Intel® QuickAssist Technology

Debugging Guide

June 2021

Revision 1.4

Document Number: 621658

Contents

Tables

Revision History

Document Number	Revision Number	Description	Revision Date
621658	1.4	Added the following section for this release: • Section 2.6 How to Enable Debug symbols for the QAT Driver • Section 2.7 How to enable debug compile for QAT_engine • Section 4.4 Kernel oops and/or segmentation fault when bringing up the driver via adf_ctl or qat_service • Section 5.9 Application failure and error when trying to run openssl (or an openssl-based application) with QAT_engine with the USDM driver with huge pages • Section 9.3 Errors observed with openssl_speed Modified the following section for this release: • Section 5.7 QAT1.7 shows a hang or slice hang but recovers automatically	June 2021
621658	1.3	Added the following section for this release: • Section 5.8 Compilation of Functional Sample Code with alternative to gcc Modified the following section for this release: • Section 12.1.1 Resolution (for Section 12.1 Possible Errors Due to BIOS Setting)	December 2020
621658	1.2	Added the following sections for this release: • 5.7 QAT1.7 shows a hang or slice hang but recovers automatically • 8.2 Core Dump Occurs During NGINX Reload Modified the following sections for this release: • Table 2 - Section 1.2 - Document location modified for Intel® QuickAssist Technology Software for Linux* – Software Drivers – Hardware Version 1.7 • Table 2 – Section 1.2 - Electronic Design Kit removed from the table 6.3 Virtualization Use Case Issues	October 2020
621658	1.1	Added the following sections for this release: • 4.3, F.W. Authentication Error • 5.5, HKDF or ECEDMONT Operations do no Succeed • 6.3, Virtualization Use Case Issues • 10.7, Issues making SSL connection against HAProxy launched with Intel® QAT configured as non-root user • 11.0, DPDK Issues	July 2020
621658	1.0	Initial release	March 2020

§

1 Introduction

This document was designed to help debug issues with Intel® QuickAssist Technology (Intel® QAT).

It contains the following sections:

1.1 Terminology

Table 1. Terminology

Term	Description
API	Application Programming Interface
BIOS	Basic Input/Output System
DC	Data Compression
GRUB	GRand Unified Bootloader
O.S.	Operating System
PCH	Platform Controller Hub
PCI	Peripheral Component Interconnect
P.F.	Physical Function
Intel® QAT	Intel® QuickAssist Technology
SoC	System-on-a-Chip

Term	Description
SRIOV	Single Root-I/O Virtualization
V.F.	Virtual Function

1.2 Reference Documents and Software

Table 2. Reference Documents and Software

Document Title	Document Number/Location
Intel® QuickAssist Technology Software for Linux* – Release Notes – Hardware Version 1.7	336211
Intel® QuickAssist Technology Software for Linux* – Getting Started Guide – Hardware Version 1.7	336212
Intel® QuickAssist Technology Software for Linux* – Programmer's Guide – Hardware Version 1.7	336210
Intel® QuickAssist Technology Software for Linux* – Software Drivers – Hardware Version 1.7	01.org
Intel® QuickAssist Technology API Programmer's Guide	330684
Intel® QuickAssist Technology – Performance Optimization Guide	330687
Using Intel® Virtualization Technology (Intel® V.T.) with Intel® QuickAssist Technology Application Note	330689
HAProxy* with Intel® QuickAssist Technology Application Note	337430
Intel® QuickAssist Technology Software for Linux* – Release Notes – H.W. version 1.7	336211
Intel® QuickAssist Technology Videos	https://software.intel.com/enus/networking/quickassist

1.3 Resources

- https://01.org/intel-quickassist-technology
- https://software.intel.com/en-us/networking/quickassist
- https://github.com/intel/QAT_Engine
- http://www.intel.com/quickassist
- https://github.com/intel/QATzip

- https://github.com/intel/asynch_mode_nginx
- https://www.haproxy.org/
- Intel® Select Solutions for NFVI

§

2 How To...

This chapter describes how to perform various status checks on Intel® QAT.

2.1 How to Determine if Intel® QAT is Installed

1. Determine if Intel® QAT is installed by running the following command:
   ```
   lsmod | grep qa
   ```
   If Intel® QAT is installed, you should see output like the following:
   ```
   ]#  lsmod | grep qa qat_c62x
   13473  0 intel_qat             141688  1
   qat_c62x authenc                17776  1
   intel_qat dh_generic             13323
   1 intel_qat rsa_generic            18819
   1 intel_qat
   ```
2. If Intel QAT is not installed, follow the instructions in 336212, *Intel® QuickAssist Technology Software for Linux* Getting Started Guide Hardware Version 1.7*, at 01.org or in the Intel® QuickAssist Technology Videos at https://software.intel.com/enus/networking/quickassist.
3. Then rerun the command above to verify Intel® QAT is installed.

2.1.1 Relevant Collateral

- 336210, Intel® QuickAssist Technology Software for Linux* – Programmer's Guide – Hardware Version 1.7, at 01.org
- 336212, Intel® QuickAssist Technology Software for Linux* – Getting Started Guide – Hardware Version 1.7, at 01.org
- Intel® QuickAssist Technology Videos at https://software.intel.com/enus/networking/quickassist

2.1.2 How to Determine if Intel® QAT is Running by Looking at Firmware Counters

Monitor the Intel® QAT firmware counters to determine if Intel® QAT is running as in the following example:
```
watch cat /sys/kernel/debug/qat_c6xx_0000\:3d\:00.0/fw_counters
```

These firmware counters are the -
`/sys/kernel/debug/qat_<devicetype>_<bus_device_function>/fw_counters.`

Intel® QAT firmware counters increase when Intel® QAT is running. If Intel® QAT is not running, the firmware counters remain at their current value.

2.1.3 Relevant Collateral

336210, Intel® QuickAssist Technology Software for Linux* – Programmer's Guide – Hardware Version 1.7, at 01.org.

2.2 How to Determine if Intel® QAT is Active

Run one of the following commands: `systemctl status qat_service`

or `service qat_service status`

You should see the resulting output similar to the following:

```
]# systemctl status qat_service
qat_service.service - LSB: modprobe the QAT modules, which loads
dependant modules, before calling the user space utility to pass
configuration parameters
Loaded: loaded (/etc/init.d/qat_service; generated)
Active: active (exited) since Fri 2019-12-20 18:32:32 UTC; 28min ago
Docs: man:systemd-sysv-generator(8)
Process: 48577 ExecStop=/etc/init.d/qat_service stop (code=exited,
status=0/SUCCESS)
Process: 48635 ExecStart=/etc/init.d/qat_service start (code=exited,
status=0/SUCCESS)
Dec 20 18:32:30 dbubuntu qat_service[48635]: Restarting all devices.
Dec 20 18:32:30 dbubuntu qat_service[48635]: Processing
/etc/c6xx_dev0.conf
Dec 20 18:32:30 dbubuntu qat_service[48635]: Processing
/etc/c6xx_dev1.conf
Dec 20 18:32:31 dbubuntu qat_service[48635]: Processing
/etc/c6xx_dev2.conf
Dec 20 18:32:32 dbubuntu qat_service[48635]: Checking status of all
devices. Dec 20 18:32:32 dbubuntu qat_service[48635]: There is 3 QAT
acceleration device(s) in the system:
Dec 20 18:32:32 dbubuntu qat_service[48635]: qat_dev0 - type: c6xx,
inst_id: 0, node_id: 0, bsf: 0000:3d:00.0, #accel: 5 #engines: 10 state:
up
Dec 20 18:32:32 dbubuntu qat_service[48635]: qat_dev1 - type: c6xx,
inst_id: 1, node_id: 0, bsf: 0000:3f:00.0, #accel: 5 #engines: 10 state:
up
Dec 20 18:32:32 dbubuntu qat_service[48635]: qat_dev2 - type: c6xx,
inst_id: 2, node_id: 1, bsf: 0000:da:00.0, #accel: 5 #engines: 10 state:
up
Dec 20 18:32:32 dbubuntu systemd[1]: Started LSB: modprobe the QAT
modules, which loads dependant modules, before calling the user space
utility to pass configuration parameters.
]# service qat_service status Checking status of all devices.
There is 3 QAT acceleration device(s) in the system: qat_dev0 - type:
c6xx, inst_id: 0, node_id: 0, bsf: 0000:3d:00.0,
#accel: 5 #engines: 10 state: up
qat_dev1 - type: c6xx, inst_id: 1, node_id: 0, bsf: 0000:3f:00.0,
#accel: 5 #engines: 10 state: up
qat_dev2 - type: c6xx, inst_id: 2, node_id: 1, bsf: 0000:da:00.0,
#accel: 5 #engines: 10 state: up
```

Note: You can also run the systemctl <start, restart or stop> qat_service command, or qat_service <start, restart or stop> to perform the specific request.

2.2.1 Relevant Collateral

336210, Intel® QuickAssist Technology Software for Linux* – Programmer's Guide – Hardware Version 1.7, at 01.org

2.3 How to Determine if the Intel® QAT Device Has Failed or Hung with Heartbeat Monitoring

You can use Heartbeat monitoring to determine if the Intel® QAT device is in a functional state.

To simulate the Heartbeat management process, run the following commands:
`cat /sys/kernel/debug/<device>/heartbeat`

If 0 is returned, it indicates the device is responding. If –1 is returned, it indicates the device is not responding.
`cat /sys/kernel/debug/<device>/heartbeat_sent`

This number will increase each time the CAT heartbeat is sent because it tracks the number of times the control process checks to see if the device is responsive.
`cat /sys/kernel/debug/<device>/heartbeat_fail`

This number will increase each time the return value of the cat heartbeat is –1 because it keeps track of the number of times the control process finds the device unresponsive.
`cat /sys/kernel/debug/<device>/heartbeat_sim_fail`

This command simulates a failure on the Intel® QAT device. The return value will be zero. In addition, you can use the `icp_sal_heartbeat_simulate_failure()` API to simulate a heartbeat failure as well. For examples of other types of applications, refer to the following subdirectory of the Intel® QAT directory where the acceleration software is unpacked:
`quickassist/lookaside/access_layer/src/sample_code/functional/common`

Note: To simulate the heartbeat failure, Intel® QAT has to be configured as follows:
`./configure --enable-icp-hb-fail-sim`

2.3.1 Relevant Collateral

336210, Intel® QuickAssist Technology Software for Linux* – Programmer's Guide – Hardware Version 1.7, Section 3.17, at 01.org.

2.4 How to Reset or Restart the Intel® QAT Device When it has Failed or Hung, Using adf_ctl

When the Heartbeat monitoring detects that the Intel® QAT device has failed or hung, the device can be reset or restarted with the `adf_ctl` utility. In addition, the Intel® QAT device can be configured for auto-reset via the configuration file. For more information, please refer to Document Number 336210, *Intel® QuickAssist Technology Software for Linux* – Programmer's Guide*. Sections 3.3 and 5.2.6 contain information on the `adf_ctl` utility. "Resetting a Failed Device," under Section 3.17.1, contains information on Intel® QAT device auto-resetting via the configuration file.

The `adf_ctl` tool is in the subdirectory `quickassist/utilities/adf_ctl` of the Intel® QAT directory, where the acceleration software is unpacked. In the following

steps, `/opt/APP/driver/QAT` is the directory where the acceleration software is unpacked.

```
/opt/APP/driver/QAT/quickassist/utilities/adf_ctl]# ./adf_ctl qat_dev0
reset
/opt/APP/driver/QAT/quickassist/utilities/adf_ctl]# ./adf_ctl qat_dev0
restart
```

The first example above resets the `QAT_dev0` device, while the second example restarts the `QAT_dev0` device. Note that if `AutoResetOnError` is set to 1 in the `[GENERAL]` section of the Intel® QAT Config file (i.e., `c6xx_dev0.conf`), the reset is done automatically, and there is no need to perform the first example.

2.4.1 Relevant Collateral

336210, Intel® QuickAssist Technology Software for Linux* – Programmer's Guide – Hardware Version 1.7, at 01.org

2.5 How to Gather Necessary Information for Debugging

The `icp_dump.sh` tool is in the `quickassist/utilities/debug_tool` subdirectory of the Intel® QAT directory, where the acceleration software is unpacked. In the following steps, the Intel® QAT directory is `/opt/APP/driver/QAT` and the tar file (created from `icp_dump.sh`) will be stored in the `/root/iss_nfvi/icp_dump` directory.

Note: Run the command `mkdir /root/iss_nfvi/icp_dump` (or the directory of your choice) before running these steps.

1. Define `ICP_ROOT` as the directory you have installed Intel® QAT
   ```
   export ICP_ROOT=/opt/APP/driver/QAT
   ```
2. Run `icp_dump.sh` with one parameter: the directory where you would like the tar file to be stored.
   ```
   debug_tool ]# ./icp_dump.sh /root/iss_nfvi/icp_dump
   ```

Note: Accept and run the debug tool, type **yes** when prompted.

3. Unzip the file and verify Intel® QAT acceleration devices in the system are up.
   ```
   iss_nfvi]# tar -xzvf ICP_debug_18h_52m_07s_17d_10m_19y.tar.gz
   iss_nfvi]# cd ICP_debug
   ICP_debug]# cat adf_ctl_status.txt
   Checking status of all devices.
   ```

   There are three Intel® QAT acceleration devices in the system:
   ```
   qat_dev0 - type: c6xx, inst_id: 0, node_id: 0, bsf: 0000:3d:00.0,
   #accel: 5 #engines: 10 state: up qat_dev1 - type: c6xx, inst_id: 1,
   node_id: 0, bsf: 0000:3f:00.0, #accel: 5 #engines: 10 state: up
   qat_dev2 - type: c6xx, inst_id: 2, node_id: 1, bsf: 0000:da:00.0,
   #accel: 5 #engines: 10 state: up
   ```
4. Verify that all Intel® QAT configuration files are the same.

The SHIM section needs to be in place when Intel® QAT SHIMs is used, and this includes the Intel® QAT Engine and QATqzip. The CPA sample code uses the default Intel® QAT configuration files that are installed along with the Intel® QAT driver.

The following is an example of the configuration that contains the [SHIM] section:

```
ICP_debug]# cd config_files/ config_files]# cat c6xx_dev0.conf …
############################################
# User Process Instance Section
############################################
[SHIM]
NumberCyInstances = 1
NumberDcInstances = 0
NumProcesses = 10

# Crypto - User space
Cy0Name = "UserCY0"
Cy0IsPolled = 1
Cy0CoreAffinity = 0
```

2.5.1 Relevant Collateral

336210, Intel® QuickAssist Technology Software for Linux* – Programmer's Guide – Hardware Version 1.7, at 01.org

2.6 How to Enable Debug Symbols for the QAT Driver

For the QAT driver, you can enable debugging with these changes:

```
quickassist/build_system/build_files/env_files/environment.mk
ICP_DEFENSES_ENABLED ?= n
quickassist/build_system/build_files/common.mk
$(PROG_ACY)_OPT_LEVEL?=0
Makefile (change it after ./configure, then make and install)
CFLAGS = -g
```

2.7 How to Enable Debug Compile for QAT_engine

For the QAT_engine / OpenSSL*, you can enable debugging with these changes:

QAT_Engine:

1. run ./configure with additional parameters such as the following:

```
--enable-qat_debug \
--enable-qat_warnings \
--enable-qat_mem_warnings \
--enable-qat_mem_debug
--with-qat_debug_file=/qat_engine_debug.log
```

2. Check QAT Engine Makefile and make the following changes:

```
Change cflags = -shared -fPIC -Wall -Wformat -Wformat-security -O2 -
D_FORTIFY_SOURCE=2 -fstack-protector to cflags = -shared -fPIC -Wall
-Wformat -Wformat-security -O0 -g -D_FORTIFY_SOURCE=0 -fstack-
protector

Change "CFLAGS = -g -O2" to "CFLAGS = -g -O0"
```

```
Change "CXXFLAGS = -g -O2" to "CXXFLAGS = -g -O0"
```

§

3 Intel® QAT Driver Installation Issues

The following sections describe steps for resolving Intel® QAT driver installation issues.

3.1 Intel® QAT Driver Does Not Compile

If you experience compile errors, try one or more of the following steps:

- Update to the latest Intel® QAT Driver version
- Study the errors and warnings
- Update driver to use the kernel functions that correspond with your kernel and structures
- Install dependencies as described in the Intel® QAT Getting Started Guide

Note: Compile errors related to the kernel version are usually observed with newer kernels.

Please update to the latest version of the Intel® QAT driver available on 01.org. If you still experience issues, consult with your Intel representative.

3.1.1 Relevant Collateral

336212, Intel QuickAssist Technology Software for Linux* Getting Started Guide Hardware Version 1.7, at 01.org.

3.2 Linux* Crypto API Does Not Use Intel® QAT

Users may be attempting to use Intel® QAT integrated into the Linux* Crypto API and looking for confirmation that Intel® QAT is being used. Users can look to the Intel® QAT FW counters and verify that they increase as crypto operations are performed. If Intel® QAT counters are not increasing, it may be due to one of the following:

- Depending on the user's version of Intel® QAT, the Linux* Crypto API may not be enabled by default. In Intel® QAT HW Version 1.7 L.4.7 and earlier, the Linux* Crypto API was enabled by default. With Intel® QAT HW Version 1.7 L.4.8 and later, the option must be enabled when installing Intel® QAT, with the following command:
  ```
  ./configure --enable-qat-lkcf
  ```
- The required algorithm may not be installed. The user may add the algorithm or ask their Intel representative to add the algorithm. The following is an example of how to determine the algorithms supported in the current installation:
  ```
  # cat /proc/crypto | grep
  qat driver       : qat-dh
  module       : intel_qat
  driver       : qat-rsa
  ```

```
module       : intel_qat
driver       :
qat_aes_cbc_hmac_sha512
module       : intel_qat
driver       :
qat_aes_cbc_hmac_sha256
module       : intel_qat
driver       :
qat_aes_cbc_hmac_sha1
module       :
intel_qat driver
: qat_aes_xts module
: intel_qat driver
: qat_aes_ctr module
: intel_qat driver
: qat_aes_cbc module
: intel_qat
```

3.2.1 Relevant Collateral

Driver code and O.S. registered functions.

3.3 Issues with the Intel® QAT Make or with Starting Intel® QAT

For the issues listed below, the root cause may be a mismatch of the install kernel and/or headers.

- Kernel Header Files Missing:

```
make[1]: Entering directory `/opt/APP/driver/QAT'
make[2]: Entering directory `/opt/APP/driver/QAT/quickassist/qat'

Makefile:66: *** ERROR: Kernel header files not found. Install the
appropriate kernel development package necessary for building external
kernel modules or run 'make oldconfig && make modules_prepare' on
kernel src to fix it. Stop.

make[2]: Leaving directory `/opt/APP/driver/QAT/quickassist/qat'
make[1]: *** [qat-driver-all] Error 2 make[1]: Leaving directory
`/opt/APP/driver/QAT' make: *** [all] Error 2
```

- Errors in Intel® QAT Make:

```
include/asm-generic/pgtable.h:632:19: note:  previous definition of
‘pud_trans_huge’ was here static inline int pud_trans_huge(pud_t pud)
^ In file included from ./arch/x86/include/asm/pgtable.h:1235:0,
from include/linux/mm.h:63,                  from
./arch/x86/include/asm/pci.h:4,                  from
include/linux/pci.h:1641,                 from
/opt/APP/driver/QAT/quickassist/qat/compat/ qat_compat.h:87,
from <command-line>:0: include/asm-generic/pgtable.h: At top level:
include/asm-generic/pgtable.h:632:19: error:  redefinition of
‘pud_trans_huge’ static inline int pud_trans_huge(pud_t pud)
```

- Unable to Start/Restart Intel® QAT:

```
Failed to restart qat_service.service: Unit not found.
```

3.3.1 Resolution

Follow these steps:

1. Use the following code to determine what kernels are installed on your system, as in the following example:
```
# yum list installed kernel
Loaded plugins: langpacks, product-id, search-disabled-repos,
subscription-manager Installed Packages kernel.x86_64
3.10.0-957.el7           @anaconda/7.6 kernel.x86_64          3.10.0-
957.12.2.el7      @rhel-7-server-rpms kernel.x86_64          3.10.0-
1062.12.1.el7     @rhel-7-server-rpms
```
2. If there is no kernel list as shown in the previous step, then install it as follows:
```
yum install kernel-devel-$(uname -r)
```
3. If multiple kernels are installed, remove the kernels that you do not need as in the following example:
```
yum remove kernel-devel-3.10.0-1062.12.1.el7.x86_64
```
4. If the only kernel installed is the one you want, then reinstall it by performing Step 3, followed by Step 2.

Reinstalling the kernel will verify the correct headers are being used (i.e., there may be a chance that Intel® QAT was previously built with a different Linux* kernel, with different headers.)

3.3.2 Relevant Collateral

336210, Intel® QuickAssist Technology Software for Linux* – Programmer's Guide – Hardware Version 1.7, at 01.org

§

4 System Configuration Issues

This section describes resolution steps for system configuration issues.

4.1 Intel® QAT Endpoint is Trained to Less than the PCIe* Max Capability

This issue includes one or more of the following symptoms:

`lspci` returns a trained value below the maximum PCIe* capability

- Intel® QAT performance is low
- Platform issues: BIOS, jumpers, or analog issues
- Intel® QAT endpoint is trained correctly, but the internal switches report at lower speeds

4.1.1 Resolution

Verify that the `cpa_sample_code` gives the expected performance.

Contact your Intel representative for the expected performance numbers, if necessary.

4.1.2 Relevant Collateral

- 336210, Intel® QuickAssist Technology Software for Linux* – Programmer's Guide – Hardware Version 1.7, at 01.org
- 330687, Intel® QuickAssist Technology – Performance Optimization Guide, at 01.org
- Intel® QuickAssist Technology Videos at https://software.intel.com/enus/networking/quickassist

4.2 "adf_ctl status" Shows Fewer than Expected Devices

If adf_ctl status shows fewer than expected devices, try the resolution steps below.

4.2.1 Resolution

Check for one or more of the following conditions:

- Intel® QAT modules were not successfully installed with `insmod`
- Intel® QAT modules were not installed with `insmod` in the correct order

4.2.2 Relevant Collateral

- 336212, Intel® QuickAssist Technology Software for Linux* – Getting Started Guide – Hardware Version 1.7, at 01.org
- Intel® QuickAssist Technology Videos at https://software.intel.com/enus/networking/quickassist

4.3 Firmware Authentication Error

If you see the following symptom, please try the resolution steps below: `dmesg` Intel® QAT: authentication error (`FCU_STATUS = 0x3`),retry = 0

4.3.1 Resolution

If there is not a PCIe AER error, double-check the firmware version. Mismatching the firmware. version and driver version will cause an authentication error.

4.3.2 Relevant Collateral

336212, Intel® QuickAssist Technology Software for Linux* – Getting Started Guide – Hardware Version 1.7, at 01.org

4.4 Kernel Oops and/or Segmentation Fault When Bringing Up the Driver Via adf_ctl or qat_service

If Kernel oops when bringing up the driver with `adf_ctl` or `qat_service`, try the resolution steps below. Oops may include the kernel message keywords "SMP PTI".

4.4.1 Resolution

Ensure that all modules were built correctly and are not a mix of assets from 01.org and kernel.org. Use modinfo on all modules returned from "lsmod | grep qa", if necessary. If following the Getting Started Guide and using the configure and make commands per the guide, this will be handled correctly.

4.4.2 Relevant Collateral

- 336212, Intel® QuickAssist Technology Software for Linux* – Getting Started Guide – Hardware Version 1.7, at 01.org

§

5 Application Issues

This section describes resolution steps for application issues.

5.1 Intel® QAT App Fails to Run

Error messages result when starting the Intel® QAT app, usually during the `userStart` function.

5.1.1 Resolution

Try one or more of the following:

- Install Intel® QAT.
- Update Intel® QAT configuration files to include the correct section name.

Note: Run the CPA Sample App first to verify that you get good results.

Please refer to Section 4.1 of the *Intel® QAT Getting Started Guide*.

5.1.2 Relevant Collateral

- 336210, Intel® QuickAssist Technology Software for Linux* – Programmer's Guide – Hardware Version 1.7, at 01.org
- 336212, Intel® QuickAssist Technology Software for Linux* – Getting Started Guide – Hardware Version 1.7, at 01.org
- Intel® QuickAssist Technology Videos at https://software.intel.com/enus/networking/quickassist

For example, Section 3, "Building and Installing Software," and Section 4, "Sample Applications," in the Getting Started Guide, will show all the necessary steps.

Also, please refer to the following entries in Section 2.0 of this document:
- How to Determine if Intel® QAT is Installed
- How to Determine if Intel® QAT is Active

5.2 Application is Not Using Intel® QAT

Intel® QAT counters are not increasing. For example,
```
watch cat /sys/kernel/debug/qat_c6xx_0000:3d:00.0/fw_counters
```

Note: Check `/sys/kernel/debug` for your applicable `qat_c6xx*` directory.

5.2.1 Resolution

Applications may not be patched or configured to use Intel® QAT. Consult the relevant documentation.

5.2.2 Relevant Collateral

- 336210, Intel® QuickAssist Technology Software for Linux* – Programmer's Guide – Hardware Version 1.7, at 01.org
- 330687, Intel® QuickAssist Technology – Performance Optimization Guide, at 01.org
- Intel® QuickAssist Technology Videos at https://software.intel.com/enus/networking/quickassist

5.3 Intel® QAT Endpoint Hangs

If the Intel® QAT device is not responsive, try the resolution steps below.

5.3.1 Resolution

Try one or more of the following:

- Step through the application to identify the operation that led to the hang, i.e., focus on replication.
- Run `adf_ctl reset` to recover.
- Verify that all Intel® QAT API operations and addresses are valid.

5.3.2 Relevant Collateral

336210, Intel® QuickAssist Technology Software for Linux* – Programmer's Guide – Hardware Version 1.7, at 01.org

5.4 Error Reading /dev/qat_dev_processes File

When testing the driver (e.g., with functional sample code), you receive the error `reading /dev/qat_dev_processes file`:

```
# ./ipsec_sample main(): Starting IPSec Sample Code App ...

ADF_UIO_PROXY err: icp_adf_userProcessToStart: Error reading

/dev/qat_dev_processes file main(): Failed to start user process SSL
```

5.4.1 Resolution Steps

1. Ensure that the configuration files match the application code, i.e., that `icp_sal_userStart` references "SSL" and that the configuration files in `/etc/` also mention "SSL" sections with a declared number of instances.
2. Restart `qat_service`.

5.4.2 Relevant Collateral

336212, Intel® QuickAssist Technology Software for Linux* – Getting Started Guide – Hardware Version 1.7, at 01.org

5.5 HKDF or ECEDMONT Operations Do Not Succeed

There are multiple options for this issue, such as the following:

```
"The device does not support ECEDMONT"
"The device does not support HKDF"
"ExtAlgChain feature not supported"
```

5.5.1 Resolution Steps

There are multiple steps you can take, such as follows:

- Ensure that you have the correct `ServicesProfile` option
- Ensure that you are on the latest release. 4.10 on the host and guest may solve the issue.

5.5.2 Relevant Collateral

- 336211, Intel QuickAssist Technology Software for Linux* Release Notes H.W. version 1.7, at 01.org
- 336210, Intel QuickAssist Technology Software for Linux* Programmers Guide Hardware Version 1.7, at 01.org

5.6 Proxy Application+QAT, No Performance Improvement Using Multi-threads

Try the resolution steps below if there is no performance improvement with 1 process and multithreading(multi workers).

5.6.1 Resolution Steps

Try setting the flag `ICP_WITHOUT_THREAD` in the USDM (`quickassist/utilities/libusdm_drv`) and recompile the USDM alone. Set the additional environment variables mentioned below to recompile USDM alone.

```
export ICP_WITHOUT_THREAD=1 export
ICP_BUILDSYSTEM_PATH=$ICP_ROOT/quickassist/build_system
export

ICP_ENV_DIR=$ICP_ROOT/quickassist/build_system/build_files/env_files
```

5.7 QAT1.7 Shows A Hang or Slice Hang but Recovers Automatically

When an automatic recovery occurs after a hang or slice hang, there is no longer a possibility to perform a register or ring dump analysis to determine the root cause of the hang. Kernel messages may be seen that mention slice hang, with a possible application error.

5.7.1 Resolution

Increase `CySymAndDcWatchDogTimer` and/or `CyAsymWatchDogTimer` (in ms) in the general section of the config file to set the watchdog timer to a high value (e.g. 1000000).

You can also disable/enable slice hang detection for a device or all devices as follows:

```
./qat_debug.sh [[--disable_slicehang_detection|--
enable_slicehang_detection] <bus>:<device>.<func>]
./qat_debug.sh [[--disable_slicehang_detection|--
enable_slicehang_detection] all]
```

Please contact your Intel representative for more information on qat_debug.sh.

5.8 Compilation of Functional Sample Code with Alternative to gcc

The functional sample code can be compiled with an alternative compiler other than the gcc compiler.

5.8.1 Resolution

Use the "CC" option when building. For example, to use the icc compiler, you would change "make all" to "make CC=icc all". (Please refer to Section 4.2.1 of the Getting Started Guide.)

5.8.2 Relevant Collateral

336212, Intel QuickAssist Technology Software for Linux* Getting Started Guide Hardware Version 1.7, at 01.org.

5.9 Application Failure and Error When Trying to Run OpenSSL* (or an OpenSSL*-based application) with QAT_engine with the USDM Driver with Huge Pages

An error message similar to the following may be received when running OpenSSL* (or an OpenSSL*-based application):

```
hugepage_mmap_phy_addr:159 qae_mmap(/dev/hugepages/qat/usdm.jxZf9Y) for
hpg_fd failed with errno:12
hugepage_alloc_slab:204 mmap on huge page memory allocation failed
ADF_UIO_PROXY err: adf_init_ring: unable to get ringbuf(v:(nil),p:(nil))
for rings in bank(0)
ADF_UIO_PROXY err: icp_adf_transCreateHandle: adf_init_ring failed
[error] SalCtrl_ServiceInit() - : Failed to initialise all service
instances
ADF_UIO_PROXY err: adf_user_subsystemInit: Failed to initialise
Subservice SAL
[error] SalCtrl_ServiceEventStart() - : Private data is NULL
ADF_UIO_PROXY err: adf_user_subsystemStart: Failed to start Subservice
SAL
[error] SalCtrl_AdfServicesStartedCheck() - : Sal Ctrl failed to start in
given time
[error] do_userStart() - : Failed to start services
ADF_UIO_PROXY err: icp_adf_subsystemUnregister: Failed to shutdown
subservice SAL.
```

5.9.1 Resolution

Ensure that huge pages are created.

```
# cat /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
(should be greater than zero)
```

Inf the number of huge pages is zero, as an example, they can be increased temporarily as follows:

```
# echo 1024 > /proc/sys/vm/nr_hugepages
```

Other things to try could be reducing the number of QAT instances allocated in the /etc config file, or adjusting the huge pages allocated on insmod of usdm:

```
# insmod ./usdm_drv.ko max_huge_pages=<adjust>
max_huge_pages_per_process=<adjust>
```

§

6 Intel® QAT Virtualization Issues

This section describes resolution steps for Intel® QAT virtualization issues.

6.1 Too Many Intel® QAT VFs are Created

When trying to create fewer virtual functions than the maximum, the maximum number always gets created.

6.1.1 Resolution

None; this is a hardware limitation, currently.

6.1.2 Relevant Collateral

- 330689, Using Intel® Virtualization Technology (Intel® V.T.) with Intel® QuickAssist Technology Application Note, at 01.org
- Videos at https://software.intel.com/en-us/networking/quickassist

6.2 Intel® QAT VFs are Not Created

If the virtual functions are not created, try resolving this issue using the resolution steps below.

6.2.1 Resolution

Check for one or more of the following causes:

- `configure` was not run with the right options and needed to be run with the correct option.
- `intel_iommu`=on is not part of the GRUB boot settings and needs to be included in the grub • Virtualization is not enabled in the BIOS and needs to be enabled

6.2.1.1 Example Outputs

1. Run `lscpu` to check if virtualization (vmx) is enabled in the BIOS:

```
# lscpu | grep vmx

Flags:                fpu vme de pse tsc msr pae mce cx8 apic sep
mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht
tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon
pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni
pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16
xtpr pdcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm
```

```
3dnowprefetch epb cat_l3 cdp_l3 invpcid_single intel_ppin intel_pt
ssbd mba ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi flexpriority
ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm
cqm mpx rdt_a avx512f avx512dq rdseed adx smap clflushopt clwb
avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 cqm_llc
cqm_occup_llc cqm_mbm_total cqm_mbm_local dtherm arat pln pts hwp
hwp_act_window hwp_epp hwp_pkg_req pku ospke avx512_vnni md_clear
spec_ctrl intel_stibp flush_l1d arch_capabilities
```

2. Check `dmesg` to see if Virtualization (DMAR) is enabled for your particular device:

```
# dmesg | grep -i DMAR | grep d8:00.0
[    5.361824] DMAR: Hardware identity mapping for device
0000:d8:00.0
```

6.2.2 Relevant Collateral

- 330689, Using Intel® Virtualization Technology (Intel® V.T.) with Intel® QuickAssist Technology Application Note, at 01.org
- Videos at https://software.intel.com/en-us/networking/quickassist

6.3 Virtualization Use Case Issues

You may encounter a kernel message such as "`PTAE Read access is not set`" and/or "`Cannot use PF with IOMMU enabled`."

6.3.1 Resolution

- Get `cpa_sample_code` working by referring to Table 2, Using Intel® Virtualization Technology (Intel® VT) with Intel® QuickAssist Technology Application Note.
- Ensure that the BIOS enables virtualization.
- Ensure that `intel_iommu=on` is set in grub, verified using "`cat /proc/cmdline`".

Note: If intel_iommu=on is not set in the grub, then it implies that QAT should be run without the configure script option enable-icp-sriov. The converse is also true.

- Ensure that host configure script was run with"`./configure --enable-icp-sriov=host`" and that the guest configure script (if applicable) was run with "`./configure --enable-icp-sriov=guest`"

6.3.2 Relevant Collateral

- 330689, Using Intel® Virtualization Technology (Intel® V.T.) with Intel® QuickAssist Technology Application Note, at 01.org
- Videos at https://software.intel.com/en-us/networking/quickassist

§

7 Intel® QAT Performance Issues

This section describes resolution steps for Intel® QAT performance issues.

7.1 CPU Performance Beats Intel® QAT Performance

If the CPU performance beats Intel® QAT performance resolve this by using the resolution steps below.

7.1.1 Resolution

Try one or more of the following steps:

- Optimize the application for memory recycling.
- Increase application concurrency and Intel® QAT configuration to use full parallelization.
- Increase buffer/packet sizes (small packets may not see the offloading benefit).
- CPU performance may beat Intel® QAT for certain algorithms, for certain packages, with enough cores.

7.1.2 Relevant Collateral

- 330687, Intel® QuickAssist Technology – Performance Optimization Guide, at 01.org
- Videos at https://software.intel.com/en-us/networking/quickassist

7.2 Intel® QAT Performance is Low

When Intel® QAT is not performing as expected try one or more of the following resolution steps to resolve the issue.

7.2.1 Resolution

Try one or more of the following steps:

- Optimize the application for memory recycling.
- Increase application concurrency and Intel® QAT configuration to use full parallelization.
- Increase buffer/packet sizes (small packets may not see the offloading benefit).
- CPU performance may beat Intel® QAT for certain algorithms, for certain packages, with enough cores.
- Remove software stack layers to verify that Intel® QAT performance at the lower-lever layers is as expected.

7.2.2 Relevant Collateral

- 330687, Intel® QuickAssist Technology – Performance Optimization Guide, at 01.org
- 336210, Intel® QuickAssist Technology Software for Linux* – Programmer's Guide – Hardware Version 1.7, at 01.org.
- Videos at https://software.intel.com/en-us/networking/quickassist

§

8 NGINX* Issues

This section describes steps to resolve NGINX* issues.

8.1 NGINX* + Intel® QAT Performance is Low

If performance is low with NGINX and Intel® QAT, follow the resolution steps below.

8.1.1 Resolution

Try one or more of the following steps:

- Use the Intel® Select Solutions for NFVI script to apply the correct settings (i.e., more worker processes, keep-alive settings, high concurrency, etc.)
- Ensure that Intel® QAT is being used with the firmware counters
- Ensure that GRUB does not have idle=poll
- Isolating cores in the GRUB has been shown to reduce performance

8.1.2 Relevant Collateral

Intel® Select Solutions for NFVI

8.2 Core Dump Occurs During NGINX Reload

The following is an example of a core dump that occurred during NGINX Reload (i.e. backtrace stack):

Core dump during Nginx reload, backtrace stack:

```
#0  0x00007f6964a20544 in Lac_MemPoolCleanUpInternal () from
/usr/local/lib/libqat_s.so

#1  0x00007f6964a208a0 in Lac_MemPoolCreate () from
/usr/local/lib/libqat_s.so

#2  0x00007f6964a3cb5a in SalCtrl_AsymInit () from
/usr/local/lib/libqat_s.so

#3  0x00007f6964a3d573 in SalCtrl_CryptoInit () from
/usr/local/lib/libqat_s.so

#4  0x00007f6964a40ac5 in SalCtrl_ServiceInit.constprop.2 () from
/usr/local/lib/libqat_s.so

#5  0x00007f6964a41918 in SalCtrl_ServiceEventHandler () from
/usr/local/lib/libqat_s.so
```

```
#6  0x00007f6964a4821d in adf_user_subsystemInit () from
/usr/local/lib/libqat_s.so

#7  0x00007f6964a48edc in adf_proxy_get_device () from
/usr/local/lib/libqat_s.so

#8  0x00007f6964a49030 in adf_proxy_get_devices () from
/usr/local/lib/libqat_s.so

#9  0x00007f6964a47688 in icp_adf_userProxyInit () from
/usr/local/lib/libqat_s.so

#10 0x00007f6964a450cc in do_userStart (process_name=0x7ffd66ddd660
"SHIM_INT_33") at
/root/qat_upstream_driver/quickassist/lookaside/access_layer/src/user/sal
_user.c:137

#11 icp_sal_userStart (process_name=<optimized out>) at
/root/qat_upstream_driver/quickassist/lookaside/access_layer/src/user/sal
_user.c:187

#12 0x00007f6964a45335 in icp_sal_userStartMultiProcess
(pProcessName=<optimized out>,
limitDevAccess=limitDevAccess@entry=CPA_FALSE) at
/root/qat_upstream_driver/quickassist/lookaside/access_layer/src/user/sal
_user.c:220

#13 0x00007f69627a4e4f in qat_engine_init (e=e@entry=0x55d014708140) at
e_qat.c:475

#14 0x00007f69627a5f30 in engine_init_child_at_fork_handler () at
qat_fork.c:91

#15 0x00007f6964d8caae in fork () from /lib64/libc.so.6

#16 0x000055d01326c90a in ngx_spawn_process
(cycle=cycle@entry=0x55d01473c3f0, proc=proc@entry=0x55d01326e380
<ngx_worker_process_cycle>, data=data@entry=0x1,
name=name@entry=0x55d0132d11db "worker process", respawn=respawn@entry=-
4)

    at src/os/unix/ngx_process.c:186

#17 0x000055d01326db10 in ngx_start_worker_processes
(cycle=cycle@entry=0x55d01473c3f0, n=32, type=type@entry=-4) at
src/os/unix/ngx_process_cycle.c:361

#18 0x000055d01326eebb in ngx_master_process_cycle (cycle=0x55d01473c3f0,
cycle@entry=0x55d0146fcfe0) at src/os/unix/ngx_process_cycle.c:246

#19 0x000055d013246605 in main (argc=<optimized out>, argv=<optimized
out>) at src/core/nginx.c:389
```

Note: Hugepage memory is used up and there is no 2MB slab memory. Neither can be checked with `cat /proc/meminfo` and `cat /proc/buddyinfo`. Usually, if check with `free -m`, the buffer/cache number will be high.

8.2.1 Resolution

There is a work around. Allocate enough Hugepage Memory for Nginx. If there are 32 Nginx worker processes, during reload, the maximum number of work processes will be 64.

```
insmod ./usdm_drv.ko max_huge_pages=N max_huge_pages_per_process=32
```

N should be larger than or equal to 32 * 64

§

9 OpenSSL*/QAT_Engine Issues

This section describes resolution steps for OpenSSL*/QAT_Engine issues.

9.1 Error with Version of OpenSSL*

If you see a result like the following:

```
[root@SR1B011 apps]# ./openssl version

./openssl: error while loading shared libraries:
libssl.so.1.1: cannot open shared object file: No such file or directory
```

Then most likely, the library path is not set up.

```
[root@SR1B011 apps]# echo $LD_LIBRARY_PATH
```

9.1.1 Resolution

Export the `$LD_LIBRARY_PATH` and rerun the command as follows:

```
[root@SR1B011 apps]# export

LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/ssl/lib

[root@SR1B011 apps]# ./openssl version

OpenSSL* 1.1.1 11 Sep 2018
```

9.1.2 Relevant Collateral

https://github.com/intel/QAT_Engine (including the Troubleshooting section)

9.2 Errors with make/make install of the Intel® QAT OpenSSL* Engine

You experience errors with `make` or `make install` as in the following:

```
qat_ciphers.c:464:26: note: each undeclared identifier is reported only
once for each function it appears in make[1]: *** [qat_rsa.lo] Error 1
qat_ciphers.c: In function 'qat_chained_ciphers_do_cipher':

qat_ciphers.c:1651:59: error: 'ASYNC_STATUS_OK' undeclared (first use in
this function) if ((job_ret = qat_pause_job(done.opDone.job,
ASYNC_STATUS_OK)) == 0)
^ qat_ciphers.c: In function 'qat_sym_perform_op':

qat_ciphers.c:1778:48: error: 'ASYNC_STATUS_EAGAIN' undeclared (first use
in this function)
if ((qat_wake_job(opDone->job, ASYNC_STATUS_EAGAIN)
== 0) ||
```

9.2.1 Resolution

The root cause could be you have cloned the `QAT_Engine` with the OpenSSL* repository. It is not normally advised to clone one git repo within another. In this case, clone the `QAT_Engine` somewhere other than in the OpenSSL* repository.

9.2.2 Relevant Collateral

https://github.com/intel/QAT_Engine (including the Troubleshooting section).

9.3 Errors observed with openssl_speed

You see errors such as the following:

```
./openssl speed -engine qatengine -elapsed -evp aes-128-cbc-hmac-sha1 -
bytes 1400
17086849.27k
140242605822464:error:0607F08A:digital envelope
routines:EVP_EncryptFinal_ex:data not multiple of block
length:crypto/evp/evp_enc.c:405:
```

9.3.1 Resolution

Two issues are present here:

1. The buffer size should be in multiple of block length (for AES128 its 16) hence the error at software and engine which then leads to:
2. Reported throughput being invalid when issuing OpenSSL* speed commands, ensure buffer sizes are in multiple of block length. This is the default behavior for OpenSSL* speed.

In this case the extra parameter -bytes was used to manually set the buffer size.

§

10 HAProxy* Issues

This section describes resolution steps for HAProxy* issues.

10.1 HAProxy* + Intel® QAT Error when Starting HAProxy*

Starting HAProxy* results in the following message:
`"ssl-engine qat: failed to get structural reference"`

10.1.1 Resolution

Review the *HAProxy* with Intel® QuickAssist Technology Application Note* to verify that all required steps were covered.

10.1.2 Relevant Collateral

337430, HAProxy* with Intel® QuickAssist Technology Application Note, on 01.org.

10.2 HAProxy* + Intel® QAT Performance is Low

If you experience a low performance of HAProxy* and Intel® QAT, refer to the resolution steps below to isolate the issue.

10.2.1 Resolution

- Use the Intel® Select Solutions for NFVI script to reapply the correct settings (i.e., more worker processes, keep-alive settings, high concurrency, etc.)
- Ensure that Intel® QAT is being used, with the firmware counters
- Ensure that GRUB does not have idle=poll
- Isolating cores in the GRUB has been shown to reduce performance

10.2.2 Relevant Collateral

Intel® Select Solutions for NFVI

10.3 Error with HAProxy* Version

If you experience the following error:

```
# ./haproxy -vv

./haproxy: error while loading shared libraries: libssl.so.1.1: cannot
open shared object file: No such file or directory
```

It is likely that the LD_LIBRARY_PATH variable is not set up.

10.3.1 Resolution

Define the LD_LIBRARY_PATH and verify that the "Built with" and "Running on" OpenSSL* versions are the same.

```
]# export LD_LIBRARY_PATH=/usr/local/ssl/lib

]# ./haproxy -vv

HA-Proxy version 1.9.4 2019/02/06 -
https://haproxy.org/
```

10.3.1.1 Build Options

- TARGET = linux2628
- CPU = generic
- CC = gcc
- CFLAGS = -O2 -g -fno-strict-aliasing -Wdeclaration-after-statement -fwrapv -Wno-unusedlabel -Wno-sign-compare -Wno-unused-parameter -Wno-old-style-declaration -Wnoignored-qualifiers -Wno-clobbered -Wno-missing-field-initializers -Wtype-limits
- OPTIONS = USE_OPENSSL=1

10.3.1.2 Default settings:

- maxconn=2000, bufsize=16384, maxrewrite=1024, maxpollevents=200
- Built with OpenSSL* version: OpenSSL* 1.1.1 11 Sep 2018
- Running on OpenSSL* version: OpenSSL* 1.1.1 11 Sep 2018

10.3.2 Relevant Collateral

337430, HAProxy* with Intel® QuickAssist Technology Application Note, at 01.org, especially the following sections:

- Section 3.0, "HAProxy* Setup and Testing for HTTP Connections"
- Section 3.1, "Installing HAProxy*"
- Section 3.2, "Verifying HAProxy* Installation"

10.4 HAProxy* Shared Libraries libssl.so.1.1. and libcrypto.so.1.1 are Not Found

The HAProxy* shared libraries libssl.so.1.1. and libcrypto.so.1.1 are not found when running the command "ldd haproxy".

```
]# ldd haproxy          linux-vdso.so.1 =>  (0x00007ffe4853e000)
libcrypt.so.1 => /lib64/libcrypt.so.1 (0x00007ff32d26e000)
libdl.so.2 => /lib64/libdl.so.2 (0x00007ff32d06a000)
```

```
libpthread.so.0 => /lib64/libpthread.so.0 (0x00007ff32ce4e000)
librt.so.1 => /lib64/librt.so.1 (0x00007ff32cc46000)
libssl.so.1.1 => not found         libcrypto.so.1.1 => not found
libc.so.6 => /lib64/libc.so.6 (0x00007ff32c878000)         libfreebl3.so
=> /lib64/libfreebl3.so (0x00007ff32c675000)

/lib64/ld-linux-x86-64.so.2 (0x00007ff32d4a5000)
```

10.4.1 Resolution

Define the `LD_LIBRARY_PATH` variable and verify that the `libssl.so.1.1` and `libcrpto.so.1.1` files point to the correct libraries.

```
]# export LD_LIBRARY_PATH=/usr/local/ssl/lib ]# ldd haproxy
linux-vdso.so.1           =>         (0x00007ffd75bbf000)
 libcrypt.so.1 => /lib64/libcrypt.so.1 (0x00007feaeb0e4000)
libdl.so.2 => /lib64/libdl.so.2 (0x00007feaeaee0000)
libpthread.so.0   =>      /lib64/libpthread.so.0 (0x00007feaeacc4000)
librt.so.1 => /lib64/librt.so.1 (0x00007feaeaabc000)
libssl.so.1.1 => /usr/local/ssl/lib/libssl.so.1.1
(0x00007feaea82a000)          libcrypto.so.1.1   =>
/usr/local/ssl/lib/libcrypto.so.1.1 (0x00007feaea345000)
libc.so.6 => /lib64/libc.so.6 (0x00007feae9f77000)         libfreebl3.so
=> /lib64/libfreebl3.so (0x00007feae9d74000)         /lib64/ld-linux-x86-
64.so.2 (0x00007feaeb31b000)
```

10.4.2 Relevant Collateral

337430, HAProxy* with Intel® QuickAssist Technology Application Note, at 01.org, especially the following sections:

- Section 3.0, “HAProxy* Setup and Testing for HTTP Connections”
- Section 3.1, “Installing HAProxy*”
- Section 3.2, “Verifying HAProxy* Installation”

10.5 Fatal Errors with HAProxy* Configuration File

If you experience fatal errors with the HAProxy* configuration file, like the following:

```
#] ./haproxy -f /etc/haproxy/allhaproxy.cfg
[ALERT] 178/155753 (38095) : ssl-engine qat: failed to get structural
reference
[ALERT] 178/155753 (38095) : parsing [/etc/haproxy/allhaproxy.cfg:3] :
(null)
[ALERT] 178/155753 (38095) : Error(s) found in configuration file :
/etc/haproxy/allhaproxy.cfg [ALERT] 178/155753 (38095) : Fatal errors
found in configuration.
```

It is likely that the `LD_LIBRARY_PATH` variable is not set up.

10.5.1 Resolution

Run the following commands:

```
]# export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/ssl/lib
```

```
#] ./haproxy -f /etc/haproxy/allhaproxy.cfg
```

10.5.2 Relevant Collateral

337430, HAProxy* with Intel® QuickAssist Technology Application Note, at 01.org, especially the following sections:

- Section 3.0, "HAProxy* Setup and Testing for HTTP Connections"
- Section 3.1, "Installing HAProxy*."
- Section 3.2, "Verifying HAProxy* Installation."

10.6 HAProxy* Test Does not Appear to Produce the Expected Results using ApacheBench as a Load Generator

If you experience this issue, you may need to use the OpenSSL* `s_time` command as a load generator, with a new HAProxy Intel® QAT configuration file.

10.6.1 Resolution

An example of a recommended HAProxy* Intel® QAT configuration file is listed below for use when running the OpenSSL* `s_time` command. Please note that the **bold** line would be removed if you were running the test without Intel® QAT (i.e., with software).

```
]# cat myhaproxy-qat.cfg  global user root group root nbproc 15 maxconn
200000 ulimit-n 700000 daemon

ssl-engine qat algo ALL ssl-mode-async

ssl-default-bind-ciphers AES128-SHA

ssl-default-bind-options no-tls-tickets no-sslv3 no-tlsv10 no-tlsv11
tune.bufsize 65536 defaults backlog 327680 balance source retries 3

frontend myfrontend
mode http

bind 127.0.0.1:4400 ssl crt /etc/ssl/myhaproxy/myhaproxy.pem  option
forceclose option httpclose option http-server-close option nolinger
timeout client 100s

timeout client-fin 0s timeout http-keep-alive 0s default_backend
mybackend backend mybackend balance roundrobin option httpclose option
http-server-close timeout connect 100s

timeout server 100s
timeout server-fin 0s
option nolinger
option forceclose
mode http

timeout http-keep-alive 0s
```

```
server myvm 127.0.0.1:80 check
```

10.6.2 Relevant Collateral

337430, HAProxy* with Intel® QuickAssist Technology Application Note, at 01.org, especially the following sections:

- Section 3.0, "HAProxy* Setup and Testing for HTTP Connections"
- Section 3.1, "Installing HAProxy*."
- Section 3.2, "Verifying HAProxy* Installation."

10.7 Issues making ssl Connection against HAProxy* Launched with Intel® QAT Configured as Non-root User.

Note: You may be able to start HAProxy*, and everything is fine. Intel® QAT reports no warnings, but issues occur as soon as a request is made.

One example of debug output:

```
[DEBUG][qat_rsa.c:911:qat_rsa_priv_enc()] - Started.
[DEBUG][qat_rsa.c:403:build_decrypt_op_buf()] - Started
[DEBUG][qat_rsa.c:415:build_decrypt_op_buf()] flen = 256, padding = 3
[WARNING][qat_asym_common.c:112:qat_BN_to_FB()] Failed to allocate fb-
>pData
[WARNING][qat_rsa.c:460:build_decrypt_op_buf()] Failed to convert
privateKeyRep2 elements to flatbuffer
[WARNING][qat_rsa.c:944:qat_rsa_priv_enc()] Failure in
build_decrypt_op_buf [DEBUG][qat_rsa.c:210:rsa_decrypt_op_buf_free()] -
Started
[DEBUG][qat_rsa.c:233:rsa_decrypt_op_buf_free()] - Finished
```

Another example:

```
[DEBUG][qat_rsa.c:845:qat_rsa_priv_enc()] - Started.
[DEBUG][qat_rsa.c:369:build_decrypt_op_buf()] - Started
[DEBUG][qat_rsa.c:381:build_decrypt_op_buf()] flen = 256, padding = 3
[MEM_DEBUG][cmn_mem_drv_inf.c:87:qaeCryptoMemAlloc()] pthread_mutex_lock
[DEBUG][cmn_mem_drv_inf.c:95:qaeCryptoMemAlloc()] Address: (nil) Size:
128 File: qat_asym_common.c:104
[MEM_DEBUG][cmn_mem_drv_inf.c:99:qaeCryptoMemAlloc()]
pthread_mutex_unlock
[WARNING][qat_asym_common.c:107:qat_BN_to_FB()] Failed to allocate fb-
>pData
[WARNING][qat_rsa.c:426:build_decrypt_op_buf()] Failed to convert
privateKeyRep2 elements to flatbuffer
[WARNING][qat_rsa.c:872:qat_rsa_priv_enc()] Failure in
build_decrypt_op_buf [DEBUG][qat_rsa.c:209:rsa_decrypt_op_buf_free()] -
Started
[DEBUG][qat_rsa.c:232:rsa_decrypt_op_buf_free()] - Finished
```

10.7.1 Resolution

- The Intel® QAT Engine/`libqat` uses `usdm_drv and mmap()`'s physical memory regions it gets from the memory driver. On some distro's with `systemd`, non-root users have a `memlock` limit set by default to a too low value, and that triggers `mmap()'` error with `-EAGAIN`.

To see if this is the case, run:

- The Linux* command `strace` to see the error.
- See the `memlock` limit for your HAProxy* process.
- If `memlock` is your problem, set a bigger value, e.g., for your `haproxy.service` by adding an override .conf to it:

```
[Service]
LimitMEMLOCK=<some value, e.g, 16M>
```

10.7.2 Relevant Collateral

337430, HAProxy* with Intel® QuickAssist Technology Application Note, at 01.org, especially the following sections:

§

11 DPDK Issues

This section describes resolution steps for DPDK issues.

11.1 DPDK cryptodev failure

If you experience the following issue, please follow the resolution steps below: There is no Intel® QAT PMD available for the DPDK application.

- If you experience a DPDK `cryptodev` failure because there is no Intel® QAT PMD available for the DPDK application, please follow the resolution steps.

11.1.1 Resolution

- Quick instructions for Intel® QAT `cryptodev` PMD are as follows:

```
cd to the top-level DPDK directory make defconfig sed –
i 's,\(CONFIG_RTE_LIBRTE_PMD_QAT_SYM\)=n,\1=y,' build/.config or/and sed
-i
's,\(CONFIG_RTE_LIBRTE_PMD_QAT_ASYM\)=n,\1=y,'
build/.config make
```

11.1.2 Relevant Collateral

https://doc.dpdk.org/guides/cryptodevs/qat.html

§

12 Miscellaneous Issues

This section describes resolution steps for otherwise uncategorized issues.

12.1 Possible Errors Due to BIOS Setting

Issues like the following may be due to BIOS settings:

- Running `make install` on the Intel® QAT Engine returns an error similar to error -14:
  `dh895xcc: probe of 0000:b1:00.0 failed with error -14`

Note: The above result may be seen in `dmesg` and/or `/var/log/syslog`.

- Error "Failed to send admin msg to accelerator":
  `dh895xcc 0000:b1:00.0: Failed to send init message`

Note: The above result may be seen in `/var/log/messages`.

- Fewer qat acceleration devices than you expect when starting Intel® QAT:

For example, you may see all the c6xx type devices, but not the dh895x device.

12.1.1 Resolution

Please refer to Section 4.4 of *QuickAssist Technology Software for Linux* - Release Notes - H.W. version 1.7* (Document ID 336211). The title of the section is, "When trying to start the Intel QuickAssist Technology driver, I see errors similar to one of the following..."

12.1.2 Relevant Collateral

336211, Intel® QuickAssist Technology Software for Linux* – Release Notes – H.W. version 1.7

§