Intel® Communications Chipset 89xx Series Software for Linux*

Getting Started Guide

July 2014

Order No.: 330750-001

Revision History

Date	Revision	Description
July 2014	001	Corresponds with software release 1.1. Updates include: • First "public" version of the document. Based on "Intel Confidential"document number 523128-1.2, with the revision history of that document retained for reference purposes. • Renamed document (was *Intel® Communications Chipset 8925 to 8955 Series Software for Linux* Getting Started Guide*). • Added information for mux package. • Removed some platform-specific content.
November 2013	1.2	Corresponds with software release 1.0.1. Updates include: • Changed document and software title (expanded SKU range to include "8955")
November 2013	1.1	Corresponds with software release 1.0. Updates include: • Modified Step 3 in Unpacking the Software on page 17 • Modified Step 5 and added new information and note to Step 18 in Installing the Hard Disk IDE Patch
October 2013	1.0	Corresponds with software release 1.0. Updates include: • Added additional debugging information to Installing the Acceleration Software on page 21 and signOfLife Tests on page 30. • Changed product branding.
June 2013	0.5	Initial release of document.

Contents

Tables

1.0 Introduction

1.1 About this Manual

This Getting Started Guide documents the instructions to obtain, build, install and exercise the Intel® Communications Chipset 89xx Series Software 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® Communications Chipset 8900 to 8920 Series and/or the Intel® Communications Chipset 8925 to 8955 Series with Intel® Xeon® Processors.

In this document, for convenience:

- *Software package* is used as a generic term for the Intel® Communications Chipset 89xx Series Software package.
- *Acceleration drivers* is used as a generic term for the software that allows the QuickAssist Software Library APIs to access the Intel® QuickAssist Accelerator(s) integrated in the Intel® Communications Chipset 8925 to 8955 Series.

The document is organized as follows:

- Installing the Operating System on the Development Board on page 11
- Building and Installing Software on page 17
- Sample Applications on page 27
- Glossary on page 35

1.2 Additional Information on Software

The software release package for Linux* has been validated with Fedora 16 32-bit and x86_64.

1.2.1 Where to Find Current Software and Documentation

The software release and associated collateral can be found on the open source website: https://01.org/packet-processing/intel%C2%AE-quickassist-technology-drivers-and-patches.

See Product Documentation on page 7 for a list of associated collateral.

1.2.2 Product Documentation

Table 1 on page 8 lists the documentation supporting this release. All documents can be accessed as described in Where to Find Current Software and Documentation on page 7.

Table 1. Product Documentation and Software

Title	Number
Intel® Communications Chipset 89xx Series Software for Linux Getting Started Guide* (this document)	330750
Intel® Communications Chipset 8900 to 8920 Series Software Programmer's Guide	330753
Intel® Communications Chipset 8925 to 8955 Series Software Programmer's Guide	330751
Intel® QuickAssist Technology Software Release Notes	330683
Intel® QuickAssist Technology API Programmer's Guide	330684
Intel® QuickAssist Technology Cryptographic API Reference Manual	330685
Intel® QuickAssist Technology Data Compression API Reference Manual	330686

1.2.3 Pre-boot Firmware

The latest release of the development board pre-boot firmware (BIOS) is also located on the Intel Business Portal.

Please refer to the Release Notes listed in Table 1 on page 8 for the correct firmware version.

1.3 Related Software and Documentation

Refer to the Development Kit User's 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.

Please follow the directions in Where to Find Current Software and Documentation on page 7 to locate this collateral.

1.4 Conventions

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

1.5 Software Overview

The software is described in the following topics:

- Features Implemented
- List of Files in Release
- Package Release Structure

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:

Files/Directory	Comments
./QATmux.L.a.b.c-n.tar.gz	Top-level QAT package
./filelist	List of files in this package
./installer.sh	Installer script
./LICENSE.GPL	Licence file
./Versionfile	Version file
./QAT1.5	Directory containing QAT1.5.L.a.b.c-n.tar.gz
./QAT1.6	Directory containing QAT1.6.L.a.b.c-n.tar.gz

Depending on the devices detected on the platform, the installer script will build the appropriate driver(s) and, if more than one driver is then built, it will also build the qat_mux.

The following extra files/directories may be present.

In the top-level folder:

Files/Directory	Comments
./cpa_mux	Directory containing qat_mux software. This will only be present if both drivers are built.
./InstallerLog.txt	Log of installer script output

In one or both of the QAT1.x directories:

Files/Directory	Comments
./QAT1.x/QAT1.x.L.a.b.c-n.tar.gz	QAT1.x driver package
./QAT1.x/filelist	List of files in the QAT1.x package
./QAT1.x/build	Build output directory
./QAT1.x/versionfile	Version of the QAT1.x driver
./QAT1.x/quickassist	Top-level acceleration software directory for QAT1.x driver
./QAT1.x/quickassist/adf	Acceleration Device Framework source files
./QAT1.x/quickassist/build_system	Build system files
./QAT1.x/quickassist/config	Configuration files

continued...

./QAT1.x/quickassist/include	Common acceleration software header files
./QAT1.x/quickassist/lookaside	Service Access Layer source files
./QAT1.x/quickassist/utilities	OSAL and Firmware loading source files

2.0 Installing the Operating System on the Development Board

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

2.1 Acquiring Fedora 16

Fedora 16 is a Linux distribution built on free and open source software. The software package from Intel does not include a distribution of Fedora or any other Linux variant. The software package includes Linux device driver source developed by Intel.

The software package has been validated with the Linux distributions listed below:

- Fedora 16 x86_64 can be obtained from:

  http://dl.fedoraproject.org/pub/archive/fedora/linux/releases/16/Fedora/x86_64/iso/

- Fedora 16 32-bit can be obtained from:

  http://dl.fedoraproject.org/pub/archive/fedora/linux/releases/16/Fedora/i386/iso/

Note: This document is written with the Fedora 16 Install Media in mind. Using the Live Media version is not recommended.

2.2 Configure the BIOS on the Development Kit

The BIOS configuration needs to be updated to allow the operating system to be successfully installed on the development kit.

If no video is available, check that the video card recommendations in the Development Kit setup chapters were followed. In certain circumstances, video can be restored by clearing CMOS.

The following steps must be performed to properly configure the BIOS:

1. Power on the development kit. Watch closely for the prompt to enter BIOS setup. Press **F2** when prompted.
2. Update the Boot order so that the DVD-ROM drive is the first boot option. The option is available under:

   Boot > Boot Option Priorities
3. Ensure the SATA mode is set to **AHCI**. The option is available under:

   Advanced > PCH-IO Configuration > SATA Configuration > SATA Mode Selection

4. Disable Intel® SpeedStep technology to achieve maximum Intel® QuickAssist Technology performance.
5. Insert the Fedora DVD into the DVD-ROM drive.
6. Press **F4** to Save and Exit. The BIOS changes are saved, and the system will boot from the device selected in the Boot Order. If the system is not booting, disconnect the power cord from the system and then reconnect.

2.3 Installing Fedora 16

For complete Fedora installation instructions, please refer to the online Installation Guide at:

http://docs.fedoraproject.org/en-US/Fedora/16/html/Installation_Guide/

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 Fedora installation disc. If not, verify the Boot Order settings described in Configure the BIOS on the Development Kit on page 11.
2. At the welcome prompt, select **Install or upgrade Fedora** and click **Enter**.
3. Select **OK** to begin testing the installation media.

   Note: It is highly recommended that the Fedora installation disc is verified prior to an installation of the OS.
4. After verifying the Fedora installation disc(s), the graphical portion of the installation is loaded. Click **Next** to continue the installation process.
5. Select the language to use during the installation process and click **Next**.
6. Select the appropriate keyboard for the system and click **Next**.
7. If specialized storage devices are not required, select Basic Storage Devices and click **Next**.

   Note: The following warning may be displayed when installing on a blank disk:

   ```
   "REINITIALIZING WILL CAUSE ALL DATA TO BE LOST!"
   ```

   If you definitely know the hard drive in use is blank, proceed. Otherwise, power down and replace the hard drive.

   If the hard drive is blank, select **Re-initialize all**, then **Next**.

   If the disk has existing Linux installation, select **Fresh Installation**, then **Next**.
8. This screen allows you to enter host name and configure your network connections. If remote access to the system is needed, it is recommended to enable **Connect Automatically** for each of the ports. To do this, click on **Configure Network**. Select your network port (for example, System p1p1) and then click **Edit**, click **Connect Automatically**, and click **Save**. Repeat these operations for each of the required ports, and then click **Close**.

9. Enter a host name for the computer and click **Next**. The default host name will be acceptable in most cases.
10. Select the nearest city to your time zone and click **Next**.
11. Enter the desired root password for the system in the box labeled **Root Password:** and reaffirm the root password by entering the same password in the **Confirm:** box. Click **Next** to continue.
12. Select the hard drive where Fedora will be installed and the drive options. Options include **Use All Space** and **Replace Existing Linux System(s)**. Click **Next** to continue.
13. If multiple drives are visible to the system, you may be asked to select a target device. Select the appropriate target device, select the right arrow, and click **Next**.
14. Select **Write changes to disk**.
15. Select the software to install. Ensure that **Software Development** is selected. Click **Next** to continue.
16. When the installation completes, the install DVD should be ejected. Remove the DVD and select **Reboot** when prompted.
17. After the reboot, some normal system setup is required, such as creating user account(s) and setting the time and date. Note that a user account is required to boot into the Fedora GUI. When asked, follow the prompt by clicking the **Forward** button.
18. When the installation completes, continue with Updating Grub Configuration File on page 13.

2.4 Updating Grub Configuration File

This section contains instructions on updating the grub configuration file.

Note: Root access is required in order to update the `/etc/default/grub` file in the following steps.

1. After completing steps in Installing Fedora 16 on page 12, log into the system.

   Note: You may observe an error message similar to `GNOME3 Failed to load` when booting to the desktop. This message can be ignored.
2. If you are booting to the desktop, Fedora 16 may automatically update your kernel version. To turn off this feature, select **Applications > Other > Software Settings**. Change **Automatically install** from **Only security updates** to **Nothing**.
3. Update the `/etc/default/grub` file to remove two Linux options `rhgb` and `quiet` and add the Linux option `acpi_enforce_resources=lax`.

   If your application does not require virtualization for the acceleration software, add the Linux option `intel_iommu=off`

   If you have made the change correctly, the line will look similar to the following:

```
GRUB_CMDLINE_LINUX="rd.md=0 rd.dm=0 rd.lvm.lv=VolGroup/llv_swap KEYTABLE=us
SYSFONT=latarcyrheb-sun16 rd.lvm.lv=VolGroup/lv_root rd.luks=0
LANG=en_US.UTF-8 acpi_enforce_resources=lax intel_iommu=off"
```

Note: Check for `intel-IOMMU: disabled` in the kernel log (dmesg) after the system is rebooted.

If your application requires virtualization, please see the *Using Intel® Virtualization Technology (Intel® VT) with Intel® QuickAssist Technology Application Note*.

4. If you are using 32-bit Linux, update the `/etc/default/grub` file to add the Linux option `vmalloc=248M`.

   If you have made the change correctly, the line will look similar to the following:

```
GRUB_CMDLINE_LINUX="rd.md=0 rd.dm=0 rd.lvm.lv=VolGroup/llv_swap KEYTABLE=us
SYSFONT=latarcyrheb-sun16 rd.lvm.lv=VolGroup/lv_root rd.luks=0
LANG=en_US.UTF-8 acpi_enforce_resources=lax intel_iommu=off vmalloc=248M"
```

5. Save the changes to the file and execute the following command to generate the grub configuration file:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

6. Reboot the system.

```
# shutdown -r now
```

2.5 Configuring Linux

Once the operating system is installed, there are a few configuration items that may need to be completed, such as updating the yum configuration files and enabling the development platform to operate within a corporate network. 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. In order 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 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.

```
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, see the *Programmer's Guide* for your platform, specifically the *Secure Architecture Considerations* section.

Securing your operating system is critical. You should consider the following items:

Note: This is not an exhaustive list.

- Employing effective security policies and tools; for instance, SELinux is configured correctly and is active
- Running and configuring the firewall(s)
- Preventing privilege escalation at boot (including recovery mode); for instance, setting a grub password. Additional details are described below.
- Removing unnecessary software packages
- Patching software in a timely manner
- Monitoring the system and the network
- Configuring and disabling (as appropriate) remote access
- Disabling network boot
- Requiring secure passwords
- Encrypting files, up to full-disk encryption
- Ensuring physical security of the system and the network
- Using mlock to prevent swapping sensitive variables from RAM to disk
- Zeroing out sensitive variables in RAM

2.6.1 Setting a Grub Password

During normal bootup, an option for recovery mode is available. When chosen, this option provides a command prompt with root access to the user without any credential authenication, in other words, root access is provided without a need to enter a root password. Intel recommends that you secure GRUB2 so it is not possible for anyone to change boot parameters or use the command line, by adding a user/ password combination to GRUB2's configuration files.

Note: In order to password-protect the grub file with an encrypted password, you may need to install version 1.98 or later of GRUB2.

The instructions below are copied from the Security section of the GRUB2 wiki here: https://wiki.archlinux.org/index.php/GRUB2#Security

1. Run the command:

```
grub2-mkpasswd-pbkdf2
```

2. Enter a password and confirm it. The output will look like this:

```
Your PBKDF2 is
grub.pbkdf2.sha512.10000.C8ABD3E93C4DFC83138B0C7A3D719BC650E6234310DA069E6FDB0
DD4156313DA3D0D9BFFC2846C21D5A2DDA515114CF6378F8A064C94198D0618E70D23717E82.50
9BFA8A4217EAD0B33C87432524C0B6B64B34FBAD22D3E6E6874D9B101996C5F98AB1746FE7C719
9147ECF4ABD8661C222EEEDB7D14A843261FFF2C07B1269AThen,
```

3. Add the following to `/etc/grub.d/00_header` file:

```
cat << EOF
set superusers="username"
password_pbkdf2 username <password>
EOF
```

where `<password>` is the string generated by the `grub2-mkpasswd_pbkdf2` command.

4. Regenerate your configuration file. Your GRUB2 command line, boot parameters and all boot entries are now protected.

This can be relaxed and further customized with more users as described in the "Security" part of the GRUB manual here:

https://www.gnu.org/software/grub/manual/grub.html#Security

In addition, to prevent recovery mode (also known as "runlevel1", or "single user mode") password-less access, the system administrator must modify the `/etc/sysconfig/init` file. In the following extract from the file:

```
# Set to '/sbin/sulogin' to prompt for password on single-user mode
# Set to '/sbin/sushell' otherwise
SINGLE=/sbin/sushell
```

Change the SINGLE= line to the following and save the file:

```
SINGLE=/sbin/sulogin
```

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. See Where to Find Current Software and Documentation for the software location.

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`.
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 <tarball name>
   ```
3. Restricting access to the files is recommended:

   ```
   # chmod -R 770 /QAT
   ```

Result: The installation script and tarball files are created under the `/QAT` directory.

3.2 Acceleration Software Installation Script

An installation script is provided that walks you through building/installing the software. Refer to Installation on a New Development Platform for instructions on installing the software on a new platform.

The installation script can also be launched with command line arguments giving the option to bypass the interactive setup. Refer to Command Line Arguments for additional information.

For details on minimizing Acceleration Software compilation time, refer to Minimizing Acceleration Software Compilation Time on page 26.

The *Programmer's Guide* for your platform describes required and optional build flags in the *Build Flag Summary* section. If you are using the installation script, the required build flags are handled by default. The optional build flags control the driver functionality and can be used with the installation script or the command line arguments.

The installation script file must be run as root. If the script is executed as a user other than root, the following error is returned: `ERROR This script must be run as root.`

Launch the script using the following command:

```
# ./installer.sh
```

A welcome message is displayed, followed by Installation Options. The table below lists some of the available installation options.

Note: If you have not unpacked the release package tarball as described in Unpacking the Software on page 17, errors will be returned.

Table 2. Installation Options

Option	Name	Description
1	Build Acceleration	Builds the acceleration software. The software is not installed.
2	Install Acceleration	Builds and installs the acceleration software. The software drivers are persistent and will be loaded on subsequent reboots.
3	Install SR-IOV Host Acceleration	Builds and installs the acceleration software for the Host OS for SR-IOV. The software drivers are persistent and will be loaded on subsequent reboots.
4	Install SR-IOV Guest Acceleration	Builds and installs the acceleration software for the Guest OS for SR-IOV. The software drivers are persistent and will be loaded on subsequent reboots.
5	Show Acceleration Device Information	Displays the number of chipset devices available on the system and the B:D:F for each device.
6	Build Acceleration Sample Code	Builds both user space and kernel space version of the Acceleration Sample Code.
7	Toggle GigE Watchdog for Acceleration Install	Changes whether or not the GigE watchdog is installed when the acceleration software is installed. *Note:* This option does not install or uninstall the GigE watchdog directly.
8	Uninstall	Uninstalls the software. There may be a sub-menu that allows you to select which software components are uninstalled.
9	Exit	Exit the installation script.

Note: The interactive installer.sh does not handle all options that may be of interest. For instance, there are a wide range of configurations that are possible, including build or install, with or without support for multiple acceleration hardware generations (i.e., using the "mux" layer), virtualization support (host or guest) or no virtualization support. Some modifications to the installer.sh may be required.

Note: Some packages may have slightly different options than those listed above.

3.2.1 InstallerLog

The `InstallerLog.txt` file is appended after each installation with the time/date and the output of the build/install. If any issues were seen during the installation, check the log file for details.

3.2.2 Command Line Arguments

The *Programmer's Guide* for your platform describes required and optional build flags in the *Build Flag Summary* section. If you are using the installation script, the required build flags are handled by default. The optional build flags control the driver functionality and can be used with the installation script or the command line arguments.

The command line takes the following arguments:

```
./installer.sh <What to Build> <Where to Build> <Kernel Source>
```

- `<What to Build>`
  - — `a` - Install Acceleration
  - — `ba` - Build Acceleration
  - — `bs` - Build Acceleration Sample Code
  - — `bamux` - Build Acceleration for QATmux
  - — `bamuxhost` - Build Acceleration QATmux for host
  - — `bamuxguest` - Build Acceleration QATmux for guest
  - — `bsmux` - Build Acceleration Sample Code for mux
  - — `iamux` - Install Acceleration for QATmux
  - — `iamuxhost` - Install Acceleration for QATmux for Host
  - — `iamuxguest` - Install Acceleration for QATmux for Guest
  - — `uamux` - Uninstall Acceleration With Mux
  - — `h` - Provide Command line help
- `<Where to Build>`
  - — Set the build location, for example, `/QAT` or `$PWD`.
- `<Kernel Source>`
  - — Set the kernel source as shown in the examples below:

    64-bit: `/usr/src/kernels/3.1.0-7.fc16.x86_64`

    32-bit: `/usr/src/kernels/3.1.0-7.fc16.i686`

  Note: The `<Kernel Source>` parameter is only required when patching the PCH drivers.

Example Usage:

```
./installer.sh a $PWD
```

```
./installer.sh bs /QAT
```

Note: Not all command line options are supported on every software package.

3.3 Installation on a New Development Platform

This section walks you through the installation of the software on a new development platform.

Note: If you require SATA IDE support, you need to install the Hard Disk IDE Patch.

Note: If you are updating from a previous release of the software, you may wish to identify the changes between the releases. See Updating the Acceleration Driver from a Previous Release on page 24 for additional information.

Note: One Acre Ethernet cards that are functional on Shumway CRBs containing Intel® Communications Chipset 8900 to 8920 Series devices are not functional on Shumway CRBs containing Intel® Communications Chipset 8925 to 8955 Series devices. A separate Ethernet card is required for network connectivity. Refer to Updating yum Configuration Files on page 14 for additional configuration information.

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 commands:

```
# service qat_service start
```

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

- Software is installed.
- Acceleration software is already running.
- If you are not using virtualization, verify the Kernel option `intel_iommu=off` has been configured as specified in Updating Grub Configuration File on page 13.
- Verify the device is enumerated properly using the `lspci` command described in Installing the Acceleration Software on page 21.

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.

Note: If the following error message is returned: `icp_qa_al err: adf_aeUcodeMap: Mapping of Firmware failed, status=0xa116 "UOF is incompatible with the chip type/revision"` you have attempted to install the software package on a system without the latest chipset device silicon.

3.5 Installing the Acceleration Software

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 the installer.sh script in the previous acceleration software package.

1. Power on the system and proceed with the instructions below.
2. Open a Terminal Window and switch to superuser.

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

3. In the `/QAT` directory, start the installation script.

```
# /QAT
# ./installer.sh
```

Select the **Install Acceleration** installation option. This installs the Acceleration software. You will be prompted for a directory location to build the package and the Build Output Directory. Use the default values provided by the installation script.

Note: If `Error: Could not open file: /etc/dh895xcc_qa_dev0.conf` is shown during the Accleration installation, it is possible that the configuration files were not copied from `/$ICP_ROOT/quickassist/config` due to the way that `installer.sh` detects the acceleration device(s) via `lspci`. Remove the line from `/usr/share/hwdata/pci.ids` that contains the string `0435` and rerun the Acceleration installation. Alternatively, copy the appropriate configuration files from`/QAT/quickassist/config` to `/etc` and then restart the service with `service qat_service restart`.

Note: If a `failed to start device` error is shown during the Acceleration installation, ensure that the kernel option `intel_iommu=off` has been configured as specified in Updating Grub Configuration File on page 13. If this kernel parameter is not specified, acceleration services are only available in a guest operating system.

Note: If **Software Development** was not selected during the OS install, you may see an error message similar to the following during the acceleration install:

```
"make: *** /lib/modules/3.1.0-7.fc16.x86_64/build/: No such file or
directory. Stop."
```

Reinstall the OS and select the **Software Development** option as described in Installing Fedora 16 on page 12, or run the following commands:

```
# yum -y install kernel-devel-$(uname -r)
# yum -y install gcc
```

Note: If an error like `cpa_mux: exports duplicate symbol cpaCyBufferListGetMetaSize` (owned by icp_qa_al) occurs, a previous version of the driver is still loaded. Run installer.sh for that package and Select Uninstall.

4. During the installation, the following message is displayed:

```
*** No error detected in InstallerLog.txt file ***
```

At the end of the installation, the following messages are displayed:

```
*** Acceleration Installation Complete ***
```

Refer to the `InstallerLog.txt` file for additional detail on the installation. It is also a good idea to 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. See Configuration Files on page 24 for more detail.

Note: After building/installing the Acceleration Software, it is highly recommended to secure the build output files (the files located in `$ICP_ROOT\build`) by either deleting them or setting permissions according to your needs.

5. Use the **0** option to exit the installation.
6. After installing the Acceleration Software, it is recommended to verify that the acceleration software kernel object is loaded and ready to use. Depending on which drivers are loaded, the objects built have different names (see the following table).

Table 3. Build Output Objects

Case	Kernel object(s)	User space object(s)	Static libraries
QAT1.5 only installed	icp_qa_al.ko	libicp_qa_al_s.so	libicp_qa_al.a
QAT1.6 only installed	icp_qa_al.ko	libicp_qa_al_s.so	libicp_qa_al.a
Mux case, i.e., both QAT1.5 and QAT1.6 installed	qat_1_5_mux.ko qat_1_6_mux.ko qat_mux.ko	libqat_1_5_mux_s.so libqat_1_6_mux_s.so libqat_mux_s.so	libqat_1_5_mux.a libqat_1_6_mux.a libqat_mux.a

The following operations can be used to verify that the correct kernel objects are loaded according to the above table:

```
# lsmod | grep qa
```

If the correct objects are not loaded then the acceleration software is not installed and is not ready for use. Refer to the Installer.log file for additional information. If necessary, run the installation script again and select Install Acceleration.

See section, "Support for Multiple Acceleration Hardware Generations", in the *Intel® Communications Chipset 8925 to 8955 Series Software Programmer's Guide* for more information.

If `icp_qa_al` is not returned, then the acceleration software is not installed and is not ready for use. Refer to the `Installer.log` file in the `/QAT` directory for additional information. If necessary, run the installation script again and select **Install Acceleration**.

Post-requisites: Once the installation/building is complete, proceed to Sample Applications to execute applications that exercise the software.

3.6 Cross-Compilation Capabilities for Intel® QuickAssist Technology

These capabilities enable users to compile the Intel® QuickAssist Technology driver on one system for the targeted architecture of another system.

If you are cross-compiling for the first time, you need to install the following packages:

- `yum -y install glibc-devel.i686`
- `yum -y install openssl-devel.i686`
- `yum -y install libgcc.i686 --setopt=protected_multilib=false`
- `cp /lib/libz.so.1.2.5 /usr/lib/libz.so`

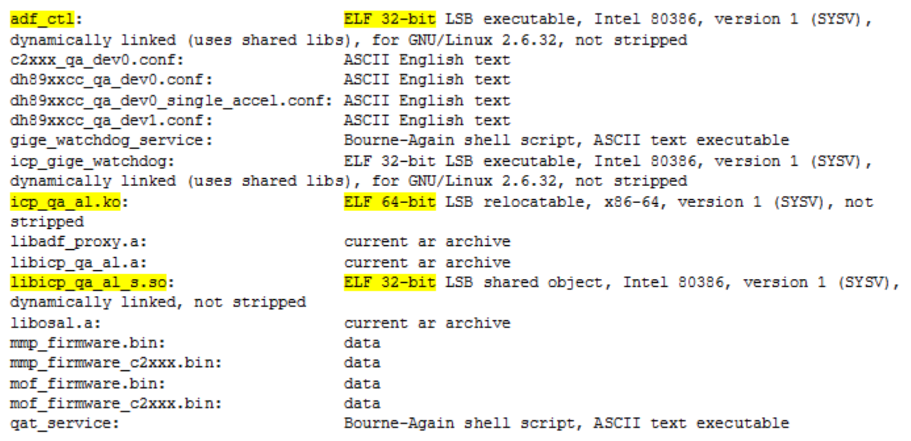
Use the following commands to cross-compile the driver:

```
# export ICP_ROOT=<your path to unzipped tar ball>
# cd quickassist
# export ICP_ENV_DIR=$ICP_ROOT/quickassist/build_system/build_files/env_files
# export ICP_BUILDSYSTEM_PATH=$ICP_ROOT/quickassist/build_system
# export ICP_BUILD_OUTPUT=$ICP_ROOT/build
# export ICP_TOOLS_TARGET=accelcomp
# export LD_LIBRARY_PATH=$ICP_ROOT/build
For i386 machines:
# make ICP_ARCH_USER=i386
For i686 machines:
# make ICP_ARCH_USER=i686
```

Next, go to the build folder and enter:

```
# cd ../build
# file *
```

Here's a sample output:



```
adf_ctl:                          ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.6.32, not stripped
c2xxx_qa_dev0.conf:               ASCII English text
dh89xxcc_qa_dev0.conf:            ASCII English text
dh89xxcc_qa_dev0_single_accel.conf: ASCII English text
dh89xxcc_qa_dev1.conf:            ASCII English text
gige_watchdog_service:            Bourne-Again shell script, ASCII text executable
icp_gige_watchdog:                ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.6.32, not stripped
icp_qa_al.ko:                     ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), not
stripped
libadf_proxy.a:                   current ar archive
libicp_qa_al.a:                   current ar archive
libicp_qa_al_s.so:                ELF 32-bit LSB shared object, Intel 80386, version 1 (SYSV),
dynamically linked, not stripped
libosal.a:                        current ar archive
mmp_firmware.bin:                 data
mmp_firmware_c2xxx.bin:           data
mof_firmware.bin:                 data
mof_firmware_c2xxx.bin:           data
qat_service:                      Bourne-Again shell script, ASCII text executable
```

As you can see, the user files, `adf_ctl` and `libicp_qa_al_s.so`, are 32-bit files while `icp_qa_al.ko` is a 64-bit file.

Cross-Compiling the Sample Code for 32-Bit User Space

Change directories to the top-level sample code directory:

```
# cd $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code
```

Compile the user space sample application:

```
For i386 machines:
# make perf_user ARCH=i386
For i686 machines:
# make perf_user ARCH=i686
```

3.7 Configuration Files

When the Acceleration software loads, it is configured based on settings in the platform-specific configuration files. The configuration files are placed in the `/etc` directory. For example, the first configuration file for Intel® Communications Chipset 8925 to 8955 Series devices is `dh895xcc_qa_dev0.conf` and the first configuration file for the Intel® Communications Chipset 8900 to 8920 Series is `dh89xxcc_qa_dev0.conf`. If more than one device of a given type is present, it will be '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 Starting/Stopping the Acceleration Software on page 20 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 will be copied to the `/etc` directory for you. If your system has more than one platform, it is recommended that you 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.

3.8 Updating the Acceleration Driver from a Previous Release

If you are upgrading from a previous release of the acceleration driver, you may wish to view the code changes between the releases. Changes would include those made by Intel for the new release as well as user changes to the previous release.

To identify the code changes between a previous release and the current release, perform the following steps:

1. If using the QATmux pkg, then extract this first:

```
#tar xzof QATmux.L.<version>.tar.gz
```

The acceleration packages are then contained in directories QAT1.5 and QAT1.6.

2. Extract the acceleration software package from the new release:

```
# tar xzof QAT*.L.<version>.tar.gz
```

3. Execute the `diff` command passing in the path to the previous release quickassist directory and the path to the new release's quickassist directory to capture the output to a file. The command would look like:

```
# diff -x'*.a' -x'*.o' -r  /<prev_version>_QAT/quickassist $ICP_ROOT/
quickassist > diff.patch
```

   where:

   - `/<prev_version>_QAT` indicates the directory where the previous release is located
   - `$ICP_ROOT` indicates the directory where the new release is located

   The results of the `diff` command are stored in the `diff.patch` file. This will include changes Intel has made between the two releases, as well as any modifications you have made. The `-x` option is used to ignore differences between intermediate files.

   Note: When first reviewing the changes between the releases, it may be beneficial to have the tool ignore changes to the Version string and Copyright string. Each source file includes these strings and they will be different between releases. Adding `-I "version:" -I "Copyright"` to the command above will prevent these differences from being included in the diff file. Once you have identified the main changes between the releases, it is strongly advised to include version string updates in your final patch. This ensures that the software is marked with proper version string.

The created patch can be used as a starting point in updating your previous release to the current release. You will need to review this file to identify the changes you have made to the driver that you wish to keep. For each change you wish to keep, remove the corresponding lines from the patch file. Once this task is completed, you can proceed with applying the patch.

Note: It is very important to review the changes called out in the patch file to identify your changes. If this is not done, your changes to the driver will be lost after applying the patch.

To apply the patch, perform the following steps:

1. Change directories to your previous quickassist directory:

```
# cd /<prev_version>_QAT/quickassist
```

2. Attempt to apply the patch. To avoid any write permission errors, this should be done by the same user who created the original folder that is being patched.

```
# patch -p1 < diff.patch
```

   If the patch does not apply cleanly, this suggests that both Intel and user have modified the same sections of code in ways that the patch utility cannot resolve. Refer to the created `.rej` files for additional information on the conflicts.

Note: It is important to compile from the proper directory after applying the patch. In the example above, the compiles must be done in the directory:

```
/<prev_version>_QAT/quickassist
```

not in the directory:

```
$ICP_ROOT/quickassist
```

Once the conflicts (if any) are resolved, you must update the firmware image. This can be done by performing the following:

- Copy the firmware images from the new release to your previous quickassist directory:

```
# cp $ICP_ROOT/quickassist/lookaside/firmware/icp_qat_ae.*
/<prev_version>_QAT/quickassist/lookassist/firmware
```

3.9 Minimizing Acceleration Software Compilation Time

When compiling/installing the Acceleration Software, a "make clean" operation is performed. This results in rebuilding every source file included in the package, even if the source files are unaltered. The "make clean" is done to ensure a proper build is performed. Here are a few items that could cause issues that would require the "make clean":

- If any compile time build flags are added which may not be reflected in the already-built object files.
- If changes are made to header files, performing make clean prior to the make would be preferred.

If you are comfortable with these constraints, you can update the `$ICP_ROOT/quickassist/Makefile` line where ALL_TARGETS is defined and remove the 'clean' from the list of targets. This will remove the clean operation from the build process.

Before the update, the line looks like:

```
ALL_TARGETS = clean lac_lib_dir libosal libosal_user hal adf adf_user

lac lac_user qat-fw install_scripts
```

After the update, the line looks like:

```
ALL_TARGETS = lac_lib_dir libosal libosal_user hal adf adf_user

lac lac_user qat-fw install_scripts
```

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 QuickAssist Acceleration 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 both Kernel Space and User Space and the following sections contain instructions for both.

Note: The memory driver included with the sample application is a sample memory driver and is not intended for actual deployment.

4.1.1 Compiling the Acceleration Sample Code

The acceleration sample code can be built from the installation script, or it can be compiled manually.

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.

To build from the installation script, do the following:

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, please refer to the "Running Applications as Non-Root User" section in the applicable Programmer's Guide (see Product Documentation on page 7).

2. In the `/QAT` directory, start the installation script.

```
# cd /QAT
# ./installer.sh
```

   Select the **Build Acceleration Sample Code** installation option, or use the 'bsmux' option to force the acceleration sample code to build with support for multiple hardware generations. This option compiles the Acceleration Sample code for both user space and kernel space. It also compiles the memory mapping driver used with the user space application.

   You will be prompted for a directory location to build the package and the Build Output Directory. Use the default value for the location to build the package. The Build Output Directory parameter is ignored.

Note: In the case where both QAT1.5 and QAT1.6 drivers are loaded, the sample code from the QAT1.6 driver should be used.

3. Proceed to signOfLife Tests on page 30 for instructions on executing the tests.

To manually compile the acceleration sample code, do the following:

1. The following environment variables must be set to build the modules:

```
# export ICP_ROOT=<QATdir>
# export ICP_BUILDSYSTEM_PATH=$ICP_ROOT/quickassist/build_system
# export ICP_ENV_DIR=$ICP_ROOT/quickassist/build_system/build_files/env_files
```

   where <QATdir> is /QAT or /QAT/QAT1.6 or /QAT/QAT1.5, depending on what hardware support is required.

2. If intermediate modules are required, the following variables must also be set:

```
# export ICP_ROOT=/QAT/QAT1.6
# export ICP_TOOLS_TARGET=accelcomp
```

3. The sample code is compiled with the default assumption that the kernel source header files are located in one of the following directories:

   64-bit: `/usr/src/kernels/3.1.0-7.fc16.x86_64`

   32-bit: `/usr/src/kernels/3.1.0-7.fc16.i686`

4. If building with mux support, set the mux environment variable:

```
# export WITH_CPA_MUX=1
```

5. If the kernel source header files are located in a different directory, create the environment variable with the directory of desired target kernel sources. For example:

```
# export KERNEL_SOURCE_ROOT=/usr/src/kernels/linux
```

6. You can compile for both Kernel space and User space at the same time using the following commands:

```
# cd $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code
# make perf_all
```

   The generated Linux kernel object and sample application are located at:

```
$ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/build
```

Proceed to signOfLife Tests on page 30 for instructions on executing the tests.

4.1.2 Loading the Sample Code

1. 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
```

Note: The kernel object names may be different depending on the devices installed on the platform. Refer to the table in Section 4.4.2 for kernel object names.

Typical output with two acceleration devices is:

```
There is 2 acceleration device(s) in the system:
 icp_dev0 - type=dh895xcc, inst_id=0, bsf=03:00:0, #accel=6, #engines=12,
state=up
 icp_dev1 - type=dh895xcc, inst_id=1, bsf=82:00:0, #accel=6, #engines=12,
state=up
```

Note: If the module is not returned from the first command, refer to Starting/ Stopping the Acceleration Software on page 20 for additional information on starting the Acceleration software.

2. The sample code is executed by installing the `cpa_sample_code` kernel object for kernel space, or by launching the application for user space.

The application allows the kernel parameters listed below.

Table 4. Sample Code Parameters

Parameter	Description
configFileVer	Version of configuration file. Can be 1 or 2 (default). If you are using the original version 1 configuration file, use 1. For configuration file details, see the *Programmer's Guide* for your platform.
cyNumBuffers=w	Number of buffers submitted for each iteration. (default=20)
cySymLoops=x	Number of iterations of all symmetric code tests. (default=5000)
cyAsymLoops=y	Number of iterations of all asymmetric code tests. (default=5000)
runTests=1	Run symmetric code tests.
runTests=2	Run RSA test code.
runTests=4	Run DSA test code.
runTests=8	Run ECDSA test code.
runTests=16	Run Diffie-Hellman code tests.
runTests=32	Run compression code tests.
runTests=63	Run all tests. (default)
runStateful=1	Enable stateful compression tests. Applies when compression code tests are run.
signOfLife=1	Indicates shorter test run that verifies the acceleration software is working. This parameter executes a subset of sample tests. Details are included in signOfLife Tests on page 30. (default=0)
wirelessFirmware	Wireless Firmware enabled. Can be 0 (default) or 1. For configuration file details, see the *Programmer's Guide* for your platform.

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 several hours to complete. In addition, for RSA 4096 and DH 4096 tests, there can be up to an hour with no perceived result activity.

Kernel Space

After building the sample code, the kernel space kernel driver, the user space application, and the memory mapping driver are located at:

```
$ICP_ROOT/quickassist/lookaside/access_layer/sample_code/build
```

To execute the sign of life test in Kernel space, use the following commands:

```
# export ICP_ROOT=/<QATdir>
# cd $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/build
# insmod ./cpa_sample_code.ko signOfLife=1
```

Note: This test takes a few minutes to complete. When the `insmod` command is executed, there is no indication on the terminal window of the activities. Instructions on viewing the results are included in Test Results on page 31.

If loading of the module fails and some messages in `/var/log/messages` show `Device 0 not found or not stated` or `There are no crypto instances`, ensure that the kernel option `intel_iommu` has been configured as specified in Updating Grub Configuration File on page 13, and ensure that the appropriate configuration files have been copied from `$ICP_ROOT/quickassist/config` to `/etc`.

User Space

After building the sample code with the installation script, the kernel space kernel driver, the user space application, and the memory mapping driver are located at:

```
$ICP_ROOT/quickassist/lookaside/access_layer/sample_code/build
```

To execute the sign of life test in User space, use the following commands:

Install the kernel memory driver `qaeMemDrv.ko`, if the module has not already been installed.

```
# insmod $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/build/
qaeMemDrv.ko
```

```
# cd $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/build
# ./cpa_sample_code signOfLife=1
```

Note: You will observe that execution time of the user space code takes longer than the kernel space code. This is due to the sample code kernel space memory management driver (`qaeMemDrv.ko`), which is slow to allocate and map memory to user space. Before beginning performance measurements, the sample code allocates memory up-front which slows execution time. This does not affect the performance of the acceleration driver itself. The acceleration driver user space and kernel space performance are equivalent, other things being equal (for instance, no throttling takes place in either case).

4.1.2.2 Wireless Tests

The software package includes a version of the firmware optimized for small cryptography packets. In order to run the sample code with this firmware, the following steps must be performed.

1. An example configuration file is included in the package in the `$ICP_ROOT/quickassist/config` folder. For instance, for the Intel® Communications Chipset 8925 to 8955 Series:

```
# cp  $ICP_ROOT/quickassist/config/dh895xcc_qa_dev0.conf.v2.wireless /etc/
dh895xcc_qa_dev0.conf
```

   Note: The commands above apply to device dev0. The same commands should be issued for device dev1.

2. Restart the acceleration service using the command:

```
# service qat_service restart
```

3. Run the sample code using the command:

```
# cd $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/build
# ./cpa_sample_code wirelessFirmware=1
```

   Note: The `wirelessFirmware=1` parameter must be specified when the wireless config file is used. The following error messages are displayed if the parameter is not specified.

```
There are no crypto instances available
setupSymmetricTest():1135 Failed to start Crypto services
main():636 Error calling setupCipherTest
```

4.1.3 Test Results

When running the application in kernel space, open a second terminal window, log in as root, and issue the following command:

```
# tail -f /var/log/messages
```

When running the application in user space, 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.

4.1.4 Unloading the Sample Code

Once the kernel space sample code test has completed, the message `Sample Code Complete` is displayed. The module can then be unloaded using the following command:

```
# rmmod cpa_sample_code.ko
```

Once the user space sample code test has completed, the kernel memory driver `qaeMemDrv.ko` can be unloaded using the following command:

```
# rmmod qaeMemDrv.ko
```

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* (see listing in Table 1 on page 8).

4.3 Acceleration Functional Sample Code

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 both Kernel Space and User Space and the following sections contain instructions for both.

Note: The memory driver included with the sample application is a sample memory driver and is not intended for actual deployment.

4.3.1 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 that 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>
```

2. The sample code is compiled with the default assumption that the kernel source header files are located in one of the following directories:
   - For 64-bit: `/usr/src/kernels/3.1.0-7.fc16.x86_64`
   - For 32-bit: `/usr/src/kernels/3.1.0-7.fc16.i686`
3. If the kernel source header files are located in a different directory, create the environment variable with the directory of desired target kernel sources. For example:

```
# export KERNEL_SOURCE_ROOT=/usr/src/kernels/`uname -r`
```

4. If building with mux support, set the mux environment variable:

```
# export WITH_CPA_MUX=1
```

5. You can compile for both Kernel space and User space at the same time 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`

4.3.2 Executing the Acceleration Functional Sample Code in Kernel Space

To execute the acceleration functional sample code in Kernel space, enter the commands:

```
# export ICP_ROOT=<QATdir>
# cd $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/functional/build
# insmod ./nrbg_sample.ko
```

Note: `nrbg_sample.ko` is one of the functional kernel modules. You can launch the other `.ko` modules in a similar fashion.

Note: You may observe an error message similar to the following when submitting the `insmod` command:

```
insmod: error inserting '<modulename>': -1 Resource temporarily unavailable
```

This error can be safely ignored. When the test application is completed, the kernel object is removed which causes this error. Test results for the application are available in `/var/log/messages`.

4.3.3 Executing the Acceleration Functional Sample Code in User Space

To execute the acceleration functional sample code in User Space, the kernel memory driver `qaeMemDrv.ko` must be installed. See QuickAssist Acceleration Sample Application on page 27 for information on compiling the performance sample code. The `qaeMemDrv.ko` kernel object is built as part of that sample code.

1. Install the kernel memory driver `qaeMemDrv.ko` as follows:

```
# insmod $ICP_ROOT/quickassist/lookaside/access_layer/src/sample_code/build/
qaeMemDrv.ko
```

2. 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
#./nrbg_sample
```

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

Appendix A Glossary

AHCI	Advanced Host Controller Interface
B2B	Back-to-back. NTB connection scenario where one NTB is connected to a second NTB on another system.
BIOS	Basic Input/Output System
BOM	Bill of Materials
Coleto Creek	Codename for the Intel® Communications Chipset 8925 to 8955 Series
DIMM	Dual Inline Memory Module
EEPROM	Electrically Erasable Programmable Read Only Memory
ESD	Electrostatic Discharge
FIPS	Federal Information Processing Standard
FPT	Flash Programming Tool
GRUB	GRand Unified Bootloader
NTB	Non-Transparent Bridge
OSAL	Operating System Access Layer
PCH	Platform Controller Hub
PCI	Peripheral Component Interconnect
SATA	Serial Advanced Technology Attachment
Shumway	Codename for Shumway with Intel® Communications Chipset 8925 to 8955 Series
SM	System Management
SPI	Serial Peripheral Interconnect
USB	Universal Serial Bus
WDT	Watchdog Timer