Migrating Applications to Linux* Kernel Version 2.6.24

For use with Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology

Application Note

June 2009

Order Number: 322282-001US

Contents

Figures

Tables

Revision History

Date	Revision	Description
June 2009	001	Initial release as public document.

§ §

1.0 Introduction

The purpose of this document is to describe the changes made to the Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology to allow it to run on a Linux* version 2.6.24 kernel. There has been no attempt to optimize the code. All software packages, detailed procedures, and documentation are listed in the Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology for Linux* Getting Started Guide.

This document describes the procedure changes that must be followed in order to build on a Vanilla Linux* distribution. This Vanilla kernel was built on a machine running CentOS 5.

1.1 Related Document

- Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology for Linux* Getting Started Guide, 320182-003 (November 2008) http://download.intel.com/design/intarch/ep80579/320182.pdf

This document will be referred to as the GSG going forward.

1.2 Additional Software Required

In addition to the software listed in the GSG, the files in Table 1 are also required.

Table 1. Additional Software Required

Software	Location
CentOS 5	http://mirror.centos.org/centos/5/isos/
2.6.24 kernel	http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.24.7.tar.gz
Patch to Linux* 2.6.24 Kernel	http://downloadcenter.intel.com/Detail_Desc.aspx?agr=Y&DwnldID=17572 filename: 2.6.24_KERNEL.patch
Patch to Intel® EP80579 acceleration software package	http://downloadcenter.intel.com/Detail_Desc.aspx?agr=Y&DwnldID=17572 filename: 2.6.24_Acceleration.patch

2.0 Installing the 2.6.24 Kernel

Install and patch the CentOS kernel as described in GSG Section 3.0, Installing the OS on a Development Board.

Note: CentOS is derived from Red Hat* Enterprise Linux*, therefore the required steps are similar. However, because the Vanilla kernel that the Security modules are built on is version 2.6.24, as opposed to the Red Hat Linux kernel version 2.6.18 described in the GSG, some changes are required.

This section lists a complete set of the required steps to build the kernel based on the GSG. Certain steps in the GSG were not followed here as the purpose of this guide is to build the Acceleration Modules. Also, Section 2.2, Step 2 includes instructions for an additional patch for the 2.6.24 kernel.

Note: The following commands assume the default bash shell is in use, as stated in the GSG.

1. Set the correct date as per the GSG Section 3.4, step 1. Then download the 2.6.24 kernel source and use tar -zxvf to extract it to /usr/src/kernels.

   ```
   tar -C /usr/src/kernels -zxvf linux-2.6.24.7.tar.gz
   ```

2. Create the /EP805XX_release directory as per GSG Section 3.4, steps 2 and 3 and copy the contents of the release package (named Security.L.1.0.1-66.tar.gz) to this new directory. The StagingArea directory is also required to be created. Refer to the example below for the location of the created directories.

```
mkdir /EP805XX_release
export ICP_ROOT=/EP805XX_release
cp -R <Path-to-Release_package> $ICP_ROOT
mkdir /EP805XX_release/StagingArea
export ICP_BUILD_OUTPUT=$ICP_ROOT/StagingArea
export ICP_BUILDSYSTEM_PATH=$ICP_ROOT/build_system
```

3. Set the environment variable KERNEL_SOURCE_ROOT to the new kernel source directory and set a soft link to the /usr/src/kernels/linux path:

```
export KERNEL_SOURCE_ROOT=<path to kernel source>
mkdir /usr/src/kernels /* if not already created*/
ln -s <path to kernel source>/ /usr/src/kernels/linux
```

4. Follow the instructions in GSG Section 3.5 Unpacking the EP80587 Security Software Linux Package in order to extract the security software package.

```
cd $ICP_ROOT
tar -xmvzf Security.L.1.0.1-66.tar.gz
tar -xmvzf Security.L.1.0.1-66_SW.tar.gz
tar -xmvzf PatchFiles_Security.L.1.0.1-66.tar.gz
```

Note: The kernel was not patched for API device recognition as this is already included in the 2.6.24 kernel, so the steps listed in Section 3.6 Patching the Kernel for PCI Device Recognition do not need to be followed.

2.1 Patching OCF and Configuring OCF Parameters

Perform the following steps to configure the OCF parameters. You do not need to perform the tasks in the GSG Section 3.7.1 Applying the OCF Patch and Section 3.7.2 Configuring OCF Parameters.

1. The archive containing the source for this patch can be obtained at this URL:

   http://heanet.dl.sourceforge.net/sourceforge/ocf-linux/ocf-linux-26-20070727.patch.gz

   Download the patch to the $ICP_ROOT directory, gunzip it and apply the patch as follows:

```
cd $ICP_ROOT
gunzip ocf-linux-26-20070727.patch.gz
cd $KERNEL_SOURCE_ROOT
patch -p1 < $ICP_ROOT/ocf-linux-26-20070727.patch
```

2. Run the following commands to modify $KERNEL_SOURCE_ROOT/crypto/make by appending the line 'obj-$(CONFIG_OCF_OCF) +=ocf/'

Note: Do not modify the file $KERNEL_SOURCE_ROOT/crypto/Kconfig as described in the GSG; it will cause errors when attempting to build the kernel in Section 2.2 of this document.

```
cd $KERNEL_SOURCE_ROOT/crypto
echo "obj-\$(CONFIG_OCF_OCF) += ocf/" >> Makefile
```

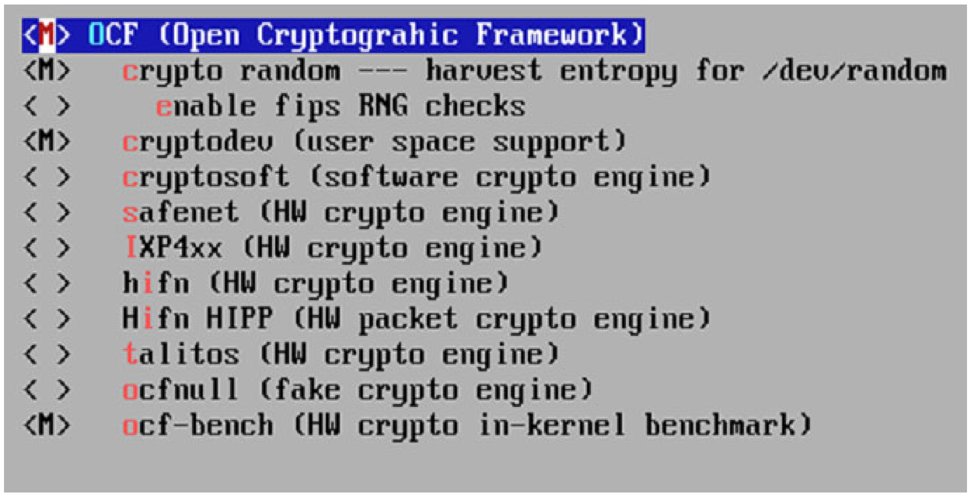
3. Run the following commands to configure OCF within the kernel

```
cd $KERNEL_SOURCE_ROOT
make mrproper
make menuconfig
```

In the graphical interface that is spawned, set the options shown in Figure 1 under Cryptographic Options -> OCF Configuration (this option will not be available unless the patch has been applied).

Note: Cryptosoft will not compile correctly and therefore should be disabled as it is not required for the acceleration modules to operate.

Figure 1. OCF Configuration GUI Settings



```
<M> OCF (Open Cryptograhic Framework)
<M>    crypto random --- harvest entropy for /dev/random
< >      enable fips RNG checks
<M>    cryptodev (user space support)
< >    cryptosoft (software crypto engine)
< >    safenet (HW crypto engine)
< >    IXP4xx (HW crypto engine)
< >    hifn (HW crypto engine)
< >    Hifn HIPP (HW packet crypto engine)
< >    talitos (HW crypto engine)
< >    ocfnull (fake crypto engine)
<M>    ocf-bench (HW crypto in-kernel benchmark)
```

2.2 Building and Patching the Kernel

1. Perform the following steps described in GSG Sections 3.7.2.1 and 3.7.3 to backport certain OCF features and also add some bug fixes.

```
mkdir -p /usr/include/crypto
cp $KERNEL_SOURCE_ROOT/crypto/ocf/cryptodev.h /usr/include/
crypto/
cd $KERNEL_SOURCE_ROOT
patch -p0 < $ICP_ROOT/OpenSourcePatches/linux-ocf-20070727-
backport.patch
patch -p0 < $ICP_ROOT/OpenSourcePatches/ocf-linux-20070727-
driver-removal.patch
```

2. Apply the patch to the kernel as follows:

```
cp <directory containing patch>/2.6.24_KERNEL.patch
$KERNEL_SOURCE_ROOT
cd $KERNEL_SOURCE_ROOT
patch -p0 < 2.6.24_KERNEL.patch
```

The kernel patch changes $KERNEL_SOURCE_ROOT/crypto/ocf/crypto.c and removes the last arg from both references to kmem_cache_create on lines 1575 and 1577 (removes the final NULL parameter for each). The kernel driver function kmem_cache_create no longer takes the parameter for cache deconstructor, making it necessary to update the parameters it takes in with the final NULL argument's removal. The kernel patch applied in this step contains these changes:

```
-   cryptop_zone = kmem_cache_create("cryptop", sizeof(struct cryptop),
-                                        0, SLAB_HWCACHE_ALIGN, NULL, NULL);
-   cryptodesc_zone = kmem_cache_create("cryptodesc", sizeof(struct cryptodesc),
-                                             0, SLAB_HWCACHE_ALIGN, NULL, NULL);
```

```
+   cryptop_zone = kmem_cache_create("cryptop", sizeof(struct cryptop),
+                                  0, SLAB_HWCACHE_ALIGN, NULL);
+   cryptodesc_zone = kmem_cache_create("cryptodesc", sizeof(struct cryptodesc),
+                                      0, SLAB_HWCACHE_ALIGN, NULL);
```

3. Follow the commands in GSG Section 3.8 Rebuilding the Kernel that are not specific to the 2.6.18 kernel. This allows the 2.6.24 kernel to be built and also enables simpler file locations and easier identification of said kernel on startup.

Alternatively, it is possible to run the following commands to build the kernel and skip the options listed in this section of the GSG:

```
make; make modules; make modules_install; make install
```

At this point, reboot the machine and boot into the newly created 2.6.24 kernel.

2.3 Known Issues

- Adding the line 'source "crypto/ocf/Kconfig"' at line 364 of the $KERNEL_SOURCE_ROOT/crypto/ocf/Kconfig file causes errors when attempting to build the kernel. No issues when removed. The sample system command to do this is:

  `sed -i 364i"source\ \"crypto/ocf/Kconfig\"" Kconfig`
- Cryptosoft will not compile with this kernel version due to a scatterlist error; hence this option is disabled as it is not required for the acceleration modules.

3.0 Installing and Patching Software Modules

This section describes the code changes made to allow the release modules to compile on the 2.6.24 kernel. It also addresses how to apply the acceleration library patch generated from these changes and how to build the resultant software modules.

3.1 Code Changes

The following code changes are implemented in the acceleration library patch to allow the release modules to compile on the 2.6.24 kernel. When applied, the acceleration library patch includes all changes documented below. Instructions for applying the patch are provided in Section 3.3.

Note: Changes made to the kernel source are **not** included in the acceleration library patch.

- The following files are required to refer to the environment variables or Makefile variables called by "EXTRA_CFLAGS" as opposed to "CFLAGS"

  `/EP805XX_release/build_system/build_files/common.mk`

  `/EP805XX_release/Acceleration/library/icp_services/linux_2.6_kernel_space.mk`

  `/EP805XX_release/Acceleration/library/icp_debugmgmt/MIL/source/Makefile`
- Depending on the Linux kernel version in use, either autoconf.h or config.h should be included. The config.h file is no longer in use when using a kernel higher than 2.6.18 and is kept only for backward compatibility. You must edit the file:

  `/EP805XX_release/Acceleration/library/icp_debugmgmt/MIL/source/mil_drv/mil_linux_kernel.c`

  to include:

```
#include <linux/autoconf.h>
```

- The unregister_chrdev function returns a void as of the 2.6.23 kernel version, so it is necessary to update the mil_fini function in /EP805XX_release/Acceleration/library/icp_debugmgmt/MIL/source/mil_drv/mil_linux_kernel.c on line 544 so that it reads:

```
-  status = unregister_chrdev (
+  unregister_chrdev (

                major_number ,
                MIL_DEVICE_NAME );
```

With the status value no longer returned, the error checking that follows is no longer required and has been deleted.

```
-      if ( status < 0 )
-      {
-         MIL_KERNEL_DEBUG_PRINT("[MIL_ks:mil_fini] "\
-            "failed to unregister the MIL device driver.\n");
-         return;
-      }
```

- As described in Section 2.2 of this document, it is necessary to update kmem_cache_create's calls in /EP805XX_release/Acceleration/shims/OCF_Shim/src/icp_common.c and in ICP_CACHE_CREATE in /EP805XX_release/Acceleration/shims/OCF_Shim/src/icp_ocf.h by removing the final NULL argument for each use.

For /EP805XX_release/Acceleration/shims/OCF_Shim/src/icp_common.c

```
drvOpData_zone =
   kmem_cache_create("ICP Op Data", sizeof(struct
icp_drvOpData) +
-      defBuffListInfo.metaSize ,0, SLAB_HWCACHE_ALIGN, NULL,
NULL);
+      defBuffListInfo.metaSize ,0, SLAB_HWCACHE_ALIGN, NULL);
```

For /EP805XX_release/Acceleration/shims/OCF_Shim/src/icp_ocf.h

```
#define ICP_CACHE_CREATE(cache_ID, cache_name)             \
kmem_cache_create(cache_ID, sizeof(cache_name),0,          \
-              SLAB_HWCACHE_ALIGN, NULL, NULL);
+              SLAB_HWCACHE_ALIGN, NULL);
```

- It is also required to edit the INIT_WORK macro at line 491 of icp_common.c. The updated kernel function has been simplified to take two arguments instead of the three passed to the function in the 2.6.18 kernel, with the function container_of used to assign the removed argument, as shown below.

```
INIT_WORK(&(workstore->work),
            icp_ocfDrvDeferedFreeLacSessionProcess)
```

icp_ocfDrvDeferedFreeLacSessionProcess on line 505 of the same file must also be edited to use the updated means of initializing the work queue using the container_of macro.

```
static void icp_ocfDrvDeferedFreeLacSessionProcess(void *arg)
 {
   struct icp_ocfDrvFreeLacSession *workstore = NULL;
+  struct workqueue_struct *wq = NULL;
   CpaCySymSessionCtx sessionToDeregister = NULL;
   int i = 0;
   int remaining_delay_time_in_jiffies = 0;
   CpaStatus lacStatus = CPA_STATUS_SUCCESS;
```

```
-   workstore = (struct icp_ocfDrvFreeLacSession *)arg;
-   if (NULL == workstore) {
+   if (NULL == arg) {
       DPRINTK("%s() function called with null parameter \n",
                __FUNCTION__);
       return;
   }
+/*retrieve session object using the pointer to its 'work'
member*/
+   wq = (struct  workqueue_struct *)arg;
+   workstore = container_of(wq,
                              struct icp_ocfDrvFreeLacSession,
                              work);
+
   sessionToDeregister = workstore->sessionToDeregister;
   kfree(workstore);
```

- The struct new_utsname is no longer required for the file /EP805XX_release/Acceleration/library/icp_utils/OSAL/platforms/EP805XX/os/linux/src/IxOsalOsOem.c, hence line 76 is deleted:

```
-   extern struct new_utsname system_utsname;
```

- In IxOsalOsOem.c (the same file as above), system_utsname from the 2.6.18 kernel has been replaced by init_utsname for the 2.6.24 kernel. Therefore, lines 129 and 136 must be altered to reflect this change:

```
-   strncpy(osName, system_utsname.sysname, maxSize);
+   strncpy(osName, (init_utsname())->sysname, maxSize);
-   strncpy(osVersion, system_utsname.release, maxSize);
+   strncpy(osVersion, (init_utsname())->release, maxSize);
```

- In /EP805XX_release/Acceleration/library/icp_utils/OSAL/common/os/linux/src/modules/ddk/IxOsalOsDdkIrq.c

```
-   local_save_flags((unsigned long)flags);
+   local_save_flags(flags);
```

- SA_SHIRQ has been deprecated and replaced with IRQF_SHARED in the 2.6.24 kernel, hence SA_SHIRQ must be replaced by IRQF_SHARED in the following files. Use vi and the string replacement command ":% s /SA_SHIRQ/IRQF_SHARED/gc" to find and replace, ensuring no variable names are changed in error:

```
Acceleration/library/icp_services/RuntimeTargetLibrary/
   Target_CoreLibs/halAe/linuxAeDrv.c
Acceleration/drivers/icp_asd/src/kernel/linux/asd_isr.c
```

- The compiled kernel does not have the asm-i386 directory. As a result, the asm directory is used in place in the following file; /EP805XX_release/Acceleration/library/icp_services/RuntimeTargetLibrary/Target_CoreLibs/halAe/include/linux/halAe_platform.h

```
-   <asm-i386/processor.h>
+   <asm/processor.h>
```

- This also applies to the sample code which can be used to verify correct operation of the security modules. The following files need to be edited to allow the sample code module to be built and insmodded:

```
/EP805XX_release/Acceleration/library/icp_crypto/
   look_aside_crypto/src/sample_code/performance/
   cpa_linux_mem_utils.c
```

```
/EP805XX_release/Acceleration/library/icp_crypto/
   look_aside_crypto/src/sample_code/performance/
   cpa_main_utils_perf.c
```

In both files, change;

```
#include <asm-i386/io.h>
```

to

```
#include <asm/io.h>
```

- Building the security modules will fail if the makefiles of qatal and asd do not contain the include path of the file core_platform.h.

  EP805XX_release\build_system\build_files\includes.mk contains all runtime tagetLibs including this path for all modules but qatal and asd. They include this file from "EXTRA_CFLAGS += $(INCLUDES)". However, these environment variables will not be included from EXTRA_CFLAGS as it does not contain the path to the core_platform.h file. You must edit the following files to resolve this. Neither ICP_IX_TOOLS_DIR nor TOOLS_DIR are used in any file in the security source package so no issues will arise from these changes.

  — Edit line 69 in /EP805XX_release/Acceleration/library/icp_crypto/QATAL/linux_2.6_kernel_space.mk,

  from

  ```
  -I$(ICP_IX_TOOLS_DIR)/include/os/linux
  ```

  to

  ```
  -I$(ICP_TOOLS_DIR)/include/os/linux
  ```

  — Edit line 103 in /EP805XX_release/Acceleration/drivers/icp_asd/src/kernel/linux/Makefile

  from

  ```
  TOOLS_INCLUDES=-I$(TOOLS_DIR)/include
  ```

  to

  ```
  TOOLS_INCLUDES=-I$(ICP_AE_TOOLS_DIR)/include
  TOOLS_INCLUDES+=-I$(ICP_AE_TOOLS_DIR)/include/os/linux
  ```

- The variable KBUILD_MODNAME cannot be shared between multiple modules and to do so will cause an error. As a result, the file pci.h which calls KBUILD_MODNAME has been removed from the files below that do not use it.

  — /EP805XX_release/Acceleration/library/icp_utils/OSAL/platforms/EP805XX/os/linux/include/IxOsalOsOem.h

  — /EP805XX_release/Acceleration/library/icp_utils/OSAL/common/os/linux/include/core/IxOsalOs.h

  — /EP805XX_release/Acceleration/library/icp_utils/OSAL/common/os/linux/src/modules/ddk/IxOsalOsDdkPci.c

  — /EP805XX_release/Acceleration/drivers/icp_asd/include/icp_accel_handle.h

  — /EP805XX_release/Acceleration/drivers/icp_asd/src/kernel/linux/asd_drv.c

  — /EP805XX_release/Acceleration/drivers/icp_asd/src/kernel/linux/asd_acpi.c

  The ASD header file /EP805XX_release/Acceleration/drivers/icp_asd/src/kernel/include/asd_drv.h has also been edited to define the KBUILD_MODNAME variable here, thus avoiding any potential error:

  ```
  + #ifndef KBUILD_MODNAME
  + #define KBUILD_MODNAME "icp_asd"
  + #endif
  + #include <linux/pci.h>
  ```

3.2 Warnings

There are some incompatible warnings when building HAL and ASD with 2.6.24 kernel that may impact on runtime:

```
warning: pci_find_device is deprecated
warning: passing argument from incompatible pointer type
```

1. warning: pci_find_device is deprecated

   Replacing pci_find_device with pci_get_device & pci_dev_put will resolve this issue, however pci_find_device is still supported under the 2.6.24 kernel so this change is not essential.

2. warning: passing argument 2 of request_irq from incompatible pointer type

   The definition of ISR has changed between the 2.6.18 kernel and 2.6.24 kernel as follows:

   ```
   typedef irqreturn_t (*irq_handler_t)(int, void *, struct
   pt_regs *); /* 2.6.18 kernel */
   typedef irqreturn_t (*irq_handler_t)(int, void *); /* 2.6.24
   kernel */
   ```

   Accordingly, the following changes have been made in /EP805XX_release/Acceleration/drivers/icp_asd/src/kernel/linux/asd_isr.c

   from:

   ```
   static irqreturn_t asd_intr(int irq, void *privdata, struct
   pt_regs *ptx)
   ```

   to

   ```
   static irqreturn_t asd_intr(int irq, void *privdata)
   ```

   Additional changes in /EP805XX_release/Acceleration/library/icp_services/RuntimeTargetLibrary/Target_CoreLibs/halAe/linuxAeDrv.c

   from:

   ```
   irqreturn_t halAe_Intr_attn_ISR_ICP(int irq, void *dev_id,
   struct pt_regs *regs)
   ```

   to:

   ```
   irqreturn_t halAe_Intr_attn_ISR_ICP(int irq, void *dev_id)
   ```

3. The definition of "ids" in "struct acpi_driver" is different between 2.6.18 kernel and 2.6.24 kernel as follows:

   ```
   /* 2.6.18 kernel */
   struct acpi_driver {
      struct list_head node;
      char name[80];
      char class[80];
      atomic_t references;
      char *ids;              /* Supported Hardware IDs */
      struct acpi_device_ops ops;
   };

   /* 2.6.24 kernel */
   struct acpi_driver {
      char name[80];
      char class[80];
      const struct acpi_device_id *ids; /*Supported Hardware IDs*/
      struct acpi_device_ops ops;
      struct device_driver drv;
   ```

```
   struct module *owner;
};

#define ACPI_ID_LEN     16 /* only 9 bytes needed here */
                           /*16 bytes are used to */
                           /*workaround crosscompile issues */

struct acpi_device_id {
   __u8 id[ACPI_ID_LEN];
   kernel_ulong_t driver_data;
};
```

It is necessary to update the file /EP805XX_release/Acceleration/drivers/icp_asd/src/kernel/linux/asd_acpi.c as described below to avoid an incompatible pointer type warning.

```
- /*#define ACPI_MEMORY_DEVICE_HID   "PNP0C02"*/
+ static const struct acpi_device_id ACPI_MEMORY_DEVICE_HID[] =
{
+  {"PNP0C02", 0},
+  {"", 0},
+ };
```

3.3 Applying the Acceleration Library Patch

1. Before applying the patch and building the security modules, ensure that all of the following environment variables are set:

```
export ICP_ROOT=/EP805XX_release
export ICP_BUILDSYSTEM_PATH=$ICP_ROOT/build_system
export ICP_BUILD_OUTPUT=$ICP_ROOT/StagingArea
export ICP_ENV_DIR=$ICP_ROOT/Acceleration/library/icp_crypto/
   look_aside_crypto
export KERNEL_SOURCE_ROOT=<Path to Kernel Source>
export ICP_OCF_SRC_DIR=$KERNEL_SOURCE_ROOT/crypto/ocf
export KERNELSRC=$KERNEL_SOURCE_ROOT
```

2. Copy the patch to the $ICP_ROOT directory and run the following commands:

```
cd $ICP_ROOT
patch -p0 < 2.6.24_Acceleration.patch
```

3. The output to the console in the case of successful patching should read:

```
patching file Acceleration/library/icp_crypto/QATAL/linux_2.6_kernel_space.mk
patching file Acceleration/library/icp_services/RuntimeTargetLibrary/Target_CoreLibs/halAe/linuxAeDrv.c
patching file Acceleration/shims/OCF_Shim/src/icp_common.c
patching file Acceleration/drivers/icp_asd/src/kernel/linux/asd_drv.c
patching file Acceleration/library/icp_utils/OSAL/common/os/linux/src/modules/ddk/IxOsalOsDdkIrq.c
patching file Acceleration/library/icp_utils/OSAL/platforms/EP805XX/os/linux/src/IxOsalOsOem.c
patching file Acceleration/shims/OCF_Shim/src/icp_ocf.h
patching file build_system/build_files/common.mk
patching file Acceleration/drivers/icp_asd/src/kernel/linux/Makefile
patching file Acceleration/library/icp_debug/DCC/source/Makefile
patching file Acceleration/library/icp_debugmgmt/MIL/source/Makefile
patching file Acceleration/library/icp_debugmgmt/MIL/source/mil_drv/mil_linux_kernel.c
patching file Acceleration/library/icp_services/linux_2.6_kernel_space.mk
patching file Acceleration/library/icp_services/RuntimeTargetLibrary/Target_CoreLibs/halAe/include/linux/
halAe_platform.h
```

```
patching file Acceleration/library/icp_crypto/look_aside_crypto/src/sample_code/performance/
cpa_linux_mem_utils.c
patching file Acceleration/library/icp_crypto/look_aside_crypto/src/sample_code/performance/
cpa_main_utils_perf.c
patching file Acceleration/drivers/icp_asd/src/kernel/linux/asd_isr.c
Hunk #2 succeeded at 276 with fuzz 2.
patching file /EP805XX_release/Acceleration/drivers/icp_asd/src/kernel/linux/asd_acpi.c
patching file /EP805XX_release/Acceleration/library/icp_utils/OSAL/platforms/EP805XX/os/linux/include/
IxOsalOsOem.h
patching file /EP805XX_release/Acceleration/library/icp_utils/OSAL/common/os/linux/include/core/IxOsalOs.h
patching file /EP805XX_release/Acceleration/library/icp_utils/OSAL/common/os/linux/src/modules/ddk/
IxOsalOsDdkPci.c
patching file /EP805XX_release/Acceleration/drivers/icp_asd/include/icp_accel_handle.h
patching file /EP805XX_release/Acceleration/drivers/icp_asd/src/kernel/include/asd_drv.h
patching file /EP805XX_release/Acceleration/drivers/icp_asd/src/kernel/linux/asd_drv.c
```

4. If a specific file has already been patched, the following message is displayed. It is not necessary to patch a second time, but it is also possible one of the files might not be patched so type n to ensure this is not the case.

   ```
   Reversed (or previously applied) patch detected!  Assume -R?
   [n]
   ```

5. The following commands will install all software modules contained in the release.

   ```
   cd $ICP_ROOT/Acceleration
   make

   make ocf

   modprobe ocf
   modprobe cryptodev

   cd ..
   make install
   ```

   Error messages for Embedded modules will be displayed at this point but the Security modules will be built and installed correctly.

3.4 Sample Code

Note: If the acceleration library patch described in Section 3.3 has already been applied, then you can go to step 2 below.

1. As stated previously, the following files need to be edited to allow the sample code module to be built and inserted into the kernel using the insmod command:

   ```
   /EP805XX_release/Acceleration/library/icp_crypto/
      look_aside_crypto/src/sample_code/performance/
      cpa_linux_mem_utils.c
   /EP805XX_release/Acceleration/library/icp_crypto/
      look_aside_crypto/src/sample_code/performance/
      cpa_main_utils_perf.c
   ```

   In both, change;

   ```
   #include <asm-i386/io.h>
   ```
   to
   ```
   #include <asm/io.h>
   ```

2. Go to the sample code directory /EP805XX_release/Acceleration/library/icp_crypto/look_aside_crypto/src/sample_code and follow the instructions in the README.txt file in the sample code directory.

```
cd performance
make module
insmod ./build/linux_2.6/kernel_space/crypto_perf.ko
```

If working directly from the workstation, the messages displayed by these tests can be seen on screen. If working from a remote login, use the **dmesg** command to see the test results. A successful build will display "Performance Code Complete- All Tests Pass" in dmesg. Once the tests have completed, the module can be removed and the Security package has been verified as operational.

§ §