Ascend 310B
23.0.RC1
黑匣子日志参考

文档版本	01
发布日期	2023-05-08

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://e.huawei.com

目 录

1 简介

为了增强昇腾AI处理器系统功能的可维护性，需要一套黑匣子机制，保证系统在发生异常时能保存必要的软硬件参数信息，方便系统故障的诊断分析，快速实现问题定位。

- Atlas 200I A2 加速模块
- Atlas 200I DK A2 开发者套件

2 配置介绍

Ascend RC 场景

黑匣子配置文件地址：/var/log/npu/conf/bbox/bbox.conf。配置项说明如下所示。

表 2-1 配置项说明

配置项	配置项说明
MNTN_PATH=/var/log/npu/hisi_logs	黑匣子日志存储路径。路径中不能包含/../相对路径，路径长度不能超过64 Bytes。
MNTN_LOGSPACE_SIZE=64	每个SOC Chip所有日志文件总大小。单位：MB。范围：10MB~10GB。

相关说明

- 修改配置文件后，需要重启log-daemon进程才会生效。
- 当日志存储路径MNTN_PATH下的日志文件总大小超过配置的日志空间MNTN_LOGSPACE_SIZE大小时，黑匣子会进行老化，删除时间最老的时间戳目录文件夹，直到总大小不超过MNTN_LOGSPACE_SIZE。
- 当配置路径的权限log-daemon进程无法访问时，黑匣子无法读写。需要用户修改路径权限或者更换路径。

3 日志介绍

本节介绍黑匣子日志文件如何使用，可以参考以下步骤操作。

步骤1 进入黑匣子日志存放目录。

Ascend RC场景下，命令示例如下：

cd /var/log/npu/hisi_logs/device-id/

说明

id表示Device设备的ID号。/var/log/npu/hisi_logs/为默认的黑匣子日志存放路径，用户可通过配置文件修改这个路径，配置文件介绍请参见**配置介绍**。

步骤2 查看history.log文件。

找到出现异常的设备device-id子目录，查看该目录下history.log日志。history.log日志格式及说明：

```
[2020-02-27-19:18:46.142527] system exception code [0x68020002]: ModuleName [DRIVER], ExceptionReason [DEVICE_HBL_EXCEPTION]，TimeStamp [20200227191842-300803183].
```

表 3-1 history.log 文件内容字段及含义

字段	含义
[2020-02-27-19:18:46.142527]	异常文件落盘的时间。
system exception code [0x68020002]	模块上报的异常码 [0x68020002]。
ModuleName [DRIVER]	异常上报的模块名称 [驱动]。
ExceptionReason [DEVICE_HBL_EXCEPTION]	模块上报的异常原因 [设备心跳丢失]。
TimeStamp [20200227191842-300803183]	异常上报的时间戳 [20200227191842-300803183]。

说明

history.log到达30000条时会启动日志老化，删除最早的20000条信息，及对应的文件夹。

步骤3 查看具体模块异常日志。

根据异常上报时间戳信息（如TimeStamp[20200227191842-300803183]）打开日志目录，目录名称即为时间戳（20200227191842-300803183），模块的具体异常信息保存在该目录下。文件说明如下所示。

表 3-2 日志文件路径及内容说明（昇腾 310B AI 处理器）

文件相对路径	文件内容
DONE	黑匣子日志记录状态。
bbox	黑匣子静态预留空间维测数据目录。
bbox/bbox_info.txt	记录黑匣子基本信息。
bbox/[module].txt	记录模块[module]的异常信息，如ts.txt。
bbox/kbox.txt	记录部分Kernel日志和内核信息，如堆栈信息。
bbox/os	OS维测信息。
bbox/os/os_info.txt	记录OS基本信息。
bbox/os/regs	记录OS关注的寄存器信息。
bbox/os/regs/reset_regs.txt	记录复位寄存器信息。
log	各类日志目录。
log/imu_boot.log	记录IMU启动日志。
log/imu_run.log	记录IMU运行日志。
log/uefi_boot.log	记录UEFI启动日志。
log/kernel.log	记录OS内核日志信息。
log/ts.log	记录TS日志信息。
mntn	模块独立维测数据目录。
mntn/ddr_mntn.txt	记录DDR维测数据。
mntn/bios_mntn.txt	记录BIOS维测数据。
mntn/pmu.reg	记录PMU寄存器信息。
mntn/tsensor.reg	记录Tsensor寄存器信息。
snapshot	快照信息目录。
snapshot/hdr.log	记录快照信息。
snapshot/hdr_status.txt	记录快照打点信息。

----结束

须知

- Device侧
  - 当黑匣子日志落盘过程中出现log-daemon进程异常时，日志内容不可控，存在丢失风险。
  - 当黑匣子日志存放路径（/var/log/npu/hisi_logs目录）所在磁盘空间不足时，无法生成黑匣子日志。
  - 当日志路径下的总文件大小超过MNTN_LOGSPACE_SIZE的配置值时，系统将循环删除时间最早的文件。
- Host侧，DONE文件记录3种状态：
  - STARTING：异常上报流程处理中，异常日志正在导出。
  - FILEDONE：异常上报流程处理完成，异常日志导出正常，异常信息保存完整。
  - PROCFAIL：异常上报流程处理完成，异常日志导出失败，异常信息保存不完整。

4 快照介绍

本章节描述snapshot目录中的快照功能相关文件，快照日志（hdr.log）。

4.1 快照日志内容

快照日志的内容可以分为头部信息区（head info）、启动区（boot region）、运行区（run region）三大块。

格式顺序为：

```
head info
boot region
   |--region config
   |--region control
   |--area 0
   |--area 1
   |·····
   └-area 7
run region
   |--region config
   |--region control
   |--area 0
   |--area 1
   |·····
   └-area 7
```

4.1.1 快照头部内容

快照头部信息内容：

```
====================head info======================
magic                    : 0xeaea2020
version                  : 0x100
reset count              : 0x97
```

表 4-1 快照头部信息字段及含义

字段	含义
magic	用来识别快照功能的魔鬼数字，固定值为：0xeaea2020。
version	版本号。例如当前为1.0。
reset count	当前环境热复位计数。

4.1.2 快照启动区和运行区内容

启动区和运行区结构相同，分为控制信息与数据两部分。

4.1.2.1 控制信息

启动区控制信息内容：

```
====================boot region====================
region offset                : 0x400
region size                  : 0x4b000

--------------------region config-------------------
total area                   : 0xa
history area                 : 0x5
error area                   : 0x2
area config:
  used module count          : 0x4

module config:
  module 0 offset            : 0x0
  module 0 size              : 0x3000

  module 1 offset            : 0x3000
  module 1 size              : 0x1000

  module 2 offset            : 0x4000
  module 2 size              : 0x1000

  module 3 offset            : 0x5000
  module 3 size              : 0x1000

--------------------region control------------------
area index                   : 0x1
error area count             : 0x2
```

运行区控制信息内容：

```
====================run region=====================
region offset                : 0x4b400
region size                  : 0x25800

--------------------region config-------------------
total area                   : 0xa
history area                 : 0x5
error area                   : 0x2
area config:
  used module count          : 0x4

module config:
  module 0 offset            : 0x0
  module 0 size              : 0x800
```

```
 module 1 offset              : 0x800
 module 1 size                : 0x800

 module 2 offset              : 0x1000
 module 2 size                : 0x1000

 module 3 offset              : 0x2000
 module 3 size                : 0x1000

--------------------region control------------------
area index                    : 0x4
error area count              : 0x0
```

表 4-2 字段及含义

分区	字段	含义
boot region	region offset	当前区域相对于快照区起始地址的偏移。
	region size	当前数据区大小。
region config	total area	可以储存数据的区域总数。
	history area	用于储存历史数据的区域数。
	error area	用于储存异常数据的区域数。
	used module count	每个区域中包含的模块数。
	module config	每个模块对应的偏移和大小。偏移值相对于所在区域的起始地址。
region control	area index	用于记录本次热复位数据的区域编号。
	error area count	本次启动时检测到的未被导出的异常数量。
	area N control info	包含每个区域的控制信息。当前只使用7个： ● 0~4用于历史队列。 ● 5、6用于异常队列。
	flag	本区域队列类型： ● 0：未使用。 ● 1：BIOS标识的历史队列。 ● 2：BIOS标识的异常队列。 ● 3：DDR历史队列。 ● 4：DDR异常队列。
	tag	本区域信息状态： ● 0：未使用。 ● 1：使用，初始化。 ● 2：使用，无异常。 ● 3：使用，有异常。

分区	字段	含义
	exception type	异常类型： • 启动区显示STARTUP_EXCEPTION（0x2c）。 • 运行区显示last reset reason。
	module id	模块ID： • 启动区显示module id。 • 运行区无该字段。
	exception id	异常码： • 启动区显示exception id。 • 运行区无该字段。 具体异常码信息请参见产品的《黑匣子错误码信息列表》。
	reset number	记录本次信息时的热复位计数。

说明

- 7个area N control info信息块被分为两个队列。0~4用于历史队列，5、6用于异常队列。历史队列遵循循环覆盖原则，覆盖使用当前的队列。异常队列遵循读清原则，只有将内容读取后，才会将队列清空重新使用。
- 快照只有在区域的error area count非零时才会被黑匣子导出。导出后，本区域对应的error area count会被清零。再次热复位后，如果无新的异常记录，则不会导出快照。
- 启动区异常，需要关注三个数据： module id、exception type、exception id。运行区只需关注exception type。
- 异常码为0xA8**EFFF的STARTUP_EXCEPTION或RUN_EXCEPTION记录，为模块默认快照异常码，代表该模块不支持快照，但BIOS启动时检测到了该模块出现过异常。更多异常类型介绍请参考**A.1 异常类型说明**。

4.1.2.2 数据

昇腾310B AI处理器启动区包含的记录信息：BIOS、DDR、TEE、ATF四个模块

昇腾310B AI处理器运行区包含的记录信息：TEE、ATF、LPM、OS四个模块

所有模块信息有相同的头信息结构，头信息之后是模块各自记录的内容。

表 4-3 数据字段及含义

字段	含义
magic	识别用魔鬼数字。模块自己定义。
version	版本号。模块自己定义。
module id	模块ID。

字段	含义
is used	本区域是否有被使用（记录数据）。 ● 0：未使用。后续数据不会被输出。 ● 1：使用。
err code	异常码。
reason	具体异常原因。
hot reset index	记录本模块信息时的热复位计数。

说明

area N的control info中记录的reset number与area N的数据区中模块记录的hot reset index值必定相等。不相等代表模块写数据的区域有误。

4.2 快照导出

Ascend RC 场景

快照日志存在主动导出方式：设备热复位时，若快照数据中有异常信息，Device侧黑匣子会上报快照数据至Host侧，Host会解析数据成明文并落盘成文件。

标志为：/var/log/npu/hisi_logs/device-0/history.log中记录有启动异常（STARTUP_EXCEPTION）或运行异常（RUN_EXCEPTION）。

说明

- 设备启动异常和心跳异常，为被动导出，不会存在实时的控制数据，需要热复位后填写。
- 快照导出的判定条件为：控制信息中error area count不为0，且控制信息队列中有flag=4。
- 快照上报为启动异常还是运行异常的判定条件为：热复位计数最小的异常。
- Ascend RC场景下，日志记录在/var/log/npu/hisi_logs/device-0/<时间戳目录>/snapshot/hdr.log。

4.3 快照异常判定

以下异常判定的快照文件都是通过主动导出生成的。

4.3.1 快照日志异常判定

1. 分别查看启动区（boot region）->控制信息（region control） 和运行区（run region）->控制信息（region control） 中的“error area count”字段的值。
   - 0：表示没有异常。
   - 1：表示area 5有异常。
   - 2：表示area 5和area 6有异常。
2. 查看有异常的区域（area 5或者area 6）下面各模块的信息（如BIOS INFO）。

- “is used”的值为非0x1，表示该模块未启用快照日志功能。
- “is used”的值为0x1，“err code”为0x0，表示该模块无异常。
- “is used”的值为0x1，“err code”为非0x0，表示该模块有异常。

3. 根据“err code”的值查询产品的《黑匣子错误码信息列表》，可以找到具体的异常描述。

A 附录

A.1 异常类型说明

说明

如下是系统已经支持的所有异常类型说明。当系统出现异常时，根据昇腾AI处理器的型号会返回对应场景的异常类型。

表 A-1 异常类型说明

异常类型	异常类型值	异常描述
DEVICE_COLDBOOT	0x0	冷启动，无异常启动。
BIOS_EXCEPTION	0x1	bios启动异常，前一次启动bios异常。
DEVICE_HOTBOOT	0x2	按键热复位。
ABNORMAL_EXCEPTION	0x10	未感知到的硬件异常，如DDR总线异常。
TSENSOR_EXCEPTION	0x1f	soc温保复位。
PMU_EXCEPTION	0x20	PMU过流、欠压、过温引起的硬件复位。
DDR_FATAL_EXCEPTION	0x22	DDR Fatal异常复位，如DDR颗粒超温复位。
OS_PANIC	0x24	Panic，如访问非法地址。
OS_OOM	0x2a	OOM异常。
OS_COMM	0x2b	通信异常。
STARTUP_EXCEPTION	0x2c	模块启动异常。
HEARTBEAT_EXCEPTION	0x2d	模块心跳异常。

异常类型	异常类型值	异常描述
RUN_EXCEPTION	0x2e	模块运行异常。
LPM_EXCEPTION	0x32	LPM异常。
TS_EXCEPTION	0x33	TS异常。
DVPP_EXCEPTION	0x35	DVPP异常。
DRIVER_EXCEPTION	0x36	DRIVER异常。
TEE_EXCEPTION	0x38	TEEOS异常。
LPFW_EXCEPTION	0x39	LPFW异常。
NETWORK_EXCEPTION	0x3a	NETWORK异常。
HSM_EXCEPTION	0x3b	HSM异常。
ATF_EXCEPTION	0x3c	ATF异常。
TOOLCHAIN_EXCEPTION	0x3f	TOOLCHAIN异常。
DEVICE_LTO_EXCEPTION	0x8a	设备启动超时。
DEVICE_HBL_EXCEPTION	0x8b	设备心跳丢失。
DEVICE_HDC_EXCEPTION	0x8e	HDC连接异常。

A.2 Stackcore 文件解析定位

A.2.1 Stackcore 文件说明

Stackcore文件主要是用来保存链接了libstackcore.so的进程（如slogd）的堆栈信息，当进程出现异常时，能够从Stackcore文件中定位出问题点。

Stackcore文件的获取方式如下：

推理场景（Ascend RC）

Stackcore文件存放在运行环境的“/var/log/npu/coredump”路径下（该路径可在**slog.conf**日志配置文件上配置StackCorePath字段修改，详细介绍请参见《日志参考》的“日志配置文件”章节），需要执行**scp**命令将该路径下的Stackcore文件拷贝到开发环境。

说明

- 堆栈深度限制：堆栈信息只打印堆栈从上往下20层，超过20层只打印20层数据。
- 文件大小：20层 * 512Byte + 2KByte（其余及预留）（12KByte）。

A.2.2 Stackcore 文件解析

工具约束说明

使用Stackcore文件解析功能，需要满足以下约束条件：

- 仅支持aarch64压fp的场景，链接了libstackcore.so的进程的异常堆栈信息导出。
- stackview.sh依赖readelf进行文件信息的获取、依赖addr2line进行堆栈函数名和行号的解析，两者都是linux系统自带工具，请确保readelf、addr2line已安装且执行该脚本的用户有权限执行。

文件解析方式

通过stackview.sh脚本调用Linux系统自带的addr2line功能对Stackcore文件进行异常堆栈信息解析，命令行格式如下：

bash stackview.sh -c *{stackcore_file_path/stackcore_file_name}* **-p** *binary_file_path*

- **stackview.sh**脚本存放在{install_path}/toolkit/tools/stacktrace/scripts目录下。
- *stackcore_file_path/stackcore_file_name*：stackcore文件所在目录及文件名。其中*stackcore_file_name*命名方式为stackcore.<execname>.<pid>.<signo>.<timestamp>
  - execname：可执行文件名。
  - pid：进程编号。
  - signo：信号值。
  - timestamp：时间戳。
- *binary_file_path*：可执行文件及依赖的so文件所在目录。
  - 可执行文件需要从产生该stackcore文件的Device侧获取，如果不是从Device侧环境上获取，可能会解析失败。如果Device侧的可执行文件编译时进行了strip，需要取相同版本没有strip的可执行文件代替。
  - so文件所在目录可以在可执行文件所在目录下执行**ldd** *binary_file(可执行文件名)命令*获取。如图A-1所示。

    图 A-1 so 文件目录获取

    ```
    @szvphicpra57061:~/workspace/test$ ldd slogd
    linux-vdso.so.1 =>  (0x00007ffc71a9e000)
    libachk.so => /lib/libachk.so (0x00007f6cb8561000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f6cb8197000)
    libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f6cb7f93000)
    libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f6cb7d76000)
    librt.so.1 => /lib/x86_64-linux-gnu/librt.so.1 (0x00007f6cb7b6e000)
    /lib64/ld-linux-x86-64.so.2 (0x0000559b8ae66000)
    ```

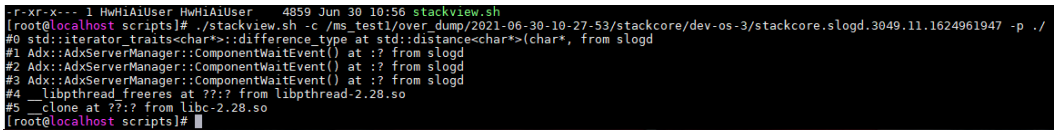
  - 需要将可执行文件和so文件从Device侧拷贝到host侧并放在同一目录下。

文件解析示例

请使用安装时配置的运行用户执行命令，命令示例如下，解析结果示例如图A-2。

bash stackview.sh -c *${HOME}/stackcore/dev-os-3/stackcore.slogd.3049.11.16249611947* **-p** *./*

图 A-2 Stackcore 文件解析结果示例



打印内容分别是函数名、文件名、行号。

说明

- 当代码中存在内联函数或宏定义函数，则这2类函数不会打印堆栈调用关系。
- 如果解析结果是??:?，原因为用户使用了非调试版本的可执行文件（即编译时进行了strip）作为stackview.sh的输入。

信号情况列表

信号名称	信号值	信号描述	stackcore文件
SIGABRT	6	Abort signal from abort	支持
SIGBUS	7	Bus error (bad memory access)	支持
SIGFPE	8	Floating-point exception	支持
SIGILL	4	Illegal Instruction	支持
SIGIOT	6	IOT trap. A synonym for SIGABRT	支持
SIGQUIT	3	Quit from keyboard	支持
SIGSEGV	11	Invalid memory reference	支持
SIGSYS	31	Bad system call (SVr4)	支持
SIGTRAP	5	Trace/breakpoint trap	支持
SIGUNUSED	31	Synonymous with SIGSYS	支持
SIGXCPU	24	CPU time limit exceeded	支持
SIGXFSZ	25	File size limit exceeded	支持

异常情况

执行stackview.sh脚本后，显示空行，无结果数据。出现这种情况的可能原因是stackcore内容不正确，例如文件内容无效，堆栈为空等。

