0          0
Media Exp Events          0          0          0
Early Media Exps          0          0          0
Exp Media Drops           0          0          0
Expired Sessions          0         23          3
Multiple OK Drops         0          0          0
Multiple OK Terms         0          0          0
Media Failure Drops       0          0          0
Non-ACK 2xx Drops        0          1          1
Invalid Requests          0          0          0
```



```

Invalid Responses      0      0      0
Invalid Messages     93     88733  9
                                     5

```

```

training1# show mbcdd add
10:33:41-151

```

	Recent	Lifetime	
		Total	PerMax
Add incoming:			
Request received	0	17	4
Duplicates received	0	0	0
Replies sent	0	17	4
Errors sent	0	0	0
Add outgoing:			
Requests sent	0	17	4
Req retransmissions	0	0	0
Replies received	0	17	4
Errors received	0	0	0

```

Avg Latency=0.000 for 0
Max Latency=0.000

```

Stats for the recent period (points to Recent column)

Lifetime stats (points to Lifetime columns)

Figure 4 – Show MBCD Transaction Output Format

The show mbcdd <transaction> command (where valid transactions are add, subtract or modify) as well as the show sipd errors command display statistics for the Recent period in the Recent column. The Lifetime columns represent statistics from the last resetting of the MBCD counters (either via reboot or issuing the reset mbcdd command).

Command	Information	Comments
Show sipd register	Display the registration information	
Show sipd invite	Used to determine if there is a problem processing a call. Look at the counters to determine if the problem is on the server side or the client side. Be sure to look for any 4xx or 5xx messages.	
Show sipd errors	Used to determine if there are any call flow problems	
Show sipd agents	Look here if a 503 was returned	
Show proc cpu	Is the CPU near or above 90% utilization?	
Show mbcdd error		
Show mbcdd realms	Look here if a 503 was returned	



Show sipd policy	404/480 returned due to local-policy issues	
-------------------------	---	--

Show dns stats	Look here if you are using FQDN in the SIP headers	
Show sipd endpoint-ip <\$AOR>	Registration cache information for a user endpoint	
Show nat by-address <source subscriber ip>	Useful for one way media issues	
Show sip acl	DOS promotons/demotions for SIP user endpoints	
Show acl untrusted	For access scenario SIP I/F starting as untrusted	
Show acl trusted	Look for promoted endpoints	
show acl denied	Displays denied acl entries	
show acl info	Displays access control statistics	
show algd/show mgcp	Displays mgcp sessions related information	
show built-in-sip-manipulations	Displays the built in sip manipulations in the SBC	
Show mbcd error	Displays the mbcd error statistics	
show mbcd realms	Displays the media statistics for particular realm	
packet-trace start/stop	To capture dump on the SBC or remote	
show packet trace	Displays the packet trace captured	
show registration	Displays the SIP registrations over a period of time/lifetime	
show sessions	Displays the active/completed SIP/H323 sessions	
show sipd tcp	Shows the sip tcp connections	
show sipd srvc	Displays information for IMS related calls	
show sipd acl	Displays the acl promotion/demotion	
show sipd client	Show the sip client transactions	
show sipd server	shows the sip server transactions	
show sipd method(Invite/Refer/prack/publish/register/refer/subscribe/update/options/info/notify/ack/bye	shows successful/unsuccessful message rate of particular method nnnnnnaaaaaai((Invite/Refer/prack/publish/register/refer/subscribe/update/options/info/notify/ack/bye	
show sipd transcode	Shows the codecs available for transcoding that are licensed	

show sipd forked	Shows the forked sessions	
show sipd rate	shows the rate at which every message is sent and received .performance related debugging	
show sipd lb-endpoints	Useful for realm specific endpoint debugging.	
Show sipd sa-nsep-burst	Displays the registration cache limit	
show sipd pooled-transcoding	Displays the transcoding for uac and uas separately	
show sipd codecs	Shows codecs available per realm	
show sipd groups	Shows the session agent groups inbound/outbound traffic	

Evaluate the SIP call flow

Many times the information contained in the ESBC log file does not adequately represent the environment that the problem is created in. For this reason, it is important to provide network level packet captures by use of either Ethereal or Wireshark to facilitate problem analysis.

For example, if you are troubleshooting a one-way media issue, you may start by looking at the "show arp" output, looking for MAC addresses to see if there is any overlapped ip addresses. If you find an issue, you will need the network packet capture to show the flow issues with RTP.

If a network packet capture is not available, the ESBC transaction logs will provide an adequate amount of data to find most problems.

Command	Information	Comments
Notify sipd siplog	Enable the daemon in debug mode	Produces the sipmsg.log file

Log-level sipd debug	Enable the daemon in debug mode	SIP full	Especially when it is necessary to check how sipd select next-hop or how HMR be triggered, etc, it is helpful to analyse log.sipd in full debug mode. Command "notify sipd debug" can have the similar effect, but it will not enable all log category into debug mode. As a result, sometimes if it is necessary to open defect, it is recommended to turn on full debug mode for Engineering to have full reference.
Log-level mbcdd debug	Enable the flow logs		

Evaluate the SIP messages

The SIP protocol (RFC3261) is a very flexible protocol that can be extended by the development of support for additional RFC's in the SIP stack or by the inclusion of custom parameters to existing SIP headers that can be used in a proprietary implementation by each end point.

The Internet Assigned Number Authority (IANA) maintains a list of all currently supported SIP methods, headers, and response codes. The list is available on the Internet at <http://www.iana.org/assignments/sip-parameters>. This list will help you find proprietary headers as well as indicate the origin (RFC or draft) of each header in the SIP messages.


In general, you should read the SIP messages paying close attention to the following:

- Header values
- Any "odd" data
- Error messages
- Description of symptom in response code text

Oracle Systems Engineering has a tool called log2cap.exe which can covert a sipmsg.log file into a PCAP file which can be read and analyzed with a network analysis tool (Ethereal/Wireshark). Use this tool so that you can pull the sipmsg.log file into the analyzer and then graph the call flow. It is very helpful to spot issues in the call flows once graphed in a ladder diagram.


Collect the ESBC log files

Setting the appropriate log-level on target process is very important and can also minimize the impact on production performance. Here below are the list of log category should be paid attention:



GENERAL
EMERGENCY
CRITICAL
MAJOR
MINOR
WARNING
PROC
IPC
SERVICE
EVENT
MESSAGE
TEST
TRIP
SIP
MBCP
FLOW
MEDIA
SESSION
TRANS
TIMER
ALG
MGCP
NPSOFT
ARP
SNMP
ANDD
XNTP
REDUNDANCY
SIPNAT
H323
ERROR
CONFIG
DNS
H248
BAND
ALI
SS8GI
COPS
ATCP
ATCPAPP
CLF
LRT

Each log category can be triggered to certain log level per process log files by use of **log-level** command. For release 4.1.1, here below is an example to trigger log category **[SIP]** and **[SIP-NAT]** to be debug level in log.sipd file and then check the changed log level on all log category for sipd:



```
Oracle-SBC# notify sipd debug
Oracle-SBC#
enabled SIP Debugging
```

```
Oracle-SBC# show loglevel sipd verbose
Log Levels for process sipd:
```

```
GENERAL=INFO
EMERGENCY=INFO
CRITICAL=INFO
MAJOR=INFO
MINOR=INFO
WARNING=INFO
PROC=INFO
IPC=INFO
SERVICE=INFO
EVENT=INFO
MESSAGE=INFO
TEST=INFO
TRIP=INFO
SIP=DEBUG
MBCP=INFO
FLOW=DEBUG
MEDIA=DEBUG
SESSION=DEBUG
TRANS=DEBUG
TIMER=INFO
ALG=INFO
MGCP=INFO
NPSOFT=INFO
ARP=INFO
SNMP=INFO
ANDD=INFO
XNTP=INFO
REDUNDANCY=INFO
SIPNAT=DEBUG
H323=INFO
ERROR=INFO
CONFIG=TRACE
DNS=TRACE
H248=INFO
BAND=INFO
ALI=INFO
SS8GI=INFO
COPS=INFO
ATCP=INFO
ATCPAPP=INFO
CLF=INFO
Oracle-SBC#
```

The highlighted shows that many log category are now in DEBUG log level. However, others are still in INFO log level. Certainly you can also only execute command **log-level sipd debug** so as to trigger all available log categories to **DEBUG** level. The drawback will then be the performance impact and the ultimate log files size.

For sufficient **SIPD** logging, below is the log categories should normally be triggered to DEBUG log level in case of problem logging necessity.

- SIP** - Generic SIP process
- TRANS** - Transaction level process
- SIPNAT** - SIP-NAT related process
- SESSION** - Session level process
- MEDIA** - SDP related process
- FLOW** - MBCD interaction interface
- TRIP** -Local Policy related process

For sufficient **MBCD** logging, below is the log categories should normally be triggered to DEBUG log level in case of problem logging necessity.

- FLOW** – MBCD interaction interface
- NPSOFT** – CAM information

Normally gathering mbcdd debug log is to trouble-shoot problem related to media in a problem call, e.g. one-way media problem. By co-relating sipmsg.log, log.sipd and log.mbcd together, SE can be possible to assure whether RTP source ip address has been latched and what exactly the source ip address is on both A and B party.

To gather sufficient **SIPD** and **MBCD** log information for trouble-shooting, below commands are suggested:

log-level sipd debug (sip trans sipnat session media flow trip)

log-level mbcdd debug (flow npsft)

notify sipd siplog

notify all rotate-logs

After finishing problem simulation, process log level should be resumed back to [NOTICE] by following commands:

log-level sipd notice
log-level mbcdd notice
notify sipd nosiplog

For efficient download of log files, it is recommended to archive all logs in a single compressed file before downloading. Here below are the procedures as reference:



```
Oracle-SBC# archives
Oracle-SBC(archives)# create LOGS test
creating directory '/code/logs'
task done
Oracle-SBC(archives)# display LOGS
test.tar.gz
Oracle-SBC(archives)# exit
```

FTP to wancom0 or any FTP-enabled front media interface and visit

/code/logs directory, file **[test.tar.gz]** can be downloaded to local PC for analysis.

For production network, it is recommended to first trigger the necessary log category to **[INFO]** log level to see whether the cause of problem can be located so as to minimize performance impact on SD.

Depending on individual trouble-shooting necessity, additional logcategory can be triggered to **[INFO]** or **[DEBUG]** level for getting sufficient logs, without necessity to trigger all log category to **[DEBUG]**.

Log Analysis

For trouble-shooting SIP related call, sipmsg.log and log.sipd are the key files to study in ESBC. Here below is the workshop analysis result on both files.

“Most of the cases I will inspect **sipmsg.log** and **log.sipd** is when I want to know how sipd decide next-hop routing,” said Mark Waugh, Oracle Chief Software Architect.

Key points for inspection on sipmsg.log and log.sipd

sipmsg.log is purely SIP message flow file for front end system engineer inspection purpose. **log.sipd** is for Engineering developer inspection purpose, which does not have documentation on explaining it in detail. They are co-related by times stamp. Normally sipmsg.log should be inspected first. Locating the problem call message time stamp, that time stamp will then be used as a key to cross-check with same location context. Here below are the key words in **log.sipd** and **log.mbcd** upon Bob and PT assistance.

This correlation can be easily seen in the consistency between the two start lines both in the sipmsg.log and log.sipd:

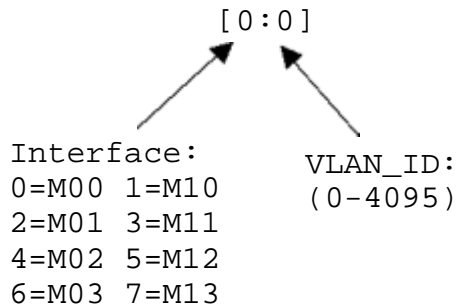
– **sipmsg.log:**

```
Jan 29 15:16:09.619 On [0:0]11.0.0.11:5060 received from 11.0.0.101:5060
```

– **log.sipd:**

```
Jan 29 15:16:09.619 [SIP] Found Socket for [0:0]11.0.0.11:5060 src=11.0.0.101:5060
```

In this first line, we can see how the interface and the VLAN_ID are identified:



After that, we can start analyzing **log.sipd** according to the SIP message we have received. There are several variables here, depending, of course, on the SIP Message received, although it is important also to consider if we have an access (HNT, no-HNT) scenario, peering scenario, SIP-NAT configuration, PBR, HMR, etc...

Access Environment Log Analysis

As an example, we are analyzing here a received REGISTER message for a HNT endpoint on a Policy-Based Routing configuration in 8.1p1 release. For your reference, you can appreciate the non configured elements are also invoked so you can have an idea of where you could look for them in case you need it:

1 - [SIP] Receipt on SIP socket

```

Jan 29 15:16:09.619 [SIP] Found Socket for [0:0]11.0.0.11:5060 src=11.0.0.101:5060
IDC-4600-1# show sipd tcp
15:39:39-114
  
```

SIP TCP Sockets	-- Period --			----- Lifetime -----		
	Active	High	Total	Total	PerMax	High
All States	0	0	0	0	0	0
TCP_INITIAL	0	0	0	0	0	0
TCP_STARTING	0	0	0	0	0	0
TCP_AVAILABLE	0	0	0	0	0	0
TCP_BOUND	0	0	0	0	0	0
TCP_CONNECTED	0	0	0	0	0	0
TCP_CONNECTING	0	0	0	0	0	0
TCP_LISTENING	0	0	0	0	0	0
TCP_DISCONNECT	0	0	0	0	0	0
TCP_CLOSED	0	0	0	0	0	0

2 - [SIP] Recognition of of Method (REGISTER)

Jan 29 15:16:09.620 [SIP] REGISTER 1 process-req: BEGIN

3 - [SIP] Parse request-URI

Jan 29 15:16:09.620 [SIP] Request-URI: sip:11.0.0.11
Jan 29 15:16:09.620 [SIP] REGISTER 1 process-req: has SIP I/F

4 - [SIP] Map received interface to realm

4.1 - Check access control (allow anonymous)

Jan 29 15:16:09.620 [SIP] UDP:11.0.0.11:5060 allow all anonymous from 11.0.0.101:5060 default-realm=access1
PE-6300-2# show acl info

Access Control List Statistics:

	# of entries	% utilization	Reserved Entry Count
Denied	1	0.0%	32768
Trusted	3	0.0%	16384
Media	0	0.0%	160000
Untrusted	2	0.0%	16384
Dynamic Trusted	0	0.0%	1000000

5 - [SIP] Check NAT-Traversal

Jan 29 15:16:09.620 [SIP] REGISTER 1 parse_via: diff IP & Port

****Note that this will only be seen if contact and via are different from ip source**

6 - [SIP] Sets Via rport to firewall's IP Address

Jan 29 15:16:09.620 [SIP] set Via rport 10.0.3.1=>11.0.0.101 rport=5060

7 - [SIP] Check for SIP-NAT

Jan 29 15:16:09.620 [SIP] REGISTER 1 use SIP-NAT Home-SA=<none>

8 - [SIP] Parse Via to determine where to send responses

8.1 - Includes Source IP/Port, ingress realm and session agent

Jan 29 15:16:09.620 [SIP] REGISTER 1 parse_via: reply=11.0.0.101:5060 from-Realm=access1 from-SA=<none>

9 - [SIP] Determination of configured proxy-mode

```
Jan 29 15:16:09.620 [SIP] determine_proxy_mode() - REGISTER 1 proxy-mode: [B2BUA]
irealm=access1:B2BUA SA=<none>:
```

10 - [SIP] Check Contact field's rport

```
Jan 15:16:09.620 [SIP] REGISTER 1 check_rport_contact: NATTED 11.0.0.101
Jan 29 15:16:09.620 [SIP] was:<sip:7001@10.0.3.1:5060>
Jan 29 15:16:09.620 [SIP]
now:<sip:7001@11.0.0.101:5060; Oracle_nat=7001+11.0.0.101@10.0.3.1:5060>
```

10.1 - Rewrite Contact rport-store both Contact and HNT rewritten Contact

11 - [SIP] Check manipulation (inbound)

```
Jan 29 15:16:09.620 [SIP] REGISTER 1 no inbound manipulation realm=access1 SA=<none>
```

12 - [SIP] REGISTER process continues

```
Jan 29 15:16:09.621 [SIP] REGISTER 1 process-req: CONT
```

```
IDC-4600-1# show registration
```

```
15:59:28-103
```

```
SIP Registrations      -- Period -- ----- Lifetime -----
      Active  High  Total   Total PerMax  High
User Entries    0    0    0      0    0    0
Local Contacts  0    0    0      0    0    0
Via Entries     0    0    0      0    0    0
AURI Entries    0    0    0      0    0    0
Free Map Ports  0    0    0      0    0    0
Used Map Ports  0    0    0      0    0    0
Forwards        -    -    0      0    0
Refreshes       -    -    0      0    0
Rejects         -    -    0      0    0
Timeouts        -    -    0      0    0
Fwd Postponed   -    -    0      0    0
Fwd Rejected    -    -    0      0    0
Refr Extension  0    0    0      0    0    0
Refresh Extended -    -    0      0    0
Surrogate Agnts 0    0    0      0    0    0
Surrogate Regs  0    0    0      0    0    0
Surrogate Sent   -    -    0      0    0
Surrogate Reject -    -    0      0    0
Surrogate Timeout -  -    0      0    0
HNT Entries     0    0    0      0    0    0
Non-HNT Entries 0    0    0      0    0    0
Secondary Interface Registrations = 0
Registrations marked for 305 = 0
```

13 - [SIP] Select the Request-URI

```
Jan 29 15:16:09.621 [SIP] Request-URI: sip:11.0.0.11
```

14 - [SIPNAT] - Check for SIP-NAT

```
Jan 29 15:16:09.621 [SIPNAT] REGISTER 1 fix_incoming 11.0.0.101:5060 no sip-nat
```

15 - [SIP] Check if it is a global URI

```
Jan 29 15:16:09.621 [SIP] non-global URI sip:11.0.0.11
```

16 - [SIP] Create cookie

```
Jan 29 15:16:09.621 [SIP] MakeCookie=[53500280]SDg9mg700-  
Jan 29 15:16:09.621 [SIP] cid=71353AE9-6864-47F2-BCFD-03BA58A0030F@11.0.0.101 cid created  
based on 10.1
```

17 - [SIPREG] Check for registry entry

```
Jan 29 15:16:09.621 [SIPREG] FindRegistry[access1|sip:7001@11.0.0.11] preferredStr =  
NULL Jan 29 15:16:09.621 [SIPREG] FindRegistry[access1|sip:7001@11.0.0.11] Not Found
```

18 - [SIPREG] Check for registry

```
Jan 29 15:16:09.621 [SIPREG] FindRegistry[access1|sip:7001@11.0.0.11] preferredStr =  
NULL Jan 29 15:16:09.621 [SIPREG] FindRegistry[access1|sip:7001@11.0.0.11] Not Found
```

19 - There is br (branch present) and also check if request-uri is SD address and
To-header is ESBC address

```
Jan 29 15:16:09.621 [SIP] REGISTER 1 process-req[B2BUA] irealm=access1 erealm=<none> br no-  
fix dest-is-SD to-is-SD
```

20 - [SIP] m (short form of Contact) and v (short form of Via)

```
Jan 29 15:16:09.621 [SIP] not-reg(m==v) not-secure SA=<none>
```

21 - [SIP] Check trusted/untrusted source

```
Jan 29 15:16:09.621 [SIP] REGISTER 1 check_trust(all) trusted 11.0.0.101:5060
```

22 - [TRANS] Note transaction process:

From Server-Initial -> Server Trying -> Server Initial -> Client Initial -> Client Trying...

```
Jan 29 15:16:09.621 [TRANS] REGISTER 1 B<S-Initial> added]
Jan 29 15:16:09.621 [TRANS] REGISTER 1 B<S-Initial> MREL=''
```

```
PE-6300-2# show sipd server
```

```
10:53:59-113
```

```
SIP Server Trans      -- Period -- ----- Lifetime -----
Active  High  Total      Total  PerMax  High
All States            0    0    0          0    0    0
<Initial>             0    0    0          0    0    0
<Queued>              0    0    0          0    0    0
<Trying>              0    0    0          0    0    0
<Proceeding>         0    0    0          0    0    0
<Cancelled>          0    0    0          0    0    0
<Established>        0    0    0          0    0    0
<Completed>          0    0    0          0    0    0
<Confirmed>          0    0    0          0    0    0
<Terminated>         0    0    0          0    0    0
```

23 - Create transaction (branch-ID) and start timers, "set timer 37000" means that a the timer is 5 secs more than the 32 sec timer to give additional time to the client transaction to complete before it is passed over to the server side.

```
Jan 29 15:16:09.621 [SIP] REGISTER 1 process-req: added server transaction 0x142a7f70(2)
B2BUA
Jan 29 15:16:09.621 [SIP]
transid=z9hg4bK0b0000650000001045be4b070000303a00000001|10.0.3.1|REGISTER|1
Jan 29 15:16:09.621 [SIP] REGISTER 1 B<S-Initial> ST-process: in-req: non-dialog
Jan 29 15:16:09.621 [TRANS] REGISTER 1 <S-Trying> WAS <S-Initial>
Jan 29 15:16:09.621 [TRANS] REGISTER 1 B<S-Trying> set timer 37000
Jan 29 15:16:09.621 [TRANS] REGISTER 1 B<S-Trying> EXPIRE in 37000 msecs
```

24 - [SIP] Request passed to core for being processed

```
Jan 29 15:16:09.622 [SIP] REGISTER 1 B<S-Trying> ST-process: request passed to Core
```

25 - [SIP]/[SIPREG] Core processing begins - check sip-config\registrar settings

```
Jan 29 15:16:09.622 [SIP] REGISTER 1 Core: PROCESS (B2BUA)
Jan 29 15:16:09.622 [SIP] target=SD=sip:11.0.0.11
Jan 29 15:16:09.622 [SIP] DestIsUs[socket] host='11.0.0.11'
Jan 29 15:16:09.622 [SIP] DestIsReg: NO MATCH host='11.0.0.11' reg='' domain=''
Jan 29 15:16:09.622 [SIP] REGISTER 1 Core: Determine Next Hop
```

```
Jan 29 15:16:09.622 [SIP]
Jan 29 15:16:09.622 [SIP]
```

```
REGISTER 1 B<S-Trying> CkRegister
                                NATTED To-RURI=SD sip:11.0.0.11
                                0x14875780 REGISTER 1 <200> load out-resp packet from in-req
                                0x142e0e10P
Jan 29 15:16:09.622 Jan 29 15:16:09.622 [SIP] dialog=0x0(0) session=0x0(0)
Jan 29 15:16:09.622 realmIn=0x142c89e0(7)access1(access1) realmOut=0x0(0)<none>( )
Jan 29 15:16:09.622 Jan 29 15:16:09.622 [SIP] reply=11.0.0.101:5060(11.0.0.101:5060)
psock=0x142e7f70(8)
(0x142e7f70(8))
[SIPREG] Register RURI=sip:11.0.0.11 min-left=34
[SIPREG] AOR=sip:7001@11.0.0.11
[SIPREG] first
contact=<sip:7001@11.0.0.101:5060;Oracle_nat=7001+11.0.0.101@10.0.3.1:5060>
Jan 29 15:16:09.622 [SIPREG] DoRegister[sip:7001@11.0.0.11]
Jan 29 15:16:09.622 [SIPREG]
contact=sip:7001@11.0.0.101:5060;Oracle_nat=7001+11.0.0.101@10.0.3.1:5060
Jan 29 15:16:09.622 [SIPREG] DoRegister[sip:7001@11.0.0.11] forward to real registrar
Jan 29 15:16:09.622 [SIP] REGISTER 1 Core: COPY REQUEST in dest-is-SD to-is-SD(keep)
Jan 29 15:16:09.623 [SIP] 0x142e11a0 REGISTER 1 copy out-req packet from in-req 0x142e0e10
Jan 29 15:16:09.623 [SIP] dialog=0x0(0) session=0x0(0) SA=0x0(0)dest-not-SD to-is-SD(keep-
to)
```

26 - [SIP] Check Local Policy Classifiers

```
Jan 29 15:16:09.623 [SIP] REGISTER 1 LocalPolicy: sip:11.0.0.11
Jan 29 15:16:09.623 [SIP] REGISTER 1 GetRoutes[access1]
Jan 29 15:16:09.623 [SIP] from-URI=<sip:7001@11.0.0.11>;tag=97788597147
Jun 13 11:02:32.288 [TRIP] (0) IRealm:ATT-Trunk FromTo:SIP:Hostname:1:*:Hostn
Jun 13 11:02:32.288 [TRIP] (0) <any> U-S 0000 2400 cost=0000 NH=[Core]SAG
Jun 13 11:02:32.288 [TRIP] (0) TRIBofRoutes::addRoute() - created ExtTRIB route
Jun 13 11:02:32.288 [TRIP] (0) IRealm:Core FromTo:SIP:Hostname:1:*:Hostname:1
Jun 13 11:02:32.288 [TRIP] (0) <any> U-S 0000 2400 cost=0000 NH=[ATT-Trun
Jun 13 11:02:32.288 [TRIP] (0) TRIBofRoutes::addRoute() - created LocTRIB route
Jun 13 11:02:32.288 [TRIP] (0) IRealm:ATT-Trunk FromTo:SIP:Hostname:1:*:Hostn
Jun 13 11:02:32.289 [TRIP] (0) <any> U-S 0000 2400 cost=0000 NH=[Core]SAG
Jun 13 11:02:32.289 [TRIP] (0) TRIBofRoutes::addRoute() - created LocTRIB route
Jun 13 11:02:32.289 [TRIP] (0) IRealm:Core FromTo:SIP:Hostname:1:*:Hostname:1
Jun 13 11:02:32.289 [TRIP] (0) <any> U-S 0000 2400 cost=0000 NH=[ATT-Trun
Jun 13 11:02:32.289 [GENERAL] (0) TRIPProcess::loadConfiguration() - done updati
Jun 13 11:02:32.289 [CONFIG] (0) Load Session Agent Groups
```

26.1 - Note that TO URI (AoR) is checked)

```
Jan 29 15:16:09.623 [SIP] to-URI=sip:11.0.0.11
Jan 29 15:16:09.623 [SIP] REGISTER 1 GetRoutes: got 1 routes
```

26.2 - Check Policy Attributes [MP = media policies, pref= preference]

```
Jan 29 15:16:09.623 [SIP] 1=<any> UMTWRFS 0000 2400 cost=0000 MP=0
NH=[backbone]172.16.0.100 SA pref=0
Jan 29 15:16:09.623 [SIP] REGISTER 1 LocalPolicy: 1 routes (access1)
Jan 29 15:16:09.623 [SIP] From=<sip:7001@11.0.0.11>;tag=97788597147
Jan 29 15:16:09.623 [SIP] To=sip:11.0.0.11
Jan 29 15:16:09.623 [SIP] DestIs-NOT-Us: host='172.16.0.100'
```

26.3 - Check for next-hop SA configuration

```
Jan 29 15:16:09.623 [SIP] REGISTER 1 LocalPolicy: check next-hop 172.16.0.100 carrier=
realm=backbone SA=172.16.0.100
Jan 29 15:16:09.623 [SIP] makeSipRouteFromSA: SAG = | SA = 172.16.0.100
Jan 29 15:16:09.623 [SIP] makeSipRouteFromSA: use LP realm=backbone
Jan 29 15:16:09.624 [SIP] sip:172.16.0.100
Jan 29 15:16:09.624 [SIP] makeSipRouteFromSA: Setting outbound port= 5060
```



```
Jan 29 15:16:09.624 [SIP] makeSipRouteFromSA: add next-hop
Jan 29 15:16:09.624 [SIP] [backbone|172.16.0.100|]sip:172.16.0.100:5060 q=1000
```

26.4 - q value is priority for each contact(per RFC 3261) -> 1(000)=highest match

```
Jan 29 15:16:09.624 [SIP] REGISTER 1 LocalPolicy: add next-hop
Jan 29 15:16:09.624 [SIP] [backbone|172.16.0.100|]sip:172.16.0.100:5060 q=1000
Jan 29 15:16:09.624 [SIPREG] REGISTER 1 use route's egress realm
Jan 29 15:16:09.624 [SIPREG] [backbone|172.16.0.100|]sip:172.16.0.100:5060 q=1000
Jan 29 15:16:09.624 [SIPREG] REGISTER 1 GetRegRoutes[sip:7001@11.0.0.11] egress=backbone
Jan 29 15:16:09.624 [SIPREG] local policy routes 1/1
Jan 29 15:16:09.624 [SIPREG] =>1=[backbone|172.16.0.100|]sip:172.16.0.100:5060 q=1000
```

27 - Add provisional Registration Cache entry (To be set as valid after 200OK response)

```
Jan 29 15:16:09.624 [SIPREG] ForwardRegister[sip:7001@11.0.0.11] update/add entry
```

```
Jan 29 15:16:09.624 [SIPREG] UA-
contact=sip:7001@11.0.0.101:5060;Oracle_nat=7001+11.0.0.101@10.0.3.1:5060
Jan 29 15:16:09.624 [SIPREG] SD-contact=sip:7001-rjum22st7e7e@172.16.0.10:5060
Jan 29 15:16:09.624 [SIPREG] Added AOR[sip:7001@11.0.0.11]
Jan 29 15:16:09.624 [SIPREG] User[sip:7001@11.0.0.11] Added URI-SD[sip:7001-
rjum22st7e7e@172.16.0.10:5060]
Jan 29 15:16:09.624 [SIPREG] SipContact[0x142a85d0]-N sip:7001@11.0.0.11 set_reg_rimer to 30
secs
Jan 29 15:16:09.624 [SIPREG] UA-
Contact=sip:7001@11.0.0.101:5060;Oracle_nat=7001+11.0.0.101@10.0.3.1:5060
Jan 29 15:16:09.624 [SIPREG] SipContact[0x142a85d0] sip:7001@11.0.0.11 SD-Contact expires in
30 secs (half=15)
Jan 29 15:16:09.624 [SIPREG] UA-
Contact=sip:7001@11.0.0.101:5060;Oracle_nat=7001+11.0.0.101@10.0.3.1:5060
Jan 29 15:16:09.625 [SIPREG] AOR[sip:7001@11.0.0.11] added Contact (not-valid)
Jan 29 15:16:09.625 [SIPREG]
sip:7001@11.0.0.101:5060;Oracle_nat=7001+11.0.0.101@10.0.3.1:5060
Jan 29 15:16:09.625 [SIPREG] Contact for <sip:7001@11.0.0.11> <not-valid> <expired>
Jan 29 15:16:09.625 [SIPREG] UA-Contact:
<sip:7001@11.0.0.101:5060;Oracle_nat=7001+11.0.0.101@10.0.3.1:5060>
Jan 29 15:16:09.625 [SIPREG] realm=access1 local=11.0.0.11:5060 UA=11.0.0.101:5060
Jan 29 15:16:09.625 [SIPREG] SD-Contact: <sip:7001-rjum22st7e7e@172.16.0.10:5060>
realm=backbone
Jan 29 15:16:09.625 [SIPREG] ForwardRegister[sip:7001@11.0.0.11] (exp=3600) to sip:11.0.0.11
Jan 29 15:16:09.625 [SIPREG] UA-
contact=sip:7001@11.0.0.101:5060;Oracle_nat=7001+11.0.0.101@10.0.3.1:5060
Jan 29 15:16:09.625 [SIPREG] SD-contact=sip:7001-rjum22st7e7e@172.16.0.10:5060
Jan 29 15:16:09.625 [SIPREG] < Request >-----
Jan 29 15:16:09.625 [SIPREG] REGISTER sip:11.0.0.11 SIP/2.0
Jan 29 15:16:09.625 [SIPREG] Content-Length: 0
Jan 29 15:16:09.625 [SIPREG] Contact: <sip:7001-rjum22st7e7e@172.16.0.10:5060>
Jan 29 15:16:09.625 [SIPREG] Call-ID: 71353AE9-6864-47F2-BCFD-03BA58A0030F@11.0.0.101
Jan 29 15:16:09.625 [SIPREG] CSeq: 1 REGISTER
Jan 29 15:16:09.625 [SIPREG] From: <sip:7001@11.0.0.11>;tag=97788597147
Jan 29 15:16:09.625 [SIPREG] Max-Forwards: 69
Jan 29 15:16:09.625 [SIPREG] To: <sip:7001@11.0.0.11>
Jan 29 15:16:09.625 [SIPREG] User-Agent: SJphone/1.60.289a (SJ Labs)
Jan 29 15:16:09.625 [SIPREG] -----
Jan 29 15:16:09.625 [SIPREG] 1/1
Jan 29 15:16:09.625 [SIPREG] =>1=[backbone|172.16.0.100|]sip:172.16.0.100:5060 q=1000
```



28 - [SIP] Start assembling egress packet and checking next-hop
an SA, set SA parameters)

(found as

```
Jan 2915:16:09.625 [SIP] REGISTER 1 B2BUA: in sip:11.0.0.11
Jan 2915:16:09.625 [SIP] MakeCookie=[53500280]SDg9mg700-
Jan 2915:16:09.625 [SIP] cid=71353AE9-6864-47F2-BCFD-03BA58A0030F@11.0.0.101
Jan 2915:16:09.625 [SIP] MakeCallID:(not-cookie)
Jan 2915:16:09.626 [SIP] id=71353AE9-6864-47F2-BCFD-03BA58A0030F@11.0.0.101
Jan 2915:16:09.626 [SIP] tag=SDg9mg700-
Jan 2915:16:09.626 [SIP] REGISTER 1 B2BUA: make Call-ID for
Jan 2915:16:09.626 [SIP] 71353AE9-6864-47F2-BCFD-03BA58A0030F@11.0.0.101
Jan 2915:16:09.626 [SIP] SDg9mg701-e8f6ba27234af2f55604dce5dbcd10bb-c540050
Jan 2915:16:09.626 [SESSION] REGISTER 1 make_callid_cookie(callid)='SDg9mg701-'
Jan 2915:16:09.626 [SIP] REGISTER 1 B2BUA: set From tag to SDg9mg701-97788597147
Jan 2915:16:09.626 [SIP] REGISTER 1 B2BUA: fix headers
Jan 2915:16:09.626 [SIP] REGISTER 1 B2BUA: keep Contact(fixed):
Jan 2915:16:09.626 [SIP] <sip:7001-rjumn22st7e7e@172.16.0.10:5060>
Jan 2915:16:09.626 [SIP] DestIsUs[socket] host='11.0.0.11'
Jan 2915:16:09.626 [SIP] REGISTER 1 Core: add LP target
Jan 2915:16:09.626 [SIP] [backbone|172.16.0.100|]sip:172.16.0.100:5060 q=1000
Jan 2915:16:09.626 [SIP] REGISTER 1 Core: sip:11.0.0.11 created context
Jan 2915:16:09.626 [SIP] REGISTER 1 <S-Trying> set_context 0x142dfc50(1)
Jan 2915:16:09.626 [SIP] REGISTER 1 SipContext::add_target:
Jan 2915:16:09.626 [SIP] [backbone|172.16.0.100|]sip:172.16.0.100:5060 q=1000
Jan 2915:16:09.626 [SIP] targets: 1/1
Jan 2915:16:09.626 [SIP] =>1=[backbone|172.16.0.100|]sip:172.16.0.100:5060 q=1000
Jan 2915:16:09.626 [SIP] routes: 0/0
Jan 2915:16:09.626 [SIP] next-hops: 0/0
Jan 2915:16:09.626 [TRANS] REGISTER 1 ctx-forward-req: <none>
Jan 2915:16:09.626 [TRANS] targets: 1/1
```

```

Jan 29 15:16:09.626 [TRANS] =>1=[backbone|172.16.0.100|]sip:172.16.0.100:5060 q=1000
Jan 29 15:16:09.627 [TRANS] routes: 0/0
Jan 29 15:16:09.627 [TRANS] next-hops: 0/0
Jan 29 15:16:09.627 [SIP] 0x1487b130 REGISTER 1 copy out-req packet from out-req 0x142e11a0
Jan 29 15:16:09.627 [SIP] dialog=0x0(0) session=0x0(0) SA=0x0(0) dest-not-SD to-is-SD(keep-
to)
Jan 29 15:16:09.627 [SIP] REGISTER 1 FindNextHop: no route; next hop is
Jan 29 15:16:09.627 [SIP] sip:172.16.0.100:5060
Jan 29 15:16:09.627 [SIPNAT] REGISTER 1 check_out_agent: 172.16.0.100 not behind SIP-NAT
Jan 29 15:16:09.627 [SIP] REGISTER 1 check_next_hop: select protocol (pref=<none>) SA
Jan 29 15:16:09.627 [SIP] sip:172.16.0.100:5060
Jan 29 15:16:09.627 [SIP] REGISTER 1 check_next_hop: need to select protocol
Jan 29 15:16:09.627 [SIP] REGISTER 1 select_proto: realm=backbone UDP pref=UDP+TCP+TLS+DTLS
supdSA=UDP; selected UDP
Jan 29 15:16:09.627 [SIP] REGISTER 1 check_natted_target: sip:172.16.0.100:5060 not-natted
Jan 29 15:16:09.627 [SIP] REGISTER 1 check_next_hop: non-dialog fix is SA (no-via-orig)
realm=backbone no-sip-nat
Jan 29 15:16:09.627 [SIP] sip:172.16.0.100:5060
Jan 29 15:16:09.627 [SIP] REGISTER 1 SipContext::find_next_hops:
Jan 29 15:16:09.627 [SIP] SipRoute [backbone|172.16.0.100|]sip:172.16.0.100:5060
q=344459084 found SA
Jan 29 15:16:09.627 [SIP] [backbone|172.16.0.100|]sip:172.16.0.100:5060 q=1000
Jan 29 15:16:09.627 [SIP] add next-hop(IP) 172.16.0.100:5060/UDP
Jan 29 15:16:09.627 [TRANS] REGISTER 1 SipContext::forward_request: (realm=backbone):
Jan 29 15:16:09.627 [TRANS] got next_hops 1/1
Jan 29 15:16:09.627 [TRANS] =>1=172.16.0.100:5060/UDP
Jan 29 15:16:09.628 [SIP] 0x14879840 REGISTER 1 copy out-req packet from out-req 0x1487b130
Jan 29 15:16:09.628 [SIP] dialog=0x0(0) session=0x0(0) SA=0x142f6710(14) dest-not-SD to-is-
SD(keep-to)
Jan 29 15:16:09.628 [SIP] REGISTER 1 check_natted_target: sip:172.16.0.100:5060 not-natted Jan 29
15:16:09.628 [SIP] REGISTER 1 ctx-forward-req: try NEXT-HOP keep-To realm=backbone To-
SA=172.16.0.100
Jan 29 15:16:09.628 [SIP] REGISTER 1 ctx-forward-req: check_outgoing
Jan 29 15:16:09.628 [SIP] 172.16.0.100:5060/UDP SA realm=backbone
Jan 29 15:16:09.628 [SIPNAT] REGISTER 1 check_outgoing 172.16.0.100:5060/UDP realm=backbone
no sip-nat
Jan 29 15:16:09.628 [SESSION] REGISTER 1 made default via-branch 9p2lmt206g20easg04g0
Jan 29 15:16:09.628 [TRANS] REGISTER 1 <C-Initial> WAS <NONE>
Jan 29 15:16:09.628 [SIP] SipTrans::set_caller origin
Jan 29 15:16:09.628 [TRANS] REGISTER 1 ctx-forward-req:
Jan 29 15:16:09.628 [TRANS] to 172.16.0.100:5060/UDP SA realm=backbone=backbone
Jan 29 15:16:09.628 [TRANS] CT:0x142a7800(1)[REGISTER:1]<C-Initial> W Q:0x14879840(3) R:0x0(0)
CX:<none>
Jan 29 15:16:09.628 [TRANS] create=15:16:09.628 last=15:16:09.628 noexp
Jan 29 15:16:09.628 [TRANS] TXID=
Jan 29 15:16:09.628 [SIP] REGISTER 1 <C-Initial> set_context 0x142dfc50(3)
Jan 29 15:16:09.628 [SIP] REGISTER 1 B<C-Initial> SipContext::check_media_forward: setup
Jan 29 15:16:09.628 [SIP] REGISTER 1 B<C-Initial> setup_media: no media directing
Jan 29 15:16:09.628 [SIP] REGISTER 1 B<C-Initial> SipContext::check_media_forward: media
setup done (send now)
Jan 29 15:16:09.629 [SIP] send_request: CLF realm_in is set
Jan 29 15:16:09.629 [SIP] realm-in is: access1
Jan 29 15:16:09.629 [SIP] continue_send_request:
Jan 29 15:16:09.629 [SIP] REGISTER 1 B<C-Initial> req-send(fix) ct-next-hop =
172.16.0.100:5060/UDP is good
Jan 29 15:16:09.629 [SIP] REGISTER 1 req-send(fix) get destination
Jan 29 15:16:09.629 [SIP] REGISTER 1 B<C-Initial> req-send: next-hop=172.16.0.100:5060/UDP
from Client Trans
Jan 29 15:16:09.629 [SIP] 1/1
Jan 29 15:16:09.629 [SIP] =>1=172.16.0.100:5060/UDP
Jan 29 15:16:09.629 [SIP] REGISTER 1 req-send(fix) 0x14879840 target=172.16.0.100:5060/UDP

```

29 - Checking if outbound SIP-NAT configured

```

Jan 29 15:16:09.629 [SIPNAT] REGISTER 1 check_outgoing 172.16.0.100:5060/UDP realm=backbone no sip-
nat

```

30 - Verifying SIP Interface settings

```

Jan 29 15:16:09.629 [SIP] REGISTER 1 req-send: get socket for target=172.16.0.100:5060/UDP
local=0.0.0.0:0 SA
Jan 29 15:16:09.629 [SIP] get_udp_socket backbone use default UDP:172.16.0.10:5060
Jan 29 15:16:09.629 [SIP] REGISTER 1 req-send: target=172.16.0.100:5060/UDP 172.16.0.100:5060
on UDP:172.16.0.10:5060
Jan 29 15:16:09.629 [SIP] REGISTER 1 check_egress_privacy: ingress side is trusted
Jan 29 15:16:09.629 [SIP] REGISTER 1 check_trust(all) untrusted SA[172.16.0.100]
172.16.0.100:5060

```

30.1 - IMS Features disabled

```
Jan 29 15:16:09.629 [SIP] convert_ppi_to_pai value of isTrusted is : disabled
Jan 29 15:16:09.629 [SIP] REGISTER 1 create_visited_network_id: no IMS; do nothing
Jan 29 15:16:09.629 [SIP] REGISTER 1 create_charging_headers
Jan 29 15:16:09.629 [SIP] P-Charging-Vector mode is :none
Jan 29 15:16:09.629 [SIP] P-Charging-Function-Addressess mode is :none
Jan 29 15:16:09.629 [SIPNAT] REGISTER 1 fix_outgoing packet 172.16.0.100:5060 no
Jan 29 15:16:09.630 [SIPNAT] REGISTER 1 set_egress_interface(backbone:1,0)
src=172.16.0.10:5060 pkt=0x142ba700(1)
```

31 - Finishes assembling packet to be sent

```
Jan 29 15:16:09.630 [SIP] via=SIP/2.0/UDP 172.16.0.10:5060
Jan 29 15:16:09.630 [SIP] REGISTER 1 req-send: added Via:
Jan 29 15:16:09.630 [SIP] SIP/2.0/UDP 172.16.0.10:5060;branch=z9hG4bK9p2lmt206g20easgo4g0.1
Jan 29 15:16:09.630 [SIP] CT:0x142a7800(2)[REGISTER:1]<C-Initial> W Q:0x14879840(3) R:0x0(0)
CX:0x142dfc50(4)
Jan 29 15:16:09.630 [SIP] create=15:16:09.628 last=15:16:09.629 noexp
Jan 29 15:16:09.630 [SIP] TXID=z9hG4bK9p2lmt206g20easgo4g0.1|172.16.0.10|REGISTER|1 Jan 29
15:16:09.630 [SIP] REGISTER 1 req-send: to 172.16.0.100:5060 on 0x142e6ef0(5) UDP:172.16.0.10:5060
fix
Jan 29 15:16:09.630 [SIP] REGISTER 1 fix contact: check(172.16.0.10) <sip:7001-
rjum22st7e7e@172.16.0.10:5060>
Jan 29 15:16:09.630 [SIP] REGISTER 1 fix contact:
Jan 29 15:16:09.630 [SIP] was=<sip:7001-rjum22st7e7e@172.16.0.10:5060>
Jan 29 15:16:09.630 [SIP] now=<sip:7001-rjum22st7e7e@172.16.0.10:5060;transport=udp> Jan 29
15:16:09.630 [SIP] REGISTER 1 fix_natted_ruri: NO-NAT
```

32 - Outbound SIP Header Manipulation

```
Jan 29 15:16:09.630 [SIP] REGISTER 1 req-send: do_header_manipulation:
local=172.16.0.10:5060; remote=172.16.0.100:5060
Jan 29 15:16:09.630 [SIP] REGISTER 1 got the outbound manipulation from 'NAT_IP'
realm=backbone
Jan 29 15:16:09.630 [SIP] Got Header rule for Header = From HeaderName = From Action =
manipulate cmpType=case-sensitive
Jan 29 15:16:09.630 [SIP] Before Manipulation From Header Value =
'<sip:7001@11.0.0.11>;tag=SDg9mg701-97788597147'
Jan 29 15:16:09.631 [SIP] From: Element[FROM] FROM replace uri-host
Jan 29 15:16:09.631 [SIP] SipElemRule[FROM] value '11.0.0.11' is an IP Address
Jan 29 15:16:09.631 [SIP] resolveValTempl: Resolved Template Value = 172.16.0.10
Jan 29 15:16:09.631 [SIP] Header From Manipulated, New Value =
'<sip:7001@172.16.0.10>;tag=SDg9mg701-97788597147'
Jan 29 15:16:09.631 [SIP] Got Header rule for Header = To HeaderName = To Action = manipulate
cmpType=case-sensitive
Jan 29 15:16:09.631 [SIP] Before Manipulation To Header Value = '<sip:7001@11.0.0.11>'
Jan 29 15:16:09.631 [SIP] To: Element[TO] TO replace uri-host
Jan 29 15:16:09.631 [SIP] SipElemRule[TO] value '11.0.0.11' is an IP Address
Jan 29 15:16:09.631 [SIP] resolveValTempl: Resolved Template Value = 172.16.0.100
Jan 29 15:16:09.631 [SIP] Header To Manipulated, New Value = '<sip:7001@172.16.0.100>'
```

33 - Packet ready to be sent

```
Jan 29 15:16:09.631 [SIP] PIPE:127.0.0.1:6000 is_local_socket tgt=[1:0]172.16.0.100:5060 is
NOT SD
Jan 29 15:16:09.632 [TRANS] REGISTER 1 B<C-Initial> req-send: save buffer; len=466
Jan 29 15:16:09.632 [TRANS] REGISTER 1 B<C-Initial> add client transaction
Jan 29 15:16:09.632 [TRANS] CT:0x142a7800(3)[REGISTER:1]<C-Initial> W Q:0x14879840(3) R:0x0(0)
CX:0x142dfc50(4)
Jan 29 15:16:09.632 [TRANS] create=15:16:09.628 last=15:16:09.629 noexp
Jan 29 15:16:09.632 [TRANS] TXID=z9hG4bK9p2lmt206g20easgo4g0.1|172.16.0.10|REGISTER|1 Jan 29
15:16:09.632 [TRANS] REGISTER 1 B<C-Initial> added
```

```

Jan 29 15:16:09.632
Jan 29 15:16:09.632
Jan 29 15:16:09.632 33.1 - Packet finally sent
Jan 29 15:16:09.632 [TRANS] REGISTER 1 B<C-Initial> SENT to
Jan 29 15:16:09.632 172.16.0.100:5060 SA=172.16.0.100
realm=backbone
[SIP] Interface backbone processMessageSentEvent
[TRANS] REGISTER 1 <C-Trying> WAS <C-Initial>
[TRANS] REGISTER 1 B<C-Trying> set timer 500
[TRANS] REGISTER 1 B<C-Trying> CTSend: sent request to
172.16.0.100:5060/UDP
Jan 29 15:16:09.632 [SESSION] Did Not Find CallIDMap C|backbone|SDg9mg701-
e8f6ba27234af2f55604dce5dbcd10bb-c540050|SDg9mg701-97788597147
Jan 29 15:16:09.632 [SESSION] REGISTER 1 out-req: get_session 0x0(0) find_callid_map (not
found)
Jan 29 15:16:09.632 [SIP] REGISTER 1 B<C-Trying> check_media_and_forward done rc=200
Jan 29 15:16:09.632 [SIP] REGISTER 1 B<C-Trying> ctx-forward-req: sent 1 requests

```

Normally, for logs when log-level is set to INFO, you can do an easy search for error as an key word in order to see if there is any "unusual" in the system. However, we can appreciate there are some essential keywords which need to be consider when you want to match a special task or problem. Here you can see a summary of the most important:

Keyword	Looking for/Meaning
Socket	Packet Received
Resp-proc	Processing a response which has been received
Process-req	Processing a request that has been received
Process-req: CONT	Continuation of processing a transaction after another process was performed (i.e. you can see after DNS)
CTX-forward- request	When the processing of the transaction is ended
prefferedSTR	Look for P-Preffered-Identity string
Determine Next Hop	If you want to check Local-Policies lookups
Targets	On Local-Policy lookups, when the original R-URI is SD then local-policy match results in targets
Route	On Local-Policy lookups, if the original R-URI isnot the ESBC (say a Registrar domain) then local-policy match results in route which is put in route header as a loose route and the original R-URI is
Next-Hop	On Local-Policy lookups, Next-Hop is determined after the R-uri/loose route is resolved

SENT	The packet has finally been sent (On wire)
------	--

Also, although not present in **log.sipd**, it is worthy to mention the following keywords present in **log.mbcd**. These can help the engineer to correlate sip and rtp traffic and they are really useful for one of the “top ten” problems found on deployments, the “one-way audio”.

Keyword	Looking for/Meaning
ADD	When flow is created (you can then search for flow)
MODIFY	When flow is modified
SUBSTRACT	When flow is deleted
Latch	When there is traffic arriving ESBC network and the flow has been latched interface

Peering Environment Log Analysis Key Words:

Inside a peering environment, a lot of key words can help us in locating problem. The idea is to avoid studying the log files line by line before locating a problem. Here below are the key words helpful to your quick log study:

Command: show sipd <Method>

INVITE (11:33:08-102)----- Argument , Timestamp, recent period timer..
 Server Column ----- ESBC acting as a User Agent Server (UAS),
 Client Column ----- ESBC acting as a User Agent Client (UAC).
 Recent column ----- the recent period (the current period plus the last period).
 Total column ----- the total for a particular metric since the last stats clearing
 PerMax column ----- the maximum for a given metric seen in any given individual (current) period

Example:

```
DFW1_1# show sip invite
INVITE (11:33:08-102)
```

Message/Event		Server	Total	PerMax	Recent	PerMax
INVITE Requests	29	1	44	29	519	4
Retransmissions	0		1	0	1	4
100 Trying	28	519	44	29	519	4
180 Ringing	25	2	33	25	405	1
181 Forwarded	1	489	3	1	10	4
		405				4
		10				3
						3
						3

```

183 Progress
200 OK                11      250   21      11      250      21
404 Not Found         1       25    4       1       25       4
480 Unavailable        4       83    8       4       83       8
486 Busy Here          1       30    5       1       30       5
487 Terminated        4       97   10       4       97      10
488 Not Acceptable     1        7    2       1        7       2
500 Internal Error      2       21    3       2       21       3
502 Bad Gateway         0        3    1       0        3       1
Response Retrans       0       22   10       0       23      11
Transaction Timeouts   -        -    -       0        0       0

```

PE- show sipd client

10:52:59-153

```

SIP Client Trans
Active  -- Period --  ----- Lifetime -----
High  Total  Total  PerMax  High
All States  4      4      1      1796    9      5
<Initial>   0      1      1      1796    9      4
<Trying>    0      1      1       719    3      1
<Calling>   0      0      0        0     0     0
<Proceeding> 0      0      0        0     0     0
<Cancelled>  0      0      0        0     0     0
<EarlyMedia> 0      0      0        0     0     0
<Completed>  3      3      0     1077    3      3
<SetMedia>  0      0      0        0     0     0
<Established> 0      0      0        0     0     0
<Terminated> 0      1      1     1792    6      1

```

PE-6300-2# show sipd transcode

Transcode Codecs:

```

PCMU clock: 8000  ptime: 10 20 30 40 50 60
PCMA clock: 8000  ptime: 10 20 30 40 50 60
G729 clock: 8000  ptime: 10 20 30 40 50 60 70 80 90
G729A clock: 8000 ptime: 10 20 30 40 50 60 70 80 90
iLBC clock: 8000  ptime: 20 30 40 60
telephone-event
T.38
G711FB clock: 8000 ptime: 10 20 30
G726 clock: 8000  ptime: 10 20 30 40 50
G726-16 clock: 8000 ptime: 10 20 30 40 50
G726-24 clock: 8000 ptime: 10 20 30 40 50
G726-32 clock: 8000 ptime: 10 20 30 40 50
G726-40 clock: 8000 ptime: 10 20 30 40 50
G723 clock: 8000  ptime: 30 60 90
G722 clock: 8000  ptime: 10 20 30 40
GSM clock: 8000  ptime: 20
AMR clock: 8000  ptime: 20 40 60 80 100
interleaving != 1 optional parameter
robust-sorting != 1 optional parameter
channels == 1 optional parameter
crc == 0 optional parameter
AMR-WB clock: 16000 ptime: 20 40 60 80 100
interleaving != 1 optional parameter
robust-sorting != 1 optional parameter
channels == 1 optional parameter

```

E-6300-2# show sipd groups

11:11:33-18

```

----- Inbound -----  ---- Outbound -----  -- Latency --  Max
SAG Active Rate ConEx Active Rate ConEx Avg Max Burst
Avaya-SM-SAG I 0 0.0 0 0 0.0 2 0.000 0.000 0

```

In **sipmsg.log**:

```

Opened Service Pipes      Maps to instantiated SIP-Interface\SIP-Ports
2006-10-16 19:13:55.245  OPENED MBC:2945
2006-10-16 19:13:55.246  STARTED UDP:11.0.0.11:5060}
2006-10-16 19:13:55.246  STARTED UDP:172.16.0.11:5060

```

Sep 4 16:48:47.809 ▼ TimeStamp

[1:0]-----received interface [<interface>:<VLAN-ID>

► Interfaces

0 – M00

1 – M10

2 – M01

3 – M11

4 – M02

5 – M12

6 – M03

7 – M13

168.18.187:5060-----Destination/port

192.168.0.8:5060 -----Source/port

e.g.

```
Sep 4 16:48:47.809 On [1:0]192.168.18.187:5060 received from 192.168.0.8:5060
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 192.168.18.187:5060;branch=z9hG4bK080gl510b8u08cga33s0.1
From: "Anonymous" <sip:192.168.18.187:5060>;tag=3366399615-31685
To: 828524878726926 <sip:828524878726926@192.168.18.187>
Call-ID: 11899931-3366399615-31613@latinomswl.latinonet.net.mx
CSeq: 1 INVITE
Content-Length: 0
```

In **log.sipd**:

CT- Client transaction

Command:

ST – Server transaction

Request

Response

CX: Context

create - Created Timestamp

last - last state change timestamp

sent - sent timestamp

rcvd - received timestamp

e.g.

```
[SIP] INVITE 101 B<S-Initial> ST-process: in-req: no session/dialog
[TRANS] CT:0x150c9d70(1)[INVITE:101]<C-Initial> W Q:0x16063ff0(3) R:0x0(0) CX:<none>
```

Process-req: CONT

- Continuation of processing a transaction after another process was performed - See this after DNS lookup CONT – means that if any DNS resolution is required.e.g.

```
[SIP] INVITE 101 process-req: CONT
```

Resp-proc - processing a response which was received

Process-req:- processing a request that was received

e.g.

```
[SIP] INVITE 101 process-req[B2BUA] irealm=access erealm=<none> br no-fix
```

Target – What will show up in the Request URI

Routes – What will show up in the route header.

Next Hop – What will show up after the resolution, what IP Address will be used.

e.g.

```
INVITE 101 ctx-forward-req: <none>
30:00.012 [TRANS] targets: 1/1
30:00.012 [TRANS] =>1=[H323_GW|172.16.205.101|]sip:01143410100@172.16.205.101:1720(i) q=1000
30:00.012 [TRANS] routes: 0/0
30:00.012 [TRANS] next-hops: 0/1
```

West- Calling party

East – Called party

e.g.

```
[MEDIA] 0x150d6320[0] East sub changed from to 127.0.0.1
31:23.017 [MEDIA] 0x150d6320[0] West sub changed from to 200.68.89.4
```



Tools

There are many and various tools we can use to verify correct operation and/or troubleshoot an issue. We generally view tools as some type of application, or third party software that is used in analysis of sessions. A tool can also be a command, a job aide, a log or an application.

Online tools

Online tool is a list of existing CLI command for log analysis purpose.

Here below are the existing available online commands.

- archives

Log files can be archived into "tar.gz" format. It is easy for download and be able to classify the different nature of log.

- display-logfiles

Log files in /logs folder can be displayed so as to know whether the log has been generated and file size has been changed.

- log-level

This command can be used to enable all log categories per log file to be debug or precisely only trigger a certain log category to be debug or any other level. Examples have been documented in previous chapter.

- notify

Log files debug level can be triggered or rotated by notify command. It is like a batch file to make use of log-level command to trigger certain important log categories to debug. Below are the examples:

```
notify sipd siplog
notify sipd debug
notify sipd nodebug
notify sipd rotate-logs
```

- show

A lot of useful statistic can help to you have an overview on existingstatus. Below are the examples:

```
show sipd invite
show sipd endpoint-ip
show sipd error
show mbcdd error
show support [available only in 4.1.1]
```

- tail-logfile-close, tail-logfile-open

If necessary, log file can be viewed from CLI. However, pay attention that tail command will introduce host processing overhead. Now it is not perfect enough as when there are a lot of traffic through ESBC, the log file will be displayed on screen continuously, disturbing your normal operation.

-Packet capture/start stop:

To start/stop the packet capture

```
IDC-4600-1# packet-trace local M00:22
```

```
File: /opt/traces/M00_22_00001_20180620170808.pcap
```

```
Packets: 131 Packets dropped: 0
```

-Show packet trace:Shows the captured trace

Off-line tools / Command Line Tools

We can debug call failures quickly using the monitor and trace option in WebGUI. Of ESBC. (<http://ESBC> Mgmt access ip)

The options present in the Monitor and Trace are

- Sessions
- Registrations
- Subscriptions
- Notable Events

Sessions

Sessions tab gives us an idea of ongoing calls. So when there is a call failure, the first point of debugging will be to find out from Sessions tab the error code which SBC/remote party is sending. There is a ladder diagram option which gives a clear picture of endpoints and SBC.

The screenshot shows the Oracle SIP Management console interface. The 'Monitor and Trace' tab is active, displaying a 'SIP Session Summary' table. The table lists various sessions with columns for Start Time, State, Call ID, Request URI, From URI, To URI, Ingress Realm, and Egress Realm. Most sessions are marked as 'UNKNOWN'.

Start Time	State	Call ID	Request URI	From URI	To URI	Ingress Realm	Egress Realm
2018-07-20 03:33:26.838	UNKNOWN	905290452-205325333-...	slp:200601197259227752...	<slp:2005011155212214...	<slp:2005011972592277...	slp-trunk	
2018-07-20 03:31:17.934	UNKNOWN	158455454-1507777128...	slp:200601197259227752...	<slp:2005011155212214...	<slp:2005011972592277...	slp-trunk	
2018-07-20 03:28:56.679	UNKNOWN	189692231-1015328573-...	slp:200401197259227752...	<slp:2004011155212214...	<slp:2004011972592277...	slp-trunk	
2018-07-20 03:26:45.482	UNKNOWN	199160299-903714955-...	slp:200301197259227752...	<slp:2003011155212214...	<slp:2003011972592277...	slp-trunk	
2018-07-20 03:24:35.881	UNKNOWN	1397423683-1340173256...	slp:200201197259227752...	<slp:2004011972592277524@155.212.214.171...	<slp:2002011972592277...	slp-trunk	
2018-07-20 03:22:27.018	UNKNOWN	674396433-90095452-19...	slp:200101197259227752...	<slp:2001011155212214...	<slp:2001011972592277...	slp-trunk	
2018-07-20 03:20:13.932	UNKNOWN	817707560-610732351-1...	slp:100001011972592277...	<slp:1000010111552122...	<slp:1000010119725922...	slp-trunk	
2018-07-20 03:18:03.745	UNKNOWN	386366578-2003466058-...	slp:100010119725922775...	<slp:1000101115521221...	<slp:1000101197259227...	slp-trunk	
2018-07-20 03:15:54.083	UNKNOWN	129062062-1411029407...	slp:100101197259227752...	<slp:1001011155212214...	<slp:1001011972592277...	slp-trunk	
2018-07-20 03:13:43.602	UNKNOWN	1986749609-460768225-...	slp:101011972592277524...	<slp:101011552122141...	<slp:1010119725922775...	slp-trunk	
2018-07-20 03:11:33.983	UNKNOWN	114553814-480307698-8...	slp:100001197259227752...	<slp:1000011155212214...	<slp:1000011972592277...	slp-trunk	
2018-07-20 03:09:40.467	UNKNOWN	175734183829583c65a-...	slp:901146423112945@1...	1001<slp:1001@155.212...	901146423112945<slp:9...	slp-trunk	
2018-07-20 03:09:24.380	UNKNOWN	4188701818-376071811-1...	slp:986995699501187259...	<slp:9869956995011154...	<slp:9869956995011973...	slp-trunk	

The screenshot shows the 'Ladder Diagram for Session - 31178' in the Oracle SIP Management console. The diagram illustrates the sequence of SIP messages between endpoints and the SBC. A red error message is displayed: 'CALL FAILURE! STATUS=403, REASON=Forbidden'. The status is noted as 'Status: 403 (1)'.

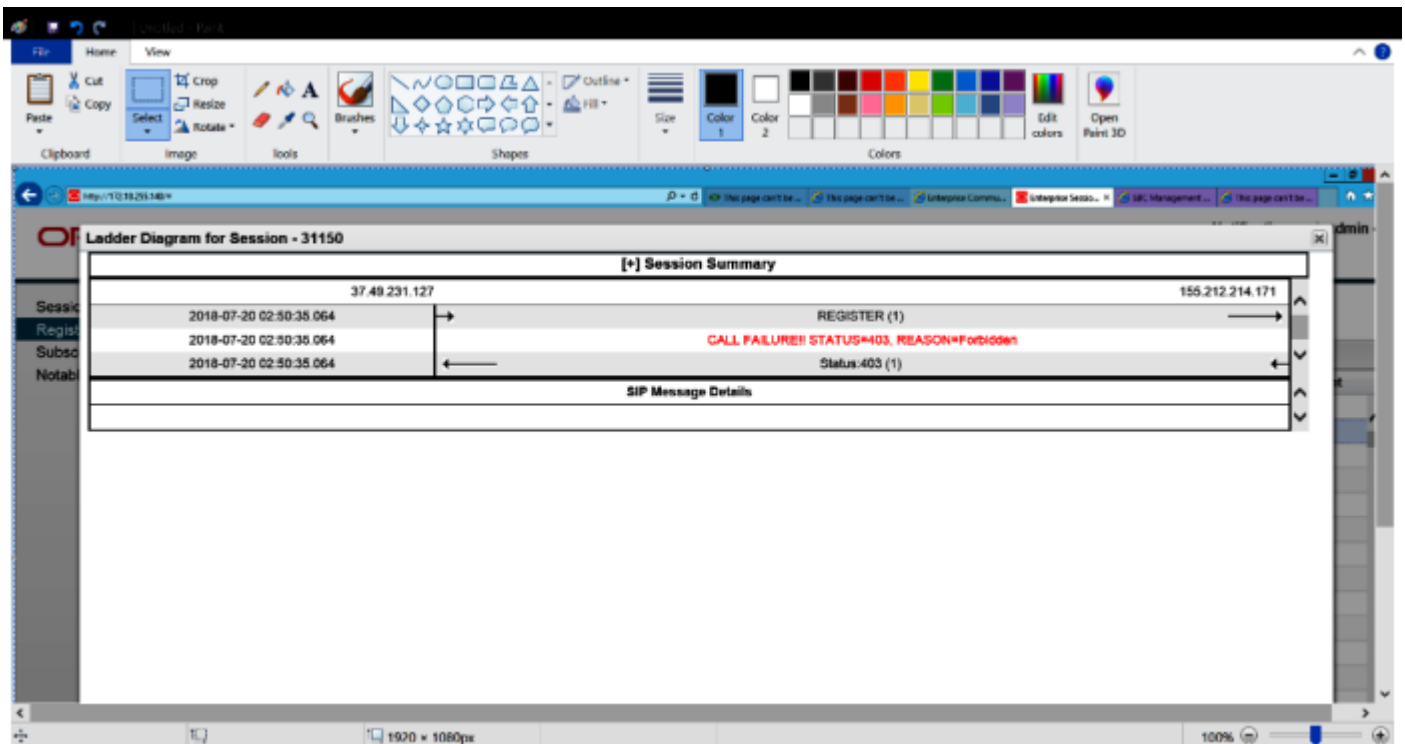
Time	Direction	Message	Status
2018-07-20 03:42:59.116	→	INVITE (1)	Status: 100 (1)
2018-07-20 03:42:59.117	←	CALL FAILURE! STATUS=403, REASON=Forbidden	Status: 403 (1)

Registrations

Registrations tab gives the details about the users registered to ESBC. If there are failures explains why the registration has failed

The screenshot shows the Oracle SIP Registration Summary page. The table contains the following data:

Start Time	Call ID	From URI	To URI	Local Expires	Remote Expires	Ingress Realm
2018-07-20 02:50:35.074	857297070	"testaccount" <sip:testacc...	"testaccount" <sip:testacc...	0	0	sip-trunk
2018-07-20 02:50:35.064	3758680688	"testaccount" <sip:testacc...	"testaccount" <sip:testacc...	0	0	sip-trunk
2018-07-20 02:50:35.053	2418665419	"testaccount" <sip:testacc...	"testaccount" <sip:testacc...	0	0	sip-trunk
2018-07-20 02:50:35.036	2436079688	"testaccount" <sip:testacc...	"testaccount" <sip:testacc...	0	0	sip-trunk
2018-07-20 02:50:35.016	502355633	"testaccount" <sip:testacc...	"testaccount" <sip:testacc...	0	0	sip-trunk
2018-07-20 02:50:35.003	3323987012	"testaccount" <sip:testacc...	"testaccount" <sip:testacc...	0	0	sip-trunk
2018-07-20 02:50:34.925	1908482469	"testaccount" <sip:testacc...	"testaccount" <sip:testacc...	0	0	sip-trunk
2018-07-20 02:50:34.900	1718724098	"testaccount" <sip:testacc...	"testaccount" <sip:testacc...	0	0	sip-trunk
2018-07-20 02:50:34.875	2612521559	"testaccount" <sip:testacc...	"testaccount" <sip:testacc...	0	0	sip-trunk
2018-07-20 02:50:34.867	1870906784	"testaccount" <sip:testacc...	"testaccount" <sip:testacc...	0	0	sip-trunk
2018-07-20 02:50:34.861	1484477351	"testaccount" <sip:testacc...	"testaccount" <sip:testacc...	0	0	sip-trunk
2018-07-20 02:50:34.852	4153920998	"testaccount" <sip:testacc...	"testaccount" <sip:testacc...	0	0	sip-trunk
2018-07-20 02:50:34.844	4149881678	"testaccount" <sip:testacc...	"testaccount" <sip:testacc...	0	0	sip-trunk



Subscriptions

Subscriptions tab gives us an idea of the subscriptions by an user like voicemail, dialog ,features (conference,message-summary, presence etc.)The details given are same as the registration tab.

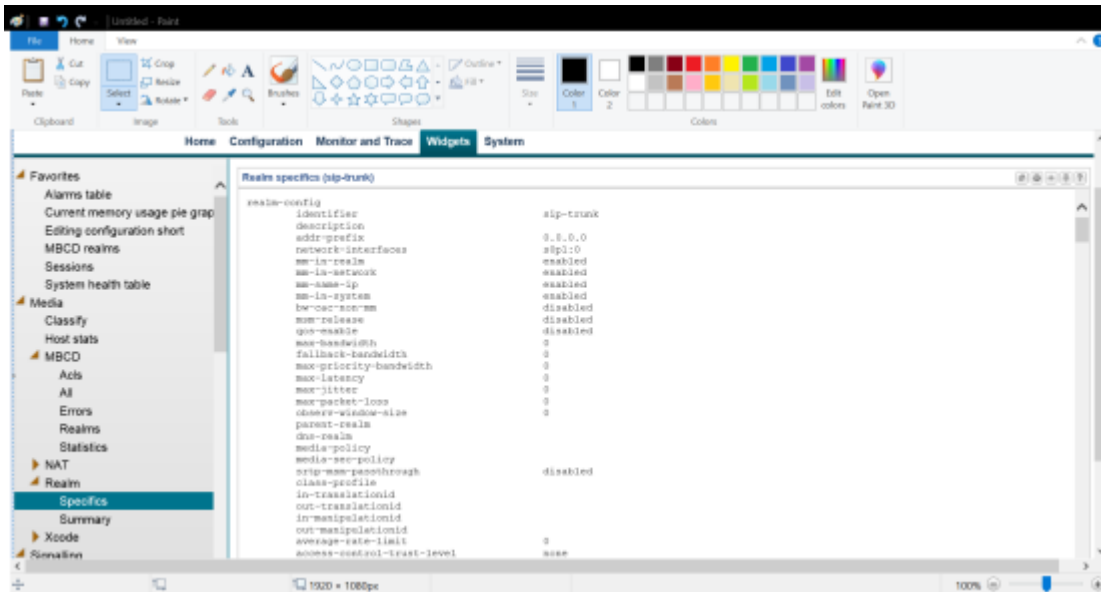
Notable Events

Indicates if a notable event has occurred on the call session. Sessions locally rejected at the E-SBC for any reason, for example, Session Agent (SA) unavailable, no route found, SIP signaling error, and so on. Session dialogue, capture media information, and termination signaling. Any event flagged as a local rejection interesting event.

Widgets

The Widgets section in the GUI displays statistics, discussed in the above chapters(same as show commands on the cli interface).

For eg :From Widgets we are gathering information about particular realm config



The same can be viewed from cli also

```

FE-0331-24 show realm-
FE-0331-24 show realm-
FE-0331-24 show realm-
FE-0331-24 show realm-
FE-0331-24 show csa realm-
realm-config realm-group

FE-0331-24 show csa realm-config
realm-config
  identifier                ATY-Trunk
  description
  addr-prefix                0.0.0.0
  network-interface         MPTC
  msi-is-rx                 enabled
  msi-is-rx-network         enabled
  msi-is-ip                  enabled
  msi-is-rytvm              enabled
  hsr-is-rx-ctrl-ssm        disabled
  msm-release                disabled
  qos-enable                 disabled
  max-bandwidth              0
  fallback-bandwidth        0
  max-priority-bandwidth    0
  max-latency                 0
  max-jitter                  0
  max-packet-loss            0
  cbserv-window-size         0
  parent-realm              MPTC
  service                    MPTC
  media-policy
  media-sec-policy           disabled
  sfp-ssm-passthrough        disabled
  class-profile
  in-translationid
  out-translationid
  in-manipulationid
  out-manipulationid
  average-rate-limit         0
  access-control-trust-level none
  invalid-signal-threshold  0
  maximum-signal-threshold  0

```

Log viewers

- Get Logs
- Xlog (to stream logs to)
- gVim (for viewing logs)
- UltraEdit (license required)

Third party software

- Ethereal/Wireshark
- FreeRADIUS (CDR collection)
- WS_FTP_LE (log files download)

Job aides

- Oracle BCP's
- ACLI Reference guide
- ACLI Configuration tree reference sheet

Trouble-shooting Procedures with sample problem cases

Trouble-shooting is a process of experience collection. Standardized procedures can give you a guideline about how you should go. However, it all depends on your creativeness and carefulness. Therefore, studying someone's case example can help to provide you reference on your own trouble-shooting.

This chapter will present three sample problem cases in below style:

Description of the symptom, or what question or complaint from customers

Steps of troubleshooting to be taken

Summary of analysis, e.g. key finding

DNS Not Reachable

Description of the problem:

Endpoint is unable to register. ESBC sends "480" for all Registration requests from this endpoint.

Troubleshooting steps performed:

Followed the troubleshooting methodology step by step and gathered the following information:

1. **show version** confirmed that latest 8.0.0 version is used.
2. **show features** showed the required licenses were available on the ESBC.
3. Checked the configuration by doing show running config. The configuration was good without any errors. **verify-config** did not show any errors with the configuration.
4. Looked at the sip stats by running show sipd all and the Registration stats confirmed that the 480 was being generated by the ESBC. The response was present in Server side stats (Recent) and not the Client side which confirmed that ESBC was not just proxying the response received from the Server but ESBC generated the response.

REGISTER (13:54:27-157)

Message/Event	Recent	Server -		Recent	Client -	
		Total	PerMax		Total	PerMax
REGISTER Requests	2	631	15	0	313	7
Retransmissions	0	0	0	0	1	1
100 Trying	0	313	7	0	313	7
200 OK	0	628	15	0	313	7
480 Unavailable	2	3	2	0	0	0

- 1) Confirmed that there are no errors in the incoming requests by doing **show sipd errors**.
- 2) show sipd agents displayed that the Session Agent with FQDN hostnameregistrar.com was marked out of service:
- 3) training2c# show sipd agents

13:57:15-250

Session Agent	----- Inbound -----			--- Outbound -----			----- Latency -- Max			
	Active	Rate	ConEx	Active	Rate	ConEx	Avg	Max	Burst	
registrar.com	0	0	0.1	0	0	0.0	0	0.001	0.001	10

- 4) From the **show run** output it shows that this agent is a next-hop in the Local-policy and a DNS server has been configured under the network-interface for this realm.
- 5) **show dns** indicates 2 Queries were made to the server which were not successful.

training2c# show dns
14:01:23-127

DNS Intf Name	---Queries---		--Successful--		---NotFound---		---TimedOut---	
	Current	Total	Current	Total	Current	Total	Current	Total
f10	2	6	0	0	0	0	1	6

- 6) The next step was to determine what the logs indicated. Turned up the sip logs to debug using following commands and collected all the logs after archiving the logs:

```
notify siplog debug
log-level sipd debug
```

- 7) sipddns.log indicated that ESBC was sending queries to the DNS server at 1.1.1.1 (a bogus IP-Address was chosen to replicate this problem) but there was no response from the server.

```
Jan 31 15:05:54.781 On 172.16.0.10:1166 sent to 1.1.1.1:53
DNS Query 47 flags=100 q=1 ans=0 auth=0 add=0 net-ttl=3600
Q:A registrar.com
```

```
0000: 00 2f 01 00 00 01 00 00 00 00 00 00 09 72 65 67  ./..... reg
0010: 69 73 74 72 61 72 03 63 6f 6d 00 00 01 00 01  istrar.com....
```



```
Jan 31 15:05:55.782 On 172.16.0.10:1166 sent to 1.1.1.1:53
DNS Query 47 flags=100 q=1 ans=0 auth=0 add=0 net-ttl=3600
Q:A registrar.com
```

```
0000: 00 2f 01 00 00 01 00 00 00 00 00 00 09 72 65 67  ./..... reg
0010: 69 73 74 72 61 72 03 63 6f 6d 00 00 01 00 01  registrar.com
```

8) Concluded that ESBC sent 480's to the Register requests because the DNS server was not responding to the DNS queries from the ESBC.

Summary of Analysis and key findings:

The problem was first identified as a ESBC issue by running the relevant show commands and then proceeded further for collecting logs. **show dns** clearly indicated that there was no response from the server for the queries. At this point it can be easily concluded that this is a non-Oracle issue. For DNS related issues it is not required to turn up all sip logs to debug (To avoid high CPU utilization). If the problem has been identified as a DNS issue turn the DNS logs to debugging **show sipd debug DNS** command to make sure the DNS queries are being answered

One-way Audio - Steering Pool IP overlapped

Description of the problem:

Customer reported occasionally it will have One-way Audio problem. An access endpoint calls to PSTN gateway through ESBC. Endpoint cannot hear any voice occasionally for most of the call, but PSTN side can hear what endpoint said. However, occasionally endpoint and PSTN can communicate properly with two-way audio

Troubleshooting steps performed:

Normally One-way Audio problem is related to mbcd. Improper configuration of steeringpool ip address can also induce this problem. Therefore, steering-pool configuration should be checked first.

For example, if someone configured the steering-pool ip as the same as network-interface default gateway ip, verify-config can pass it properly, but it will still introduce one-way audio because all RTP will be destined on default gateway ip and will not reach ESBC network-interface. However, this time the problem is that occasionally call can work properly but most of the time it is one-way audio. If only simply configured the steering-pool ip the same as default gateway ip, all calls will have the same one-way audio problem.

Check mbcd error statistic by command **show mbcd error**.

If verified the steering-pool ip is correct, we can check mbcd error statistic by command show mbcd error. See whether there would be any internal mbcd error.

Turn on minimum internal debug level

Normally sniffer capture is the most effective way to check the real status of SIP and RTP communication on line. However, if it is not available for taking sniffer capture, and sip layer communication is fine, we can only turn on sipmsg.log by command: notify sipd siplog

For mbcd, if there is no special mbcd error, it will then be important to check whether RTP packet has really been arrived ESBC network-interface. Therefore, following command can be enough:

log-level mbcd debug (flow npsft)

Therefore, it should only introduce minimum overhead to ESBC host processor. First of all, from sipmsg.log, we can filter out the problem call by either phone number or call-id. Pay attention to the portion about mbcd communication in sipmsg.log of the filtered call. Try to find some key information, such as timestamp, **Context =**, **idest** port number and esource port number and then search the log.mbcd to see which portion has those value. If there are multiple log.mbcd files, we can make use of "Grep" or "WinGrep" tool to search those text files. The key word latch on can help us to quickly locate all latched RTP flow. With information from sipmsg.log, we can easily filter out the target latched flow related to our target call. The key target is to check with [NPSOFT]. If we can see something like below, we can then assure that both way RTP has arrived ESBC network interface

```

Jan 30 10:41:36.091 [FLOW] Ingress CX=65536.1E ID=65536-A <1way=17> UDP/2n noQoS med=audio
<lg>
Jan 30 10:41:36.091 [FLOW] I=<access1=s0/p0:0>0.0.0.0,11.0.0.11:21000
Jan 30 10:41:36.091 [FLOW] O=<backbone=s1/p0:0>172.16.0.10:10000,172.16.0.10:10002
Jan 30 10:41:36.091 [FLOW] 30 10:41:33.794 last=30 10:41:36.051 next=65538 other=65537; 4
ports:
Jan 30 10:41:36.091 [FLOW] 11.0.0.11:21000+21001
Jan 30 10:41:36.091 [FLOW] 172.16.0.10:10000+10001
Jan 30 10:41:36.091 [MEDIA] <2833>Check Interworking: 2833 not enabled ix=1E ID=65536
Jan 30 10:41:36.091 [NPSOFT] NAT_flow_update:
Jan 30 10:41:36.091 [NPSOFT] access_type : MEDIA

```

```

Jan 30 10:41:36.091 [NPSOFT] DA_flow_key : 011.000.000.011 DA_prefix : 32
Jan 30 10:41:36.091 [NPSOFT] SP_flow_key : 49152 SP_prefix : 16
Jan 30 10:41:36.091 [NPSOFT] DP_flow_key : 21000 DP_prefix : 16

```

```

Jan 30 10:41:36.091 [NPSOFT] VLAN_flow_key : 0
Jan 30 10:41:36.091 [NPSOFT] Protocol_flow_key : 17
Jan 30 10:41:36.091 [NPSOFT] Ingress_flow_key : 0
Jan 30 10:41:36.091 [NPSOFT]XSA_data_entry: 011.000.000.012
Jan 30 10:41:36.091 [NPSOFT]XDA_data_entry: 011.000.000.102
Jan 30 10:41:36.091 [NPSOFT]XSP_data_entry: 22000
Jan 30 10:41:36.091 [NPSOFT]XDP_data_entry: 49152
Jan 30 10:41:36.092 [NPSOFT]Egress_data_entry : 0
Jan 30 10:41:36.092 [NPSOFT]flow_action: 0
Jan 30 10:41:36.092 [NPSOFT]optional_data : 0
Jan 30 10:41:36.092 [NPSOFT]VLAN_data_entry : 0
Jan 30 10:41:36.092 [NPSOFT]host_table_index : 17
Jan 30 10:41:36.092 [NPSOFT]Switch ID : 0x00000002
Jan 30 10:41:36.092 [NPSOFT]sustained-rate : 0
Jan 30 10:41:36.092 [NPSOFT]peak-rate : 0
Jan 30 10:41:36.092 [NPSOFT]max-burst-size : 0
Jan 30 10:41:36.092 [NPSOFT]FPGA handle : 0xffffffff
Jan 30 10:41:36.092 [NPSOFT]init_flow_guard : 300
Jan 30 10:41:36.092 [NPSOFT]inact_flow_guard : 300
Jan 30 10:41:36.092 [NPSOFT]max_flow_guard: 86400
Jan 30 10:41:36.092 [NPSOFT]assoc_FPGA_handle : 0xffffffff

Jan 30 10:41:41.689 [FLOW]NAT 15 set latch on 65539@11.0.0.12:22000 to 11.0.0.102:49152
Jan 30 10:41:41.689 [FLOW]Ingress CX=65538.1W ID=65539-A <1way=15> UDP/2n noQoS med=audio
<ld>
Jan 30 10:41:41.689 [FLOW]I=<access2=s0/p0:0>11.0.0.102:49152,11.0.0.12:22000
Jan 30 10:41:41.689 [FLOW]O=<backbone=s1/p0:0>172.16.0.10:10002,172.16.0.10:10000
Jan 30 10:41:41.689 [FLOW]30 10:41:33.832 last=30 10:41:36.069 other=65538
Jan 30 10:41:41.689 [MEDIA]<2833>Check Interworking: 2833 not enabled ix=1W ID=65539
Jan 30 10:41:41.689 [NPSOFT]NAT_flow_update:
Jan 30 10:41:41.689 [NPSOFT]access_type : MEDIA
Jan 30 10:41:41.689 [NPSOFT]SA_flow_key : 011.000.000.102 SA_prefix: 32
Jan 30 10:41:41.689 [NPSOFT]DA_flow_key : 011.000.000.012 DA_prefix: 32
Jan 30 10:41:41.689 [NPSOFT]SP_flow_key : 49152SP_prefix : 16
Jan 30 10:41:41.689 [NPSOFT]DP_flow_key : 22000DP_prefix : 16
Jan 30 10:41:41.689 [NPSOFT]VLAN_flow_key : 0
Jan 30 10:41:41.689 [NPSOFT]Protocol_flow_key : 17
Jan 30 10:41:41.689 [NPSOFT]Ingress_flow_key : 0
Jan 30 10:41:41.689 [NPSOFT]XSA_data_entry: 172.016.000.010
Jan 30 10:41:41.689 [NPSOFT]XDA_data_entry: 172.016.000.010
Jan 30 10:41:41.689 [NPSOFT]XSP_data_entry: 10002
Jan 30 10:41:41.689 [NPSOFT]XDP_data_entry: 10000
Jan 30 10:41:41.689 [NPSOFT]Egress_data_entry : 1
Jan 30 10:41:41.689 [NPSOFT]flow_action: 0
Jan 30 10:41:41.689 [NPSOFT]optional_data : 0
Jan 30 10:41:41.690 [NPSOFT]VLAN_data_entry : 0
Jan 30 10:41:41.690 [NPSOFT]host_table_index : 15
Jan 30 10:41:41.690 [NPSOFT]Switch ID : 0x00000004
Jan 30 10:41:41.690 [NPSOFT]sustained-rate : 0
Jan 30 10:41:41.690 [NPSOFT]peak-rate : 0
Jan 30 10:41:41.690 [NPSOFT]max-burst-size : 0
Jan 30 10:41:41.690 [NPSOFT]FPGA handle : 0xffffffff
Jan 30 10:41:41.690 [NPSOFT]init_flow_guard : 300
Jan 30 10:41:41.690 [NPSOFT]inact_flow_guard : 300
Jan 30 10:41:41.690 [NPSOFT]max_flow_guard: 86400
Jan 30 10:41:41.690 [NPSOFT]assoc_FPGA_handle : 0xffffffff

```

```

costello# show mbcid nat
08:43:14-192
NAT Entries      ---- Lifetime ----
      Recent  Total PerMax
Adds           0    19   14
Deletes        0    16   16
Updates        0     7    7
Non-Starts     0     1    1
Stops          0     0    0
Timeouts       0     0    0

```

However, for the reported problem, you should only be able to find one latched flow for occasional one-way audio call. If you can still find two latched flow information, then it can be classified as either endpoint or PSTN gateway problem, not related to

ESBC because ESBC has done his job to receive and forward RTP. Maybe the ip provided in SDP from either endpoint or PSTN gateway is incorrect. Or maybe that ip was being duplicated in their networks. When ip address is duplicated,

it would be possible that RTP will then be hijacked by the duplicated ip device.

Check ARP table.

By analyzing all available information up to this point, we may suspect whether there is any ip address duplicated with ESBC. It would be possible to be happened. For all ip address configured in **sip-interface or steering-pool**, they are recognized as virtual ip address. Once they are duplicated, there is no alarm or warning in advance from existing release. Even if some late coming device mis-configured their ip the same as the ESBC steering-pool ip, both ESBC and that device will not be noticed, as there is no ARP reply when that device was plugged into the same subnet network as ESBC.

By checking show arp table, we can pay attention whether there is any ip address in ARP table, where the ip supposed should be belonging to ESBC but actually the MAC address is not belonging to ESBC. For ESBC, all MAC address should be started with a prefix **00:08:25**. If any steering pool ip or sip-interface was found in ARP table, but the MAC address was not started with **00:08:25**, it is obvious that ESBC ip address was duplicated by some device in the same network.

Summary of analysis

For the reported occasional one-way media problem, it can be easily committed by any duplicated ip address device in the same network. During call communication, the MAC address of the duplicated device will be recorded in either router or layer-2 switch. Therefore when ingress RTP supposed should be delivered to ESBC steering-pool ip and MAC address, however, because of such device, RTP will then be delivered to it by those router or layer-2 switch. But occasionally when this device was not in network, everything will be working fine. As existing release of ESBC software does not have any mechanism to prevent it or check it. Therefore, I suggest configuring the steering-pool ip in HIP-ip-list, even it is not necessary in normal operation. When an ip address was listed in HIP-ip-list, whenever there is any duplicated ip address in the network, warning message will then be noticed from command line. Therefore, one of the one-way media problem cause can then be avoided.

DOS protection parameter misconfiguration

Description of the symptoms

The endpoint not registering/able to make calls. Simulated a customer scenario where DOS settings were too stringent and hence the endpoint got demoted/denied and hence was not able to register or make calls.

Steps followed for troubleshooting the problem reported:

Issued following commands to narrow down the problem to be related to DOS.

1. Check which software version you are running

```

training2c# sho ver
Acme Packet 6300 ECZ8.0.0 Patch 1 (Build 72)
Oracle Linux branches-7/el7-u4 {2017-08-24T07:00:00+0000}
Build Date=02/23/18
01/18/07

```

2. Check if the ESBC has all the required features loaded

```

training2c# sho features
Total session capacity: 32000
Enabled features: SIP, MGCP, H323, IWF, QOS, ACP, Routing, Load Balancing, Accounting, High Availability, PAC

```

3. Scanning through the output of "show running" for sanity check reveals the DOS settings.

```

Look at the DOS settings in realm access2
training2c# sho run
    average-rate-limit          2700
    access-control-trust-level  low
    invalid-signal-threshold    20
    maximum-signal-threshold    5
    untrusted-signal-threshold  20
    deny-period                 30

```

4. Check if the end-point is registered

```

training2c# sho sipd endpoint-ip 7007
Entry not found

```

5. Search how the promotion and demotion occurred. In the example below the endpoint got promoted on successful registration and later was demoted.

```

training2c# sho sip acl

```

```

14:10:44-135
SIP ACL Status
Active High Total Lifetime - High
Total Entries 1 1 1 1 1 1
Trusted 0 1 1 1 1 1
Blocked 0 0 0 0 0 0

ACL Operations
Recent Total PerMax
ACL Requests 2 2 1
Bad Messages 0 0 0
Demotions 1 1 1

```

6. Search if the end-point has been denied.

```

training2c# sho acl denied
deny entries:
intf:vlan source-ip/mask:port/mask dest-ip/mask:port/mask prot type index
0/0:0 11.0.0.107:5060 11.0.0.12:5060 UDP dynamic 18
Total number of deny entries = 1
Denied Entries not allocated due to ACL constraints: 0

```

7. After it has been determined that the end-point was demoted/denied we need to find which threshold was exceeded which caused its demotion and its move to the deny list. In order to do that we need to enable logs.
8. As in a production network it is not always possible to enable debug level logs for SIP or MBCD as the amount of traffic going through the ESBC could be overwhelming. So, in order to get debug level logs related to DOS promotion and demotion without taxing the CPU turn the MINOR log category of the logs.

```

training2c# log-level sipd debug minor
Completed

```

9. The enabling of debug mode for MINOR log category can be verified using the command:

```
training2c# sho loglevel sipd verbose
```

```
Log Levels for process sipd:
```

```
GENERAL=NO TI CE
EMERGENCY=NO TI CE
CRITICAL=NOTICE
MAJOR=NOTICE
MINOR=DEBUG
WARNING=NO TI CE
PROCESS=NOTICE
IPC=NO TI CE
SERVICE=NOTICE
EVENT=NO TI CE
MESSAGE=NOTICE
TEST=NOTICE
TRIP=NOTICE
SIP=NOTICE
MBCP=NOTICE
FLOW=NOTICE
MEDIA=NOTICE
SESSION=NOTICE
TRANS=NOTICE
TIMER=NOTICE
ALG=NOTICE
MGCP=NOTICE
NPST=NOTICE
ARP=NOTICE
SNMP=NOTICE
ANDD=NOTICE
XNTP=NOTICE
REDUNDANCY=NOTICE
SI PNAT=NOTICE
H323=NOTICE
ERROR=NOTICE
CONFIG=NOTICE
DNS=NOTICE
H248=NOTICE
BAND=NOTICE
AL I=NO TI CE
SS8GI=NOTICE
CO PS=NOTICE
ATCP=NOTICE
ATCPAPP=NOTICE
CLF=NOTICE
LRT=NOTICE
```

```
training2c#
```

10. Gather log.sipd and search for keyword "exceeded message threshold of". And then compare that value with parameter values in realm configuration. In the example below the value 5 refers to max-signal-threshold.

```
Feb 1 14:10:37.843 [MINOR] SigAddr[access2:11.0.0.107:5060=low:PERMIT] ttl=152
exp=115 exceeded message threshold of 5
Feb 1 14:10:37.843 [MINOR] recent(38): msgs=6 errs=0
Feb 1 14:10:37.844 [MINOR] lifetime: msgs=6 errs=0
```

11. In order to get more details enabling FLOW log category to debug is an option. But as FLOW log category takes care of all the media flow establishment so, enabling FLOW to debug could be taxing in a high traffic production environment.

In order to enable FLOW log category to debug, uses the command:

```
training2c# log-level sipd debug flow
Completed
```

Log.sipd gives the following information and key words to search are "exceeded message threshold of", "White-List", "Grey-List", "Black-List".

```

Jan 31 17:23:47.948 [FLOW] ACL(low) Promote SigAddr[access2:11.0.0.102:5060=low:NONE] ttl=152
to White-List
Jan 31 17:23:47.948 [FLOW] SipSigAccessList::incAclCount(2) SIP
Jan 31 17:23:47.948 [FLOW] New ACL Request [1:PERMIT]
access2:11.0.0.102:5060#[0:0]11.0.0.12:5060/UDP/SIP(2700,0,0) SP=6000 exp=152
Jan 31 17:23:47.949 [FLOW] Send ACL Request:
Jan 31 17:23:47.949 [FLOW] [1:PERMIT]
access2:11.0.0.102:5060#[0:0]11.0.0.12:5060/UDP/SIP(2700,0,0) SP=6000 exp=152
Jan 31 17:23:47.949 [FLOW] SipSigAccessList::incAclCount(0) SIP
Jan 31 17:23:47.949 [FLOW] SigAddr[access2:11.0.0.102:5060=low:NONE] ttl=152 exp=152 now
PERMIT send to peer

Jan 31 17:24:09.608 [FLOW] ACL(low) Demote SigAddr[access2:11.0.0.102:5060=low:PERMIT]
ttl=152 exp=130 to Grey-List
Jan 31 17:24:09.608 [FLOW] SipSigAccessList::incAclCount(3) SIP
Jan 31 17:24:09.608 [FLOW] New ACL Request [3:REMOVE]
access2:11.0.0.102:5060#[0:0]11.0.0.12:5060/UDP/SIP SP=6000 exp=0
Jan 31 17:24:09.608 [FLOW] Send ACL Request:
Jan 31 17:24:09.608 [FLOW] [3:REMOVE] access2:11.0.0.102:5060#[0:0]11.0.0.12:5060/UDP/SIP
SP=6000 exp=0
Jan 31 17:24:09.609 [FLOW] SipSigAccessList::incAclCount(0) SIP
Jan 31 17:24:09.609 [FLOW] SipSigAccessList::decAclMeter(1) SIP
Jan 31 17:24:09.609 [FLOW] SigAddr[access2:11.0.0.102:5060=low:PERMIT] ttl=152 guard=60 now
NONE send to peer

Jan 31 17:24:36.324 [MINOR] SigAddr[access2:11.0.0.102:5060=low:NONE] ttl=152 guard=33 exceeded message
threshold of 20
Jan 31 17:24:36.324 [MINOR] recent(30): msgs=21 errs=0
Jan 31 17:24:36.325 [MINOR] lifetime: msgs=27 errs=0
Jan 31 17:24:36.325 [FLOW] ACL(low) Demote SigAddr[access2:11.0.0.102:5060=low:NONE] ttl=152
guard=33 to Black-List
Jan 31 17:24:36.325 [FLOW] SipSigAccessList::incAclCount(3) SIP
Jan 31 17:24:36.325 [FLOW] New ACL Request [2:DENY]
access2:11.0.0.102:5060#[0:0]11.0.0.12:5060/UDP/SIP SP=6000 exp=30
Jan 31 17:24:36.325 [FLOW] Send ACL Request:
Jan 31 17:24:36.325 [FLOW] [2:DENY] access2:11.0.0.102:5060#[0:0]11.0.0.12:5060/UDP/SIP
SP=6000 exp=30
Jan 31 17:24:36.326 [FLOW] SipSigAccessList::incAclCount(0) SIP
Jan 31 17:24:36.326 [FLOW] SigAddr[access2:11.0.0.102:5060=low:NONE] ttl=152 guard=33 exp=30
now DENY send to peer
Jan 31 17:24:36.326 [FLOW] SipSigAccessList::incAclMeter(2) SIP
Jan 31 17:24:36.326 [FLOW] SigAddr[access2:11.0.0.102:5060=low:DENY] ttl=152 guard=33 exp=30
set timer to 29999
Jan 31 17:24:36.326 [FLOW] SigAddr[access2:11.0.0.102:5060=low:DENY] ttl=152 guard=33 exp=30
Demoted to Black-List; send SNMP trap
Jan 31 17:24:36.327 [FLOW] SigAddr[access2:11.0.0.102:5060=low:DENY] ttl=152 guard=33 exp=30
found
Jan 31 17:24:36.327 [FLOW] ACL(low) Promote SigAddr[access2:11.0.0.102:5060=low:DENY] ttl=152
guard=33 exp=30 to Grey-List

```

Packets drop can be monitored using the following command and searching for "deny list drop"

training2c# sho media classify 0 0

```

Slot 0 Port 0 Microcode Statistics
DIX/IP 0x00: [ 0x0007d8d2.00000bd2]
SNAP/IP 0x01: [ 0x00000000.00000000]
VLAN-DIX/IP 0x02: [ 0x000001aa.00000000]
VLAN-SNAP/IP 0x03: [ 0x00000000.00000000]
DIX/ARP 0x04: [ 0x0000ff6e.000009a3]
SNAP/ARP 0x05: [ 0x00000000.00000000]
VLAN-DIX/ARP 0x06: [ 0x00000285.00000000]
VLAN-SNAP/ARP 0x07: [ 0x00000000.00000000]
DIX/ICMP 0x08: [ 0x00000b8a.00000000]
SNAP/ICMP 0x09: [ 0x00000000.00000000]
VLAN-DIX/ICMP 0x0A: [ 0x00000000.00000000]
VLAN-SNAP/ICMP 0x0B: [ 0x00000000.00000000]
PCCAM Drops 0x0C: [ 0x00005f87.00000000]
NAT/XSM Miss Drops 0x0D: [ 0x0007cb9d. 00000000]
PAUSE Frames 0x0E: [ 0x00000000.00000000]
Deny list Drops 0x0F: [ 0x00000020.00000000]
IP Fragment Drops 0x10: [ 0x0000002a.00000000]
ARP/MAC Drops 0x11: [ 0x00000000.00000000]

```

```

NAT/Latches      0x12: [ 0x00000005.00000000 ]
NAT/XSM Miss Key (lower) 0x13: [ 0x0a00030c.0a000307 ]
NAT/XSM Miss Key (upper) 0x14: [ 0x06000000.001708f1 ]
ARP Miss Key     0x15: [ 0x00000000.00000000 ]
Phy Stats       0x16: [ 0x00000003.00000000 ]
Undefined Stats Register 0x17: [ 0x00000000.00000000 ]
Undefined Stats Register 0x18: [ 0x00000000.00000000 ]
Undefined Stats Register 0x19: [ 0x00000000.00000000 ]
Undefined Stats Register 0x1a: [ 0x00000000.00000000 ]
Undefined Stats Register 0x1b: [ 0x0b000065.0b00000b ]
Undefined Stats Register 0x1c: [ 0x82000003.00450000 ]
Undefined Stats Register 0x1d: [ 0x00000000.00000000 ]
Standby lookup skip flag 0x1e: [ 0x00000000.00000000 ]
Pkt Capture config flag 0x1f: [ 0x00000000.00000000 ]

```

Summary of Analysis:

The show commands do a wonderful job in identifying the promotion/demotion and packet loss information. After that has been identified enabling MINOR log category to debug can help in determining which threshold is getting exceeded.

Recommendation to Engineering for Enhancement

In most cases it is more time consuming to find and retrieve the problem related data from the amount and variety of logs provided, than to analyze the data found.

The following suggestions should help to retrieve and minimize the amount of data needed to be investigated by "implementing" keywords and codes to search for.

Tools helping to link keyword / timestamps of related data from different logs will also help to reduce time searching for the very same keywords in over and over again different logs.

Prior to this, some enhancement requests to Engineering shall help to avoid errors caused by wrong configuration and/or not activation of parameters.

ESBC OS Enhancement Recommendation

- Consistency between logging element start lines between sipmsg.log and log.sipd
- Promote useful log entries to TRACE level to reduce the number of messages that the SE's have to review when looking at logs
- Possibility to set a trigger to start logging, driven by below scenarios:
 - Event-driven
 - Timer-driven
 - Alert-driven
 - Duration-driven
- Logging session should be closed when session ends.

Helpful Tools Enhancement Recommendation

- Tool that allows retrieving all or filtering out one single request related data from all logs. Filter criteria should be flexible enough to trigger ESBC only logging expected target.
- Tool to link and compare different log files next to each other.
 - For example: For one specific SIP call, it collects the sipmsg.log, the log.sipd and log.mbcd.
- Open the files in three independent columns next to each other. In a column when I select a "key", request or response message, the related data are also listed in the other one or two columns. In addition, the use of colors will help to visualize the keyword in each log file.
- Tool can be integrated with EMS for centralized management. Therefore log files can be redirected to or collected by EMS server. Operator can centrally manage the ESBC and analyze its log.
- Viewing Log files tool should also be available from EMS.
- Logs should be possible to be cleared by online command or EMS GUI command button. Only rotate-logs is not enough.
- EMS Virtualization tool for configuration EMS Virtualization tool to retrieve cached subscribers list.

Author's Address

- i. Gayathri Balakrishnan(gayathri.balakrishnan@oracle.com)
- ii. Bhaskar Reddy Gaddam(bhaskar.gaddam@oracle.com)
- iii. Mark Ansley(mark.ansley@oracle.com)
- iv. Sana Guntupalli (sanaa.guntupalli@oracle.com)
- v. Glen Mchugh(glen.mchugh@oracle.com)

Disclaimer

The content in this document is for informational purposes only and is subject to change by Oracle without notice. While reasonable efforts have been made in the preparation of this publication to assure its accuracy, Oracle assumes no liability resulting from technical or editorial errors or omissions, or for any damages resulting from the use of this information. Unless specifically included in a written agreement with Oracle, Oracle has no obligation to develop or deliver any future release or upgrade or any feature, enhancement or function.

Full Copyright Statement

Copyright © Oracle (2018). All Rights Reserved. Oracle, Session-Aware Networking, and related marks are trademarks of Oracle. All other brand names are trademarks or registered trademarks of their respective companies.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implantation may be prepared, copied, published and distributed, in whole or in part, given the restrictions identified in section 2 of this document, provided that the above copyright notice, disclaimer, and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to Oracle or other referenced organizations. The limited permissions granted above are perpetual and will not be revoked by Oracle or its successors or assigns. This document and the information contained herein is provided on an "AS IS" basis and ORACLE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

