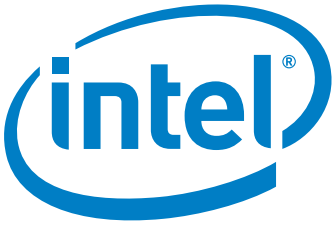

Intel® QuickAssist Technology Software for Linux*

Getting Started Guide

July 2020

Document Number: 336212-007

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 claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

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.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting: http://www.intel.com/design/literature.htm.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at http://www.intel.com/ or from the OEM or retailer.

No computer system can be absolutely secure.

Intel, Intel Atom, Xeon, and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2020, Intel Corporation. All rights reserved.

Contents

Tables

Revision History

Date	Revision	Description
July 2020	007	Minor edits to Appendix A
June 2019	006	Minor edits.
March 2019	005	Added instructions for installing software using yum.
December 2018	004	Updated configuration details.
June 2018	003	Updated information on build instructions and software dependencies.
August 2017	002	Added support for Intel Xeon Processor D Family devices.
July 2017	001	Initial public release.

§

1.0 Introduction

1.1 About This Manual

This getting started guide documents the instructions to obtain, build, install and exercise the Intel® QuickAssist Technology Software for Hardware Version 1.7 package. Additionally, this document includes brief instructions on configuring the supported development board.

Note: The software described in this document relates to a platform that pairs the Intel® C62x Chipset (also known as Platform Controller Hub, or PCH) with Intel® Xeon® Processor D Family System-on-a Chip (SoC) or Intel Atom® C3000 Processor Product Family SoC. While the bulk of the software relates to the PCH or SoC, some components such as the embedded drivers are platform-specific.

In this document, for convenience:

- *Software package* is used as a generic term for the Intel® QuickAssist Technology Software for Hardware Version 1.7 package.
- *Acceleration drivers* is used as a generic term for the software that allows the Intel® QuickAssist Software Library APIs to access the Intel® QuickAssist Accelerator(s) integrated in the PCH and SoC.

Note: The software package also works on the Intel® Communications Chipset 8925 to 8955 Series.

Sections specific to all covered products include:

- Section 2.0, “Installing the Operating System” on page 11
- Section 3.0, “Building and Installing Software” on page 15
- Section 4.0, “Sample Applications” on page 21

Sections specific to the Intel Atom® C3000 Processor Product Family SoC include:

- Section 5.0, “Installing, Building, and Running Yocto*” on page 25

1.2 Additional Information on Software

The software release package for Linux* has been validated with CentOS* 7 x86_64. It has also been validated with Yocto* for the Intel Atom® Processor 3000 SoC.

Collateral can be found on https://01.org/intel-quickassist-technology

1.2.1 Accessing Additional Content from the Intel Business Portal

1. In a web browser, go to www.intel.com/ibl.
2. Enter your login ID in the **Login ID** box. Check **Remember my login ID** only if you are not using a shared computer. Click **Submit**.

   Note: To acquire a new Intel Business Portal account, please contact your Intel Field Sales Representative.
3. Enter your password in the **Password** box. Click **Submit**.

For the Intel® C62x Chipset PCH:

4. Within the design kit categories, under the **Platform & Solutions** heading, click **Server and Workstation**. Under the **Products** heading, click **Server and Workstation Platforms Codenamed Purley, including Skylake Server and Cannonlake Server processors, with Lewisburg PCH** then, under the Associated Collateral Lists heading, click **Server and Workstation Platforms: Purley - Lewisburg Chipset Intel QuickAssist Technology Software.**

For the Intel Atom® Processor 3000 SoC:

- Within the Design Kit Categories, under the **Platform & Solutions** heading, click **Embedded**.
- Under the **Pre-Launch Products** heading, click **Embedded Platform Code Named Denverton-NS**. Under the **Associated Collateral Lists** heading, click **Microserver Platform Code Named Harrisonville - Denverton and Denverton-NS Intel QuickAssist Technology Software**.

1.2.2 Product Documentation

Table 1 lists the documentation supporting this release. All documents can be accessed as described in Section 1.2.1, "Accessing Additional Content from the Intel Business Portal".

Table 1. Product Documentation and Software

Title	Number/Location
Intel® QuickAssist Technology Software for Linux - Release Notes - Hardware Version 1.7*	336211 01.org
Intel® QuickAssist Technology Software for Linux - Getting Started Guide - Hardware Version 1.7* (this document)	336212 01.org
Intel® QuickAssist Technology Software for Linux - Programmer Guide - Hardware Version 1.7*	336210 01.org
Intel® QuickAssist Technology Software for Linux - Software Drivers - Hardware Version 1.7*	562366 01.org
Intel® QuickAssist Technology Cryptographic API Reference Manual	330685 01.org
SoC Yocto BSP - PV	565774
Electronic Design Kit	565762
Intel® QuickAssist Technology API Programmer's Guide	330684 01.org
Intel® QuickAssist Technology Data Compression API Reference Manual	330686 01.org
Intel® QuickAssist Technology - Performance Optimization Guide	330687 01.org
Using Intel®Virtualization Technology (Intel® VT) with Intel® QuickAssist Technology Application Note	330689 01.org

1.3 Related Software and Documentation

Refer to the Development Kit User Guide for your hardware for additional information on the development board including board layout, components, connectors, jumpers, headers, power and environmental requirements, and pre-boot firmware.

Follow the directions in Section 1.2.1, "Accessing Additional Content from the Intel Business Portal" to locate this collateral.

1.4 Conventions and Terminology

The following conventions are used in this manual:

- `Courier` font - code examples, command line entries, API names, parameters, filenames, directory paths, and executables
- **Bold** text - graphical user interface entries and buttons
- *Italic* text – key terms and publication titles

The following terms and acronyms are used in this manual:

Table 2. Terminology

Term	Definition
API	Application Programming Interface
BIOS	Basic Input/Output System
BOM	Bill of Materials
CY	Cryptography
DC	Data Compression
GRUB	GRand Unified Bootloader
OS	Operating System
PCH	Platform Controller Hub
PCI	Peripheral Component Interconnect
PF	Physical Function
PKE	Public Key Encryption
Intel® QAT	Intel® QuickAssist Technology
SoC	System-on-a-Chip
SRIOV	Single Root-I/O Virtualization
VF	Virtual Function

1.5 Software Overview

The software is described in the following topics:

- Section 1.5.1, “Features Implemented” on page 9
- Section 1.5.2, “List of Files in Release” on page 9
- Section 1.5.3, “Package Release Structure” on page 9

1.5.1 Features Implemented

Note: For feature details and limitations, if any, refer to the release notes.

1.5.2 List of Files in Release

A Bill of Materials (BOM) is included as a text file in the software package(s).

1.5.3 Package Release Structure

After unpacking the tar file, the directory should contain:

Table 3. Package Release Structure

Files/Directory	Comments
`QAT1.7.Upstream.L.<version>.tar.gz`	Top-level Intel® QAT package
`./filelist`	List of files in this package

Table 3. Package Release Structure

Files/Directory	Comments
`./LICENSE.GPL`	License file
`./versionfile`	Version file
`./quickassist`	Top-level acceleration software directory

§

2.0 Installing the Operating System

The section describes the process of obtaining, installing, and configuring the operating system (OS) on the development board.

Although this document describes how to install and configure CentOS*7, Intel® QuickAssist Technology can work with other Linux* distributions, such as Ubuntu* and Fedora*. Refer to notes in Section 3.2.1 for specific installation commands.

2.1 Acquiring CentOS* 7

Note: CentOS* 7 is based on Red Hat Enterprise Linux* 7.3 (or later) and may also be referred to as CentOS 7.3. In general, using the latest version of CentOS 6 or CentOS 7 is recommended, though it is possible that changes to the Linux kernel in the most recent versions may break some functionality for particular releases.

CentOS 7 is a Linux distribution built on free and open source software. The software package from Intel does not include a distribution of CentOS or any other Linux variant. The software package includes Linux device driver source developed by Intel.

CentOS 7 x86_64 can be obtained from: https://www.centos.org/download.

Note: This document is written with the CentOS 7 DVD Install Media in mind. Using any Live Media versions is not recommended.

2.2 Configure the BIOS

Update to the latest stable BIOS for your platform.

If the performance achieved from the acceleration software is not meeting the advertised capability, some BIOS changes (or other changes) may be required:

- Set the links to train to the highest possible speed, e.g., PCIe* Gen3 instead of PCIe Gen2 or Gen1.
- Ensure that the link has trained to the expected width, e.g., x8 or x16.
- Disable some CPU power-saving features.

Performance numbers matching the expected performance may not be achievable for all platform configurations or with the default configuration files. Refer to the *Intel® QuickAssist Technology - Performance Optimization Guide* (Table 1) for more information on achieving best performance.

In the BIOS setup, set the first boot device to be the DVD-ROM drive and the second boot device to be the drive on which CentOS* 7 is to be installed.

2.3 Installing CentOS* 7

Note: If you encounter issues installing the operating system, refer to Appendix A, "Avoiding Kernel Crashes with PCH Intel® SKUs" for a possible resolution.

For complete additional (and non-standard) CentOS* installation instructions, refer to the online installation guide at:

https://wiki.centos.org/HowTos

This section contains basic installation instructions. For the purposes of this getting started guide, it is assumed that the installation is from a DVD image.

Note: If the hard drive already has an operating system, some of the following steps may be slightly different.

1. When the development board starts, it should begin booting from the CentOS 7 installation disc. If not, verify the Boot Order settings described in Section 2.2, "Configure the BIOS" on page 11.
2. At the welcome prompt, select **Test this Media & Install CentOS 7** and click **Enter**.
3. Select **OK** to begin testing the installation media.

Note: It is recommended that the CentOS 7 installation disc is verified prior to an installation of the OS.

4. After verifying the CentOS 7 installation disc(s), the graphical portion of the installation is loaded. Click **Next** to continue the installation process.
5. Update **DATE & TIME** options if required, including the time zone, network time (if desired).
6. **SOFTWARE SELECTION**. For the best evaluation experience and to avoid any build issues with the acceleration software package, it is recommended to select "Development and Creative Workstation" as the "Base Environment". Select the following "Add-Ons for Selected Environment" as well: "Additional Development", "Development Tools", and "Platform Development". Also select "Virtualization Hypervisor" if virtualization is required (and supported by the acceleration software package).
7. Select **INSTALLATION DESTINATION**. If the correct target device is not selected, select it. Click Done.
8. Select **NETWORK & HOST NAME**. In most cases, changing the Ethernet connection from "OFF" to "ON" is desired. Also set the Host name, if required. Click **Done**. Ensure DHCP is selected.
9. Once all items on the **INSTALLATION SUMMARY** are configured, select **Begin Installation**.
10. Set the root password and create a user. When creating a user, select "Make this user administrator" to enable sudo. Select **Done** and wait for the installation to complete.
11. When the installation completes, the install DVD should be ejected. Remove the DVD and select **Reboot** when prompted.

Note: If you encounter issues booting after installing the operating system, refer to Appendix A, "Avoiding Kernel Crashes with PCH Intel® SKUs" for a possible resolution.

When the installation completes, continue with Section 2.4, "Updating grub Configuration File" on page 13.

2.4 Updating grub Configuration File

This section contains instructions on updating the Grand Unified Bootloader (grub) configuration file.

Note: Root access is required to make grub changes.

If the acceleration software will be used with a virtualized (SRIOV) environment (if supported by the acceleration software package), update grub to add `intel_iommu=on` to the boot options, using the following guide:

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/ch-working_with_the_grub_2_boot_loader

Consult the following document for more information on using the acceleration software in a virtualized environment: *Using Intel®Virtualization Technology (Intel® VT) with Intel® QuickAssist Technology Application Note* (refer to Section 1.2.2, "Product Documentation" on page 8).

Using the boot flag `intel_iommu=on` prevents using the QAT physical function (PF) on the host. To use Intel® QAT on the host with this flag, refer to the virtualization app note cited above for the instructions to use Intel® QAT virtual functions (VFs) on the host.

2.5 Configuring Linux*

Once the operating system is installed, a few configuration items may need to be completed, such as updating the yum configuration files. This section describes these items.

2.5.1 Updating yum Configuration Files

yum is an application that can be used to perform operating system updates. To use yum in a corporate network, the following change may be required.

2.5.1.1 /etc/yum.conf

If the system needs to connect to the internet through a corporate firewall, yum needs to be updated to use the proxy server. Add a line similar to the following in the `/etc/yum.conf` file. The line can be added to the end of the file. Contact your network administrator for details on the proxy server.July 2020

```
proxy=http://<proxy_server:portnum>
```

where `<proxy_server:portnum>` is replaced with your server information.

2.6 System Security Considerations

This section contains a high-level list of system security topics. Specific OS/filesystem topics are outside of the scope of this document. For more information, refer to the programmer guide for your platform, specifically the Secure Architecture Considerations section.

Securing your operating system is critical. Consider the following items:

Note: This is not an exhaustive list.

- Employ effective security policies and tools; for instance, SELinux* is configured correctly and is active.
- Run and configure the firewall(s).
- Prevent privilege escalation at boot (including recovery mode); for instance, set a grub password. Additional details are described below.
- Remove unnecessary software packages.
- Patch software in a timely manner.
- Monitor the system and the network.
- Configure and disable remote access, as appropriate.
- Disable network boot.
- Require secure passwords.
- Encrypt files, up to full-disk encryption.
- Ensure physical security of the system and the network.
- Use `mlock` to prevent swapping sensitive variables from RAM to disk.
- Zero out sensitive variables in RAM.

§

3.0 Building and Installing Software

This chapter provides details on building and installing the software on the development kit.

3.1 Unpacking the Software

The software package comes in the form of a tarball. Refer to Section 1.2.1, "Accessing Additional Content from the Intel Business Portal" for the software location.

The software package can also be installed under Linux using `yum`. Refer to Section 3.3, "Package Installation Using yum" for additional information.

The instructions in this document assume that you have super user privileges.

```
# su
<enter password for root>
```

1. Create a working directory for the software. This directory can be user defined, but for the purposes of this document, a recommendation is provided.

   ```
   # mkdir /QAT
   # cd /QAT
   ```

Note: In this document, the working directory is assumed to be `/QAT`. This directory is the `ICP_ROOT`.

2. Transfer the tarball to the development board using any preferred method, for example USB memory stick, CDROM, or network transfer in the `/QAT` directory. Unpack the tarball using the following command:

   ```
   # tar -zxof <QAT tarball name>
   ```

3. Restricting access to the files is recommended:

   ```
   # chmod -R o-rwx *
   ```

Result: The package is unpacked and the installation script and other items are created in the `/QAT` directory. Refer to Section 1.5.3, "Package Release Structure" on page 9.

3.2 Installation Overview

The installation procedure handles a number of tasks that would otherwise have to be done manually, including the following:

- Create the shared object (`.so`) files by building the source code.
- Copy the shared object (`.so`) files to the right directory (e.g., `/lib` or `/lib64`).
- Build adf_ctl and copy it to the right directories (`$ICP_ROOT/build and /usr/sbin`).
- Copy the config files to `/etc`.
- Copy the firmware files to `/lib/firmware`.
- Copy the modules to the appropriate kernel source directory for loading by qat_service.
- Start the qat_service, which inserts the appropriate modules as required and runs `adf_ctl` to bring up the devices.
- Set up the qat_service to run on future boots (copy to `/etc/init.d`, run `chkconfig` to add the service).

On recent Linux* kernels, there is an upstreamed version of the Intel® QuickAssist Technology driver, and it will interfere with the loading of the driver included with the software package assumed in this document. The `qat_service` accounts for this by removing the upstreamed kernel modules, but if `qat_service` is not used, errors may be displayed when trying to load the driver.

3.2.1 Installation Commands

Note: If the OS was not installed with the correct software packages (refer to Section 2.3, "Installing CentOS* 7" on page 12), build error messages appear during the acceleration install. If CentOS was not installed correctly, reinstall the OS and select the correct **SOFTWARE SELECTION** option as described in Section 2.3, "Installing CentOS* 7" on page 12, or run the following commands:

```
# yum -y groupinstall "Development Tools"
# yum -y install pciutils
# yum -y install libudev-devel
# yum -y install kernel-devel-$(uname -r)
# yum -y install gcc
# yum -y install openssl-devel
```

Note: If you are installing Ubuntu*, run these commands instead:

```
apt-get update
apt-get install pciutils-dev
apt-get install g++
apt-get install pkg-config
apt-get install libssl-dev
```

Note: If you are installing a recent distribution of Fedora*, run these commands instead:

```
dnf groupinstall "Development Tools"
dnf install gcc-c++
dnf install systemd-devel
dnf install kernel-devel-`uname -r`
dnf install elfutils-devel
dnf install openssl-devel
```

3.2.2 Starting the Driver

To start the driver, use the qat_service, or rmmod the upstreamed modules (`qat_*`, `intel_qat`) and insert the modules built with the software package assumed in this document before starting the driver.

The acceleration software package supports the standard Linux software installation process.

```
# ./configure [OPTION]... [VAR=VALUE]
# make
# make install
```

Run the following command to see the list of available options:

```
# ./configure --help
```

Note: If you are installing an Intel® C629 Series Chipset or any Intel® device that does not support Public Key Encryption (PKE), enter the following configuration option:

```
# ./configure --enable-icp-dc-sym-only
```

This option enables the data compression and symmetric code services and disables cryptographic services.

Note: To enable the Intel® QuickAssist API in Kernel space, enter the following configuration option:

```
# ./configure --enable-kapi
```

Note: Current implementation of Intel® QuickAssist API for Kernel space only supports Symmetric cryptography and Data compression services.

3.2.3 Installation Procedure

When installing acceleration software on a system that had a previous or modified version of the acceleration software installed, it is strongly recommended to uninstall the previous acceleration software first, using `make uninstall` in the acceleration software package.

1. Open a Terminal Window and switch to superuser.

   ```
   # su
   <enter root password>
   # cd /QAT
   ```

2. Enter the following commands to build and install the acceleration software and sample code using default options:

   ```
   # ./configure
   # make install
   # make samples-install
   ```

Note: To build 64-bit kernel components and 32-bit user-space components, run the `make install` command with the following build parameter:

```
# make ICP_ARCH_USER=i686 install
```

Note: After building/installing the acceleration software, secure the build output files by either deleting them or setting permissions according to your needs.

Note: The messages "Can't read private key" can be safely ignored. These are generated because the modules are not signed by the private key of OS distribution.

3. After installing the Acceleration Software, it is recommended to verify that the acceleration software kernel object is loaded and ready to use.

```
# lsmod | grep qa
```

Depending on the specific hardware present, this command returns something similar to the following:

```
# lsmod | grep qa
qat_c62x               13581  0
qat_dh895xcc           13581  0
intel_qat             141557  3 usdm_drv,qat_dh895xcc,qat_c62x
authenc                17776  1 intel_qat
uio                    19259  1 intel_qat
```

Not all modules are required, depending on the specific hardware present. If the acceleration software is not installed, all of these modules are typically not present.

Note: If the `./configure` command would have been run with option `--enable-kapi`, the QuickAssist API Kernel module would be loaded as part of the installation and the console command `# lsmod | grep qa` would return something similar to:

```
# lsmod | grep qa
qat_api               561152  0
qat_c62x               20480  0
intel_qat             225280  3 qat_c62x,qat_api,usdm_drv
authenc                16384  1 intel_qat
dh_generic             16384  1 intel_qat
uio                    20480  1 intel_qat
```

The Kernel module `qat_api` exports the Intel® Quickassist APIs as symbols allowing a Kernel space application to use them.

Applications need to make use of the static library (`libqat.a`) or the shared object (`libqat_s.so`). If the installation procedure described in this chapter is not used, the shared object needs to be copied manually, or other steps (e.g., setting `LD_LIBRARY_PATH`) need to be taken to link to this file.

Check `/var/log/messages` or `dmesg` to make sure that the acceleration service started. Warning messages related to `Invalid core affinity`can be addressed by modifying the configuration files so that no core numbers are referenced beyond the core count of the system. Refer to Section 3.5, "Configuration Files" on page 20 for more detail.

Once the installation/building is complete, proceed to Section 4.0, "Sample Applications" on page 21 to execute applications that exercise the software.

3.3 Package Installation Using yum

The Intel® QAT software package can be installed using the standard Linux `yum` command on operating systems that support this command. Using `yum` will identify and install any missing required packages prior to installation of the package and then setup the software for non-root access.

Note: Users without root access must be added to the `qat` group.

The following steps are required:

1. Update packages on the system.

   ```
   # sudo yum update
   ```

2. Create a file named `intel-qat.repo` in `/etc/yum.repos.d` with the following contents:

   ```
   [intel-qat]

   name=Intel QAT

   baseurl=https://download.01.org/QAT/repo

   gpgcheck=0
   ```

3. Clean the `yum` cache by running:

   ```
   # sudo yum clean  all
   ```

4. Install the acceleration software using the command:

   ```
   # sudo yum install QAT
   ```

5. Ensure the acceleration software is running by using the command:

   ```
   # sudo service qat_service status
   ```

To remove the acceleration software, use the command:

```
# sudo yum remove QAT
```

3.4 Starting/Stopping the Acceleration Software

When the acceleration software is installed, a script file titled `qat_service` is installed in the `/etc/init.d` directory.

The script file can be used to start and stop the acceleration software. To start the software, issue the following command:

```
# service qat_service start
```

Note: If the `service qat_service start` command fails, verify the following:

- Software is installed.
- Acceleration software is already running.
- For the Platform Controller Hub, verify the device is enumerated properly using the `lspci` command:
  ```
  # lspci -d 8086:37c8
  ```
- For the Intel Atom® C3000 Processor SoC, verify the device is enumerated properly using the `lspci` command:
  ```
  # lspci -d 8086:19e2
  ```
- For the Intel® Xeon® Processor D Family SoC, verify the device is enumerated properly using the `lspci` command:
  ```
  # lspci -d 8086:6f54
  ```

To stop the software, issue the following command:

```
# service qat_service stop
```

To stop the software and remove the kernel driver, issue the following command:

```
# service qat_service shutdown
```

When the acceleration software is installed, it is set to load automatically when the operating system loads.

3.5 Configuration Files

When the Acceleration software loads, it is configured based on settings in the platform-specific configuration files. The configuration files are in the `/etc` directory. Specifically:

- The name of the first configuration file for Intel® Communications Chipset 8925 to 8955 Series devices is `dh895xcc_dev0.conf`.
- The first configuration file for the Platform Controller Hub is `c6xx_dev0.conf`.
- The first configuration file for Intel Atom® C3000 Processor SoC is `c3xxx_dev0.conf`.
- The first configuration file for Intel ® Xeon® Processor D SoC is `d15xx_dev0.conf`.

Note: If more than one device of a given type is present, its name includes '`dev1`', '`dev2`', etc.

The files are processed when the system boots. If changes are made to the configuration file, the Acceleration software must be stopped and restarted for the changes to take effect. Refer to Section 3.2.3, "Installation Procedure" on page 17 for detailed instructions.

The software package includes multiple types of platform-specific configuration files. Depending on your installation options and SKU, a valid configuration file is copied to the `/etc` directory. If your system has more than one type of hardware device or SKU, verify that the correct configuration files were copied.

Note: The software package has been validated with the default configuration files. Changes to the configuration files could have adverse effects.

Refer to the Programmer's Guide for your platform for additional information on the configuration files.

§

4.0 Sample Applications

This section describes the sample code that can be executed on the target platform along with instructions on their usage.

4.1 Intel® QuickAssist Accelerator Sample Application

The software package contains a set of sample tests that exercises acceleration functionality. This section describes the steps required to build and execute the sample tests.

The sample application is provided for the user space.

4.1.1 Compiling the Acceleration Sample Code

Note: These instructions assume the software package was untarred in the /QAT directory and the kernel source files were placed in the directory specified in this guide.

1. Open a Terminal Window and switch to superuser:

```
# su
<enter root password>
```

Note: For details on running user space applications as non-root user, refer to the "Running Applications as Non-Root User" section in the applicable programmer guide (refer to Section 1.2.2, "Product Documentation" on page 8).

2. Switch to the `/QAT` directory and compile the installation samples.

```
# cd /QAT
# make samples-install
```

This compiles the acceleration sample code for user space. It also compiles the memory mapping driver used with the user space application.

Proceed to "Running the chained hash and compression test code requires:" on page 23 for instructions on executing the tests.

4.1.2 Loading the Sample Code

Note: In user space, before launching the `cpa_sample_code` application, the environmental variable `LD_LIBRARY_PATH` may need to be set to the path where `libqat_s.so` is located. This may be `/usr/local/lib` or `/QAT/build`.

The acceleration kernel module must be installed and the software must be started before attempting to execute the sample code. This can be verified by running the following commands:

```
# lsmod | grep "qa"
# service qat_service status
```

Typical output is similar to the following:

```
# 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, bsf: 88:00.0, #accel: 5 #engines: 10 state: up
 qat_dev1 - type: c6xx, inst_id: 1, bsf: 8a:00.0, #accel: 5 #engines: 10 state: up
 qat_dev2 - type: c6xx, inst_id: 2, bsf: 8c:00.0, #accel: 5 #engines: 10 state: up
```

Note: If the module is not returned from the first command, refer to Section 3.2.3, "Installation Procedure" on page 17 for additional information on starting the Acceleration software.

In user space the sample code is executed with the command: `./build/cpa_sample_code`

In Kernel space the sample code is executed with the command: `insmod ./build/cpa_sample_code.ko`

Note: The output of the cpa_sample_code application is available in dmesg.

Should the user decide to run the sample code more than once, unload the `cpa_sample_code` module as follows before rerunning it:

```
rmmod cpa_sample_code
```

The application allows the run-time parameters listed below.

Table 4. Sample Code Parameters (Sheet 1 of 2)

Parameter	Supported in user space	Supported in kernel space	Description
cyNumBuffers=w	Yes	Yes	Number of buffers submitted for each iteration. (default=20)
cySymLoops=x	Yes	Yes	Number of iterations of all symmetric code tests. (default=5000)
cyAsymLoops=y	Yes	No	Number of iterations of all asymmetric code tests. (default=5000)
runTests=1	Yes	Yes	Run symmetric code tests.
runTests=2	Yes	No	Run RSA test code.
runTests=4	Yes	No	Run DSA test code.
runTests=8	Yes	No	Run ECDSA test code.
runTests=16	Yes	No	Run Diffie-Hellman code tests.
runTests=32	Yes	Yes	Run compression code tests.
runTests=63	Yes	Yes	Run all tests except the chained hash and compression tests. (default)
runTests=128	Yes	Yes	Run chained hash and compression test code.
runStateful=1	Yes	Yes	Enable stateful compression tests. Applies when compression code tests are run.

Table 4. Sample Code Parameters (Sheet 2 of 2)

Parameter	Supported in user space	Supported in kernel space	Description
signOfLife=1	Yes	Yes	Indicates shorter test run that verifies the acceleration software is working. This parameter executes a subset of sample tests. Details are included in "Running the chained hash and compression test code requires:" on page 23. (default=0)
getLatency=1	Yes	No	Measures the processing time for the request being processed. Requires NumberCyInstances=1 and NumberDcInstances=1 to be configured in [SSL] section of the driver configuration file.
getOffloadCost=1	Yes		Measures the average number of cycles spent for single-request offloading. Requires NumberCyInstances=1 and NumberDcInstances=1 to be configured in [SSL] section of the driver configuration file.

Note: Running the chained hash and compression test code requires:

- Updating the configuration file to set StorageEnabled=1
- Restarting qat_service
- Possibly setting ServicesEnabled to dc or dc;sym.

4.1.2.1 signOfLife Tests

The signOfLife parameter is used to specify that a subset of the sample tests are executed with smaller iteration counts. This provides a quick test to verify the acceleration software and hardware are set up correctly.

Note: If the signOfLife parameter is not specified, the full run of tests can take a significant amount of time to complete.

User Space

After building the sample code with the installation script, the user space application is located at:

```
$ICP_ROOT/build
```

Then run the following commands:

```
# cd $ICP_ROOT/build/
# export LD_LIBRARY_PATH=`pwd`
# ./cpa_sample_code signOfLife=1
```

Kernel Space

After building the sample code with the installation script, the kernel space application is located at:

```
$ICP_ROOT/build
```

Then run the following commands:

```
# cd $ICP_ROOT/build/
# insmod ./cpa_sample_code.ko signOfLife=1
```

On a second terminal window, run dmesg to view the output of the cpa_sample_code Kernel application.

4.1.3 Test Results

When running the application, the results are printed to the terminal window in which the application is launched.

Example

Here is an example of the log messages created during the test:

```
--------------------------------------
Algorithm Chaining - AES256-CBC HMAC-SHA512
Number of threads    2
Total Submissions    20
Total Responses      20
Packet Size          512
--------------------------------------
```

A similar pattern is repeated for each of the tests.

4.2 Intel® QuickAssist API Sample Code

The software package contains sample code that demonstrates how to use the Intel® QuickAssist APIs and build the structures required for various use cases.

For more details, refer to the *Intel® QuickAssist Technology API Programmer's Guide* (refer to listing in Table 1, "Product Documentation and Software" on page 8).

4.2.1 Compiling the Acceleration Functional Sample Code

The acceleration functional sample code can be compiled manually.

Note: These instructions assume the software package has been untarred to the `/QAT` directory and the kernel source files were placed in the directory specified in this guide.

1. The following environment variable must be set to build the modules:
   ```
   export ICP_ROOT=<QATdir>
   ```
   where `<QATdir>` is `/QAT` or the directory where the package was untarred.
   ```
   # export WITH_UPSTREAM=1
   # export WITH_CMDRV=1
   ```
2. Compile for the user space using the following commands:
   ```
   # cd $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/functional
   # make all
   ```

Result: The generated Linux* kernel objects and sample applications are located at:
```
$ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/functional/build
```

4.2.2 Executing the Acceleration Functional Sample Code in User Space

1. To execute the acceleration functional sample code in user space, use the following commands:
   ```
   # cd $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/functional/
build
   "./hash_file_sample
   ```

Note: The hash_file_sample is one of the functional user space applications. You can launch the other user space applications in a similar fashion.

§

5.0 Installing, Building, and Running Yocto*

The Yocto Project* is an open-source collaboration project focused on embedded Linux*. Yocto includes a set of tools to build a custom Linux distribution. The process to create your custom Linux distribution using Yocto involves creating your own image on a software development workstation. The steps in Section 5.1, "Building the Yocto* SDK Image" on page 25 should be done on a software development workstation, not the Harcuvar CRB.

The steps to build and copy the image on Ubuntu* 14.04 are included here. If using a different Linux distribution, consult the Yocto Project website (http://www.yoctoproject.org) for more information and documentation, including:

- Yocto Quick Start Guide: http://www.yoctoproject.org/docs/latest/yocto-project-qs/yocto-project-qs.html
- Git repository: http://git.yoctoproject.org/cgit/cgit.cgi/meta-intel/

Note: The pre-built image has a Time-Limited-Kernel (TLK), which means that the image is restricted to a 10-day uptime and the image will be auto-rebooted after that time. TLK is added to encourage end-users to build their own image for production.

5.1 Building the Yocto* SDK Image

Follow the instructions below to create the Yocto* SDK image. A pre-built image Yocto SDK image is provided in the Electronic Design Kit (document number 565762). (refer to listing in Table 1, "Product Documentation and Software").If using the pre-built image, proceed to Section 5.2, "Creating the Linux* Boot Disk" on page 27 for instructions on creating OS boot image.

Note: If you are upgrading the acceleration drivers, proceed to Section 5.4, "Upgrading Acceleration Software" on page 29.

Prerequisites: The build process using sato consumes about 100 GB of disk space. Therefore, at least 200 GB of free disk space is recommended.

Note: If script build errors appear to be syntax errors, it is likely that the script is being passed to the wrong shell. Many Yocto scripts call `/bin/sh`, which may be symbolically linked to `/bin/dash`. To resolve the problem, remove `/bin/sh` ("`sudo rm /bin/sh`") and link to bash instead ("`sudo ln -s /bin/bash /bin/sh`").

1. Install Ubuntu* 14.04 (64-bit).

Note: Always execute the following instructions as a non-root user.

2. If required, add the following proxy settings for your network environment to `/etc/environment`:

```
https_proxy='https://<proxy_server>:<proxy_port>/'
http_proxy='http://<proxy_server>:<proxy_port>/'
ftp_proxy='http://<proxy_server>:<proxy_port>/'
GIT_PROXY_COMMAND=/usr/bin/git_proxy_command
```

Configure ssh configuration file for proxy settings. Add following lines to `~/.ssh/config`:

```
host *
ProxyCommand connect-proxy -s %h %p
```

Create the `/usr/bin/git_proxy_command` file and add the following lines:

```
#!/bin/sh
connect-proxy -s $@
```

Change the `git_proxy_command` file to be executable:

```
sudo chmod +x /usr/bin/git_proxy_command
```

Log out of the current user account and login, to allow the environment changes to take effect.

3. Update `apt-get`:

```
# sudo apt-get -y update
```

4. Install the required software components:

```
# sudo apt-get -y install gawk wget git-core diffstat unzip texinfo \
build-essential chrpath libsdl1.2-dev xterm socat connect-proxy
```

Note: If the `apt-get` command does not succeed completely, try it again, since the specific mirror selected may not have transferred the files correctly.

5. Download the BSP (document number 565774) and copy this file to the `/home/<userid>` directory on the build system. Then change that directory.
6. Decompress and extract the archived package. For example:

```
<package_name> = harcuvar_PV
    # tar xzvf <package_name>.tar.gz
```

The `<package_name>` package consists of setup folder and setup scripts. The setup script has test logic to check if your build system is able to access the Internet.

The setup folder consists of BSP-related patches inside `setup/patchset/bsp`, `setup/patchset/meta`.

```
# cd <package_name>
# chmod +x setup.sh setup/combo-layer
# ./setup.sh
```

7. Verify that the following directory structure is present, at a minimum:

```
pwd: ~/<package_name>
|-- bitbake
|
|-- meta-intel
|   |
|   |--meta-isg
|--meta-openembedded
|--meta-virtualization
|--meta-yocto
```

8. Run the script to build the environment essentials:

```
# cd ~/<package_name>
# source oe-init-build-env build
```

   This also changes the current working directory to `/<package_name>/build`.

Note: Be sure to source the file while in `/poky` since the build directory is created based on the current working directory.

9. By default, the 64-bit version of the OS is built. If the 32-bit OS is required, edit the file `~/<package_name>/build/conf/local.conf` and update the MACHINE line as follows: `MACHINE ?= "intel-core2-32"`.
10. Build the SDK image by running the commands:

```
# cd ~/<package_name>/build
# bitbake core-image-sato-sdk
```

Note: The `bitbake` cannot run as a root user. Note the following:

- If build errors appear to be script errors such as the one shown in the following example, verify that `/bin/sh` is linked to `/bin/bash` as described in Step 7.

```
[: 128: cmdline: unexpected operator |
```

- If the build fails and `bitbake`command needs to be executed again, repeat Step 7 to source `oe-init-build-env`.
- Log files (including errors and warnings) for the build are included in the `~/<package_name>/build/tmp/log directory`.
- Warning messages may be observed during the build process. These can be safely ignored.
- If an error is returned stating `bitbake` is not installed, verify that you sourced the `oe-init-build-env`file in `~/<package_name>` as described in Step 7.
- The command may take several hours to complete, depending on the particular software development machine and network speed.

11. Verify that the `hddimg` is created in `~/<package_name>/build/tmp/deploy/images/<intel-corei7-64 | intel-core2-32>`. The 64-bit version is titled `core-image-sato-sdk-intel-corei7-64.hddimg`.

5.2 Creating the Linux* Boot Disk

Note: If you are upgrading the acceleration drivers, proceed to Section 5.4, "Upgrading Acceleration Software" on page 29.

5.2.1 Locating the hddimg

If creating your own `hddimg` via the process in Section 5.1, "Building the Yocto* SDK Image" on page 25, your `hddimg` is located in `~/<package_name>/build/tmp/deploy/images/<intel-corei7-64 | intel-core2-32>`.

5.2.2 Creating the Boot Disk

Note: Special care must be taken when creating the boot disk, since any misidentification of the target disk can overwrite critical data. Back up your data if there is any doubt about which disk you will be writing to in the following steps.

A script file is included in the Yocto* BSP for creating the disk image. The script is called `mkefidisk.sh` and is located in the following directory:

```
~/<package_name>/scripts/contrib
```

Usage is:

```
mkefidisk.sh HOST_DEVICE image.hddimg TARGET_DEVICE
```

where:

`HOST_DEVICE` => Device to install image to.

`TARGET_DEVICE` => Name of the device as target device sees it. This would likely be `/dev/sds`.

1. Identify the device name for your `HOST_DEVICE`. This is the drive the Yocto image is being installed to. Study the output of one or more of the following commands to give confidence as to which disk is which:

   ```
   # sudo parted -l
   # df -h
   # cat /proc/partitions
   ```

2. Launch the `mkefidisk.sh` script using the HOST_DEVICE identified in Step 1, the hard drive image created in Section 5.1 or pre-built image, and the TARGET_DEVICE (likely `/dev/sda`). The following commands show the command for building the target image to the `/dev/sdb` drive with the 64-bit image created in Section 5.1.

   ```
   # cd ~<package_name>
   # sudo ./scripts/contrib/mkefidisk.sh /dev/sdb ./build/tmp/deploy/images/
intel-corei7-64/core-image-sato-sdk-intel-corei7-64.hddimg /dev/sda
   ```

   Answer "**y**" when prompted to prepare the EFI image on the `HOST_DEVICE` (`/dev/sdb` in this example).

   During the process, an error dialog may appear that states Unable to open a folder for 3.3 GB Volume or Unable to open a folder for ROOT. These errors can be safely ignored. Click **OK** to close the error dialog.

   When the image preparation is complete, the following message appears:

   ```
   Installation completed successfully
   ```

3. Power down the build system when the image preparation is complete. Insert the newly created SATA drive or USB stick into the target system and boot to the Yocto OS.

5.3 Additional Information on Software

Note: If you are upgrading the acceleration drivers, proceed to Section 5.4, "Upgrading Acceleration Software" on page 29.

The software package contains a set of sample tests that exercises acceleration functionality. This section describes the steps required to build and execute the sample tests.

The sample application is provided for the user space.

5.3.1 Loading the Sample Code

The acceleration kernel module must be installed and the software must be started before attempting to execute the sample code. This can be verified by running the following commands:

```
# lsmod | grep "qa"
# /etc/init.d/qat_service status
```

Typical output is similar to the following:

```
~# /etc/init.d/qat_service status Checking status of all devices.
There is 1 QAT acceleration device(s) in the system:
qat_dev0 - type: c3xxx, inst_id: 0,  bsf: 01:00:0,  #accel: 2 #engines: 6
state: up
```

Install the memory driver.

```
# insmod /lib/modules/4.1.8-yocto-standard/updates/drivers/crypto/qat/
usdm_drv.ko
```

The sample application is executed by launching the application for user space. The application is located in the `/usr/bin` directory. Section 4.1.2, "Loading the Sample Code" on page 21 includes a list of run-time parameters for the application. To execute the sign of life test, use the following command:

```
# ./cpa_sample_code signOfLife=1
```

5.4 Upgrading Acceleration Software

This section describes the steps required to update the existing Yocto* image with the updated acceleration software. It enables usage of the newer acceleration software package without rebuilding the target image.

1. Perform the following commands to install kernel header files:

   ```
   # cd /usr/src/kernel/
   # make oldconfig && make modules_prepare && make scripts
   # ln -s /usr/src/kernel /lib/modules/4.4.13-yocto-standard/build
   ```

2. Create a working directory for the software. This directory can be user defined, but for the purposes of this document a recommendation is provided.

   ```
   # mkdir /QAT
   # cd /QAT
   ```

Note: In this document, the working directory is assumed to be `/QAT`. This directory is the `ICP_ROOT`.

3. Transfer the tarball to the development board using any preferred method, for example, the USB memory stick, CDROM, or network transfer in the `/QAT` directory.

   Unpack the tarball using the following command:

   ```
   # tar -zxof <QAT tarball name>
   ```

4. Restricting access to the files is recommended:

   ```
   # chmod -R o-rwx *
   ```

5. Uninstall the existing acceleration software using the command:

```
# make uninstall
```

6. Install the acceleration software and acceleration sample code using the following commands:

```
# ./configure
# make install
# make samples-install
```

7. Execute sample code.

```
# cd build
# export LD_LIBRARY_PATH=/usr/lib64
# ./cpa_sample_code
```

Refer to Section 4.1.2, “Loading the Sample Code” on page 21 for list of run-time parameters for the application.

§

6.0 Physical Function to Virtual Function Communications

In a virtualized environment, a Physical Function (PF) device driver runs on the host and Virtual Function (VF) drivers on the guests. The PF driver initializes the device and loads firmware.

The PF driver communicates with the VF drivers to exchange information and state. Information communicated to the VF driver includes:

- Device capabilities. Capabilities may be filtered by fuses, soft straps, or firmware.
- Driver compatibility. Different driver versions may be running on host and guest. Driver versions may not always be compatible. New features may be developed which, when enabled on the PF, introduce an incompatibility with older VF drivers. If the drivers are incompatible, the VF driver does not complete initialization.
- Event notification. Events, such as errors, which are detected on the PF may need to send notifications to the VFs.

§

Appendix A Avoiding Kernel Crashes with PCH Intel® SKUs

Some Linux* versions, including RHEL/CentOS 7.3, will not boot/install with PCH Intel® QAT SKUs (E/M/T/L). This is due to a software driver bug in the in-kernel drivers for Intel® QAT. Pre-QS SKUs and 1G/2/4 SKUs are not affected.

Note: Intel is committed to moving away from non-inclusive terms such as 'blacklist' and will do so at the soonest possible opportunity, governed by dependencies on other software, such as `modprobe`.

To avoid the problem, change the grub boot options to block the loading of the Intel® in-kernel driver for PCH (`qat_c62x`) at installation time and for future boots until the driver is updated and/or the in-kernel `qat_c62x.ko` file is deleted.

Note: These instructions may not cover the case in which the kernel source is updated. Before updating the kernel, be sure to understand if kernel crashes may result due to this in-kernel Intel® QAT issue.

Note: Before uninstalling a Intel® QAT package, ensure that the package will not restore an in-kernel `qat_c62x.ko` file that has the issue.

Note: This issue is present approximately from the Linux* kernel 4.5 to kernel 4.9 and derivations thereof. The fixes are documented here:

- https://patchwork.kernel.org/patch/9485107/
- https://patchwork.kernel.org/patch/9485109/

Note: Some Linux* kernels after 4.9 may also experience kernel crashes with `qat_c62x.ko` or other `qat` modules. Follow the same recommended procedures described below.

A.1 Recommended Procedures

A.1.1 Installing the Operating System

Follow these instructions to safely install the operating system:

1. Boot platform from installation source (DVD, CD, USB)
2. Select “Install CentOS Linux v7.3” (or equivalent) menu from the GRUB list (*but don’t press* Enter).
3. Enter the grub options edit mode. This may be done by pressing **Tab** or **e**.
4. Add `modprobe.blacklist=qat_c62x` to the boot options. If you are not sure where exactly to make the edit, you can just find the word `quiet` and change it to `modprobe.blacklist=qat_c62x`.
5. Press **Enter** to boot.
6. Continue with installation process. The platform reboots when installation is complete.

A.1.2 Operating System First Boot

At the next reboot select the kernel you want to boot from the GRUB list (*but don't press* **Enter**).

1. Press **e**.
2. Append to the kernel command line `modprobe.blacklist=qat_c62x` (line that starts with `linuxefi` or perhaps `linux16`). If you are not sure where exactly to make the edit, you can just find the word `quiet` and change it to `modprobe.blacklist=qat_c62x`.
3. Press **Ctrl+x** to boot

Note: If you still see a kernel crash including the keywords `qat` or `adf`, refer to Section A.2, "Alternative Module Load Blocking Procedures" for additional ideas to reach a command prompt.

Upon reaching the command prompt, remove the in-kernel `qat_c62x.ko` file that can lead to a kernel crash when inserted into the kernel:

```
# rm /usr/lib/modules/`uname -r`/kernel/drivers/crypto/qat/qat_c62x/qat_c62x.ko
```

This prevents the offending module from being reloaded in all cases except the case in which a new kernel is loaded.

Reboot and verify that no kernel crash is observed.

```
# shutdown -r now
```

A.2 Alternative Module Load Blocking Procedures

Depending on your operating system and environment, some alternative methods of module load blocking can be attempted or utilized in order to get to a command prompt or for avoiding the kernel crash in the long term.

Note: Some methods may have unintended consequences (e.g. out-of-kernel QuickAssist packages may not install).

A.2.1 Grub Options

Try these options one at a time:

- `qat_c62x.blacklist=yes`
- `rdblacklist=qat_c62x`
- `module_blacklist=qat_c62x`

A.2.2 Changes to Configuration File

Append the following to `/lib/modprobe.d/dist-blacklist.conf` after booting into rescue mode:

```
blacklist intel_qat

blacklist qat_c62x

blacklist qat_dh895xcc
```

§