# Troubleshooting StarOS Facility Crashes

## Contents

## Introduction

This document describes how to find and troubleshoot StarOs Facility Crashes.

## Overview

At times, the system logic can fail, causing a software task to restart in order to restore proper functionality. This can lead to a process crash. Task facility crashes are frequently reported in StarOS, and the necessary actions can be taken based on the root cause of the crash. To identify crashes on the node, you can use this CLI command:

```
 ******** show crash list *******
Saturday April 15 05:05:56 SAST 2023
=== ==================== ======== ========== =============== =======================
#           Time         Process  Card/CPU/        SW          HW_SER_NUM
                                     PID         VERSION        CF / Crash Card
=== ==================== ======== ========== =============== =======================

1   2022-Dec-02+14:08:46 confdmgr 02/0/19342 21.26.13        NA
2   2022-Dec-02+14:48:08 confdmgr 02/0/31546 21.26.13        NA
3   2022-Dec-04+19:10:50 sessmgr  03/0/12321 21.26.13        NA
4   2022-Dec-21+03:34:13 sessmgr  04/0/12586 21.26.13        NA
```

Similar crashes are consolidated into one record. The record does display the number of times this crash type has occurred.

```
********************* CRASH #02 ***********************
SW Version          : 21.26.13
Similar Crash Count : 33 >>>>
Time of First Crash : 2022-Dec-02+14:10:05

Assertion failure at confdmgr/src/confdmgr_fsm.c:870
  Note: State machine failure, state = 3
  Function: confdmgr_fsm_state_wait_p0_handler()
  Expression: 0
  Code: CRASH
  Proclet: confdmgr (f=1900,i=0)
  Process: card=2 cpu=0 arch=X pid=31546 argv0=confdmgr
```

In `show snmp trap history verbose`output shows that some process has crashed:

```
Fri Dec 26 08:32:20 2014 Internal trap notification 73 (ManagerFailure) facility sessmgr
instance 188 card 7 cpu 0
Fri Dec 26 08:32:20 2014 Internal trap notification 150 (TaskFailed) facility sessmgr instance
188 on card 7 cpu 0
Fri Dec 26 08:32:23 2014 Internal trap notification 1099 (ManagerRestart) facility sessmgr
instance 139 card 4 cpu 1
Fri Dec 26 08:32:23 2014 Internal trap notification 151 (TaskRestart) facility sessmgr
instance 139 on card 4 cpu 1
```

## Scenario of Crashes

There can be multiple different reasons for crashes:

1. Different call flow scenarios

2. Memory issues

3. Configuration issue

4. Hardware failures

## Reason Behind Crash

You have multiple task facilities in StarOS having their individual functionality so based on the functions, whenever the facility encounters any such input where it is getting into a problematic state, it crashes the facility to recover from that error state.

## Different Types of Crash

1. Assertion failure:

```
******************** CRASH #22 **********************
SW Version          : 21.26.13
Similar Crash Count : 33
Time of First Crash : 2023-Apr-12+22:40:01

Assertion failure at sess/smgr/sessmgr_snx.c:9568 >>>>
  Function: sessmgr_snx_send_drop_call()
  Expression: result == SN_STATUS_SUCCESS
  Proclet: sessmgr (f=87000,i=261)
  Process: card=5 cpu=0 arch=X pid=12724 cpu=~0% argv0=sessmgr
```

2. Segmentation fault:

```
******************** CRASH #69 **********************
SW Version : 21.13.3
Similar Crash Count : 2
Time of First Crash : 2019-Nov-25+07:53:54
Fatal Signal 11: Segmentation fault >>>>
Faulty address: 0x7ff6b4801036
Signal from: kernel
Signal detail: address not mapped to object
Process: card=8 cpu=1 arch=X pid=7316 argv0=vpp
Crash time: 2020-Feb-11+04:04:23 UTC
Build_number:
```

3. Fatal Signal:

```
******************** CRASH #01 **********************
SW Version          : 21.23.12
Similar Crash Count : 2
Time of First Crash : 2023-Jan-27+05:22:46

Fatal Signal 11: 11  >>>>>
  PC: [04be6859/X] sessmgr_pgw_create_bearers()
  Faulty address: 0x297116e4
  Signal from: kernel
  Signal detail: address not mapped to object
  Process: card=9 cpu=1 arch=X pid=10383 cpu=~8% argv0=sessmgr
```

## Initial Logs Requirement

The crash log serves as a valuable source of crash event information. When a software crash occurs, StarOS captures and stores relevant data that can help determine the cause of the crash. This information can be stored in system memory or transferred and saved on a network server.

Core File or Mini Core File: Note that core files correspond to the PID(s) where the crash occurred. Core files are named in the format "crash-<card no>-<cpu>-<pid>-<unixtime>-core". You can find this information in the "show crash list" command output.

Minicore File: This file contains information about the failing task, including the current stack trace, past profiler samples, past memory-activity samples, and other bundled data in a proprietary file format.

Core Dump (or Full Core): A core dump provides a complete memory dump of the process immediately after the crash occurred. This memory dump is often essential in identifying the root cause of the software crash.

Crash Signatures: The crash signatures can be reviewed from the shared Show Support Details (SSD) or other relevant sources.

```
 ******** show crash list *******
```

```
Saturday April 15 05:05:56 SAST 2023
=== =================== ======== ========== ============== ======================
#          Time         Process  Card/CPU/        SW          HW_SER_NUM
                                    PID        VERSION       CF / Crash Card
=== =================== ======== ========== ============== ======================

1   2022-Dec-02+14:08:46 confdmgr 02/0/19342 21.26.13        NA
2   2022-Dec-02+14:48:08 confdmgr 02/0/31546 21.26.13        NA
3   2022-Dec-04+19:10:50 sessmgr  03/0/12321 21.26.13        NA
```

Now if you want to know the signature for crash 1, search in SSD with **CRASH #01** or in CLI use **show crash number 1.**

```
From SSD

******************** CRASH #01 ***********************
SW Version          : 21.26.13
Similar Crash Count : 1
Time of First Crash : 2022-Dec-02+14:08:46

Assertion failure at confdmgr/src/confdmgr_fsm.c:758
  Note: State machine failure, state = 5
  Function: confdmgr_fsm_state_wait_p1_handler()
  Expression: 0
  Code: CRASHâ€‹

Using CLI

[local]abc# show  crash number 1
Friday June 09 06:41:53 CDT 2023
******************** CRASH #01 ***********************
SW Version          : 21.12.20.77760
Similar Crash Count : 1
Time of First Crash : 2021-Mar-31+15:58:06

Fatal Signal 6: Aborted
  PC: [ffffe430/X] __kernel_vsyscall()
  Note: User-initiated state dump w/core.
  Signal from: sitmain pid=6999 uid=0
  Process: card=9 cpu=0 arch=X pid=9495 cpu=~0% ar
```

Examine the Show Support Details (SSD) and syslogs during the specific timestamp when the problem occurred.

## Analysis Steps

1. Need to check the crash stack/signature and to check if there are any bugs for that particular crash signature.

2. Need to parse the corefile/minicore to analyze the backtrace and to get a clue for what function the facility crashed.

3. Once corefile debugging is done, you need to verify the symptoms with the software defect and if there is any existing software defect for a similar crash signature and backtrace.

## Session Recovery

StarOs software is designed to handle both foreseen conditions/events and unforeseen conditions/events. While Cisco strives to have perfect software, inevitably mistakes do exist and crashes are possible. That is why the session recovery feature is so important.

The Session Recovery feature provides seamless failover and reconstruction of subscriber session information in the event of a hardware or software fault within the system preventing a fully connected user session from being disconnected. Session recovery is performed by mirroring key software processes (for example, session manager and AAA manager) within the system. These mirrored processes remain in an idle state (standby mode) wherein they perform no processing until they can be needed in the event of a software failure (for example, a session manager task aborts).

Tasks such as demux tasks, AAA manager, and VPN manager have built-in automatic recovery mechanisms in our systems, specifically for handling subscriber information. Session recovery primarily refers to scenarios where there is a failure in the sessmgr task or a card-level failure, and sessions need to be recovered without any call loss.

- In the system, a standby session manager is active on each processing card and can quickly take over as the primary sessmgr in case of a failure. A new standby sessmgr is then created on the processing card.
- When a sessmgr process unexpectedly fails, the standby sessmgr retrieves backed-up information from the aaamgr (AAA manager) and rebuilds its sessions.
- If the aaamgr fails, the standby aaamgr queries the sessmgr to synchronize the relevant subscriber information.
- In the event of demux-mgr failure, it queries all sessmgrs and reconstructs its database of call distribution information.

- To ensure card-level redundancy, one processing card serves as a standby, ready to take over in case of hardware or software failure. The standby card then recovers the sessions from a peer aaamgr running on a different card.

```
 ******** show session recovery status verbose *******
Saturday April 15 05:11:17 SAST 2023
Session Recovery Status:
  Overall Status        : Ready For Recovery  >>>>
  Last Status Update    : 5 seconds ago

              ----sessmgr---  ----aaamgr----  demux
 cpu state    active standby  active standby  active  status
---- -------  ------ -------  ------ -------  ------  -------------------------
 3/0 Active   40     1        40     1        0       Good
 4/0 Active   40     1        40     1        0       Good
 5/0 Active   40     1        40     1        0       Good
 6/0 Active   40     1        40     1        0       Good
 7/0 Active   0      0        0      0        10      Good (Demux)
 8/0 Active   40     1        40     1        0       Good
 9/0 Active   40     1        40     1        0       Good
10/0 Active   40     1        40     1        0       Good
11/0 Standby  0      40       0      40       0       Good
```