

Troubleshoot 3650/3850 Stack Manager Reloads

Contents

[Introduction](#)

[Prerequisites](#)

[Requirements](#)

[Components Used](#)

[Background Information](#)

[System Report Versus Switch Reports](#)

[Where to Gather System/Switch Reports](#)

[Noteworthy Sections in the System Report](#)

[Types of Failures](#)

[Related Bugs](#)

[Diagnose a Potential Stack Cable or Port Issue](#)

[Additional Tips](#)

[1. Archive Crashinfo Directories](#)

[2. Recover an Unstable Stack](#)

[3. Generate System Reports Manually](#)

[Related Information](#)

Introduction

This document describes how to leverage system reports to diagnose stack issues.

Prerequisites

Requirements

There are no specific requirements for this document.

Components Used

This document is not restricted to specific software and hardware versions.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

If you troubleshoot stack reloads through a system report in the absence of a crash is commonly done on NGWC switching platforms you use the stackwise technology. The current documentation is limited on the uses of a system report and this guide explains how you can leverage these reports to diagnose problems typically found with stacking issues. This guide is particularly focused for the Catalyst 3650/3850 Switch


architectures that run Cisco IOS® XE with support stacking capabilities.

The majority of issues with stackwise technology stem from a communication problem between the members within a stack. Any inconsistency of information between the members or loss of connectivity can result in a problem that permeates through the entire stack and ultimately leads to a reset with stack manager. This document can highlight some of the common types of failures seen with stack-manager reloads, uses of a system report, and relevant CLIs available to diagnose and different types of problems.

System Report Versus Switch Reports

A system-report is a comprehensive report of the member from how it perceives the state of the stack. This is **not** a crashinfo (which can dump out memory for further debugging), but instead, it is a report that has logs and debugging information for various services that run under Cisco IOS XE that would be useful for development to track the state of that service. A system-report can be generated when the switch is reloaded by stack manager, a process crash has occurred, or manually generated by the user during live operation.

In many cases, there are situations in which a single switch can fail in a stack, but the rest of the members can remain intact. To gather as information on the state of the stack at the given time, switch_reports were introduced so that current members can generate one when it detects that a member has gone down. The switch_report can be a local report of how that member perceives the current state of the stack.

 **Note:** These reports are written and compressed so they cannot be printed to the terminal with **more**. They can need to be transferred off the switch and decompressed to view the log.

Where to Gather System/Switch Reports

System reports can typically be written in the crashinfo: directory of the member in that stack. For instance, if there is an x-member switch stack, then each switch can have their own crashinfo directory which can be accessible with **dir crashinfo-x** where x corresponds to that member within the stack.

```
3850#dir crashinfo-1:
```

```
Directory of crashinfo-1/
```

```
11 -rwx      355 Aug 14 2015 07:48:17 -04:00 last_systemreport_log
```


```
12 -rwx    724015 Oct 15 2014 07:14:32 -04:00 system-report_1_20141015-111342-UTC.gz
```

```
3850#dir crashinfo-2:
```

```
Directory of crashinfo-2/
```

```
11 -rwx      357 Aug 14 2015 07:50:49 -04:00 last_systemreport_log
```

```
12 -rwx    751340 Oct 15 2014 06:41:12 -04:00 system-report_1_20141015-104022-UTC.gz
```

 **Note:** Be sure to gather the output for **dir crashinfo-x** for every switch in that stack because the **show tech** can not list out the available file systems or the crashinfo files. It is important that you have the entire picture of each and every member in that stack. Update: As of newer Cisco IOS XE releases >3.6E, the show tech can reflect the **dir crashinfo: + show file systems** output. See Cisco bug ID [CSCun50428](#).

Note: Only registered Cisco users can access internal Cisco bug information and tools.

Noteworthy Sections in the System Report

From a TAC perspective, these are some of the more commonly viewed entries within the system report that can help diagnose events of a specific issue. There are other logs from other services contained in here that development can find want to review.

log file: /mnt/pss/sup_sysmgr_reset.log

This is a short output to very generically understand why a reset was seen. See the next types of failures section to look the meaning and context in how these reasons can vary.

log file: Cisco IOS

This is the log buffer maintained from within Cisco IOSd. Any commands that were issued by the user or syslogs generated within Cisco IOSd can be found in this section. Most recent logs are towards the end of this output.

Trace Buffer: stack-mgr-events

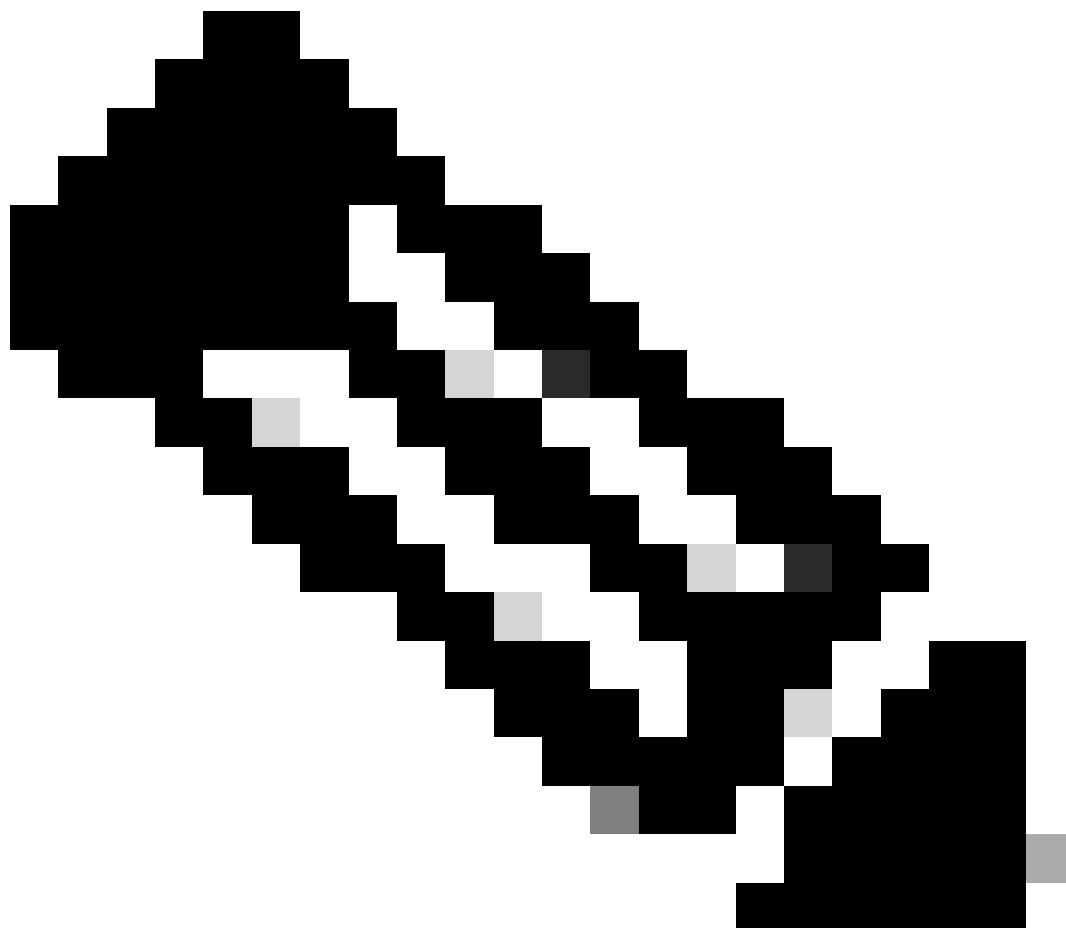
Keeps track of events seen from stack manager which can include when other members are joining/removed from the stack or which stack port the messages come in through.

Trace Buffer: redundancy-timer-ha_mgr

Displays keep alive events between switches in the stack. The timestamps can help determine when the breakdown in communication started.

Types of Failures

This section can highlight some commonly seen resets from a system report which are typically invoked by the stack manager process. Stack manager is a Linux process that manages the members in the stack and can oversee any changes in roles between switches in the stack. If stack manager detects a problem during initialization or role election, it can send a reload signal to individual switches in order for the stack to reset. The next list can also list known bugs that have been associated to the respective failure type.



Note: Not all stack-manager reloads are attributed to a software problem. In fact, it is more common to see these problems manifest as a secondary/victim issue to a core hardware problem.

Reset Reason:Reset/Reload requested by [stack-manager]. [ISSU Incompatibility]

You can see this type of reset when there is a bulk sync failure when you try to synchronize the configuration on the active between all the members in the stack. Check the logfile: Cisco IOS or the the logs from the active switch can highlight the configurations that contributed to this reset.

Reset Reason:Service [iosd] pid:[xyz] terminated abnormally [11].

This seen when the switch crashes in Cisco IOSd process. Look at the crashinfo directories for any crashinfo files + core dumps can be used to debug this failure further.

hap_sup_reset: Reset Reason:Reset/Reload requested by [stack-manager]. [stack merge]

A stack merge is seen when there are two or more switches that believe they are the active switch of the stack. This can be seen when there is a break in the ring of a stack (that is, two cables are disconnected from the stack) so both the active and standby loses communication to the other members. The addition of an already powered switch to a current stack can cause a stack merge as there can be two active switches in the stack.

[Cisco bug ID CSCuh58098](#) - 3850 stack can reload when stack cabling issues are present

[Cisco bug ID CSCui56058](#) - Enables debounce logic for stack cable

[Cisco bug ID CSCup53338](#) - 3850 IOSD crash | Signal=SIGSEGV(11) @ pm_port_data_from_swidb

hap_sup_reset: Reason Code:[4] Reset Reason:Reset/Reloadrequested by [stack-manager]. [stack merge due to incompatibility]

This has been seen in situations when an active and standby switch exists in the stack. If the active switch loses communication to the standby, the standby can attempt to take over as the active even though the active is still up.

[Cisco bug ID CSCuo49555](#)

[Cisco bug ID CSCup58016](#) - 3850/3650 crashes due to unicast flood on mgmt port

[Cisco bug ID CSCur07909](#) - Stack merge due to active and standby lost connectivity

Reset Reason:Reset/Reload requested by [stack-manager]. [Wrong neighbor encountered after ASIC ballot]

Switches participate in an ASIC ballot during boot up to determine its neighboring switches within the stack. This reset can be seen when a timer expires for a neighbor to declare itself or if there is a logic error during the nbrcast conversation. This has been seen in context of faulty stack cables which have been resolved through replacement.

[Cisco bug ID CSCun60777](#) - Switch reloaded due to Wrong neighbor encountered after ASIC ballot

[Cisco bug ID CSCud93761](#) - Switch reloaded due to Wrong neighbor encountered after ASIC ballot

[Cisco bug ID CSCun17506](#) - Amur:ng3k:same neighbor is seen on both stack ports on a 3 member stack

hap_sup_reset: Reason Code:[4] Reset Reason:Reset/Reload requested by [stack-manager]. [lost both active and standby]

This is typically seen from a member on the stack that is not in an either an active or standby role. When the active fails, if there is no standby switch to assume the active role for the stack, then the entire stack can reset. If the stack is in an unstable state or redundancy policy have not synced yet, this can be seen. This is likely a victim of the why the active/standby switches went down or potentially an indication that redundancy does not sync correctly. This can also be seen in when stacks are configured in a half-ring setup.

[Cisco bug ID CSCup53882](#)- Member switches in a 3850 stack reboot - [lost both active and standby]

hap_sup_reset: Reason Code:[1] Reset Reason:Reset/Reload requested by [stack-manager]. [Keepalive_Timeout]

Seen when keep alive messages are not received from the switch in the stack. Look at **Trace Buffer: redundancy-timer-ha_mgr** and it shows the exchange of keep alive messages and provide a perspective of time for when the breakdown in communication began. If you gather switch reports from the rest of the stack and look at logs throughout the time frame can help.

hap_sup_reset: Reset Reason:Reset/Reload requested by [stack-manager]. [Reload command]

This is a pretty intuitive reset reason – this is seen when stack-manager receives a reload request which could be invoked through CLI or externally via management device (SNMP). In cases of the Cisco bug ID [CSCuj17317](#), this also shows up as a **reload** command issued as well. From the log file: Cisco IOS you can see:

```
CMD: 'reload'
%SYS-5-RELOAD: Reload requested by console. Reload Reason: Reload command.
%STACKMGR-1-RELOAD_REQUEST: 1 stack-mgr: Received reload request for all switches, reason Reload command
%STACKMGR-1-RELOAD: 1 stack-mgr: Reloading due to reason Reload command
```

Related Bugs

[Cisco bug ID CSCur76872](#) - Stack manager goes down when the system runs out of SDP buffer.

[Cisco bug ID CSCup49704](#) - 3850 FED Crash - Waits for SPI channels FED_SPI_FLCD,FED_SPI_FAST
...

Diagnose a Potential Stack Cable or Port Issue

Symptom 1) Any signs of a stack cable issue is apparent by any flapping of the stack port prior to the reset. Look at the **logfile: Cisco IOS** report prior to a reset is typically a good place to start. Here is an example of where you see flapping of the stack port which is registered on both the current SW2 and the standby SW1. This same stack port was flapping each in each instance of the reset and was resolved by when the stack cable was replaced:

```
===== log file: Cisco IOS =====
```

```
.
.
Aug  8 21:40:14.532 UTC: %STACKMGR-1-STACK_LINK_CHANGE: STANDBY:1 stack-mgr: Stack port 1 on switch 1
Aug  8 21:40:17.242 UTC: %STACKMGR-1-STACK_LINK_CHANGE: STANDBY:1 stack-mgr: Stack port 1 on switch 1
Aug  8 21:46:11.194 UTC: %STACKMGR-1-STACK_LINK_CHANGE: 2 stack-mgr: Stack port 2 on switch 2 is down
Aug  8 21:46:12.937 UTC: %STACKMGR-1-STACK_LINK_CHANGE: 2 stack-mgr: Stack port 2 on switch 2 is up
Aug  8 21:48:23.063 UTC: %STACKMGR-1-STACK_LINK_CHANGE: 2 stack-mgr: Stack port 2 on switch 2 is down
```

```
Aug 8 21:48:24.558 UTC: %STACKMGR-1-STACK_LINK_CHANGE: 2 stack-mgr: Stack port 2 on switch 2 is up
Aug 8 21:50:40.666 UTC: %STACKMGR-6-SWITCH_REMOVED: 2 stack-mgr: Switch 1 has been removed from the s
Aug 8 21:50:40.671 UTC: Starting SWITCH-DELETE sequence, switch 1
```

Symptom 2) Based on the stackwise setup is used (180, 480, plus), the number of transmission rings per port ASIC varies. These commands poll global registers that maintain a total that increments by how many read errors are seen for each transmission ring. Port-asic 0 corresponds to stack port 1 and port-asic 1 corresponds to stack port 2. This must be issued for every switch and any signs of counts that increment can isolate whether there can be a problem at the port or with the stack cable.

These can be collected several times over a few minutes to compare the deltas in the count:

show platform port-asic <0-1> read register SifRacDataCrcErrorCnt switch <switch#>

- Segment with data CRC error.

show platform port-asic <0-1> read register SifRacRwCrcErrorCnt switch <switch#>

- Incremented on any failed CRC check.

show platform port-asic <0-1> read register SifRacPcsCodeWordErrorCnt switch <switch#>

- Incremented on invalid PCS code, unknown PCS codeword, that run a detected disparity error.

show platform port-asic <0-1> read register SifRacInvalidRingWordCnt switch <switch#>

- Bit error on stack caused ringword CRC error.

For Polaris (16.X code) these are the commands:

```
show plat hardware fed sw <#/active/standby> fwd-asic register read register-name SifRacDataCrcErrorCnt
asic <0-1>
```

```
show plat hardware fed sw <#/active/standby> fwd-asic register read register-name SifRacRwCrcErrorCnt
asic<0-1>
```

```
show plat hardware fed sw <#/active/standby> fwd-asic register read register-name
SifRacPcsCodeWordErrorCnt asic <0-1>
```

```
show plat hardware fed sw <#/active/standby> fwd-asic register read register-name
SifRacInvalidRingWordCnt asic <0-1>
```

This example shows where you had a stack merge event seen both members of a 2-member stack without any signs of a flapping stack port. You see the ring[0] increment with CRCs on stack port-1 of switch 1 and to get past this issue, replace the stack cable.

```
3850#show platform port-asic 0 read register SifRacRwCrcErrorCnt switch 1
Load for five secs: 11%/4%; one minute: 11%; five minutes: 12%
Time source is NTP, 14:02:49.119 EDT Thu Aug 20 2015
```

For asic 0

```
SifRacRwCrcErrorCnt on Asic 0
```

```
[0]
```

```
count 0x00000031 <<
```

```
[1]          count 0x00000001
[2]          count 0x00000000
[3]          count 0x00000001
[4]          count 0x00000000
[5]          count 0x00000001
```

```
3850#show platform port-asic 0 read register SifRacRwCrcErrorCnt switch 1
Load for five secs: 9%/4%; one minute: 11%; five minutes: 12%
Time source is NTP, 14:02:53.550 EDT Thu Aug 20 2015
```

For asic 0

```
SifRacRwCrcErrorCnt on Asic 0
[0]          count 0x000000c9 <<
[1]          count 0x00000001
[2]          count 0x00000000
[3]          count 0x00000001
[4]          count 0x00000000
[5]          count 0x00000001
```




Note: Based on the register that is reviewed, the mask can be different in each case. In the previous example, the mask can wrap around on the last 14 bits. Thus, when the counter reaches 0x00003FFF, it can wrap back to 0x00000000.

Additional Tips

1. Archive Crashinfo Directories

More switches in the stack means that there can be more report files to be collected. It is easy to get overwhelmed by the number of reports that are generated. Organization is vital to separate out a failure. Find a consistency with timestamps of when each switch wrote report file for a given incident if possible. From there, ask for those very specific reports from those given switches so the client does not upload several files. The crashinfo directory can also be archived so that the Cisco user can send a single archive contains the interested reports. The next example can create an archive named **crashinfo-archive.tar** in the flash directory:

```
F340.03.10-3800-1#archive tar /create ?
```

WORD Tar filename

```
F340.03.10-3800-1#archive tar /create crashinfo-archive.tar ?  
WORD Dir to archive files from
```

```
F340.03.10-3800-1#archive tar /create crashinfo-archive.tar crashinfo ?  
WORD File or Dir  
<cr>
```

```
F340.03.10-3800-1#archive tar /create crashinfo-archive.tar crashinfo:
```

2. Recover an Unstable Stack

There can be some situations where you see a several members in a stack reload during boot up after the stack election process takes place. If a reloaded switch believes itself to be the active then this can often lead to a stack merge event and can enter into a boot loop state. In this situation, it can be advisable to ask the Cisco client:

- Power down the entire stack and reset all the stack cables firmly.
- Power-on each member switch in the stack one by one until all members have converged to its expected state.
- In cases where a member fails to join the stack, remove this from the stack and try to boot this individual as a standalone to troubleshoot further.

3. Generate System Reports Manually

A manually created system reports requires **service internal** to be enabled. This writes a system report as a text file which can be done per switch basis.

```
3800-1#conf t  
Enter configuration commands, one per line. End with CNTL/Z.  
3800-1(config)#service internal  
3800-1(config)#exit
```

```
3800-1#resource create_system_report ?  
WORD system report filename
```

```
3800-1#resource create_system_report sysreport.txt ?  
switch Switch number  
<cr>
```

```
3800-1#resource create_system_report sysreport.txt switch ?  
<1-1> Switch number
```

```
3800-1#resource create_system_report sysreport.txt switch 1 ?  
<cr>
```

Related Information

- [Cisco Technical Support & Downloads](#)