Lanner AVE-ICE02 Coleto Creek Acceleration Module Implementation Manual

Rev 1.0.1 May 27, 2014

Revision History

Rev	Date	Changes
1.0.1	May 27, 2014	Initial version

Table of Contents

ABOUT THIS DOCUMENT **3**

PURPOSE **3**

INTENDED AUDIENCE **3**

CONVENTIONS USED **3**

CHAPTER 1. INTRODUCING THE LANNER COLETO CREEK ACCELERATION MODULES **4**

CHAPTER 2. INSTALLING THE OS AND PREPARING FOR THE SOFTWARE INSTALLATION .. **6**

2.1 CONFIGURING THE BIOS ON THE SYSTEM 6

2.2 INSTALLING FEDORA 16 7

2.3 UPDATING GRUB CONFIGURATION FILE 7

CHAPTER 3. BUILDING AND INSTALLING THE SOFTWARE **10**

3.1 UNPACKING THE SOFTWARE 10

3.2 INSTALLATION SCRIPT AND OPTIONS 11

3.3 STARTING/STOPPING THE ACCELERATION SOFTWARE 16

CHAPTER 4. RUNNING SAMPLE APPLICATIONS **18**

4.1 COMPILING THE ACCELERATION SAMPLE CODE 18

4.2 LOADING THE SAMPLE CODE 20

CHAPTER 5. APPENDIX A: SUPPORTED APIS AND DIRECTORY OF FUNCTION DEFINITIONS FILES 26

CHAPTER 6. APPENDIX B: ADDITIONAL INFORMATION AND GUIDES 29

About this document

Purpose

The purpose of this document is to provide implementation information for Intel QuickAssist software on Lanner AV-ICE02/AV-ICE04/AV-ICE05 and NCS2-ICE03A.

Intended audience

This document is for individuals who install and configure networking appliances with the above mentioned functionality.

Conventions used

Following are all the special characters and typographical conventions used in this manual:

Convention	Meaning
Press Enter	Means press the Enter or Return key or its equivalent on your computer.
📝	Note: introduces important additional information.
⚠	Caution: warns that a failure to follow the recommended procedure could result in loss of data or damage to equipment.

Chapter 1. Introducing the Lanner Coleto Creek Acceleration Modules

Lanner' products which utilize Intel Coleto Creek technology include network modules NCS2-ICE03A and PCIe acceleration card AV-ICE02/AV-ICE04/AV-ICE05. These modules utilize the latest Intel® Communications Chipset 8925 or 8950 Series silicon and are designed to efficiently increase performance for communications infrastructure systems.

More specifically, they are I/O interfaces with Intel QuickAssist Accelerator to optimize I/O performance on data compression and encryption on platforms powered by Intel Xeon and Core Processors.

This document provides information on how to quickly set up an environment and build and install the QuickAssist software.

For each supported acceleration service (Cryptographic, Data Compression), the following client access usage models are supported:

- Kernel mode, where both the client and the service(s) are running in kernel space.
- Direct user space access to services running in user space. In this model, both the client and service(s) are running in user space and access to the hardware is also performed from user space.

The Intel® Communications Chipset 8925/8950 Series Software for Linux has been validated with Fedora 16 32-bit and x86_64.

Note:

1. This document is written based on Intel® Communications Chipset 895x Series

Software for Linux package DH895xCC.L.1.0.1_31. Other software versions may have features or porting guides that are different from those described here.

2. For further assistance such as obtaining the SDK and the associated documentation, please contact Intel support directly.

Chapter 2. Installing the OS and Preparing for the Software Installation

2.1 Configuring the BIOS on the System

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

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 (or other command as instructed by your system's manual) when prompted.
2. Update the Boot order so that the DVD-ROM drive is the first boot option (if you the operating system is to be installed from the DVD-ROM). The option is available under:

   Boot > Boot Option Priorities (the menu directory depends on your BIOS)
3. Disable Intel® SpeedStep technology to achieve maximum Intel® QuickAssist Technology performance (if this option is configurable)

   Advanced >CPU Configuration>CPU Power Management Configuration> Intel (R) SpeedStep(^{TM}) or EIST (Enhanced Intel SpeedStep Technology)

Save your changes and exit. Then reboot the system.

2.2 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/

Select the software to install. Ensure that ***Software Developme***nt is selected.

Note:

1. When installing the Fedora, ensure that ***Software Development*** is selected or an error message may appear when installing the QuickAssist software.

2. The document is written with the Fedora 16 Install Media in mind. Using the Live Media version is not recommended.

2.3 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 Installing Fedora 16, 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, then 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=on"
```

Note: When virtualization is enabled, the acceleration services can only be used in the guest operating systems unless the IOMMU Remapping. Functions are used. See the Programmer’s Guide for your platform, specifically the “IOMMU Remappping Functions” section for more information.

Note: In rare instances, the system may hang with the Linux option acpi_enforce_resources=lax. If this occurs, this boot option should not be used.

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
   ```

Chapter 3. Building and Installing the Software

3.1 Unpacking the Software

The software package comes in the form of a tarball. 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
   ```

2. Transfer the tarball to the development board using any preferred method, for example USB memory stick, CDROM, etc. Unpack the tarball using the following command:

   ```
   #tar –zxof <tarball name>
   ```

The installation script and accelerations software tarball will be created under the QAT directory.

3.2 Installation Script and Options

The installation script is provided to walk you through building/installing the software. The installation script can also be launched with command line arguments giving the option to bypass the interactive setup. Refer to 3.2.3 Command Line Arguments or more information.

Launch the script as root:

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

A welcome message is displayed, followed by Installation Options. The table below lists the available 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 Host 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 Intel® Communications Chipset 89xx Series devices available on the system and the B:D.F for each device.
6	Builds Acceleration Sample Code	Builds both user space and kernel space version of the Acceleration Sample Code.
7	Install Hard Disk IDE Patch (*)	Performs kernel patch that enables IDE SATA mode. Kernel recompile is required when this option is selected.
8	Build Sample Drivers	Builds the sample drivers, including Watchdog Timer driver, Memory Scrubbing driver, NTB, and SPI driver.
9	Uninstall	Uninstalls the software. A sub-menu appears that allows you to select which software components are uninstalled.

0	Exit	Exit the installation script.
(*) not applicable to the AV-ICE02 and other Lanner Coleto Creek Acceleration cards and network modules.		

3.2.1 Acceleration Software Installation

When you run the installation script, select the **Install Acceleration** Installation option to install the Acceleration software. Type the following commands:

1. In the /QAT directory, start the installation script:

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

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:

1. 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 /QAT/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 configurationfiles from/QAT/quickassist/config to /etc and then restart the service with service qat_service restart.

2. 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 the GRUB configuration file. If this kernel parameter is not specified, acceleration ervices are only available in a guest operating system (in a virtualized

environment).

2. During the instllation, 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 dh89xxcc_aq_dev0.conf and h89xxcc_qa_dev1.conf) so that no core numbers are referenced beyond the core count of the system.

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.

3. Use the `0` option to exit the installation.
4. After installing the Acceleration software, it is recommended to verify that the acceleration software kernel object is loaded and ready to use. This can be done by performing the following operation:

```
# lsmod | grep icp_qa_al
```

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**.

3.2.2 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.

3.2.3 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
- 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

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

Example Usage:

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

3.2.4 Configuration Files

When the Acceleration software loads, it is configured based on settings in the configuration files (*dh895xcc_qa_dev0.conf* and *dh895xcc_qa_dev1.conf*). Both configuration files are placed in the /etc directory.

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.

The Intel Communications Chipset 8925/8950Series 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 Intel Communications Chipset 89xx Series, it is recommended that you verify that the correct configuration files were copied.

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

3.3 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
```

Note: 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 2.3 Updating Grub Configuration File

-If you are not using virtualization, verify you are starting the service in a guest Operating System and not the host Operating System.

-Verify the Intel Communications chipset 895x is enumerated properly by using the *lspci* command.

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 Intel® Communications Chipset 895x Series silicon.

Chapter 4. Running Sample Applications

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

The software package contains a set of sample tests that exercises the Intel Communications Chipset 895x Series 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 Compiling the Acceleration Sample Code

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

To build from the installation script, do the following:

1. Open a Terminal Window and switch to superuser:

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

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

   ```
   # cd /QAT
   ```

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

Select the **Build Acceleration Sample Code** installation option. 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.

3. Proceed to ***signOfLife*** Tests next page 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=/QAT
# export ICP_BUILDSYSTEM_PATH=$ICP_ROOT/quickassist/build_system
# export ICP_ENV_DIR=$ICP_ROOT/quickassist/build_system/build_files/env_files
```

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

```
# export ICP_BUILD_OUTPUT=$ICP_ROOT/build
# export ICP_TOOLS_TARGET=accelcomp
```

2. 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

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/linux
```

4. 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 next page for instructions on executing the tests.

4.2 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 icp_qa_al
# service qat_service status
```

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 3.3 Starting/Stopping the Acceleration Software 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.

Parameter	Description

SignOfLife=v	Indicates shorter test run that verifies the acceleration software is working. This parameter executes a subset of sample tests. Details are included in signOfLifeTests in the next section. (default=0)
cyNumBuffers=w	Number of buffers submitted for each iterations. (default =20)
cySymLoops=x	Number of iterations of all symmetric 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 tests.
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
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 *Intel® Communications Chipset 8925 to 8955 Series Software Programmer's Guide.*
wirelessFirmware	Wireless Firmware enabled. Can be 0 (default) or 1. For configuration file details, see the *Intel® Communications Chipset 8925 to 8950 Series Software Programmer's Guide*.

4.2.1 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=/QAT

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

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

Note:

1. 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 on next page.

2. 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**. And ensure that the appropriate configuration files have been copied from /QAT/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
```

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).

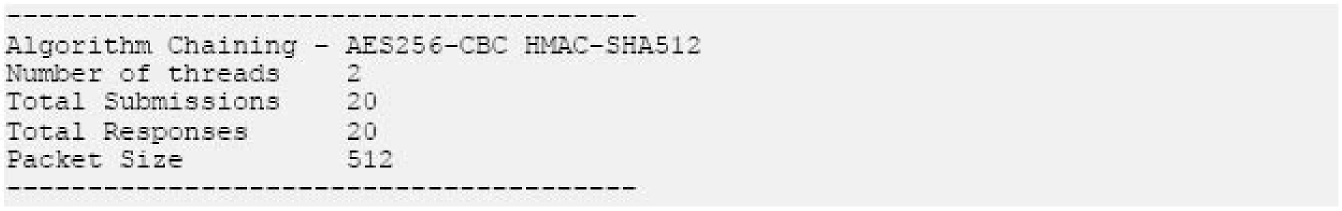
4.2.2 Test Results

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.

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
---------------------------------------
```

4.2.3 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
```

Chapter 5. Appendix A: Supported APIs and directory of function definitions files

Intel® QuickAssist Technology APIs

- Cryptographic Functions

API definitions are located in: $ICP_ROOT/quickassist/include/lac,

where *$ICP_ROOT* is the directory where the Acceleration software is unpacked.

- Data Compression Functions:

API definitions are located in: $ICP_ROOT/quickassist/include/dc.

Base API definitions

Base API definitions that are common to the API libraries are located in: $ICP_ROOT/quickassist/include.

DATA Plane APIs

The data plane APIs are intended for use in user space applications that take advantage of the functionality provided of the Intel Data Plane Development Kit (Intel DPDK). The APIs are recommended for applications that are executing in a data plane environment where the cost of offload (that is, the cycles consumed by the driver sending requests to the hardware) needs to be minimized. To minimize the cost of offload, several constraints have been placed on the APIs. If these constraints are too restrictive for your application, the traditional APIs can be used instead (at a cost of additional IA cycles).

- Cryptographic DATA Plane Functions

The definition of the Cryptographic Data Plane APIs are contained in:

$ICP_ROOT/quickassist/include/lac/cpa_cy_sym_dp.h

- Data Compression Data Plane Functions

The definition of the Data Compression Data Plane APIs are contained in:

$ICP_ROOT/quickassist/include/dc/cpa_dc_dp.h

Additional APIS

- Polling Functions:

These functions are intended for retrieving response messages that are on the rings and dispatching the associated callbacks.

All Polling function definitions are located in:
$ICP_ROOT/quickassist/lookaside/access_layer/include/icp_sal_poll.h

- Random Number Generation Functions

All random number generation function definitions are located in the following header files:

$ICP_ROOT/quickassist/lookaside/access_layer/include/icp_sal_drbg_impl.h

$ICP_ROOT/quickassist/lookaside/access_layer/include/icp_sal_drbg_ht.h

$ICP_ROOT/quickassist/lookaside/access_layer/include/icp_sal_nrbg_ht.h

- User Space Access Configuration Functions

Functions that allow the configuration of user space access to the Intel® QuickAssist Technology services from processes running in user space.

All user space access configuration function definitions are located in $ICP_ROOT/quickassist/lookaside/access_layer/include/icp_sal_user.h.

- User Space Heartbeat Functions

These functions allow the user space application to check the status of the firmware/hardware of the Intel® Communications Chipset 8925 to 8950 Series device as part of the Heartbeat functionality.

All user space heartbeat function definitions are located in $ICP_ROOT/quickassist/lookaside/access_layer/include/icp_sal_user.h.

- Version Information Function

A function that allows the retrieval of version information related to the software and hardware being used.

The version information function definition is located in: $ICP_ROOT/quickassist/lookaside/access_layer/include/icp_sal_versions.h.

Chapter 6. Appendix B: Additional Information and Guides

The Intel quickAssist Technology APIs provide the interface to acceleration services including cryptographic and data compression functionalities. These functionalities have been documented in the following software library documentation:

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 following guides:

Intel® Communications Chipset 8925 to 8955 Series Software package (No. 523127)

Intel® Communications Chipset 8925 to 8955 Series Software for Linux* Getting Started Guide (No. 523128)

Intel® Communications Chipset 8925 to 8955 Series Software Release Notes (No. 523129)

Intel® Communications Chipset 8925 to 8955 Series Software Programmer's Guide (No.523126)

Intel® QuickAssist Technology API Programmer's Guide (No. 442844)

Intel® QuickAssist Technology Cryptographic API Reference Manual (No. 410923)

Intel® QuickAssist Technology Data Compression API Reference Manual (No. 410925)

Intel® Xeon® Processor E5-2658 and E5-2448L with Intel® Communications Chipset 8920 Development Kit User Guide (No. 470433)

To obtain these manuals, go to www.intel.com/ibl.

You will need an Intel Business Portal account and a subscription to the Electronic Design Kit (EDK) to access the Intel Business Portal.

OpenSSL Library Inclusion and Usage

The Intel Coleto Creek chipset software Linux package is distributed with an OpenSSL library file. Customer applications may connect to the services through the supported open source frameworks and associated patches.

Software packages for patches, such as OpenSSL and zlib* are distributed separately:

libcrypto* (OpenSSL*) Sample Patch for Intel(R) QuickAssist Technology

Please see document number *477629* or *540272* (available at www.intel.com/ibl) to download the libcrypto Sample Patch for Intel QuickAssist Technology.

zlib* Sample Patch for Intel(R) QuickAssist Technology for Intel(R) Communications Chipset 89xx Series Software

Please see document number 480522 (available at www.intel.com/ibl) to download the zlib* Sample Patch for Intel QuickAssist Technology.

The Intel® Communications Chipset 89xx Series Software for **FreeBSD*** package

https://downloadcenter.intel.com/Detail_Desc.aspx?agr=Y&DwnldID=20864

Intel Virtualization Technology

If you are using the hardware-assisted Intel® Virtualization Technology (Intel® VT) on the system, the document "*Using Intel Virtualization Technology with the Intel Communications Chipset 89xx Series Software*" and "*Using Intel® Virtualization Technology (Intel® VT) with Intel® QuickAssist Technology Application Note*" provide detailed information on deploying acceleration software in a virtualized environment.