Troubleshoot Silent Reloads on Catalyst 9300/3850/3650s

Contents

Introduction

This document describes how to troubleshoot commands/registers for issues specifically related to stacking port/cable problems and silent reloads.

Troubleshoot/Show Commands

Collect and analyze useful registers (for each ASIC and Core). There are three main ones:

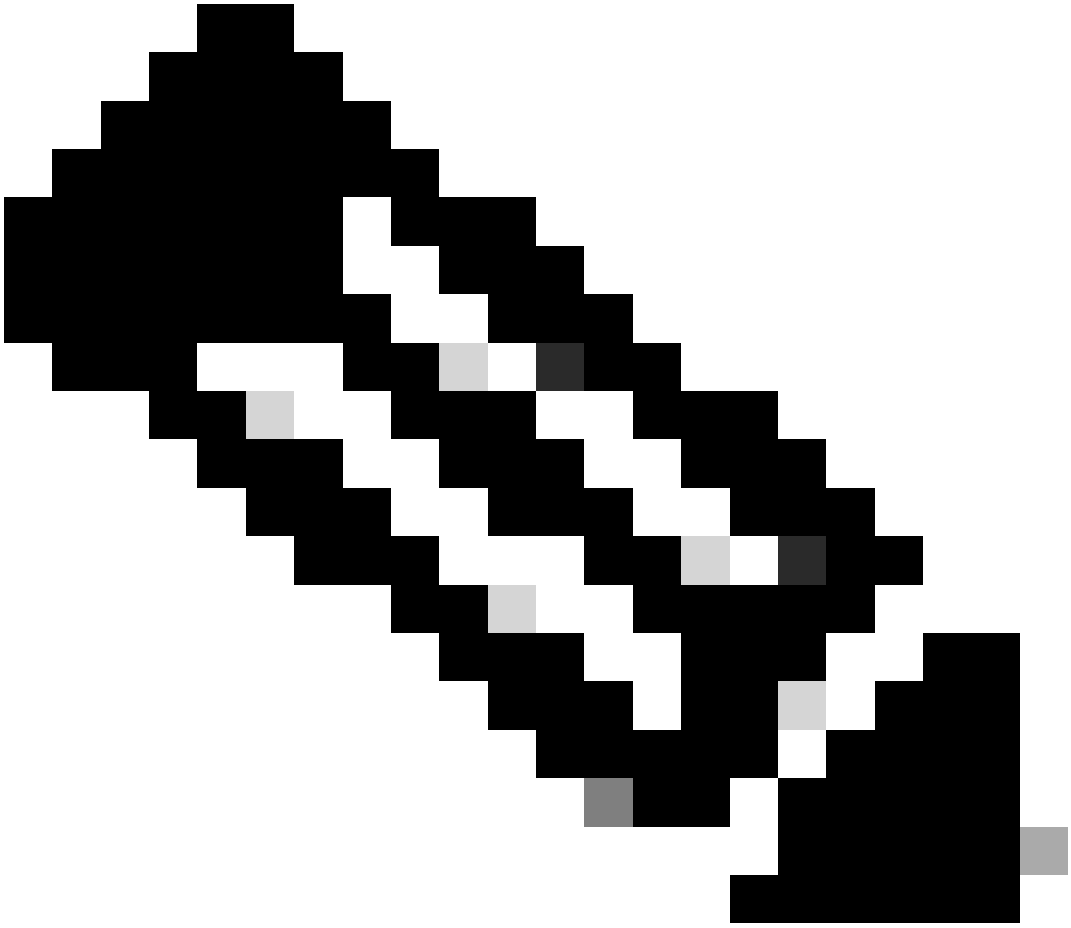
- SifInfo
- SifRacStatus
- SifRacControl

```
show platform hardware fed switch active fwd-asic register read register-name <name>
```

SifInfo

The first bit tells us if asic is available or not. It is set as 0x1. If it is set as 0x0, then there are forward issues. Error counters or box are not able to recover packets properly.

```
Switch#sh platform hardware fed switch active fwd-asic register read register-name SifInfo

For asic 0 core 0

Module 0 - SifInfo[0][0]


available                : 0x1 <---- should be 0x1 indicating balloting is completed
headerVersion            : 0x0
nodeAllLinksAvailable    : 0x1
nodeId                   : 0x4 <---- asic ID (unique across all asics in the stack)
numNodes                 : 0x8 <---- how many asics are there in whole stack
serdesSpeed              : 0x2
sifAllLinksAvailable     : 0x1
sifSupStall              : 0x0
wrappedAtRac0            : 0x0 <---- If a single stack port is down, 3 of 6 should wrap w/ value
wrappedAtRac1            : 0x0       of 0x1. Will appears in groups for 0, 2 and 4 or 1, 3 and 5.
wrappedAtRac2            : 0x0
wrappedAtRac3            : 0x0
wrappedAtRac4            : 0x0
wrappedAtRac5            : 0x0
```



Note: Each stack cable has six rac rings (ring access control), three outgoing/three incoming at 40Gig each. **WrappedAtRac** zero to five corresponds whether any stack link is down or not. If things are good, then it is showed as 0x0 (six links per asic, three outgoing, three incoming. For example, odd numbers are outgoing and even numbers are incoming or vice versa).

SifRacStatus

To check in detail each of the Racs, Critical aspects to verify are displayed; active/linkOk/syncOk bits which tell us if specific Rac has linked up or not (if OK then it is showed as 0x1).

```
Switch#sh plat hardware fed sw active fwd-asic register read register-name SifRacStatus

For asic 0 core 0

Module 0 - SifRacStatus[0][0]

active                    : 0x1 <----
available                 : 0x1
copyOk                    : 0x1
```

```
disabled                  : 0x0
insertOk                  : 0x1
linkOk                    : 0x1 <----
messageOk                 : 0x1
noDataOnRing              : 0x0
pcsAlignmentOk            : 0x1
pcsCodewordSync           : 0xf
reOrderOk                 : 0x1
slapId                    : 0x0
stripOk                   : 0x1
syncOk                    : 0x1 <----
toPbcOk                   : 0x1
transmitOk                : 0x1
```

SifRacControl

See if Rac is powered down or not. Check for the **greenPowerDisable** parameter. This shows 0x0 for all Racs (at least for Nyquist platform). There are some exceptions where it is expected to see Racs power down or **greenPowerDisable** parameter showed as 0x1 due to HW limitation on the stack cable itself, such as 3650 switch which is the lower end box. Then the stack cable just supports two Racs per asic. The remaining two Racs are powered down.

```
Switch#sh plat hardware fed sw active fwd-asic register read register-name SifRacControl

For asic 0 core 0

Module 0 - SifRacControl[0][0]

copyEn                    : 0x1
deployToken               : 0x0
disablePmaChecks          : 0x0
forceSync                 : 0x0
greenPowerDisable         : 0x0 <----
init                      : 0x0
initRacInfoLinkedList     : 0x0
insertEn                  : 0x1
messageEn                 : 0x1
reOrderEn                 : 0x1
stripEn                   : 0x1
toPbcEn                   : 0x1
transmitEn                : 0x1
```

SifExceptionInterruptA4

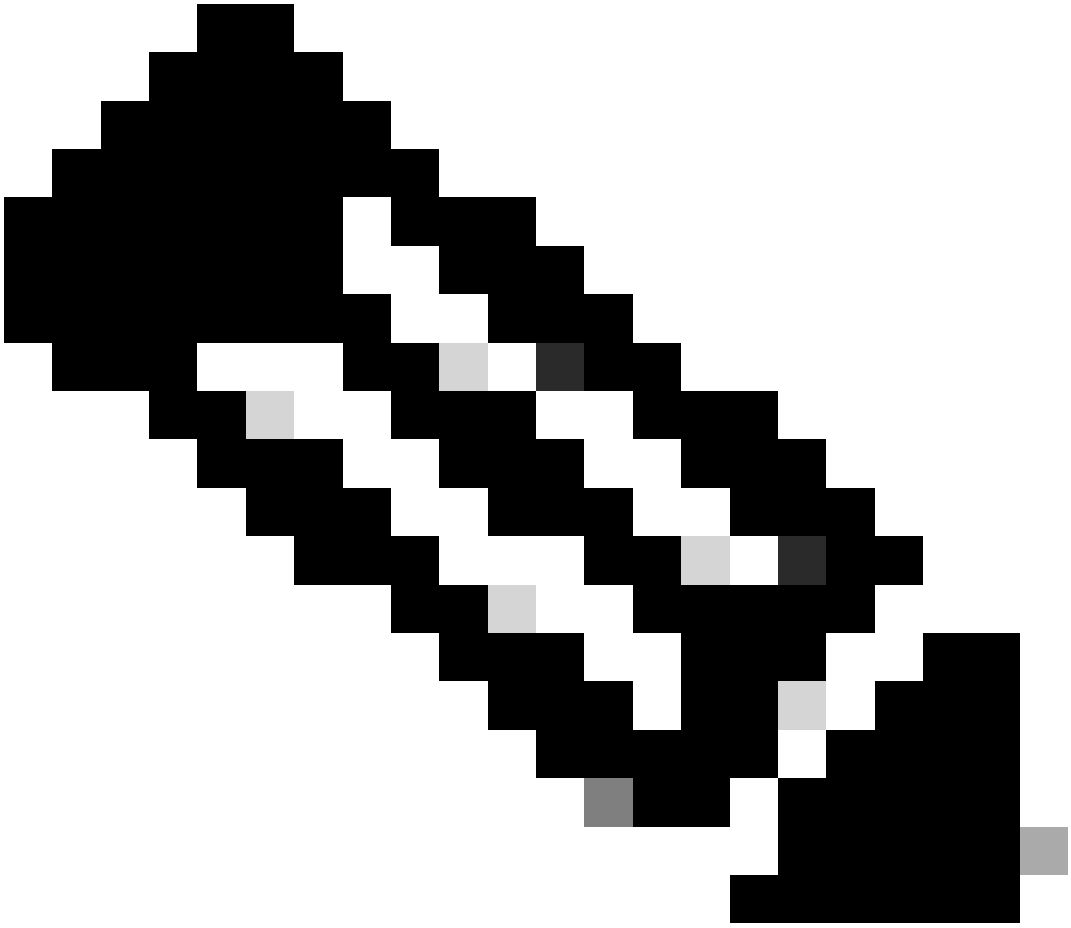
This is triggered because there is a link change in the system (Up/Down situation). The interrupt is handled at the software level. It is processed to see if there is any link related changes, and then it is published (log generated).

```
Switch#sh plat hardware fed sw active fwd-asic register read register-name SifExceptionInterruptA4
```

```
For asic 0 core 0

Module 0 - SifExceptionInterruptA4[0][0]

sifRac0LinkOkChange       : 0x0
sifRac0LinkedListSpill    : 0x0
sifRac0SyncOkChange       : 0x1
sifRac0TransitFifoSpill   : 0x0
sifRac1LinkOkChange       : 0x0
sifRac1LinkedListSpill    : 0x0
sifRac1SyncOkChange       : 0x1
sifRac1TransitFifoSpill   : 0x0
sifRac2LinkOkChange       : 0x0
sifRac2LinkedListSpill    : 0x0
sifRac2SyncOkChange       : 0x1
sifRac2TransitFifoSpill   : 0x0
sifRac3LinkOkChange       : 0x0
sifRac3LinkedListSpill    : 0x0
sifRac3SyncOkChange       : 0x1
sifRac3TransitFifoSpill   : 0x0
sifRac4LinkOkChange       : 0x0
sifRac4LinkedListSpill    : 0x0
sifRac4SyncOkChange       : 0x1
sifRac4TransitFifoSpill   : 0x0
sifRac5LinkOkChange       : 0x0
sifRac5LinkedListSpill    : 0x0
sifRac5SyncOkChange       : 0x1
sifRac5TransitFifoSpill   : 0x0
```

SifExceptionInterruptA8

This is the hardware interrupt which give us details when balloting is done (balloting = asic initialization process). After A8 is completed, the system checks if asic available bit is properly set. If not, then balloting is run again.



Note: When the maximum number is reached, the switch is reloaded with some error saying **HW available bit was not set** or **Balloting did not complete**.

```
Switch#sh plat hardware fed sw active fwd-asic register read register-name SifExceptionInterruptA8

For asic 0 core 0

Module 0 - SifExceptionInterruptA8[0][0]

sifBallotDone             : 0x0
sifBallotOverallTimerExpires : 0x0
sifBallotPerStateTimerExpires : 0x0
sifBallotSpeedChangeNeeded : 0x0
sifBallotStart            : 0x1
sifDebugSent              : 0x0
sifEastNeighborChange     : 0x1
sifMessageReceiveBufferCreditsEmpty : 0x0
sifMessageReceived        : 0x1
sifMessageSent            : 0x1
sifNodeIdChanged          : 0x1
sifOob3in2DropCntOverflow : 0x0
```

```
sifOobFlushDropCntOverflow : 0x0
sifOobStackSifCreditDropCntOverflow : 0x0
sifOobStackSifMtuDropCntOverflow : 0x0
sifOobSupSifMtuDropCntOverflow : 0x0
sifRacInfoLinkedListInitDone0 : 0x1
sifRacInfoLinkedListInitDone1 : 0x1
sifRacInfoLinkedListInitDone2 : 0x1
sifRacInfoLinkedListInitDone3 : 0x1
sifRacInfoLinkedListInitDone4 : 0x1
sifRacInfoLinkedListInitDone5 : 0x1
sifSegmentBuffer0LinkedListSpill : 0x0
sifSegmentBuffer1LinkedListSpill : 0x0
sifSegmentBufferLinkedListInitDone0 : 0x1
sifSegmentBufferLinkedListInitDone1 : 0x1
sifStackTopologyChange    : 0x1
sifUnmappedDestIndex      : 0x0
sifWestNeighborChange     : 0x1
```

The next command displays **SIF Counters** that involve SDP messages and SIF management messages. Focus on the failed messages, if any.

```
Switch#show platform software sif switch active r0 counters
Stack Interface (SIF) Counters

------------------------------


Stack Discovery Protocol (SDP) Messages
---------------------------------------
Message                 Tx Success    Tx Fail       Rx Success    Rx Fail
------------------------------------------------------------------------------
Discovery               0             0             0             0
Neighbor                0             0             0             0
Forward                 455966        0             1355818       107

---------------------------------------

SIF Management Messages
-----------------------
Message                 Success       Fail
-------------------------------------------------
Link Status             16            0
Link Management         0             0
Chassis Num             1             0
Topo Change             3             0
Active Declare          1             0
Template set            2             0
```

There is an additional command that could be run and displays information only when an **interrupt goes beyond the threshold**. The command is show platform software sif switch active R0 exceptions. Here is the output when no issues are present on the Interrupts:

```
Switch#
Switch#show platform software sif switch active R0 exceptions
```

```
Switch#
```

When interrupts are present, the output is similar to the next script. Keep in mind interrupts are expected in some scenarios (bootup, plug/unplug, and so on), so if there is a real issue and continous interrupts, execute the command repeatedly for a period of seconds/minutes.

```
Switch#show platform software sif switch active r0 exceptions
*******************************
Asicnum: 0
SIF INT : SIFEXCEPTIONINTERRUPTA1_SIFRAC5PMARECEIVEFIFOSPILL3_FIELD_IDX
Occurred count: 1
First Time: Fri May 18 08:03:23 2018
Last Time: Fri May 18 08:03:23 2018
--------------------------------
SIF INT : SIFEXCEPTIONINTERRUPTA1_SIFRAC5PMARECEIVEFIFOSPILL2_FIELD_IDX
Occurred count: 1
First Time: Fri May 18 08:03:23 2018
Last Time: Fri May 18 08:03:23 2018
--------------------------------
SIF INT : SIFEXCEPTIONINTERRUPTA1_SIFRAC5PMARECEIVEFIFOSPILL1_FIELD_IDX
Occurred count: 1
First Time: Fri May 18 08:03:23 2018
Last Time: Fri May 18 08:03:23 2018
--------------------------------
SIF INT : SIFEXCEPTIONINTERRUPTA1_SIFRAC5PMARECEIVEFIFOSPILL0_FIELD_IDX
Occurred count: 1
First Time: Fri May 18 08:03:23 2018
Last Time: Fri May 18 08:03:23 2018
```

This table details the most common SIF exceptions from show platform software sif switch active R0 exceptions:

Exception#	FieldName	Severity	Usage	Description
0	sifRac{0:5}PmaTransmitFifoSpill{0:3}	major	Statistic	This fires if the push-pull FIFO between the system clock and serdes clock spills. This cannot occur. If it does, it is likely an indicator that the Serdes clock has been disabled (eithe by programming or a faulty Serdes.) I this is not due to a programming issue it is a major issue. But the SIF self-heals. And the net result of a small issue is a lost segment or in extreme cases a re-init. If this was not a small issue, and it is still occuring, then afte processing this CHIEF, it re-fires, telling you the condition is still occurring at this point. This transmit link is toast.
1	sifRac{0:5}PmaReceiveFifoSpill{0:3}	major	Statistic	This fires if the push-pull FIFO between the system clock and serdes clock spills. This cannot occur. If it does, it is likely an indicator that the Serdes clock has been disabled (eithe

				by programming or a faulty Serdes.) I this is not due to a programming issue it is a major issue. But the SIF self-heals. And the net result of a small issue is a lost segment or in extreme cases a re-init. If this was not a small issue, and it is still occuring, then afte processing this CHIEF, it re-fires, telling you the condition is still occurring at this point. This transmit link is toast.
2	sifRac{0:5}SerdesLossOfLock{0:3}	major	Statistic	To be used in correlation with sifRac{0:5}PmaReceiveFifoSpill{0:3 to inform about the condition of the received Serdes clocks w.r.t. normal operation condition. If they are out-of spec, then the IdleDensity Timer cannot compensate for the difference. In general, this is an problem checker to ensure the assumption that the receiever Serdes are working properly is actually true.
3	sifRac{0:5}ClockLossOfLock{0:3}	major	Statistic	To be used in correlation with sifRac{0:5}PmaReceiveFifoSpill{0:3 to inform about the condition of the received Serdes clocks w.r.t. normal operation condition. If they are out-of spec, then the IdleDensity Timer cannot compensate for the difference. In general, this is an problem checker to ensure the assumption that the receiever Serdes are working properly is actually true.
4	sifRac{0:5}syncOkChange	minor	Monitor	Indication of link-flap
	sifRac{0:5}linkOkChange	minor	Monitor	Indication of link-flap
	sifRac{0:5}linkedListSpill	major	Monitor	Rac linked lists that are part of the reorder algorithm have exceeded the maximum entries possible. This is rea bad and means that the reorder is now tail-dropping data segments and OOB messages on this RAC. This cannot occur unless the stack is mis-configured or the linked list has experienced a soft error. See Exceptio 9 and10.
	sifRac{0:5}transitFifoSpill	major	Statistic	The transitFifo reponsible for moving data through the SIF to other nodes ha spilled this is likely due to a mis-configuration of the IdleDensityTime w.r.t. to the actual Serdes clock ppm (parts per million) offset for this switc vs. its neighbor.
5	sifRac{0:5}missingToken	major	Statistic	The Stack conch shell has been lost, corrupted, re-depolyed, and so. This i

				likely an indication that a bit-hit on th stack hit a SifTokenDesc. This is a very unlikely thing to happen. The SI can be configured to deal with this in different ways. Either re-ballot and start over, re-deploy a token, or allow the SIF to re-deploy.
	sifRac{0:5}duplicateToken	major	Statistic	
	sifRac{0:5}tokenDeployed	info	Statistic	
6	sifRac{0:5}RwCrcErrorCntOverflow	minor	Statistic	Likely all indicators of the stack cable or neighbor box being comproimsed. Broken out to this detail largely for debugging. In the course of normal operation syncOkChange and linkOkChange are all you need to know. In collecting **LONG-TERM-BER**, you have to monitor and count these when the counters roll-over for proper counting of bit-errors. It is possible that when an **invalidRw** or **pcsCodeWordError** is present, the CRC is not checked. That way you ca sum all these registers for BER.
	sifRac{0:5}DataCrcErrorCntOverflow	minor	Statistic	
	sifRac{0:5}InvalidRwErrorCntOverflow	minor	Statistic	
	sifRac{0:5}PcsCodeWordErrorCntOverflow	minor	Statistic	
7	sifRac{0:5}RdispErrorCntOverflow	minor	Statistic	
	sifRac{0:5}PrbsUnLockErrorCntOverflow	info	Statistic	Bring up stats for use in helping find the best configuration of the IBM HS macros to find the optimal programmings.
	sifRac{0:5}PrbsBitErrorCntOverflow	info	Statistic	
	sifRac{0:5}ErrorCaptureCntOverflow	info	Lab	Bring up stats for capturing the form errored ringWords for inspection to s what is happening on the Stack.
8	sifRacInfoLinkedListInitDone{0:5}	info	Monitor	HW initialization of the RAC linked list is complete.
	sifDroppedSegmentCntOverflow	info	Statistic	
	sifPbcInconsistentSopEopCntOverflow	info	Statistic	Worst case scenario. Check for data arrival as per protocol form from PBC
	sifPbcErrorCntOverflow	info	Statistic	
	sifSupInconsistentSopEopCntOverflow	info	Statistic	Worst case scenario. Check for data arrival as per protocol form from SUI (OOBM).
	sifSupErrorCntOverflow	info	Statistic	
	sifReorderInconsistentSopEopCntOverflow	info	Statistic	Indication that the missing segment indicator has rolled over.
	sifDebugSent	info	Lab	Bring up indication for insertion of debug Segments onto the stack.
	sifMessageSent	info	Lab	Due to the automated nature of the OOBM, these are really only useful i lab situations.

	sifMessageReceived	info	Lab	
	sifMessageDropped	info	Lab	
	sifMessageReceiveBufferCreditsEmpty	minor	Monitor	Please refresh Credits if this fires. Th credit level is actively monitored so that this does not trip.
	sifUnmappedDestIndex	minor	Statistic	During the **Copy/Strip**, it couldn not map the destIndex and a **portCop** was set to '0' and portStrip set to '1'. This indicates a configuration issue.
	sifSegmentBuffer{0:1}linkedListSpill	major	Monitor	Segment linked lists that are part of reorder have exceeded the maximum entries possible. This is an indication that the reorder is now tail-dropping data segments and OOB messages. This cannot occur unless the stack is mis-configured or the linked list has experienced a soft-error. See Excepti 9 and 10.
	sifSegmentBufferLinkedListInitDone{0:1]	info	Monitor	HW initialization of the segment link list is complete.
	sifBallotDone	info	Monitor	Indication Balloting has completed.
	sifBallotSpeedChangeNeeded	info	Monitor	Since the last successful ballot, a new speed is required on the stack link. This means that a Node has entered th stack has changed the dynamic of the stack speed. Either by being slower than the current speed, the stack has t adjust down. Or by being faster than i was previously. It can be the result of new shorter cable.
	sifEastNeighborChange	info	Monitor	Monitor for stack bring-up, merge, an wrap scenarios.
	sifWestNeighborChange	info	Monitor	
	sifNodeIdChanged	info	Monitor	Indication that as a result of the last ballot, the **SifInfo.nodeId** has been changed.
	sifStackTopologyChange	info	Monitor	Monitor for stack bring-up, merge, an wrap scenarios.
9	sifRacInfoBuffer{0:5}EccCorrected	major	Monitor	**sifRacInfoBuffer{0:5}** was hit with a soft-error. This is bad, but the worst-case result is some out-of-order packe or later packet drops in the egress datapath. Resetting Doppler is not required here.
	sifRacInfoBuffer{0:5}EccDetected	major	Monitor	
	sifRacInfoLinkedListBuffer{0:5}EccCorrected	major	Monitor	**sifRacInfoLinkedListBuffer{0:5}** w hit with a soft-error. Depending on th over-arching HA guideline for this SW load, you want to Reset Doppler. This can cause performance issues to SifReorder.
	sifRacInfoLinkedListBuffer{0:5}EccDetected	major	Monitor	
	sifSegmentLinkedListBuffer{0:1}EccCorrected	major	Monitor	**sifRacInfoLinkedListBuffer{0:5}** w

				hit with a soft-error. Depending on th over-arching HA guideline for this SV load, you want to Reset Doppler. This can cause performance issues to SifReorder.
	sifSegmentLinkedListBuffer{0:1}EccDetected	major	Monitor	
10	DestinationIndexTabeParityError	major	Monitor	Memory was hit with a parity error. Reload contents and recognize that some packets can have mis-copied/stripped as a result. Reset Doppler is probably not required.
	GlobalToLocalPortTable	major	Monitor	
	CpuIndexTable	major	Monitor	
	HashTableA	major	Monitor	
	HashTableB	major	Monitor	
	MessageQueueFifo	major	Monitor	Message control memories were hit with a soft-error. This is a transient issue that can lead to a mis-forwarded or out-of-order OOB. This can self he and does not require a Doppler reset a new users of the entries here can overwrite the old ones.
	MessageQueueLinkBuffer	major	Monitor	

This is found in **EDCS-757121:NG3K SIF Driver Software Functional Specification**.

Other Stacking Registers

- SifRacStatus
- SifStatistics
- SifRacInsertedCnt
- SifRacCopiedCnt
- SifRacPmaControl
- SifBallotWatchDogTimer
- SifPbcSifErrorCnt
- SifMessageStatus
- SifControl
- SupStackInterfaceControl
- SifSifPbcCnt0
- SifSifPbcCnt1
- SifSifPbcDroppedCnt
- SifSerdesHssMacroStatus
- SifSerdesHssChannelStatusRx
- SifSerdesHssChannelStatusTx

to understand details for each register.

Cli to monitor health of stack ports:

show platform hardware fed switch <> fwd-asic register read register-name SifSerdesHssMacroStatus
show platform hardware fed switch <> fwd-asic register read register-name SifInfo
show platform hardware fed switch <> fwd-asic register read register-name SifRacStatus
show platform hardware fed switch <> fwd-asic register read register-name SifRacControl

show platform hardware fed switch <> fwd-asic register read register-name SifExceptionInterruptA8
show platform hardware fed switch <> fwd-asic register read register-name SifExceptionInterruptA4
show platform hardware fed switch <> fwd-asic register read register-name SifStatistics
show platform hardware fed switch <> fwd-asic register read register-name SifRacInsertedCnt
show platform hardware fed switch <> fwd-asic register read register-name SifRacCopiedCnt
show platform hardware fed switch <> fwd-asic register read register-name SifRacPmaControl
show platform hardware fed switch <> fwd-asic register read register-name SifBallotWatchDogTimer
show platform hardware fed switch <> fwd-asic register read register-name SifPbcSifErrorCnt
show platform hardware fed switch <> fwd-asic register read register-name SifMessageStatus
show platform hardware fed switch <> fwd-asic register read register-name SifControl
show platform hardware fed switch <> fwd-asic register read register-name SupStackInterfaceControl
show platform hardware fed switch <> fwd-asic register read register-name SifSifPbcCnt0
show platform hardware fed switch <> fwd-asic register read register-name SifSifPbcCnt<>
show platform hardware fed switch <> fwd-asic register read register-name SifSifPbcDroppedCnt
show platform hardware fed switch <> fwd-asic register read register-name SifSerdesHssChannelStatusRx
show platform hardware fed switch <> fwd-asic register read register-name SifSerdesHssChannelStatusTx
show platform hardware fed switch <> fwd-asic register read register-name SifRacDataCrcErrorCnt
show platform hardware fed switch <> fwd-asic register read register-name SifgRacRwCrcErrorCnt
show platform software sif switch <> R0 counters
show platform software sif switch <> R0 exceptions

Reading Registers from Linux Kernel

.

After you are in the Linux Shell, proceed with the next script:

<#root>

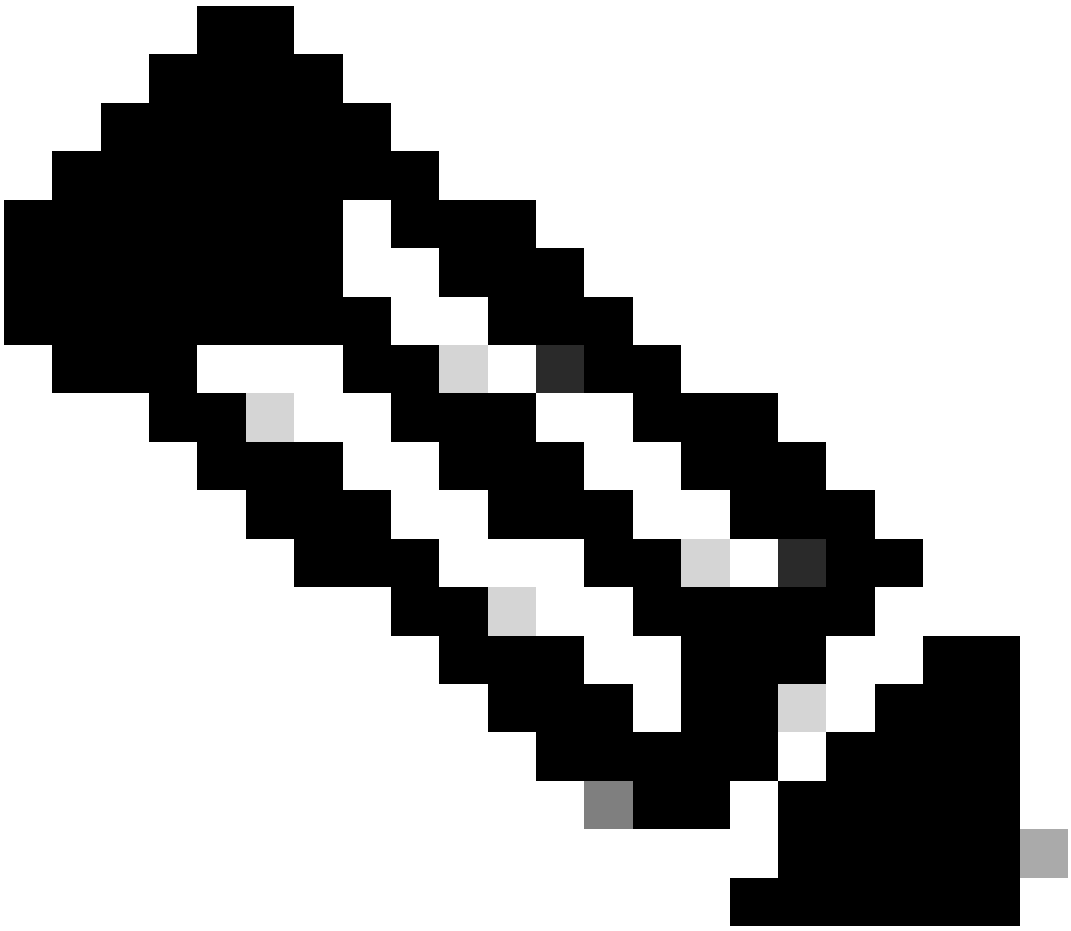
```
[Switch_2_RP_0:~]$ dope.sh
Num Asics: 0
Cat9300 platform dope vft
        ****************************************************
                DOPpler Examiner

         http://wwwin-dopplersdk.cisco.com
        ****************************************************
Detecting number of asics...found 1 asics
asic-0: phy_addr=0x87f80000000 virt_addr=0x7f84d746f000
Loading Library : libasd2_DL.so ... Success. (null)
ASIC Layer libraries successfully loaded!!!
ASIC version: 0x448
Starting ASIC Driver create
Driver and Device Init Completed.
dope[0,0]> rdsp SifControl <------- rdsp <register name>
```

Changing ASIC in Dope.sh

The previous script is reading switch one, asic zero. Change this performing this script:

```
dope[0,0]> asic 1 <--- changes to asic 1
```

```
dope[1,0]>
```



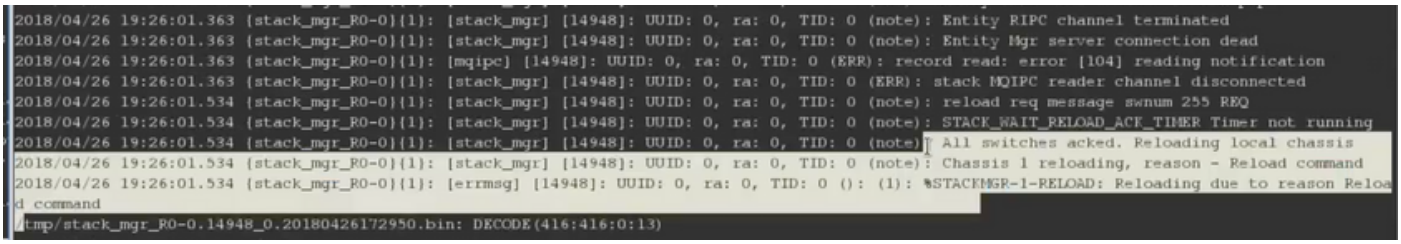
> **Note**: Dope.sh (Doppler shell) is the lowest level in the hardware programming. This is how you read the ring values directly from the hardware. Use the **Other Stacking Registers** in the prior script after the rdsp command to get the most granular data (if needed).

Silent Reloads Problems

Whenever there is a silent reload (**no crashdump/system_report generated**), there are crash tracelogs displaying some specific files to get more information related to what could cause the event.

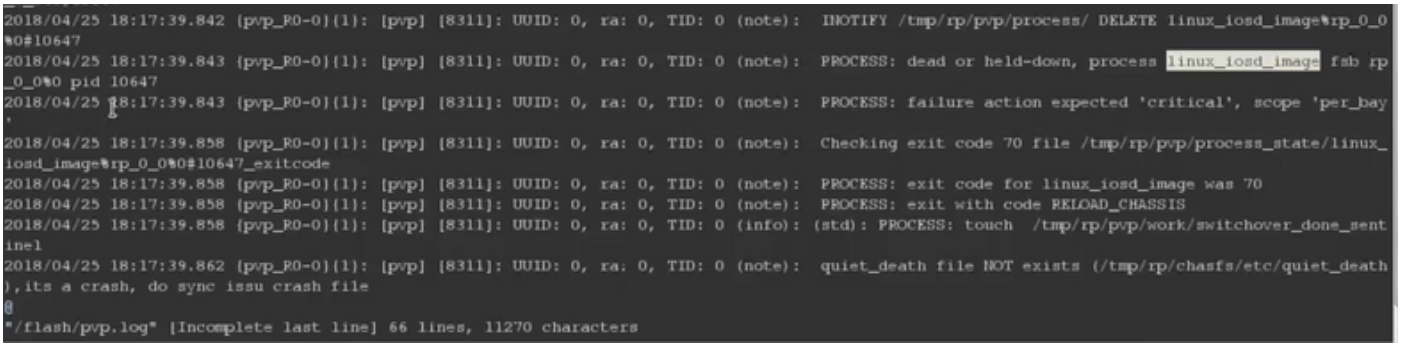
Step 1

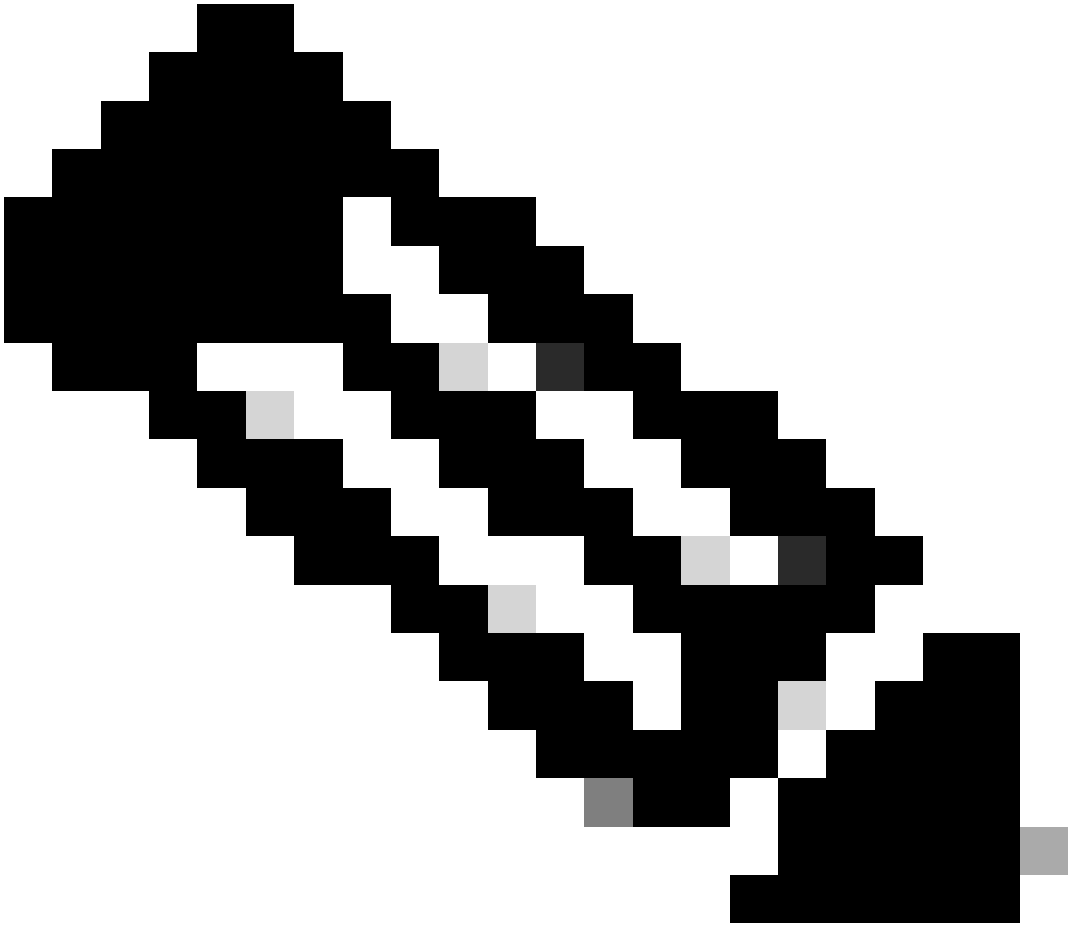
We can start looking at **stack_mgr_R0** first and see from its perspective the reason for the reload. Such as:



Step 2

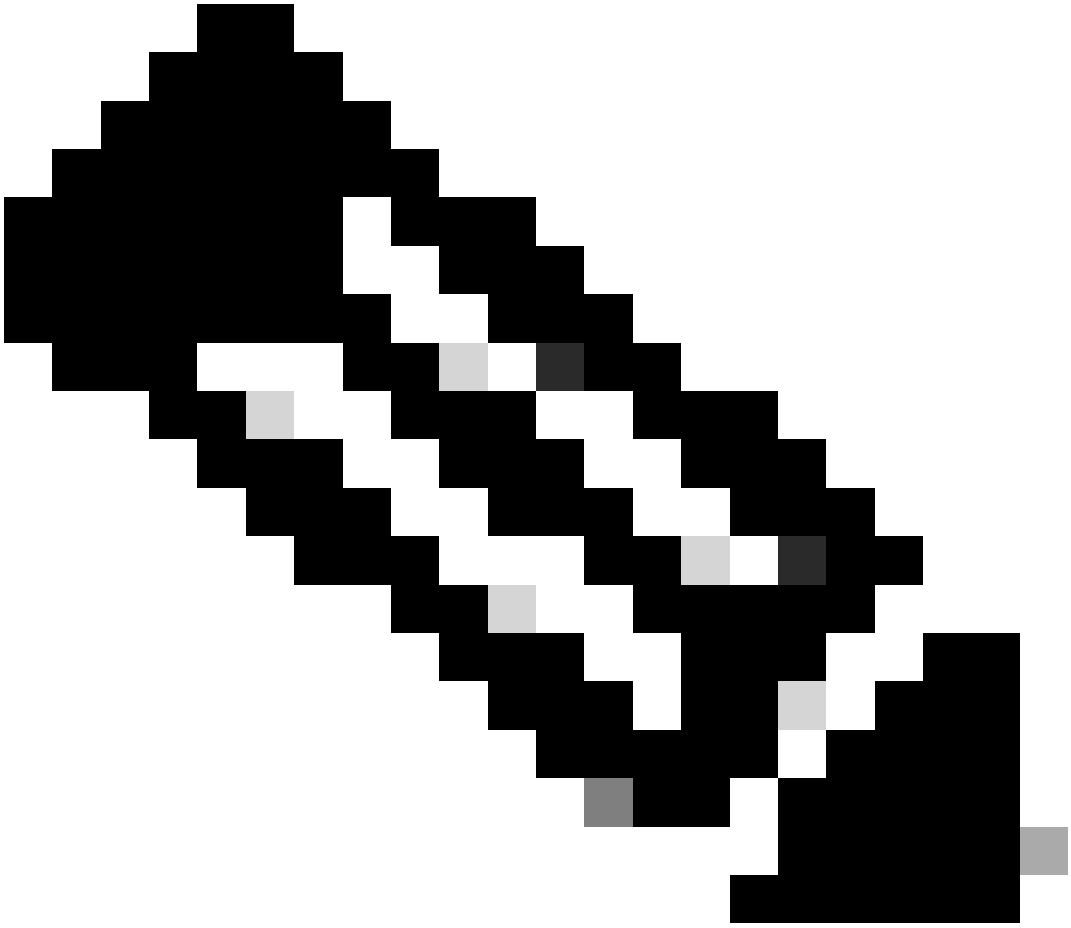
We can now move to pvp logs. Use the timestamps extracted from **stack_mgr_R0** (specifically when reload occurred) and look through **pvp_F0** and **pvp_R0** to identify when the processes termination sequence started before it executes all the reload orchestration sequence. Such as:





Note: It can show **pvp_F0** and **pvp_R0**.

```
-rw-r--r-- 1 root root  4476 Apr 24 21:38 pvp_F0-0.13136_0.20180424012429.bin.gz
-rw-r--r-- 1 root root  4405 Apr 24 01:12 pvp_F0-0.14840_0.20180403072736.bin.gz
-rw-rw-rw- 1 root root 10094 Apr 25 22:36 pvp_R0-0.8079_0.20180425223247.bin.gz
-rw-rw-rw- 1 root root  2938 Apr 26 17:26 pvp_R0-0.8079_1.20180425223618.bin.gz
```



Note: Make sure to check both because you could see **linux_iosd_image** process terminating in **pvp_R0**, but a different process within **pvp_F0** was terminated before. This is a key factor because the very first process that is killed. Then it can point to the trigger of the problem.

Step 3

Within **pvp_F0** and **pvp_R0**, there is also an exit code provided after the process dead/held-down. For real process crashes, exit codes 129 and so are used. This is how pvp is aware that **crashdump/system_report** needs to be created. With no **crashdump/system_report**, the exit code normally is zero. Such as:

```
2018/04/25 18:17:39.843 {pvp_R0-0}{1}: [pvp] [8311]: UUID: 0, ra: 0, TID: 0 (note):  PROCESS: failure action expected 'critical', scope 'per_bay
'
2018/04/25 18:17:39.858 {pvp_R0-0}{1}: [pvp] [8311]: UUID: 0, ra: 0, TID: 0 (note):  Checking exit code 70 file /tmp/rp/pvp/process_state/linux_
iosd_image%rp_0_0%0#10647_exitcode
2018/04/25 18:17:39.858 {pvp_R0-0}{1}: [pvp] [8311]: UUID: 0, ra: 0, TID: 0 (note):  PROCESS: exit code for linux_iosd_image was 70
```

Step 4

After identifying the culprit process, go to the process related btrace logs and check for more details.

Stack Member Timeouts/Reloads - Case Study

It is possible for a single bad cable between two switches to cause any switch in the stack to reload due to lost keepalives.

Symptoms

Stack traces, or switches, actively experiencing the issue produces these errors:

- 9300-1# show platform software trace message stack-mgr switch active R0 | inc not responding
- 2018 <tel:2018>/05/10 13:57:30.397 [stack_mgr] [24459]: UUID: 0, ra: 0, TID: 0 (note): Peer 4 not responding, for 8000 <tel:8000> msec. Bookkeep=3EFDD last_msg = 3EFD5
- 2018 <tel:2018>/05/10 13:57:29.396 [stack_mgr] [24459]: UUID: 0, ra: 0, TID: 0 (note): Peer 6 not responding, for 8000 <tel:8000> msec. Bookkeep=3EFDC last_msg = 3EFD4

The bookkeep checks every one second for the last time it heard from each switch in the stack (from the perspective of the switch running the bookkeep). After 8000 msec of no keepalives, we start printing traces that peers have not been heard. At 16000 msec, the switches in question reload for lost keepalives.

```
9300-1#sh switch stack-ports sum
Load for five secs: 8%/4%; one minute: 9%; five minutes: 9%
Time source is NTP, 11:53:11.196 EDT Thu May 17 2018

Sw#/Port#  Port Status  Neighbor  Cable Length   Link OK   Link Active   Sync OK   #Changes to LinkOK
-------------------------------------------------------------------------------------------------
1/1        OK           2         100cm          Yes       Yes           Yes       2
1/2        OK           8         300cm          Yes       Yes           Yes       143
2/1        OK           3         50cm           Yes       Yes           Yes       1
2/2        OK           1         100cm          Yes       Yes           Yes       1
3/1        OK           4         50cm           Yes       Yes           Yes       1
3/2        OK           2         50cm           Yes       Yes           Yes       1
4/1        OK           5         50cm           Yes       Yes           Yes       1
4/2        OK           3         50cm           Yes       Yes           Yes       1
5/1        OK           6         50cm           Yes       Yes           Yes       1
5/2        OK           4         50cm           Yes       Yes           Yes       1
6/1        OK           7         50cm           Yes       Yes           Yes       1
6/2        OK           5         50cm           Yes       Yes           Yes       1
7/1        OK           8         50cm           Yes       Yes           Yes       1
7/2        OK           6         50cm           Yes       Yes           Yes       1
8/1        DOWN         NONE      300cm          No        No            Yes       116
8/2        OK           7         50cm           Yes       Yes           Yes       1
```

This timeout was also seen where there was a large amount of instability in the stack link between to switches, eventally causing one switch to believe the stack port was up and able to pass traffic, but the other thinking it was down.

The stack-ring operates in both a clockwise and counter-clockwise direction. Traffic on the ring can take either path regardless of its destination. This means that if switch 2 wants to send a keepalive to switch 1, it can go through switches 3, 4, 5, 6, 7, 8 and then 1, or just from from 2 directly to 1. Return traffic from switch 1 to switch 2 that happens to hash toward switch 8 would have been dropped, leading to the timeouts seen in the previous script.

Acronyms

- OOB: Out of Band
- SIF: Stack Interface
- RAC: Ring Access Controller