ASR5x00 Session Manager Tasks - Description of Function, Crash, Recovery Operations, and Crash Logs

TAC **Document ID: 119224**

Contributed by Krishna Kishore D V, Cisco TAC Engineer.

Aug 25, 2015

Contents

Introduction

This document describes and explains the software reliability, service availability, and failover features for the Cisco Aggregation Services Router (ASR) 5x00 Series. It presents the definition for a software crash on ASR5x00 and the effects of the software crash. The article goes on to establish that even in case of unexpected software crashes, how the ASR5x00 is able to deliver the goal of "carrier-class" availability due to the inherent software resiliency and availability features. The mobile subscriber should never have to think about the availability of the service. Cisco's goal is no session loss due to any single hardware or software failures, which include loss of a complete system, in other words - voice grade reliability. The software reliability features on ASR5x00 are targeted to be able to achieve the goals for "carrier-class" service availability even in cases where unforeseen failures might occur in an operator's network.

Software Architecture: Designed for Resiliency

The ASR5x00 has a collection of software tasks distributed across the Packet Services Card (PSC) or Data Processing Card (DPC) and System Management Card (SMC) or Management and I/O (MIO) cards that are designed to perform a variety of specific functions.

For example, the session manager task is responsible for handling the sessions for a set of subscribers and to perform inline services such as Peer-to-peer (P2P), Deep Packet Inspection (DPI), and so on, on user traffic. The Authentication, Authorization, and Accounting (AAA) manager task is responsible for the generation of billing events in order to record subscriber traffic usage and so on. The session manager and AAA manager tasks run on the PSC/DPC card.

The SMC/MIO card is reserved for Operation and Maintenance (O&M) and platform related tasks. The ASR5x00 system is virtually compartmentalized into different software subsystems such as the session

subsystem for processing subscriber sessions and the VPN subsystem that is responsible for IP address assignment, routing, and so on. Each subsystem has a controller task that oversees the health of the subsystem it controls. The controller tasks run on the SMC/MIO card. The session manager and AAA manager tasks are paired together in order to handle a subscriber's session for control, data traffic, and billing purposes. When session recovery is enabled in the system, each session manager task backs up the state of its set of subscriber states with a peer AAA manager task to be recovered in the event of a session manager crash.

What is a crash?

A task in the ASR5x00 can potentially crash if it encounters a fault condition during normal operation. A crash or software fault in the ASR5x00 is defined to be an *unexpected* exit or termination of a task in the system. A crash can happen if the software code attempts to access memory areas that are prohibited (such as corrupted data structures), encounters a condition in the code that is not expected (such as an invalid state transition), and so on. A crash can also be triggered if the task becomes unresponsive to the system monitor task and the monitor attempts to kill and restart the task. A crash event can also be explicitly triggered (as opposed to unexpected) in the system when a task is forced to dump its current state by a CLI command or by the system monitor in order to analyze the task state. An expected crash event can also happen when the system controller tasks restart themselves in order to potentially correct a situation with a manager task that repeatedly fails.

Effects of a Session Manager Crash

Under normal operation, a session manager task handles a set of subscriber sessions and associated data traffic for the sessions along with a peering AAA manager task that handles billing for those subscriber sessions. When a session manager crash occurs it ceases to exist in the system. If session recovery is enabled in the system, a standby session manager task is made to become active in the same PSC/DPC card. This new session manager task reinstates the subscriber sessions as it communicates with the peer AAA manager task. The recovery operation ranges from 50 msec to a few seconds dependent upon the number of sessions that were active in the session manager at the time of the crash and overall CPU load on the card and so on. There is no loss in subscriber sessions that were already established in the original session manager in this operation. Any subscriber session that was in the process of establishment at the time of the crash will likely also be restored due to protocol retransmissions and so on. Any data packets that were in transition through the system at the time of the crash can assumed to be associated with a network loss by the communicating entities of the network connection and will be retransmitted and the connection will be carried on by the new session manager. Billing information for the sessions carried by the session manager will be preserved in the peer AAA manager.

When should the operator get concerned?

When a session manager crash occurs, the recovery procedure happens as described previously and the rest of the system remains unaffected by this event. A crash in one session manager does not impact the other session managers. As a guidance to the operator, if multiple session manager tasks *on the same PSC/DPC card* crash simultaneously or within 10 minutes of each other, there might be loss of sessions as the system might not be able to start new session managers fast enough to take the place of the crashed tasks. This corresponds to a double fault scenario where loss of sessions can occur. When recovery is not feasible, the session manager is simply restarted and is ready to accept new sessions.

When a given session manager crashes repeatedly (such as it encounters the same fault condition over and over), the session controller task takes note and restarts itself in an attempt to restore the subsystem. If the session controller task is unable to stabilize the session subsystem and restarts itself continously over in this effort, the next step in the escalation is for the system to switch over to a standby SMC/MIO card. In the unlikely event that there is no standby SMC/MIO card or if a failure is encountered in the switchover

operation, the system reboots itself.

Session managers also maintain statistics for each Access Point Name (APN), Services, functionalites, and so on that will be permanently lost when a crash occurs. Therefore an external entity that collects bulkstats periodically will observe a dip in the statistics when one or more crashes occur. This can manifest as a dip in a graphical representation of the statistics drawn over a time axis.

Note: A typical chassis populated with 7-14 PSC or 4-10 DPC cards has about 120-160 session managers, dependent upon the number of PSC/DPC cards, and a single crash will result in the loss of about 1/40th or 1/80th of the statistics. When a standby session manager takes over, it begins to accumulate the statistics again from zero.

How to know if a crash occurred?

A crash will trigger an SNMP trap event to a network monitoring station, such as the Event Monitoring Service (EMS) and by syslog events. The crashes that have occurred in the system can also be observed with the **show crash list** command. Note that this command lists both unexpected and expected crash events as described earlier. These two types of crash events can be distinguished by means of a header that describes each crash.

A task crash followed by successful session recovery is indicated by this log message:

```
"Death notification of task <name>/<instance id> on <card#>/<cpu#> sent to parent
 task <parent name>/<instance id> with failover of <task name>/<instance id>
 on <card#>/<cpu#>"
```

A task crash that could not recover is indicated by this log message:

```
"Death notification of task <name>/<instance id> on <card#>/<cpu#> sent to parent
 task <parent name>/<instance id>"
```

In summary, with session recovery enabled, in most cases the crashes will not be noticed because they have no subscriber impact. One has to enter the CLI command, or look at the logs or SNMP notification in order to detect any occurrence of crashes.

For example:

```
 ******** show crash list *******
Tuesday May 26 05:54:14 BDT 2015
=== ==================== ======== ========== =============== ======================
#           Time         Process  Card/CPU/        SW          HW_SER_NUM
                                     PID         VERSION       MIO / Crash Card
=== ==================== ======== ========== =============== ======================

1   2015-May-07+11:49:25 sessmgr  04/0/09564 17.2.1          SAD171600WS/SAD172200MH
2   2015-May-13+17:40:16 sessmgr  09/1/05832 17.2.1          SAD171600WS/SAD173300G1
3   2015-May-23+09:06:48 sessmgr  03/1/31883 17.2.1          SAD171600WS/SAD1709009P
4   2015-May-25+15:58:59 sessmgr  09/1/16963 17.2.1          SAD171600WS/SAD173300G1
5   2015-May-26+01:15:15 sessmgr  04/0/09296 17.2.1          SAD171600WS/SAD172200MH

 ******** show snmp trap history verbose *******
 Fri May 22 19:43:10 2015 Internal trap notification 1099 (ManagerRestart) facility
 sessmgr instance 204 card 9 cpu 1
 Fri May 22 19:43:29 2015 Internal trap notification 73 (ManagerFailure) facility
 sessmgr instance 204 card 9 cpu 1
 Fri May 22 19:43:29 2015 Internal trap notification 150 (TaskFailed) facility
 sessmgr instance 204 on card 9 cpu 1
 Fri May 22 19:43:29 2015 Internal trap notification 151 (TaskRestart) facility
```

```
 sessmgr instance 204 on card 9 cpu 1
 Fri May 22 19:43:30 2015 Internal trap notification 183 (SessMgrRecoveryComplete)
 Slot Number 9 Cpu Number 1 fetched from aaa mgr 1755 prior to audit 1755 passed
 audit 1754 calls recovered 1754 all call lines 1754 time elapsed ms 1108.
 Fri May 22 19:43:32 2015 Internal trap notification 1099 (ManagerRestart) facility
 sessmgr instance 204 card 9 cpu 1
 Fri May 22 19:44:49 2015 Internal trap notification 73 (ManagerFailure) facility
 sessmgr instance 236 card 7 cpu 0
 Fri May 22 19:44:49 2015 Internal trap notification 150 (TaskFailed) facility
 sessmgr instance 236 on card 7 cpu 0
 Fri May 22 19:44:49 2015 Internal trap notification 151 (TaskRestart) facility
 sessmgr instance 236 on card 7 cpu 0
 Fri May 22 19:44:51 2015 Internal trap notification 183 (SessMgrRecoveryComplete)
 Slot Number 7 Cpu Number 0 fetched from aaa mgr 1741 prior to audit 1741 passed audit
 1737 calls recovered 1737 all call lines 1737 time elapsed ms 1047.
 Fri May 22 19:44:53 2015 Internal trap notification 1099 (ManagerRestart) facility
 sessmgr instance 236 card 7 cpu 0
 Fri May 22 19:50:04 2015 Internal trap notification 73 (ManagerFailure) facility
 sessmgr instance 221 card 2 cpu 1
: Fri May 22 19:50:04 2015 Internal trap notification 150 (TaskFailed) facility
 sessmgr instance 221 on card 2 cpu 1
 Fri May 22 19:50:04 2015 Internal trap notification 151 (TaskRestart) facility
 sessmgr instance 221 on card 2 cpu 1
Fri May 22 19:50:05 2015 Internal trap notification 183 (SessMgrRecoveryComplete)
 Slot Number 2 Cpu Number 1 fetched from aaa mgr 1755 prior to audit 1755 passed
 audit 1749 calls recovered 1750 all call lines 1750 time elapsed ms 1036.

 ******** show snmp trap history verbose *******
 Fri May 22 19:43:10 2015 Internal trap notification 1099 (ManagerRestart) facility
 sessmgr instance 204 card 9 cpu 1
 Fri May 22 19:43:29 2015 Internal trap notification 73 (ManagerFailure) facility
 sessmgr instance 204 card 9 cpu 1
 Fri May 22 19:43:29 2015 Internal trap notification 150 (TaskFailed) facility
 sessmgr instance 204 on card 9 cpu 1
 Fri May 22 19:43:29 2015 Internal trap notification 151 (TaskRestart) facility
 sessmgr instance 204 on card 9 cpu 1
 Fri May 22 19:43:30 2015 Internal trap notification 183 (SessMgrRecoveryComplete)
 Slot Number 9 Cpu Number 1 fetched from aaa mgr 1755 prior to audit 1755 passed
 audit 1754 calls recovered 1754 all call lines 1754 time elapsed ms 1108.
 Fri May 22 19:43:32 2015 Internal trap notification 1099 (ManagerRestart) facility
 sessmgr instance 204 card 9 cpu 1
 Fri May 22 19:44:49 2015 Internal trap notification 73 (ManagerFailure) facility
 sessmgr instance 236 card 7 cpu 0
 Fri May 22 19:44:49 2015 Internal trap notification 150 (TaskFailed) facility
 sessmgr instance 236 on card 7 cpu 0
 Fri May 22 19:44:49 2015 Internal trap notification 151 (TaskRestart) facility
 sessmgr instance 236 on card 7 cpu 0
 Fri May 22 19:44:51 2015 Internal trap notification 183 (SessMgrRecoveryComplete)
 Slot Number 7 Cpu Number 0 fetched from aaa mgr 1741 prior to audit 1741 passed
 audit 1737 calls recovered 1737 all call lines 1737 time elapsed ms 1047.
 Fri May 22 19:44:53 2015 Internal trap notification 1099 (ManagerRestart) facility
 sessmgr instance 236 card 7 cpu 0
 Fri May 22 19:50:04 2015 Internal trap notification 73 (ManagerFailure) facility
 sessmgr instance 221 card 2 cpu 1
: Fri May 22 19:50:04 2015 Internal trap notification 150 (TaskFailed) facility
 sessmgr instance 221 on card 2 cpu 1
 Fri May 22 19:50:04 2015 Internal trap notification 151 (TaskRestart) facility
 sessmgr instance 221 on card 2 cpu 1
Fri May 22 19:50:05 2015 Internal trap notification 183 (SessMgrRecoveryComplete
) Slot Number 2 Cpu Number 1 fetched from aaa mgr 1755 prior to audit 1755 passed
 audit 1749 calls recovered 1750 all call lines 1750 time elapsed ms 1036.

 ******** show logs  *******
2015-May-25+23:15:53.123 [sitmain 4022 info] [3/1/4850 <sitmain:31> sittask.c:4762]
 [software internal system critical-info syslog] Readdress requested for facility
 sessmgr instance 5635 to instance 114
```

```
2015-May-25+23:15:53.122 [sitmain 4027 critical] [3/1/4850 <sitmain:31>
 crash_mini.c:908] [software internal system callhome-crash] Process Crash Info:
 time 2015-May-25+17:15:52(hex time 556358c8) card 03 cpu 01 pid 27118 procname
 sessmgr  crash_details
Assertion failure at acs/acsmgr/analyzer/ip/acs_ip_reasm.c:2970
  Function: acsmgr_deallocate_ipv4_frag_chain_entry()
  Expression: status == SN_STATUS_SUCCESS
  Proclet: sessmgr (f=87000,i=114)
  Process: card=3 cpu=1 arch=X pid=27118 cpu=~17% argv0=sessmgr
  Crash time: 2015-May-25+17:15:52 UTC
  Recent errno: 11 Resource temporarily unavailable
  Stack (11032@0xffffb000):
    [ffffe430/X] __kernel_vsyscall() sp=0xffffbd28
    [0af1de1f/X] sn_assert() sp=0xffffbd68
    [0891e137/X] acsmgr_deallocate_ipv4_frag_chain_entry() sp=0xffffbde8
    [08952314/X] acsmgr_ip_frag_chain_destroy() sp=0xffffbee8
    [089d87d1/X] acsmgr_process_tcp_packet() sp=0xffffc568
    [089da270/X] acs_process_tcp_packet_normal_path() sp=0xffffc5b8
    [089da3fd/X] acs_tcp_analyzer() sp=0xffffc638
    [0892fb39/X] do_acsmgr_process_packet() sp=0xffffc668
    [08940045/X] acs_ip_lean_path() sp=0xffffc6b8
    [0887e309/X] acsmgr_data_receive_merge_mode() sp=0xffffc9d8
    [0887f323/X] acs_handle_datapath_events_from_sm_interface() sp=0xffffca08
    [037c2e1b/X] sessmgr_sef_initiate_data_packet_ind() sp=0xffffca88
    [037c2f50/X] sessmgr_pcc_intf_send_data_packet_ind() sp=0xffffcaf8
    [061de74a/X] sessmgr_pcc_fwd_packet() sp=0xffffcb58
    [0627c6a4/X] sessmgr_ipv4_process_inet_pkt_part2_slow() sp=0xffffcf68
    [06318343/X] sessmgr_ipv4_process_inet_pkt_pgw_ggsn() sp=0xffffd378
    [0632196c/X] sessmgr_med_ipv4_data_received() sp=0xffffd418
    [0633da9a/X] sessmgr_med_data_receive() sp=0xffffd598
    [0afb977c/X] sn_epoll_run_events() sp=0xffffd5e8
    [0afbdeb8/X] sn_loop_run() sp=0xffffda98
    [0ad2b82d/X] main() sp=0xffffdb08

2015-May-25+23:15:53.067 [rct 13038 info] [5/0/7174 <rct:0> rct_task.c:305]
 [software internal system critical-info syslog] Death notification of task
 sessmgr/114 on 3/1 sent to parent task sessctrl/0 with failover of sessmgr/5635 on 3/1
2015-May-25+23:15:53.065 [evlog 2136 info] [5/0/7170 <evlogd:0> odule_persist.c:3102]
 [software internal system critical-info syslog] Evlogd crashlog: Request received to
 check the state of persistent crashlog.
2015-May-25+23:15:53.064 [sitmain 4099 info] [3/1/4850 <sitmain:31> crash_mini.c:765]
 [software internal system critical-info syslog] have mini core, get evlogd status for
 logging crash file 'crashdump-27118'
2015-May-25+23:15:53.064 [sitmain 4017 critical] [3/1/4850 <sitmain:31> sitproc.c:1544]
 [software internal system syslog] Process sessmgr pid 27118 died on card 3 cpu 1
 signal=6  wstatus=0x86
2015-May-25+23:15:53.048 [sitmain 4074 trace] [5/0/7168 <sitparent:50> crashd.c:1130]
 [software internal system critical-info syslog] Crash handler file transfer starting
 (type=2 size=0 child_ct=1 core_ct=1 pid=23021)
2015-May-25+23:15:53.047 [system 1001 error] [6/0/9727 <evlogd:1> evlgd_syslogd.c:221]
 [software internal system syslog] CPU[3/1]:  xmitcore[21648]: Core file transmitted to
 card 5 size=663207936 elapsed=0sec:908ms
2015-May-25+23:15:53.047 [system 1001 error] [5/0/7170 <evlogd:0> evlgd_syslogd.c:221]
 [software internal system syslog] CPU[3/1]:  xmitcore[21648]: Core file transmitted to
 card 5 size=663207936 elapsed=0sec:908ms
2015-May-25+23:15:53.047 [sitmain 4080 info] [5/0/7168 <sitparent:50> crashd.c:1091]
 [software internal system critical-info syslog] Core file transfer to SPC complete,
 received 8363207936/0 bytes

 ******** show session recovery status verbose *******
Tuesday May 26 05:55:26 BDT 2015
Session Recovery Status:
  Overall Status        : Ready For Recovery
  Last Status Update    : 8 seconds ago

              ----sessmgr---  ----aaamgr----  demux
```

```
 cpu state    active standby  active standby  active  status
---- -------  ------ -------  ------ -------  ------  -------------------------
 1/0 Active   24     1        24     1        0       Good
 1/1 Active   24     1        24     1        0       Good
 2/0 Active   24     1        24     1        0       Good
 2/1 Active   24     1        24     1        0       Good
 3/0 Active   24     1        24     1        0       Good
 3/1 Active   24     1        24     1        0       Good
 4/0 Active   24     1        24     1        0       Good
 4/1 Active   24     1        24     1        0       Good
 5/0 Active   0      0        0      0        14      Good (Demux)
 7/0 Active   24     1        24     1        0       Good
 7/1 Active   24     1        24     1        0       Good
 8/0 Active   24     1        24     1        0       Good
 8/1 Active   24     1        24     1        0       Good
 9/0 Active   24     1        24     1        0       Good
 9/1 Active   24     1        24     1        0       Good
10/0 Standby  0      24       0      24       0       Good
10/1 Standby  0      24       0      24       0       Good
```

Crash Logging Architecture

Crash logs record all possible information that pertain to a software crash (full core dump). Due to their size, they cannot be stored in system memory. Therefore, these logs are only generated if the system is configured with a URL that points to a local device or a network server where the log can be stored.

The crash log is a persistent repository of crash event information. Each event is numbered and contains text associated with a CPU (minicore), network processing unit (NPU), or kernel crash. The logged events are recorded into fixed length records and stored in /flash/crashlog2.

Whenever a crash occurs, this crash information is stored:

1. The event record is stored in /flash/crashlog2 file (the crash log).
2. The associated minicore, NPU, or kernel dump file is stored in the /flash/crsh2 directory.
3. A full core dump is stored in a user configured directory.

Synchronization of Crash Events and Minicores between Management Cards

The crashlog is unique to each of the management cards, so if a crash occurs when card "8" is active it will be logged on card "8". A subsequent switchover would no longer display the crash in the log. In order to retrieve this crash, a switch back over to card "8" has to be done. The crash event log and dumps are unique to active and standby management cards, so if a crash occurs on an active card then the crash event log and related dumps will be stored on an active card only. This crash information is not available on the standby card. Whenever the cards switchover due to a crash in the active card, and crash information is no longer displayed on the card which takes over, crash information can be retrieved only from the current active card. In order to retrieve the crash list of the other card, a switchover is required again. In order to avoid this switchover and to obtain the crash information from the standby card, synchronization between two management cards and maintainence of the latest crash information is required.

The arriving crash event will be sent over to the standby SMC/MIO and saved in the standby's crashlog file in the similar manner. Minicore, NPU, or kernel dumps on flash of active SMC/MIO needs to be synchronized to standby SMC/MMIO with the **rsync** command. When a crashlog entry or the whole list is deleted through the CLI command, it should be erased on both active and standby SMCs/MIOs. There is no impact on memory. All the crash related synchronization activity will be done by the evlogd of the standby SMC/MIO card, as the standby evlogd is less loaded and the standby card has enough room for synchronization activity. Therefore

the performance of the system will not be affected.

Commands

These commands can be used in order to troubleshoot issues:

```
#show support details

#show crash list

#show logs

#show snmp trap history verbose

#show session recovery status verbose

#show task resources facility sessmgr instance <>

#show task resources facility sessmgr all
```

Corefiles are generated after a crash. Usually operators store them in an external server. The corefile name usually looks like crash-<Cardnum>-<CPU Num>-<Hex timestamp>-coree.gcrash-09-00-5593a1b8-core.

Whenever a crash occurs, this crash information is stored:

- The event record is stored in /flash/crashlog2 file (the crash log).
- The associated minicore, NPU, or kernel dump file is stored in the /flash/crsh2 directory.

Summary

All of the ASR5x00 software is designed to handle both foreseen conditions/events and unforeseen conditions/events. While Cisco strives to have perfect software, inevitably mistakes will exist and crashes will be possible. That is why the session recovery feature is so important. Cisco's strive for perfection will minimize the occurrences of crashes, and session recovery will allow the sessions to continue after a crash. Nonetheless, it is important that Cisco continues to strive to achieve perfect software. Fewer crashes will reduce the likelihood of multiple crashes that happen simultaneously. While session recovery seamlessly heals a single crash, the recovery from multiple simultaneous crashes is designed a bit differently. Operators should rarely (or never) experience multiple simultaneous crashes, but if such were to occur, the ASR5x00 is designed to recover system integrity as the highest priority, possibly at the sacrifice of some subscriber sessions.

Updated: Aug 25, 2015 Document ID: 119224