ALL ABOUT CORES in ASR5500/ASR5000

What are core files?

A core file is created when a program terminates unexpectedly, due to a bug, or a violation of the operating system's or hardware's protection mechanisms. The operating system kills the program and creates a core file.

When does it get generated on ASR5x00?

A core file gets generated on the ASR5x00 under two circumstances listed below.

1. A process crashes generating a core.
2. A core is forced to be generated using cli.

How is it useful for TAC/BU ?

After the OS kills the program/process and creates the core file, it can be used by the programmers to figure out what went wrong. It contains a detailed description of the state that the program/process when it died/crashed. This information is used by the engineering team to re-create the issue in the local lab to find the issue and fix it or figure out the cause for the program/process to crash.

What is the configuration for getting a good core file ?

In the unlikely event of a software crash, the system stores information that could be useful in determining the reason for the crash. This information can be maintained in system memory or it can be transferred and stored on a network server.

Crash logs record all possible information pertaining 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 Universal Resource Locator (URL) pointing to a local device or a network server where the log can be stored.

Whenever a crash occurs, the following 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.
Important: The crashlog2 file along with associated minicore, NPU and kernel dumps are automatically synchronized across redundant management cards (SMC, MIO/UMIO). Full core dumps are not synchronized across management cards.
The following behaviors apply to the crash logging process.
· When a crash event arrives on an active management card, the event record is stored in its crashlog2 file along with the minicore, NPU, or kernel dump file in /flash/crsh2. The crash event and dump file are also automatically stored in the same locations on the standby management card.
· When a crash log entry is deleted via CLI command, it is deleted on both the active and standby management cards.
· When a management card is added or replaced, active and standby cards will automatically synchronize crash logs and dump files.
· When a crash event is received and the crash log file is full, the oldest entry in the crash log and its related dump file will be replaced with the latest arrived event and dump file on both management cards. Information for a maximum of 120 crash events can be stored on management cards.
· Duplicate crash events bump the count of hits in the existing record and update the new record with the old crash record. Additions to the count use the timestamp for the first time the event happened.

Crash log files (full core dumps) are written with unique names as they occur to the specified location. The name format is **crash-card-cpu-time-core**. Where card is the card slot, cpu is the number of the CPU on the card, and time is the Portable Operating System Interface (POSIX) timestamp in hexadecimal notation.

Use the following example to configure a software crash log destination in the Global Configuration mode:

```
configure
 crash enable [ encrypted ] url crash_url
 end
```

```
[local]lab# show configuration|grep -i crash
  crash enable encrypted url
+A0mkhq8vovt9r839m92n24ujeve3cvcauqg6n3od1n8zaed1czuwg3jqh2ocwct8t92w1dktko63ief1z3
ib8g595xaj2ukt1u9mpscd5
[local]lab#
```

How to force a core file ?

A core can be forced to be generated by using the below command after getting into the hidden mode. Please contact the Cisco Support team to get assistance to generate the core file.

Below is an example.

```
[local]lab# show crash list
==          ====          =======  ========== ========== ================
#           Time          Process   Card/CPU/     SW         HW_SER_NUM
                                        PID      VERSION   VPO / Crash Card
==          ====          =======  ========== ========== ================

1  2015-May-21+12:47:54 vpnmgr   01/0/03360 15.0(51852) NA  >>> Generated crash

Total Crashes : 1
```

Impact of forcing a core file ?

There is no business impact of forcing a core. By forcing a core we are getting a snapshot of the current state of the system process.

Caveats and solution

Core files created from a crash on an ASR5000/ASR5500 sometimes become corrupted due to the size of the core file being created.

By default the maximum size of a core file is 1024 megabytes (1 Gigabyte) in ASR5000 and 2048 MB in ASR5500. This means that when the core file is being created in the location specified, the creation of the core file will be terminated if the core file size is larger than 1024 Megabytes and 2048 MB on the ASR5000 and ASR5500 respectively. This can be changed with a simple hidden configuration. Please contact Cisco Support for the related configuration. This configuration basically allows the operator to increase the maximum size of the core file to 2048 MB and 4096MB respectively in the ASR5000 and ASR5500 respectively.

When core files are written to let's say an EMS server, upon creation, core files are automatically gzipped, but are written without the .gz extension. On the EMS server, after a set period of time, the core files are moved to an archived directory, where the core file is gzipped again, but this time has the .gz extension appended.

How to check the sanity of a core file?

Testing of core files can be facilitated in a simple manner. Your testing is based upon whether the core file that is supplied to you is in a .gz format, or without the .gz format.

Procedure

In this instance the core file has no .gz extension, but as stated, was compressed when written to the EMS server.

```
$ ls -ltr
-rwx------+ 1 Administrators Domain Users 6967918 Mar  2 08:51 crash-02-01-53133543-
core <crash-card-cpu-time-core>
```

1) Copy the core to a temp directory for posterity.

2) Rename the core file adding a .gz extension to the file.

```
$ mv <crash-card-cpu-time-core> <crash-card-cpu-time-core>.gz
```

3) gunzip (unzip) the core file. If no errors are seen, the core file should not be corrupted, providing there is no internal corruption.

```
$ gunzip <crash-card-cpu-time-core>.gz
(no error returned)
```

Using a file that is known to be corrupted the following is seen.

```
$ ls -lt
-rwx------+ 1 Administrators Domain Users 98467840 Feb 27 12:43 <crash-card-cpu-time-
core>.gz

$ gunzip <crash-card-cpu-time-core>.gz
gzip: <crash-card-cpu-time-core>.gz:                    unexpected end of
file  <<<<<<
```

If you have an error returned when gunzipping the core file, it is corrupted.

If the core file has a .gz extension, you will first have to gunzip the file, rename it adding on the .gz extension again, and then gunzip it a second time for testing of corruption.

Core File

```
$ ls -lt
-rwx------+ 1 Administrators Domain Users 94981317 Feb 27 12:43 <crash-card-cpu-time-
core>.gz
```

First gunzip:

```
$ gunzip <crash-card-cpu-time-core>.gz
```

Rename the file adding on the .gz extension

```
$ mv <crash-card-cpu-time-core <crash-card-cpu-time-core>.gz
```

View the core file

```
$ ls -ltr
-rwx------+ 1 Administrators Domain Users 98467840 Feb 27 12:43 <crash-card-cpu-time-
core>.gz
```

Gunzip the file second time

```
$ gunzip <crash-card-cpu-time-core>.gz
gzip: <crash-card-cpu-time-core>.gz:             unexpected end of file  <<<<<
```