Atlas 300I 推理卡
1.0.11

黑匣子日志参考（型号 3000, 3010）

文档版本	01
发布日期	2021-07-31

华为技术有限公司

版权所有 © 华为技术有限公司 2022。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明

HUAWEI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 黑匣子日志参考（Linux 系统）.....1
1.1 简介.....1
1.2 配置介绍.....1
1.3 日志介绍.....2
1.4 快照日志介绍.....5
1.4.1 快照日志内容.....5
1.4.1.1 快照头部内容.....5
1.4.1.2 快照启动区和运行区内容.....5
1.4.1.2.1 控制信息.....5
1.4.1.2.2 数据.....8
1.4.2 快照导出.....9
1.5 附录.....9
1.5.1 msnpureport 工具使用.....10
1.5.2 连续导出 Device 侧的日志和文件.....14
1.5.3 异常类型说明.....16
1.5.4 Stackcore 文件解析定位.....17
1.5.4.1 Stackcore 文件说明.....17
1.5.4.2 Stackcore 文件解析.....17
1.6 FAQ.....19
1.6.1 kernel log 解析失败.....20
2 黑匣子日志参考（Windows 系统）.....21
2.1 简介.....21
2.2 日志介绍.....21
2.3 快照日志介绍.....24
2.3.1 快照日志内容.....24
2.3.1.1 快照头部内容.....24
2.3.1.2 快照启动区和运行区内容.....25
2.3.1.2.1 控制信息.....25
2.3.1.2.2 数据.....27
2.3.2 快照导出.....28
2.4 附录.....28
2.4.1 msnpureport 工具使用.....29

2.4.2 异常类型说明 33

1 黑匣子日志参考（Linux 系统）

1.1 简介
1.2 配置介绍
1.3 日志介绍
1.4 快照日志介绍
1.5 附录
1.6 FAQ

1.1 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。目前，支持黑匣子功能的模块有：

- 昇腾310 AI处理器：包括BIOS、LPM、TEE、ATF、TS、Driver、DVPP、OS。

1.2 配置介绍

Ascend EP 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 1-1 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=10	每个SOC Chip所有日志文件可占用的空间大小值。单位：MB。范围：10MB~10GB。

版权所有 © 华为技术有限公司

Ascend RC 场景（Atlas 200 DK）

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 1-2 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所有日志文件可占用的空间大小值。单位：MB。范围：10MB~10GB。

相关说明

- Ascend EP场景下，该配置文件存放在Device侧，用户需要有登录Device侧的权限，才能修改配置文件。
- 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。

1.3 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。

黑匣子日志是通过msnpureport工具导出的，导出方法请参见**1.5.1 msnpureport工具使用**。存放目录是运行msnpureport工具的所在路径（如“/var/log/npu/report”）。

命令示例如下：

cd /var/log/npu/report/2020-10-20-03-12-05/hisi_logs

说明

- "/var/log/npu/report/"：运行msnpureport工具的所在路径，请根据实际情况修改。
- 2020-10-20-03-12-05：运行msnpureport工具生成的时间戳目录，请根据实际情况修改。
- hisi_logs下面存在*device-id*文件夹及device_info.txt文件
  - *device-id*表示Device设备的ID号，保存对应设备的异常信息。
  - device_info.txt记录了设备信息。

Ascend RC场景（Atlas 200 DK）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 1-3 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。
- Ascend EP场景下，对于 ModuleName [AP], ExceptionReason [OS_OOM]异常，history.log中TimeStamp记录的是运行导出工具时获取到的时间戳，实际异常上报的时间戳需要在步骤3中的bbox/bbox_info.txt内查看。

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 1-4 日志文件路径及内容说明（昇腾 310 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，如挂死任务栈。

文件相对路径	文件内容
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/kernel.log	记录OS内核日志信息。
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。

----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
  - Ascend EP场景下，运行msnpureport工具的所在路径（如“/var/log/npu/report”）的磁盘空间不足时，无法生成黑匣子日志。
  - 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。

1.4 快照日志介绍

1.4.1 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
```

1.4.1.1 快照头部内容

快照头部信息内容：

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

表 1-5 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	当前环境热复位次数。

1.4.1.2 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。

1.4.1.2.1 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 1-6 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。

分区	字段	含义
	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：L2BUF 历史队列。 ● 2：L2BUF 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。
	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。
	reset number	记录本次信息时的热复位计数。

说明

1. 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。
2. 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。
3. 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。
4. 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**1.5.3 异常类型说明**。

1.4.1.2.2 数据

启动区包含的记录信息如下。

- 昇腾310 AI处理器：BIOS、DDR、TEE、ATF四个模块

运行区包含的记录信息如下。

- 昇腾310 AI处理器：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 1-7 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。
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。0为本模块没有异常。
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。

1.4.2 快照导出

Ascend EP 场景

快照日志存在以下导出方式：

- 被动导出
  - msnpureport工具检测到Device设备启动异常后，导出快照。

    标志为：导出路径+/hisi_logs/device-x/history.log中记录有设备丢失异常（DEVICE_LTO_EXCEPTION）。
  - msnpureport工具检测到Device设备心跳丢失异常后，导出快照。

    标志为：导出路径+/hisi_logs/device-x/history.log中记录有设备心跳异常（DEVICE_HBL_EXCEPTION）。
- 主动导出

  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用户态进程，用户态进程会解析数据成明文并落盘成文件。执行msnpureport工具，导出快照。

  标志为：导出路径+/hisi_logs/device-x/history.log中记录有启动异常（STARTUP_EXCEPTION）或运行异常（RUN_EXCEPTION）。

Ascend RC 场景（Atlas 200 DK）

快照日志存在以下导出方式：

- 主动导出

  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）。

相关说明

- 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：热复位次数最小的异常。
- Ascend EP场景下，日志记录在<msnpureport导出路径>/hisi_logs/device-x/<时间戳目录>/snapshot/hdr.log。
- Ascend RC场景（Atlas 200 DK）下，日志记录在/var/log/npu/hisi_logs/device-0/<时间戳目录>/snapshot/hdr.log。

1.5 附录

1.5.1 msnpureport 工具使用

工具介绍

msnpureport工具部署在Host侧，该工具有以下用途：

- 导出Device侧的相关日志和文件，包括slog日志、syslog日志、黑匣子和Stackcore文件。导出的日志和文件将存储到运行msnpureport工具的路径下以时间戳命名的子目录中，且slog日志、syslog日志、黑匣子和Stackcore文件分别存储到slog、message、hisi_logs和stackcore文件夹下。
- 查询和设置Device侧slog系统类日志的级别。

须知

- msnpureport工具仅适用于Ascend EP场景。
- msnpureport工具不支持容器场景，部署容器时，禁止将msnpureport工具映射到容器内。
- 不支持多个用户同时运行msnpureport工具。
- msnpureport工具不支持增量导出日志，如果用户想避免老旧日志的影响，提高问题定位的效率，可以定期通过msnpureport工具获取Device侧日志，并自行制作脚本实现日志去重处理。

导出 Device 侧的相关日志和文件

步骤1 登录Host侧服务器。

步骤2 获取msnpureport工具。

msnpureport工具在驱动Driver的安装目录下，路径为“*Driver安装目录*/driver/tools/msnpureport”。

步骤3 在某个有读、写、执行权限的目录（如“/var/log/npu/report”）下执行如下命令，运行msnpureport工具。

Driver安装目录/**driver/tools/msnpureport** [options]

须知

在加锁的目录下（使用**lsattr**命令查看目录属性，有“i”选项的为加锁目录），用户没有权限运行msnpureport工具。如果用户想在该目录下运行msnpureport工具，可以通过**chattr -i** *<加锁的目录>*命令，将目录的“i”选项撤销。工具运行完后建议通过**chattr +i** *<加锁的目录>*命令，将目录的“i”选项加上。为了安全起见，不建议在加锁目录中运行工具。

其中[options]支持的参数及解释请参见**表1-8**。

表 1-8 参数说明

参数	说明	举例
不指定任何参数	导出slog日志、syslog日志、黑匣子日志和Stackcore文件。	***Driver安装目录*/driver/tools/msnpureport**
-a或--all	导出slog日志、syslog日志、黑匣子日志、黑匣子设备事件信息和Stackcore文件。	***Driver安装目录*/driver/tools/msnpureport -a**
-f或--force	导出slog日志、syslog日志、黑匣子日志、黑匣子的设备事件信息、黑匣子存储空间中的历史维测信息和Stackcore文件。	***Driver安装目录*/driver/tools/msnpureport -f**

运行成功后，Device侧的相关日志和文件被导出到Host侧，并存储到当前目录（如“/var/log/npu/report”）下以时间戳命名的文件夹中，具体如下：

- slog日志：“/var/log/npu/report/2020-10-20-03-12-05/slog”，其具体日志目录如下：

存储目录	说明
dev-os-*id*/device-os/device-os_*.log	Device侧Control CPU上的系统类日志，包括用户态日志和内核态日志。
dev-os-*id*/device-*id*/device-*id*_*.log	Device侧非Control CPU上的系统类日志。
dev-os-*id*/device-app-*pid*/device-app-*pid*_*.log	Device侧应用类日志。仅当Device侧应用类日志回传到Host侧失败时，才会在Device侧存储该日志。
dev-os-*id*/slogd/slogdlog	维测日志。记录日志工具自身的运行信息，用于日志工具自身问题定位。

- syslog日志：“/var/log/npu/report/2020-10-20-03-12-05/message”

  syslog日志表示调用syslog接口记录Device侧其他组件产生的日志。

- 黑匣子：“/var/log/npu/report/2020-10-20-03-12-05/hisi_logs”

  导出的黑匣子日志、黑匣子的设备事件信息和黑匣子存储空间中的历史维测信息均保存在该目录下。

- Stackcore文件：“/var/log/npu/report/2020-10-20-03-12-05/stackcore”

  通过msnpureport工具导出Stackcore文件后，可以参考**1.5.4 Stackcore文件解析定位**对Stackcore文件进行后续的分析处理。

----结束

查询和设置 Device 侧 slog 系统类日志级别

步骤1 登录Host侧服务器。

步骤2 获取msnpureport工具。

msnpureport工具在驱动Driver的安装目录下，路径为“*Driver安装目录*/driver/tools/msnpureport”。

步骤3 在某个有读、写、执行权限的目录（如“/var/log/npu/report”）下，执行如下命令查询和设置日志级别。

Driver安装目录/**driver/tools/msnpureport** [options]

Driver安装目录/**driver/tools/msnpureport** [options] **-d** <*device-id*>

其中[options]为查询和设置日志级别的各个参数，[options]各个参数及其他参数的解释请参见**表1-9**。

表 1-9 参数说明

参数	说明	举例
-g <*level*>或 **--global** <*level*>	设置全局级的日志级别。 ● debug：表示DEBUG级别。 ● info：表示INFO级别。 ● warning：表示WARNING级别。 ● error：表示ERROR级别。 ● null：表示NULL级别，不输出日志。	***Driver安装目录*/driver/tools/msnpureport -g info**
-m <*module:level*>或 **--module** <*module:level*>	设置模块级的日志级别。 ● *module*：模块名称。例如SLOG等。 ● *level*：模块级别。取值为： – debug：表示DEBUG级别。 – info：表示INFO级别。 – warning：表示WARNING级别。 – error：表示ERROR级别。 – null：表示NULL级别，不输出日志。	***Driver安装目录*/driver/tools/msnpureport -m SLOG:error**
-e <*level*>或 **--event** <*level*>	设置是否开启Event日志。 ● enable：开启Event日志。 ● disable：不开启Event日志。	***Driver安装目录*/driver/tools/msnpureport -e disable**

参数	说明	举例
-d *<device-id>*或 **--device** *<device-id>*	指定Device ID（逻辑ID），默认为0。通过指定Device ID设置对应Device的日志级别。 指定的Device ID是指逻辑上的Device ID而不是物理上的Device ID。须先判断逻辑ID后再设置对应的日志级别，详细步骤请参见**逻辑ID判断**。	***Driver安装目录*/driver/tools/msnpureport -g warning -d 1**
-r或**--request**	查询Device侧slog系统类日志的级别，包括全局级、模块级和是否开启Event日志。如果不指定Device ID，默认查询Device 0的日志级别。	***Driver安装目录*/driver/tools/msnpureport -r** 查询后显示的级别示例如下： The system log level of device_id:0 is as follows: [global] INFO [event] ENABLE [module] SLOG:INFO IDEDD:INFO DVPP:INFO CCE:INFO HDC:INFO DRV:INFO MDCDEFAULT:INFO DEVMM:INFO KERNEL:INFO LIBMEDIA:INFO ASCENDDK:INFO ROS:INFO HCCP:INFO ROCE:INFO PROFILING:INFO APP:INFO TDT:INFO MD:INFO MB:INFO ME:INFO BBOX:INFO TS:INFO TSDUMP:INFO LP:INFO
-h或**--help**	用于打印帮助信息。	***Driver安装目录*/driver/tools/msnpureport -h**

----结束

逻辑 ID 判断

步骤1 查询物理ID。

使用npu-smi info命令查看到的设备上的NPU Device ID即为物理ID。假设物理ID的取值范围是0~7，Device0~Device3四个为一组Device4~Device7四个为一组。由于同组的Device在同一个OS上，共用一个日志配置文件，所以日志级别也相同，只需要修改其中一个Device日志级别，那么同组其他的Device日志级别同样被修改。

物理ID的分组情况请以设备实际情况为准。

步骤2 判断逻辑ID。

查询到的物理ID会按照数字大小从小到大自上而下排列，那么对应的逻辑ID则从0开始按顺序为0~n。假设查询到的物理ID为“0，1，4，5”，那么对应逻辑ID则为“0，

1，2，3”。根据上一步的分组情况可判断物理ID“0，1”为一组，“4，5”为一组，那么可以推断出逻辑ID“0，1”为一组，“2，3”为一组。

步骤3 设置日志级别。

根据前面判断的逻辑ID“0，1”为一组，“2，3”为一组。假设设置所有Device的日志级别为error，则需要配置两次：

Driver安装目录/driver/tools/msnpureport -g error -d 0

Driver安装目录/driver/tools/msnpureport -g error -d 2

----结束

1.5.2 连续导出 Device 侧的日志和文件

当执行模型推理时，如果Device侧出现异常，会导致无法连接Device侧，因此将不能通过msnpureport工具导出Device侧的日志和文件（msnpureport工具导出的Device侧日志和文件具体请参见**1.5.1 msnpureport工具使用**章节）。此时可以在执行模型推理之前，在Host侧运行msnpureport_auto_export.sh脚本连续导出Device侧的日志和文件，确保能够获取到Device异常前的所有日志文件，便于定位问题。

须知

- 该功能仅适用于Ascend EP场景。
- msnpureport_auto_export.sh脚本不支持容器场景，部署容器时，禁止将msnpureport_auto_export.sh脚本映射到容器内。
- 不支持多个用户同时运行msnpureport_auto_export.sh脚本。
- 在异常场景下，可能会出现连续导出Device侧日志和文件失败的情况。
- 如果用户想要终止导出Device侧日志和文件或者模型推理完成，需要使用“Ctrl +c”或**kill -15** *pid*命令结束进程。

  *其中pid*代表连续导出日志和文件的脚本进程ID，可以通过**ps -elf | grep msnpureport_auto_export.sh**命令进行查询。

操作步骤

步骤1 登录Host侧服务器。

步骤2 获取msnpureport_auto_export.sh脚本。

msnpureport_auto_export.sh脚本在驱动Driver的安装目录下，路径为“*Driver安装目录*/driver/tools/msnpureport_auto_export.sh”。

步骤3 在某个有执行权限的目录（如“/home/work”）下执行如下命令，运行脚本。

*Driver安装目录***/driver/tools/msnpureport_auto_export.sh** *<timeInterval> <logAbsolutePathCapacity> <logAbsolutePath>*

命令示例：**/usr/local/Ascend/driver/tools/msnpureport_auto_export.sh 2 10 /home/log/**

须知

在加锁的目录下（使用**lsattr**命令查看目录属性，有“i”选项的为加锁目录），用户没有权限运行msnpureport_auto_export.sh脚本。如果用户想在该目录下运行该脚本，可以通过**chattr -i** *<加锁的目录>*命令，将目录的“i”选项撤销。脚本运行完后建议通过**chattr +i** *<加锁的目录>*命令，将目录的“i”选项加上。为了安全起见，不建议在加锁目录中运行脚本。

其中各参数解释如表1-10所示：

表 1-10 参数说明

参数	说明
<timeInterval >	导出Device侧日志和文件的间隔时间。取值为大于0的整数，单位是s，如：2s。
<logAbsolutePathCapacity>	导出Device侧日志和文件的存储目录容量。取值为大于等于2的整数，单位是G，如：10G。
<logAbsolutePath>	导出Device侧日志和文件的存储路径（任意的绝对路径）。如：“/home/log/”。

脚本运行成功后，Device侧日志和文件将存储在运行脚本时指定的存储路径下（如“/home/log/”），如果不存在该目录，会自动进行创建；如果存在则直接存储。该目录下会自动创建如下子目录：

目录	说明
msnpureport_log_new	导出Device侧日志和文件的存储目录。该目录下包含如下子目录： ● hisi_logs ● message ● slog ● stackcore 其中hisi_logs目录下的history.log文件和message目录下的message.log文件是在Device侧的同名文件中进行老化的，所以每次导出之后需要将导出内容追加到同一目录下的history_new.log和message_new.log文件中并进行去重，来获取所有日志文件。 其他目录的日志文件在Device侧是以时间戳命名的，是通过删除较早时间戳的日志文件进行老化的，所以每次导出之后只需将导出内容拷贝覆盖就可以获取所有日志文件。

目录	说明
msnpureport_log_old	老化的日志和文件的存储目录。 如果msnpureport_log_new目录存储的日志容量超过运行脚本时指定的存储目录容量的一半（如10/2=5G），将自动清空msnpureport_log_old目录下的日志和文件，再将msnpureport_log_new目录下存储的日志和文件全部移动到msnpureport_log_old目录下。

----结束

1.5.3 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 1-11 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总线挂死。
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。
RUN_EXCEPTION	0x2e	模块运行异常。
LPM_EXCEPTION	0x32	LPM异常。
TS_EXCEPTION	0x33	TS异常。

异常类型	异常类型值	异常描述
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。

1.5.4 Stackcore 文件解析定位

1.5.4.1 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、adda、tsdaemon、aicpu_scheduler）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 EP）

Device侧产生的Stackcore文件通过msnpureport工具导出到Host侧，导出方法及存储路径请参见**1.5.1 msnpureport工具使用**。

推理场景（Ascend RC）

Stackcore文件存放在运行环境的“/var/log/npu/coredump”路径下，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）。

1.5.4.2 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1-1所示。

    图 1-1 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1-2。

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

图 1-2 Stackcore 文件解析结果示例

```
-r-xr-x--- 1 HwHiAiUser HwHiAiUser    4859 Jun 30 10:56 stackview.sh
[root@localhost scripts]# ./stackview.sh -c /ms_test1/over_dump/2021-06-30-10-27-53/stackcore/dev-os-3/stackcore.slogd.3049.11.1624961947 -p ./
#0 std::iterator_traits<char*>::difference_type at std::distance<char*>(char*, from slogd
#1 Adx::AdxServerManager::ComponentWaitEvent() at :? from slogd
#2 Adx::AdxServerManager::ComponentWaitEvent() at :? from slogd
#3 Adx::AdxServerManager::ComponentWaitEvent() at :? from slogd
#4 __libpthread_freeres at ??:? from libpthread-2.28.so
#5 __clone at ??:? from libc-2.28.so
[root@localhost scripts]#
```

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	9	Illegal Instruction	不支持
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。

1.6 FAQ

1.6.1 kernel log 解析失败

现象故障

解析kernel log时存在错误打印，时间戳目录下未生成kernel.log文件，查看DONE文件内容不是FILEDONE。

图 1-3 解析 kernel log 时的错误打印信息

```
2020-12-23 11:42:51.099 [BBOX][INFO] [device-0] bbox runtime data, dump end.
2020-12-23 11:42:51.100 [BBOX][INFO] [device-0] bbox exception event(OOM), dump start.
2020-12-23 11:42:53.389 [BBOX][ERROR] invalid param, log_start[0x0], log_end[0x0], buf_size[0x0], length[2068480]
2020-12-23 11:42:53.389 [BBOX][ERROR] check klog msg failed.
2020-12-23 11:42:53.389 [BBOX][ERROR] parse data[kernel_log] failed.
2020-12-23 11:42:53.390 [BBOX][ERROR] [device-0] dump type[kernel_log] failed.
2020-12-23 11:42:53.406 [BBOX][INFO] [device-0] dump type[bbox_ddr_dump] succeed.
2020-12-23 11:42:53.406 [BBOX][INFO] [device-0] bbox exception event(OOM), dump end.
2020-12-23 11:42:53.406 [BBOX][ERROR] [device-0] dump oom event data failed.
2020-12-23 11:42:53.406 [BBOX][INFO] [device-1] bbox runtime data, dump start.
```

原因分析

当系统中存在模块频繁刷dmesg日志时，会出现存储kernel log的内存与cache中数据不一致的情况，此时通过黑匣子导出的内存存在未及时刷新的错误数据，导致解析失败。出现此问题的典型场景之一即为OOM异常导出场景（存在频繁刷日志的情况，并且会在此时导出kernel log数据）。

解决方案

通过kbox.txt文件中的内容查看OOM信息。

2 黑匣子日志参考（Windows 系统）

2.1 简介
2.2 日志介绍
2.3 快照日志介绍
2.4 附录

2.1 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。目前，支持黑匣子功能的模块有：BIOS、LPM、TEE、ATF、TS、Driver、DVPP、OS。

2.2 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。

黑匣子日志是通过msnpureport工具导出的，导出方法请参见**2.4.1 msnpureport工具使用**。存放目录是运行msnpureport工具的所在路径（如“C:\Program Files\Huawei\Ascend\Driver\tools”）。

命令示例如下：

cd C:\Program Files\Huawei\Ascend\Driver\tools\2020-10-20-03-12-05\hisi_logs

版权所有 © 华为技术有限公司

说明

- "C:\Program Files\Huawei\Ascend\Driver\tools"：运行msnpureport工具的所在路径，请根据实际情况修改。
- 2020-10-20-03-12-05：运行msnpureport工具生成的时间戳目录，请根据实际情况修改。
- hisi_logs下面存在*device-id*文件夹及device_info.txt文件
  - *device-id*表示Device设备的ID号，保存对应设备的异常信息。
  - device_info.txt记录了设备信息。

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 2-1 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

1. history.log到达30000条时会启动日志老化，删除最早的20000条信息。
2. 对于 ModuleName [AP], ExceptionReason [OS_OOM]异常，history.log中TimeStamp记录的是运行导出工具时获取到的时间戳，实际异常上报的时间戳需要在**步骤3**中的bbox\bbox_info.txt内查看。

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 2-2 日志文件路径及内容说明

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。

文件相对路径	文件内容
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，如挂死任务栈。
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\kernel.log	记录OS内核日志信息。
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。

----结束

版权所有 © 华为技术有限公司

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时，系统将老化最早生成的文件。
- Host侧
  - 运行msnpureport工具的所在路径（如“C:\Program Files\Huawei\Ascend\Driver\tools”）的磁盘空间不足时，无法生成黑匣子日志。
  - 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。

2.3 快照日志介绍

2.3.1 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
```

2.3.1.1 快照头部内容

快照头部信息内容：

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

表 2-3 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	当前环境热复位次数。

2.3.1.2 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。

2.3.1.2.1 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 2-4 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：L2BUF 历史队列。 ● 2：L2BUF 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。
	reset number	记录本次信息时的热复位计数。

说明

1. 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。
2. 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。
3. 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。
4. 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2.4.2 异常类型说明。

2.3.1.2.2 数据

启动区包含BIOS、DDR、TEE、ATF四个模块记录的信息。

运行区包含TEE、ATF、LPM、OS四个模块记录的信息。

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 2-5 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。

版权所有 © 华为技术有限公司

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。0为本模块没有异常。
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。

2.3.2 快照导出

快照日志存在以下导出方式：

- 被动导出
  - msnpureport工具检测到Device设备启动异常后，导出快照。

    标志为：导出路径+\hisi_logs\device-x\history.log中记录有设备丢失异常（DEVICE_LTO_EXCEPTION）。
  - msnpureport工具检测到Device设备心跳丢失异常后，导出快照。

    标志为：导出路径+\hisi_logs\device-x\history.log中记录有设备心跳异常（DEVICE_HBL_EXCEPTION）。
- 主动导出

  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用户态进程，用户态进程会解析数据成明文并落盘成文件。执行msnpureport工具，导出快照。

  标志为：导出路径+\hisi_logs\device-x\history.log中记录有启动异常（STARTUP_EXCEPTION）或运行异常（RUN_EXCEPTION）。

说明

1. 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。
2. 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。
3. 快照上报为启动异常还是运行异常的判定条件为：热复位次数最小的异常。
4. 日志记录在<msnpureport导出路径>\hisi_logs\device-x\<时间戳目录>\snapshot\hdr.log。

2.4 附录

2.4.1 msnpureport 工具使用

工具介绍

msnpureport工具部署在Host侧，该工具有以下用途：

- 导出Device侧的相关日志和文件，包括slog日志、syslog日志、黑匣子和Stackcore文件。导出的日志和文件将存储到运行msnpureport工具的路径下以时间戳命名的子目录中，且slog日志、syslog日志、黑匣子和Stackcore文件分别存储到slog、message、hisi_logs和stackcore文件夹下。
- 查询和设置Device侧slog系统类日志的级别。

须知

msnpureport工具不支持增量导出日志，如果用户想避免老旧日志的影响，提高问题定位的效率，可以定期通过msnpureport工具获取Device侧日志，并自行制作脚本实现日志去重处理。

导出 Device 侧的相关日志和文件

步骤1 获取msnpureport工具。

msnpureport工具在驱动Driver的安装目录下，路径为“C:\Program Files\Huawei\Ascend\Driver\tools\msnpureport.exe”。

步骤2 运行msnpureport工具，导出日志和文件。有如下两种方式，选择其一即可。

- 将**msnpureport.exe**程序拷贝到某一个有执行权限的目录下，选择**msnpureport.exe**程序，右键单击“以管理员身份运行”。

  运行成功后，Device侧的相关日志和文件被导出到当前目录下以时间戳命名的子目录中。

- 在开始菜单旁的搜索框中输入**cmd**，选择“命令提示符”，右键“以管理员身份运行”。在命令行界面执行如下命令并回车。

  C:\"Program Files"\Huawei\Ascend\Driver\tools\msnpureport.exe [options]

  说明

  如果msnpureport工具所在路径某一目录中包含空格，需要增加双引号才能运行。

  其中[options]支持的参数及解释请参见表2-6。

表 2-6 参数说明

参数	说明	举例
不指定任何参数	导出slog日志、syslog日志、黑匣子日志和Stackcore文件。	**C:\"Program Files"\Huawei\Ascend\Driver\tools\msnpureport.exe**

版权所有 © 华为技术有限公司

参数	说明	举例
-a或--all	导出slog日志、syslog日志、黑匣子日志、黑匣子设备事件信息和Stackcore文件。	**C:\"Program Files"\Huawei\Ascend\Driver\tools\msnpureport.exe -a**
-f或--force	导出slog日志、syslog日志、黑匣子日志、黑匣子的设备事件信息、黑匣子存储空间中的历史维测信息和Stackcore文件。	**C:\"Program Files"\Huawei\Ascend\Driver\tools\msnpureport.exe -f**

运行成功后，Device侧的相关日志和文件被导出到当前目录（如“C:\Windows\System32”）下以时间戳命名的子目录中。

导出的日志和文件包括：

- slog日志：位于slog文件夹，例如：“C:\Windows\System32\2020-12-22-20-26-01\slog”，其具体日志目录如下：

存储目录	说明
dev-os-*id*/device-os/device-os_*.log	Device侧Control CPU上的系统类日志，包括用户态日志和内核态日志。
dev-os-*id*/device-*id*/device-*id*_*.log	Device侧非Control CPU上的系统类日志。
dev-os-*id*/device-app-*pid*/device-app-*pid*_*.log	Device侧应用类日志，即应用程序的执行日志。仅当Device侧应用类日志回传到Host侧失败时，才会在Device侧存储该日志。
dev-os-*id*/slogd/slogdlog	维测日志。记录日志工具自身的运行信息，用于日志工具自身问题定位。

- syslog日志：位于message文件夹，例如：“C:\Windows\System32\2020-12-22-20-26-01\message”

  syslog日志表示调用syslog接口记录Device侧其他组件产生的日志。

- 黑匣子：位于hisi_logs文件夹，例如：“C:\Windows\System32\2020-12-22-20-26-01\hisi_logs”

  导出的黑匣子日志、黑匣子的设备事件信息和黑匣子存储空间中的历史维测信息均保存在该目录下。

- Stackcore文件：位于stackcore文件夹，例如：“C:\Windows\System32\2020-12-22-20-26-01\stackcore”

----结束

查询和设置 Device 侧 slog 系统类日志级别

步骤1 获取msnpureport工具。

msnpureport工具在驱动Driver的安装目录下，路径为“C:\Program Files\Huawei\Ascend\Driver\tools\msnpureport.exe”。

步骤2 在开始菜单旁的搜索框中输入**cmd**，选择“命令提示符”，右键“以管理员身份运行”。

步骤3 在命令行界面执行如下命令并回车，查询和设置日志级别。

C:\"Program Files"\Huawei\Ascend\Driver\tools\msnpureport.exe [options]

C:\"Program Files"\Huawei\Ascend\Driver\tools\msnpureport.exe -d *<device-id>* [options]

说明

如果msnpureport工具所在路径某一目录中包含空格，需要增加双引号才能运行。

其中[options]为查询和设置日志级别的各个参数，[options]各个参数及其他参数的解释请参见表2-7。

表 2-7 参数说明

参数	说明	举例
-g *<level>* 或 **--global** *<level>*	设置全局级的日志级别。 ● debug：表示DEBUG级别。 ● info：表示INFO级别。 ● warning：表示WARNING级别。 ● error：表示ERROR级别。 ● null：表示NULL级别，不输出日志。	**C:\"Program Files"\Huawei\Ascend\Driver\tools\msnpureport.exe -g info**
-m *<module:level>*或 **--module** *<module:level>*	设置模块级的日志级别。 ● *module*：模块名称。例如SLOG等。 ● *level*：模块级别。取值为： – debug：表示DEBUG级别。 – info：表示INFO级别。 – warning：表示WARNING级别。 – error：表示ERROR级别。 – null：表示NULL级别，不输出日志。	**C:\"Program Files"\Huawei\Ascend\Driver\tools\msnpureport.exe -m SLOG:error**

参数	说明	举例
-e *<level>* 或 **--event** *<level>*	设置是否开启Event日志。 ● enable：开启Event日志。 ● disable：不开启Event日志。	**C:\"Program Files"\Huawei\Ascend\Driver\tools\msnpureport.exe -e disable**
-d *<device-id>*或 **--device** *<device-id>*	指定Device ID（逻辑ID），默认为0。通过指定Device ID设置对应Device的日志级别。 指定的Device ID是指逻辑上的Device ID而不是物理上的Device ID。须先判断逻辑ID后再设置对应的日志级别，详细步骤请参见**逻辑ID判断**。	**C:\"Program Files"\Huawei\Ascend\Driver\tools\msnpureport.exe -d 1 -g warning**
-r或**--request**	查询Device侧slog系统类日志的级别，包括全局级、模块级和是否开启Event日志。如果不指定Device ID，默认查询Device 0的日志级别。	**C:\"Program Files"\Huawei\Ascend\Driver\tools\msnpureport.exe -r** 查询后显示的级别示例如下： The system log level of device_id:0 is as follows: [global] INFO [event] ENABLE [module] SLOG:INFO IDEDD:INFO DVPP:INFO CCE:INFO HDC:INFO DRV:INFO MDCDEFAULT:INFO DEVMM:INFO KERNEL:INFO LIBMEDIA:INFO ASCENDDK:INFO ROS:INFO HCCP:INFO ROCE:INFO PROFILING:INFO APP:INFO TDT:INFO MD:INFO MB:INFO ME:INFO BBOX:INFO TS:INFO TSDUMP:INFO LP:INFO
-h或**--help**	用于打印帮助信息。	**C:\"Program Files"\Huawei\Ascend\Driver\tools\msnpureport.exe -h**

----结束

逻辑 ID 判断

步骤1 查询物理ID。

使用npu-smi info命令查看到的设备上的NPU Device ID即为物理ID。假设物理ID的取值范围是0~7，Device0~Device3四个为一组Device4~Device7四个为一组。由于同组的Device在同一个OS上，共用一个日志配置文件，所以日志级别也相同，只需要修改其中一个Device日志级别，那么同组其他的Device日志级别同样被修改。

版权所有 © 华为技术有限公司

物理ID的分组情况请以设备实际情况为准。

步骤2 判断逻辑ID。

查询到的物理ID会按照数字大小从小到大自上而下排列，那么对应的逻辑ID则从0开始按顺序为0~n。假设查询到的物理ID为“0，1，4，5”，那么对应逻辑ID则为“0，1，2，3”。根据上一步的分组情况可判断物理ID“0，1”为一组，“4，5”为一组，那么可以推断出逻辑ID“0，1”为一组，“2，3”为一组。

步骤3 设置日志级别。

根据前面判断的逻辑ID“0，1”为一组，“2，3”为一组。假设设置所有Device的日志级别为error，则需要配置两次：

Driver安装目录/driver/tools/msnpureport -g error -d 0

Driver安装目录/driver/tools/msnpureport -g error -d 2

----结束

2.4.2 异常类型说明

表 2-8 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总线挂死。
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。
RUN_EXCEPTION	0x2e	模块运行异常。
LPM_EXCEPTION	0x32	LPM异常。
TS_EXCEPTION	0x33	TS异常。

异常类型	异常类型值	异常描述
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。
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

版权所有 © 华为技术有限公司