Intel® In-Memory Analytics Accelerator (Intel® IAA) User Guide

Document ID: 354834
June 2023

Notices & Disclaimers

Intel technologies may require enabled hardware, software, or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claims thereafter drafted which includes subject matter disclosed herein.

All product plans and roadmaps are subject to change without notice.

The products described may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exception that a) you may publish an unmodified copy, and b) code included in this document is licensed subject to the Zero-Clause BSD open source license (0BSD), https://opensource.org/licenses/0BSD. You may create software implementations, based on this document and in compliance with the foregoing, that are intended to execute on one or more Intel products referenced in this document. No rights are granted to create modifications or derivatives of this document.

REVISION HISTORY

Date	Revision	Description
June 2023	Initial Release	The initial release of the document.

GLOSSARY

Acronym	Term	Description
AECS	Analytics Engine Configuration and State	A data structure. • Passes configuration data that did not fit into the descriptor to the accelerator. • Passes state information between descriptor executions when a job consists of multiple descriptors.
DSA	Intel® Data Streaming Accelerator	An accelerator designed by Intel to accelerate streaming operations like memory copy.
QPL	Intel® Query Processing Library	An Intel library to interface between applications and the hardware.

CONTENTS

1. CONFIGURING ACCELERATORS

1.1 Introduction

Intel® Fourth Generation Xeon® scalable processors have multiple accelerators, including:

- Intel® Data Streaming Accelerator (Intel® DSA): for data movement.
- Intel® Dynamic Load balancer (Intel® DLB): for load balancing.
- Intel® In-Memory Analytics Accelerator (Intel® IAA): for compression/decompression and encryption/decryption.

This document provides instructions for configuring the Intel® In-Memory Analytics Accelerator (Intel® IAA), which offloads compression/decompression operations from the CPU.

1.2 Prerequisites

- Specific models of Intel® Fourth Generation Xeon® scalable processor or later, which include Intel® IAA.
- Linux Kernel with ENQCMD support enabled. For example, Linux Kernel 5.18 or later.
- Instructions are for CentOS Stream8 and Ubuntu Operating Systems.

2. BIOS SETUP FOR ENABLING THE ACCELERATORS

2.1 Entering the BIOS Setup

To enter the BIOS Setup using a keyboard (or emulated keyboard), press the F2 key during boot time when the OEM or Intel Logo Screen or the POST Diagnostic Screen is displayed.

1. Once in the BIOS menu, click **EDKII Menu.**

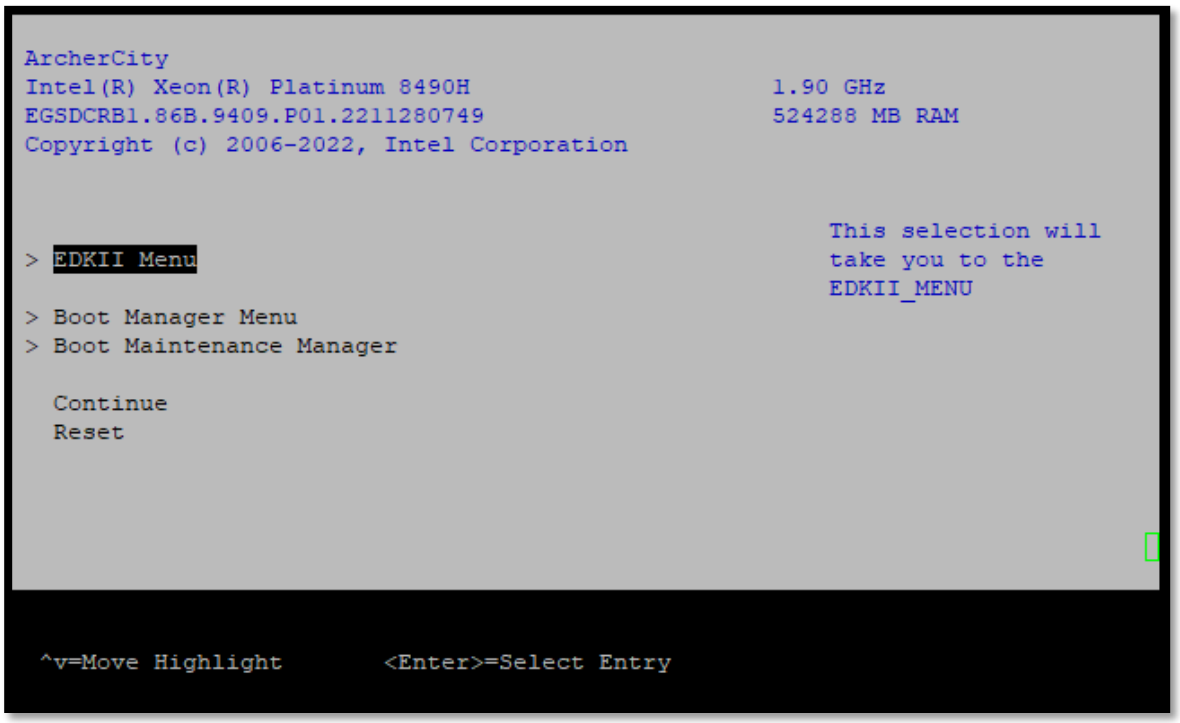

Figure 2-1: EDKII Menu within the BIOS menu.

2. Select Socket Configuration.

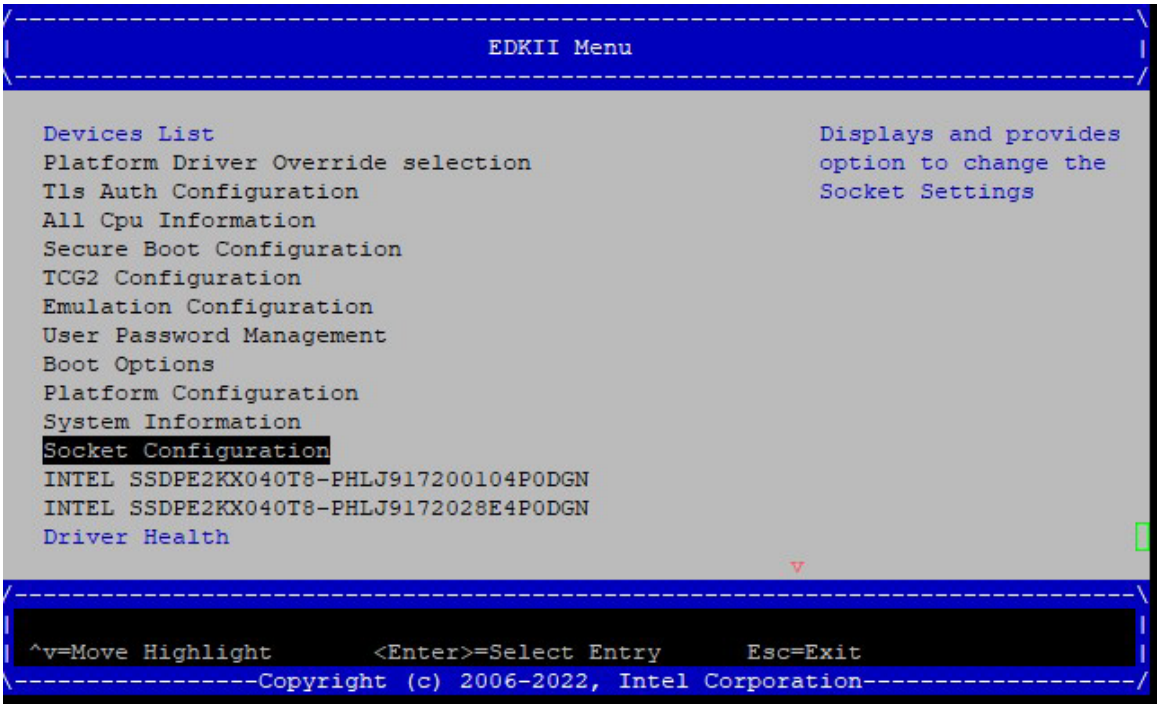

Figure 2-2: Select Socket Configuration

3. Select IIO Configuration.

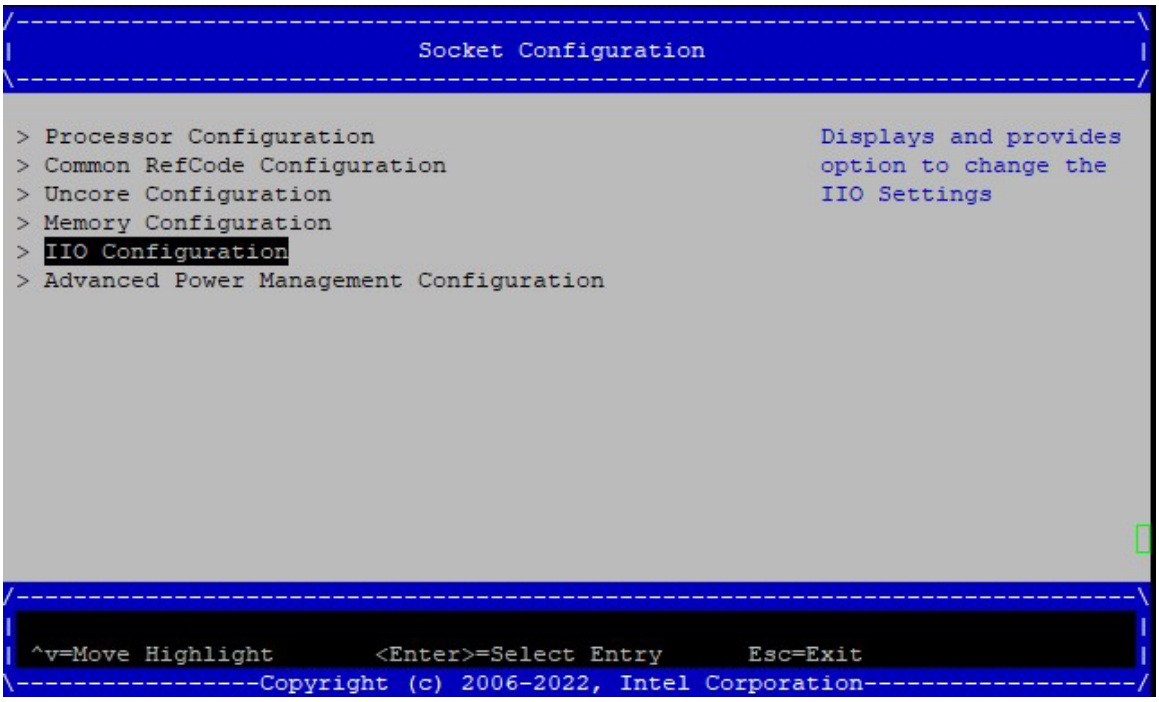

Figure 2-3: Select IIO Configuration

4. Select IOAT Configuration.

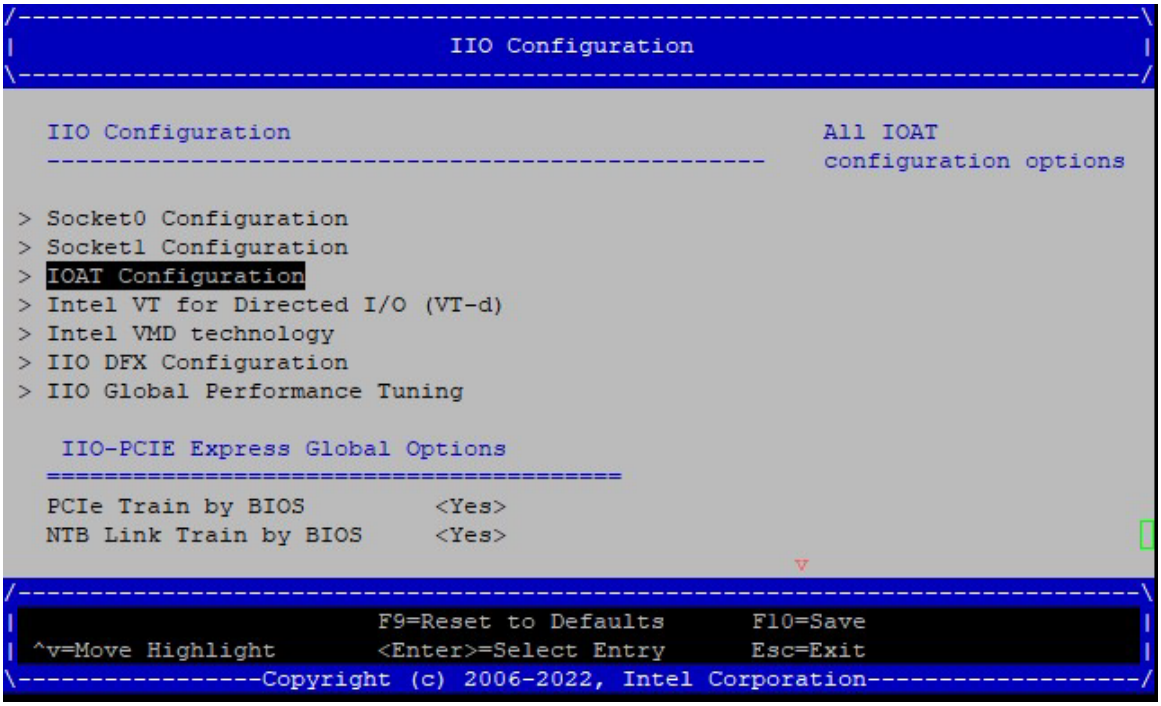

Figure 2-4: Select IOAT Configuration

5. Select Socket 0 (Sck0) IOAT Config.

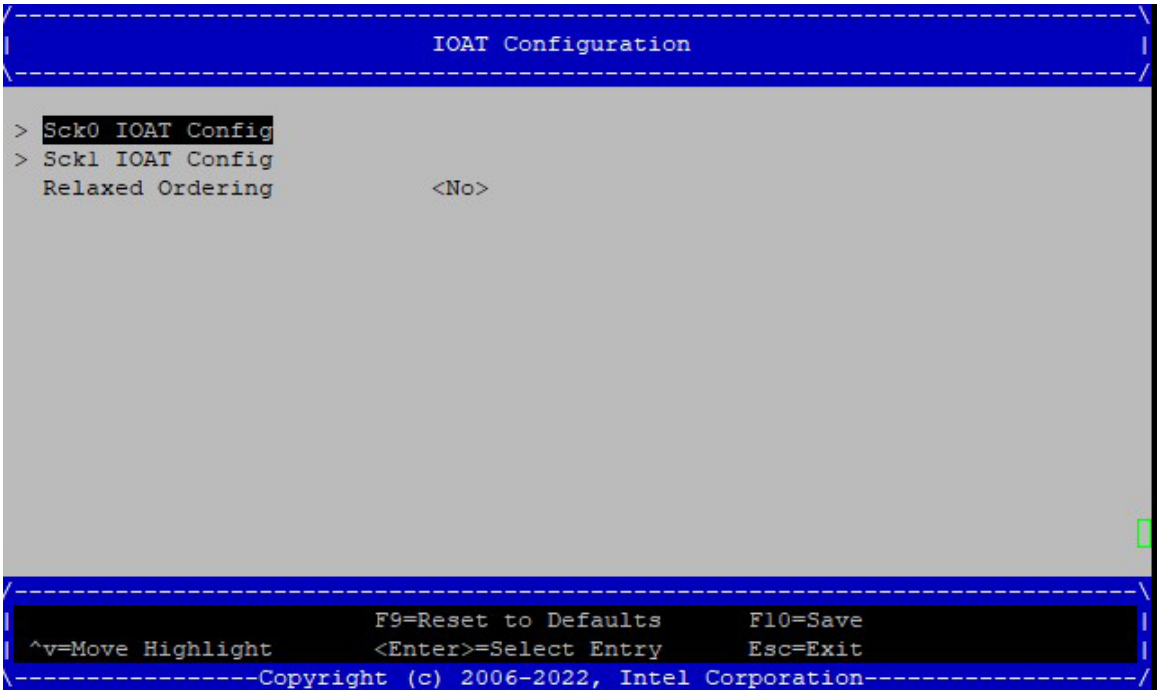

Figure 2-5: Select Socket 0 (Sck0) IOAT Config

6. Select **<Enable>** for IAX.

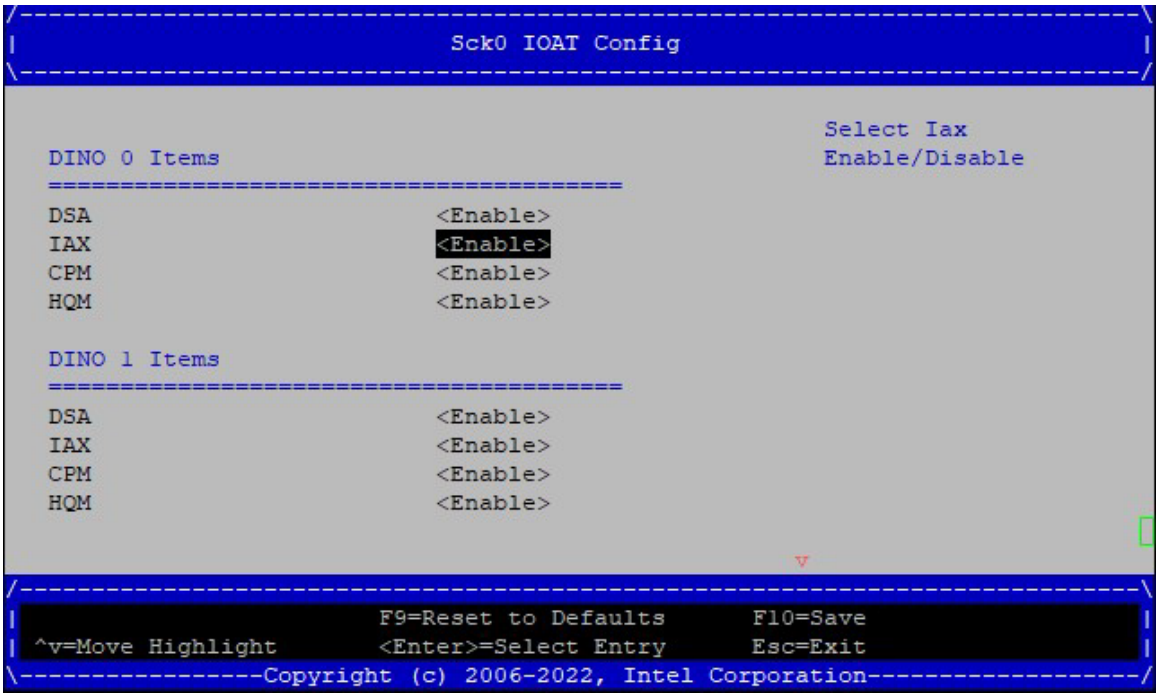

Figure 2-6: Select <Enable> for IAX on the (Sck0) IOAT Config Screen

3. ACCEL-CONFIG UTILITY

The Intel® Fourth Generation Xeon® Scalable processor accelerator devices are configured using a user-space utility called **accel-config**. This utility is available for installation via the accel-config-GitHub (https://github.com/intel/idxd-config)

3.1 Configure the Intel® In-Memory Analytics Accelerator Device

1. Enable Intel® IAA as in the BIOS Setup for Enabling the Accelerators. In addition to that, configure the following parameters.
    - EDKII Menu -> Socket Configuration -> Processor Configuration -> VMX: Enable
    - EDKII Menu -> Socket Configuration -> IIO Configuration -> Intel VT for Directed I/O (VT-d) -> Intel VT for Directed I/O: Enable
    - EDKII Menu -> Socket Configuration -> IIO Configuration -> Interrupt Remapping: Enable
    - EDKII Menu -> Socket Configuration -> IIO Configuration -> Opt-Out Illegal MSI Mitigation: Enable
        - Note: *Opt-Out Illegal MSI Mitigation* enablement depends on processor stepping and required for steppings older than E5.
2. Enable IOMMU in the kernel. For example,

```
$vim /etc/default/grub
# Add  intel_iommu=on,sm_on for the default kernel parameters
# For example, GRUB_CMDLINE_LINUX_DEFAULT=" intel_iommu=on,sm_on"
# save and exit from vim
# For CentOS
$grub2-mkconfig -o /boot/grub2/grub.cfg
$grub2-mkconfig -o /boot/efi/EFI/centos/grub.cfg
# For Ubuntu
$update-grub
$reboot
```

3. Download and install accel-config using Accel-config Utility instructions as in accel-config-GitHub (https://github.com/intel/idxd-config)

To configure the Intel IAA device

1. Disable all accelerator devices first.
2. Configure each engine and group, then assign them a group_id.
3. Configure one or more work queues (depending on device mapping).
4. Enable one or more accelerator devices and engine.

The following script summarizes the steps and provides an example. In this example, all available Intel IAA devices are configured as a separate work queue, and all engines in each device are configured by default. Devices can be configured in the “shared” or “dedicated” depending on the scheduling mode.

Please copy the following section of the code and create an executable file to run it.

```
#!/usr/bin/env bash

# Script configure IAA devices

# Usage : ./configure_iaa_user <mode> <start,end> <wq_size>
# mode: 0 - shared, 1 - dedicated
# devices: 0 - all devices or start and end device number.
#  For example, 1, 7 will configure all the Socket0 devices in host or 0, 3 will configure all the Socket0 devices in guest
#               9, 15  will configure all the Socket1 devices and son on
#               1  will configure only device 1
# wq_size: 1-128
#
# select iax config
#
dir="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"
#echo ${dir}
#
# count iax instances
#
iax_dev_id="0cfe"
num_iax=$(lspci -d:${iax_dev_id} | wc -l)
echo "Found ${num_iax} IAX instances"

dedicated=${1:-0}; shift
device_num=${1:-$num_iax}; shift
wq_size=${1:-128}; shift

if [ ${dedicated} -eq 0 ]; then
    mode="shared"
else
    mode="dedicated"
fi

#set first,step counters to correctly enumerate iaa devices
first=1 && step=2
```

```
#
# disable iax wqs and devices
#
echo "Disable IAX"

for ((i = ${first}; i < ${step} * ${num_iax}; i += ${step})); do
    echo disable wq iax${i}/wq${i}.0 >& /dev/null
    accel-config disable-wq iax${i}/wq${i}.0 >& /dev/null
    echo disable iax iax${i} >& /dev/null
    accel-config disable-device iax${i} >& /dev/null
done
echo "Configuring devices: ${device_num}"

if [ ${device_num} == $num_iax ]; then
    echo "Configuring all devices"
    start=${first}
    end=$(( ${step} * ${num_iax} ))
else
    echo "Configuring devices ${device_num}"
    declare -a array=($(echo ${device_num}| tr "," " "))
    start=${array[0]}
    if [ ${array[1]}  ];then
        end=$((${array[1]} + 1 ))
    else
        end=$((${array[0]} + 1 ))
    fi
fi
#
# enable all iax devices and wqs
#
echo "Enable IAX ${start} to ${end}"
for ((i = ${start}; i < ${end}; i += ${step})); do
    # Config  Engines and groups

    accel-config config-engine iax${i}/engine${i}.0 --group-id=0
    accel-config config-engine iax${i}/engine${i}.1 --group-id=0
    accel-config config-engine iax${i}/engine${i}.2 --group-id=0
```

```
    accel-config config-engine iax${i}/engine${i}.3 --group-id=0
    accel-config config-engine iax${i}/engine${i}.4 --group-id=0
    accel-config config-engine iax${i}/engine${i}.5 --group-id=0
    accel-config config-engine iax${i}/engine${i}.6 --group-id=0
    accel-config config-engine iax${i}/engine${i}.7 --group-id=0


    # Config WQ: group 0, size = 128, priority=10, mode=shared, type = user,
name=iax_crypto, threashold=128, block_on_fault=1, driver_name=user
    accel-config config-wq iax${i}/wq${i}.0 -g 0 -s $wq_size -p 10 -m
${mode} -y user -n user${i} -t $wq_size -b 1 -d user



    echo enable device iax${i}
    accel-config enable-device iax${i}
    echo enable wq iax${i}/wq${i}.0
    accel-config enable-wq iax${i}/wq${i}.0
done
```

Verify if the devices are configured correctly using the `accel-config list`.

For the sake of simplicity, the expected results from a single Intel IAA device are shown. The display for the remaining Intel IAA devices will be the same since they all are configured as in the preceding script. Intel IAA devices are counted as odd numbers, while Intel® Data Streaming Accelerator (Intel® DSA) devices are counted as even numbers.


```
{
 {
    "dev":"iax1",
    "max_groups":4,
    "max_work_queues":8,
    "max_engines":8,
    "work_queue_size":128,
    "numa_node":0,
    "op_cap":[
      "0xd",
      "0x7f331c",
      "0",
      "0"
    ],
    "gen_cap":"0x71f10901f0105",
```

```
  "version":"0x100",
  "state":"enabled",
  "max_batch_size":1,
  "max_transfer_size":2147483648,
  "configurable":1,
  "pasid_enabled":1,
  "cdev_major":237,
  "clients":1,
  "groups":[
    {
      "dev":"group1.0",
      "traffic_class_a":1,
      "traffic_class_b":1,
      "grouped_workqueues":[
        {
          "dev":"wq1.0",
          "mode":"shared",
          "size":128,
          "group_id":0,
          "priority":10,
          "block_on_fault":0,
          "max_batch_size":32,
          "max_transfer_size":2097152,
          "type":"kernel",
          "name":"iaa_crypto",
          "driver_name":"crypto",
          "threshold":10,
          "ats_disable":0,
          "state":"enabled",
          "clients":1
        }
      ],
      "grouped_engines":[
        {
          "dev":"engine1.0",
          "group_id":0
        },
```

```
        {
          "dev":"engine1.1",
          "group_id":0
        },
        {
          "dev":"engine1.2",
          "group_id":0
        },
        {
          "dev":"engine1.3",
          "group_id":0
        },
        {
          "dev":"engine1.4",
          "group_id":0
        },
        {
          "dev":"engine1.5",
          "group_id":0
        },
        {
          "dev":"engine1.6",
          "group_id":0
        },
        {
          "dev":"engine1.7",
          "group_id":0
        }
      ]
    },
   {
      "dev":"group1.1",
      "traffic_class_a":1,
      "traffic_class_b":1
    },
    {
      "dev":"group1.2",
```

```
        "traffic_class_a":1,
        "traffic_class_b":1
      },
      {
        "dev":"group1.3",
        "traffic_class_a":1,
        "traffic_class_b":1
      }
    ]
  },
  { …. Next device … and son on
  }
}
```

3.2 Operation Offloading to IAA Devices: Intel® Query Processing Library (Intel® QPL)

The Intel® Query Processing Library (Intel® QPL) can improve the performance of databases, enterprise data, communications, and scientific/technical applications. Intel QPL provides interfaces for several commonly used algorithms. This library enables your application to tune automatically with generations of processors without unnecessary changes to the application. The Intel QPL provides high-performance implementations of data processing functions for an existing hardware accelerator and/or software path if no hardware accelerator is available. Code written with the library automatically takes advantage of available modern CPU capabilities. This can provide tremendous development and maintenance savings. The goal of the Intel QPL is to provide an application programming interface (API) with the following:

- C and C++-compatible interfaces and data structures to enhance usability and portability.
- Faster time to market.
- Scalability with Intel® In-Memory Analytics Accelerator (Intel® IAA) hardware.

More details on Intel QPL use are available in the QPL GitHub (https://intel.github.io/qpl/index.html)

4. ADDITIONAL RESOURCES

Resource Name	Description
Intel® In-Memory Analytics Accelerator (IAA) Architecture Specification	This document describes the architecture of the Intel® In-Memory Analytics Accelerator (Intel® IAA).