WIND RIVER

Wind River® Intelligent Device Platform XT

IDP XT VALIDATION PLAN

2.0

EDITION 1

Copyright Notice

Corporate Headquarters

Wind River
500 Wind River Way
Alameda, CA 94501-1153
U.S.A.
Toll free (U.S.A.): 800-545-WIND
Telephone: 510-748-4100
Facsimile: 510-749-2010

For additional contact information, see the Wind River Web site:

www.windriver.com

For information on how to contact Customer Support, see:

www.windriver.com/support

18 Jul 2014

1

Introduction

Wind River Intelligent Device Platform XT (IDP XT) provides integrated development and management support for distributed systems that use smart services with cloud computing. It includes secure remote management layer for cloud-based smart services, including automated customer interaction and support. Wind River® Intelligent Device Platform XT is a scalable, sustainable, and secure development environment that simplifies the development, integration, and deployment of Internet of Things (IoT) gateways. It is based on Wind River industry-leading operating systems and the Wind River development tools.

The platform provides device security, smart connectivity, rich network options, and device management. Intelligent Device Platform XT is part of the Intel® Gateway Solutions for IoT, a family of platforms that enables companies to seamlessly interconnect industrial devices and other systems into a system of systems. Intel Gateway Solutions for IoT enables customers to securely aggregate, share, and filter data for analysis. It helps ensure that federated data generated by devices and systems can travel securely and safely from the edge to the cloud and back—without replacing existing infrastructure.

Intel Gateway Solutions for IoT offers companies a key building block to enable the connectivity of legacy industrial devices and other systems to IoT. It integrates technologies and protocols for networking, embedded control, enterprise-grade security, and easy manageability on which application-specific software can run.

Purpose of this Document

This document is intended to be used by third parties who have made changes to the delivered platform software using the Devolpment Kit's software tools. In particular the target audience is ODMs who have performed a BSP port and want to re-validate the IDP XT capabilities. This document describes the test environment and the steps to verify the functionality of the IDP XT functional blocks.

This document is written for a software and hardware tester audience who need to validate the capabilities of IDP XT. The document contains brief guidelines for validating tasks such as project and system configuration, security and connectivity testing, and application development.

Where to Find Information

The following is a list of the documentation included with Wind River Intelligent Device Platform XT.

Wind River Intelligent Device Platform XT Programmer's Guide

provides instructions for installing, configuring the Intelligent Device Platform and modifying it for your specific requirements (this document).

Wind River Intelligent Device Platform XT Security Guide

provides guidance on performing a security analysis and matching IDP XT capabilities with assessed needs.

Wind River Intelligent Device Platform XT Release Notes

provides general product information, changes in this release, usage caveats, and known problems.

Wind River OPC for IDP Programmer's Guide

provides guidance on using Wind River OPC with IDP XT.

Wind River Linux Getting Started Guide, 5.0.1

provides instructions for creating, modifying, deploying, and debugging platform and application projects using the command-line and Workbench.

Wind River Linux User's Guide, 5.0.1

provides command-line instructions for configuring, building, and developing platform projects as well as detailed information on the development environment and build system.

Wind River Workbench by Example Guide (Linux Version), 3.3

provides procedures and examples for using Workbench to configure, build, and debug Wind River Linux application, platform, and kernel module projects.

NOTE: This list represents the primary documents for developing an Intelligent Device platform target system and is not complete. For the full set of documents that come with Wind River Linux, see the *Wind River Linux User's Guide, 5.0.1* .

McAfee Embedded Control User Guide

provides an overview of McAfee Embedded Control as well as installation and configuration information and examples for getting started.

McAfee Application Control Product Guide

provides details of McAfee Application Control including installation and licensing, capabilities, and troubleshooting.

McAfee Application Control Command Line Interface Guide

provides details of McAfee Application Control commands and arguments.

McAfee Change Control Product Guide

provides details of McAfee Change Control including installation and licensing, capabilities, and troubleshooting.

McAfee Change Control Command Line Interface Guide

provides details of McAfee Change Control basic and advanced commands.

Accessing Documentation Online

You can also read Wind River Intelligent Device Platform XT documentation online on on the *Wind River Online Support Web* site. To do so, log on to Wind River Online Support and select:

Products > **Wind River Intelligent Device Platform XT** > **Manuals**.

2

Prerequisites

Before you can validate Wind River Intelligent Device Platform XT capabilities, you must first create, and then deploy a platform project.

Configuring a Platform Project for Quark Boards

Configure and create a Wind River Intelligent Device Platform XT platform project.

IDP XT supports the following Intel Quark boards:

- Clanton Hill
- Cross Hill
- Galileo

There are several differences between Quark boards and Intel boards supported for previous versions of IDP XT:

- Quark boards require that the system files in Flash on the device match the system files on the boot media.
- Quark boards do not have video output capability.
- Quark boards use a special 3.5MM audio to DB9 serial cable.
- Quark boards can boot from either a USB Flash drive or from an MMC card.

For more information, see your BSP documentation.

NOTE: Wind River recommends that you exclude the **grsecurty** feature from your project during development. **grsecurity** is automatically included; to exclude **grsecurity**, add the following option to your configure line:

--with-template=non_grsec

Intel Galileo boards do not support security features. When you configure a project for a Galileo board, add the following option:

--without-layer=wr-srm

Step 1 Set the Wind River Linux environment variables on your host machine.

```
$ <installDir>/wrenv.sh -p wrlinux-5
```

Step 2 Create a platform project directory, *projDir*.

```
$ mkdir <projDir>
$ cd <projDir>
```

Step 3 Configure the platform project.

The following configure command uses **intel-quark** as the board type:

```
$ $WIND_LINUX_CONFIGURE --enable-board=intel-quark --enable-addons=wr-idp \
--enable-kernel=standard --enable-rootfs=glibc-idp --enable-bootimage=ext3,hdd \
--enable-parallel-pkgbuilds=4 --enable-jobs=4
```

NOTE: $WIND_LINUX_CONFIGURE is an environment variable set by **wrenv.sh** for the location of the **configure** command.

Step 4 Build the project.

```
$ make fs
```

Building the project generates the following items in the ***projDir*/export/images** folder:

bzImage-initramfs-intel-quark.bin

The kernel image with **initramfs** bundled inside it. Use this image if you are using the IDP XT SRM capabilities. (Wind River recommends this image.)

bzImage-intel-quark.bin

The kernel image without **initramfs**. SRM capabilities will not work if you use this kernel image.

wrlinux-image-glibc-idp-intel-quark-dist-srm.tar.bz2

A tar file containing the root file system image with the SRM capabilities enabled by default. The image is signed with default keys, which cannot be used for production systems. (Wind River recommends this image for development.)

wrlinux-image-glibc-idp-intel-quark.tar.bz2

A tar file containing the root file system image with the SRM capabilities disabled by default. The image is not signed.

You are now ready to deploy your platform project to the device.

Building Advantech UTX-3115 Platform

Use the following procedure to configure and build the Advantech UTX-3115 platform

Building and Configuring Advantech

Configure and build a platform project for an Advantech UTX-3115 board.

IDP XT supports the Advantech UTX-3115 board.

NOTE: Wind River recommends that you exclude the grsecurty feature from your project during development. grsecurity is automatically included; to exclude grsecurity, add **--with-template=non_grsec** to your configure line.

Step 1 Set the Wind River Linux environment variables on your host machine.

```
$ <installDir>/wrenv.sh -p wrlinux-5
```

Step 2 Create a platform project directory, *projDir*.

```
$ mkdir <projDir>
$ cd <projDir>
```

Step 3 Configure the platform project.

Use the following configure command:

```
$ $WIND_LINUX_CONFIGURE --enable-board=intel-atom-baytrail --enable-addons=wr-idp \
--enable-kernel=standard --enable-rootfs=glibc-idp \
--enable-parallel-pkgbuilds=4 --enable-jobs=4
```

NOTE: $WIND_LINUX_CONFIGURE is an environment variable set by **wrenv.sh** for the location of the **configure** command.

Step 4 Build the project.

```
$ make fs
```

Building the project generates the following items in the ***projDir*/export/images** folder:

bzImage-initramfs-intel-atom-baytrail.bin

The kernel image with **initramfs** bundled inside it. Use this image if you are using the IDP XT SRM capabilities. (Wind River recommends this image.)

bzImage-intel-atom-baytrail.bin

The kernel image without **initramfs**. SRM capabilities will not work if you use this kernel image.

wrlinux-image-glibc-idp-intel-atom-baytrail-dist-srm.tar.bz2

A tar file containing the root file system image with the SRM capabilities enabled by default. The image is signed with default keys, which cannot be used for production systems. (Wind River recommends this image for development.)

wrlinux-image-glibc-idp-intel-atom-baytrail.tar.bz2

A tar file containing the root file system image with the SRM capabilities disabled by default. The image is not signed.

Step 5 Deploy the kernel image and root file system on the USB drive.

Follow the procedure appropriate to your configuration.

Deploying Quark Boards Using a vfat-Formatted USB Drive

Manually format a USB drive with vfat and tar the system files to it. Flash additional system files to the device and boot the device from the USB drive.

After configuring and building your platform project with IDP XT, you must tar the system files to a USB drive and install the bios and boot loader from the same project to Flash. The files in Flash and on the USB drive must be from the same project for the boot to succeed.

Step 1 Format the USB drive to VFAT.

Ubuntu host:

Use **fdisk** and **mkfs.vfat** to format the USB drive; mount the USB drive to **/mnt**

Windows host:

Format the USB drive to **vfat32**.

Step 2 Tar ***projDir*/export/images/intel-quark-idp-srm-bundle.tar.bz2** or *projDir*/**export/images/intel-quark-idp-bundle.tar.bz2** to the USB drive.

Step 3 Unmount the USB drive.

```
# umount /dev/sdb1
```

Step 4 Insert the USB drive in the board USB port on the target.

Step 5 Update the target's Flash firmware.

For more information, see: *Updating Flash Firmware for Quark Boards*

Step 6 Boot the target.

The boot behavior is as follows:

- When the Quark board has a USB Flash drive attached, but no MMC card, the board boots from USB and uses USB Flash drive as the root device.

- When the Quark board has an MMC card attached, but no USB Flash drive, the board boots from the MMC card and uses it as root device.
- When the Quark board has both a USB Flash drive and an MMC card connected at boot time, the default is to boot from USB and use the USB Flash drive as the root device.

NOTE: Only one USB Flash drive is supported. Do not attach more than one USB Flash drive to the target at boot time.

NOTE: Boot time for Cross Hill boards with IDP is four to six minutes.

Deploying Quark Boards Using a Script

Use the **deploy.sh** script to place the system files on the boot media. Flash additional system files to the device and boot the device.

After configuring and building your platform project with IDP XT, you must deploy the system files to the boot media and install the BIOS and boot loader from the same project to Flash. The files in Flash and on the USB drive must be from the same project for the boot to succeed.

Step 1 Deploy the file to the USB drive using **deploy.sh**.

You can use either of the following files:

- ***projDir*/export/intel-quark-glibc-idp-standard-dist-srm.tar.bz2**
- ***projDir*/export/intel-quark-idp-standard-dist.tar.bz2**

For example, using a Cross Hill board and assuming your device is **/dev/sdb**:

```
$ sudo ./deploy.sh -f \
<projDir>/export/intel-quark-glibc-idp-standard-dist-srm.tar.bz2 \
-d /dev/sdb -b cross-hill -u
```

After deployment your USB device is formatted with two partitions, a VFAT and an EXT3 file system. For a detailed list of options for **deploy.sh**, execute the script without any parameters by typing the following command:

```
$ ./deploy.sh -h
```

Step 2 Insert the USB drive in the board USB port on the target.

Step 3 Update the target's Flash firmware.

For more information, see: *Updating Flash Firmware for Quark Boards*

Step 4 Boot the target.

The boot behavior is as follows:

- When the Quark board has a USB Flash drive attached, but no MMC card, the board boots from USB and uses USB Flash drive as the root device.
- When the Quark board has an MMC card attached, but no USB Flash drive, the board boots from the MMC card and uses it as root device.

- When the Quark board has both a USB Flash drive and an MMC card connected at boot time, the default is to boot from USB and use the USB Flash drive as the root device.

NOTE: Only one USB Flash drive is supported. Do not attach more than one USB Flash drive to the target at boot time.

NOTE: Boot time for Cross Hill boards with IDP is four to six minutes.

Deploying Advantech Boards Manually

Deploy your image to a USB storage device manually. Use it to boot your board.

To deploy your image to a USB storage device manually:

Step 1 Format the USB drive if you have not already done so.

Create two partitions, one formatted with vfat and one with ext3. For more information, see *Preparing USB Boot Media*

Step 2 Mount the USB drive.

```
$ su
# mkdir /mnt/sdb_vfat
# mount /dev/sdb1 /mnt/sdb_vfat
# mkdir /mnt/sdb_ext3
# mount /dev/sdb2 /mnt/sdb_ext3
```

Step 3 Extract the rootfs tar file.

```
# tar xjvf projDir/export/images/ \
wrlinux-image-glibc-idp-intel-atom-baytrail-dist-srm.tar.bz2 –C /mnt/sdb_ext3
```

Step 4 Update the vfat partition.

```
# mkdir –p /mnt/sdb_vfat/EFI/BOOT

# cp /mnt/sdb_ext3/boot/bzImage /mnt/sdb_vfat/
# cp /mnt/sdb_ext3/boot/grub/grub.efi /mnt/sdb_vfat/EFI/BOOT/BOOTIA32.EFI
# cp /mnt/sdb_ext3/boot/grub/grub.conf /mnt/sdb_vfat/EFI/BOOT/BOOTIA32.conf
```

Modify **/mnt/sdb_vfat/EFI/BOOT/BOOTIA32.conf** as follows:

```
root (hd0, 0)
kernel    /bzImage root=/dev/sdb2 rootdelay=5
```

Step 5 Optional (if you want to boot from legacy GRUB): Modify **/mnt/sdb_ext3/boot/grub/menu.lst** as follows:

```
title       Wind River Intelligent Device Platform
root        (hd0,1)
kernel      /boot/bzImage root=/dev/sdb2 rw,noatime rootwait reboot=bios
```

Step 6 Optional (if you want to boot from legacy GRUB): Install GRUB

$ ./grub-0.97 -batch

```
Probing devices to guess BIOS drives. This may take a long time.


    GNU GRUB  version 0.97  (640K lower / 3072K upper memory)

 [ Minimal BASH-like line editing is supported.  For the first word, TAB
   lists possible command completions.  Anywhere else TAB lists the possible
   completions of a device/filename. ]
grub> device (hd0) /dev/sdb
grub> root (hd0,1)
grub> setup (hd0)
grub> quit
```

Step 7 Unmount the USB drive.

```
# umount /dev/sdb1
# umount /dev/sdb2
# exit
```

Step 8 Unplug the USB drive from your host machine and plug it into the target.

Step 9 Connect the HDMI cable to a display monitor and to the target.

Step 10 Connect the power adaptor and power on the target.

Step 11 Check the BIOS configuration (for more information see *Configuring the BIOS for UTX-3115 and other Bay Trail Boards*) and boot the target.

NOTE: Do not use the traditional Wind River Linux approach of issuing the command **make usb-image-burn** to prepare the USB image because **make usb-image-burn** uses **syslinux** as the boot loader instead of GRUB; GRUB 0.97 is required for SRM to function properly.

NOTE: The default user ID and password for Wind River targets are:

User ID: **root**
Password: **root**

Deploying Advantech UTX-3115 Boards Using a Script

Deploy your image to a USB storage device using a Wind River script. Use it to boot your board.

NOTE: The **deploy.sh** script is intended for use only with USB storage devices. The script does not support SD card deployment.

Step 1 Insert the USB storage device in the host machine.

Step 2 Execute the **deploy.sh** script from your project directory.

This command deploys the image on your USB storage device (**/dev/sdb**):

```
$ sudo ./deploy.sh -f <projDir>/export/images/<fileName>.tar.bz2 -d \
/dev/sdb –y -u
```

where ***fileName*.tar.bz2** is one of the following:

- **wrlinux-image-glibc-idp-intel-atom-baytrail-dist-srm.tar.bz2**
- **wrlinux-image-glibc-idp-intel-atom-baytrail.tar.bz2**

NOTE: Occasionally, you might need to boot from a legacy GRUB boot loader. For example, if your board has a 64-bit UEFI BIOS and you do not want to update the BIOS image to 32-bits, you have to use a legacy GRUB boot loader to boot your board.

The following command depolys the image to your USB storage device (**/dev/sdb**) for both UEFI and legacy GRUB boot operations. (Select UEFI boot or legacy GRUB boot in the Bios. For more information see *Configuring the BIOS for UTX-3115 and other Bay Trail Boards*.)

```
$ sudo ./deploy.sh -f <projDir>/export/images/<fileName>.tar.bz2 -d\
/dev/sdb -y -u -g ./grub-0.97
```

For a detailed list of options execute the script without any parameters, type the following command:

```
$ ./deploy.sh -h
```

For detailed information on using the **deploy.sh** script, see the **README** file located at:

***projDir*/layers/wr-idp/wr-idp-devkit/recipes-devtools/deploy-tool/files**

Step 3 Unplug the USB storage device from your host machine.

Step 4 Plug the USB storage device into the target and boot from it.

NOTE: The default user ID and password for Wind River targets are:

User ID: **root**
Password: **root**

3

System Configuration

Confirm that all hardware, software, and networking configuration is properly set up, that IDP XT detects hardware devices, and that the proper IDP XT processes and services are started.

Configuring Hardware

Confirm correct hardware detection.

To confirm the IDP XT has correctly detected hardware, run the **lspce** and the **lsmod** commands to show connection information for your peripheral hardware components and loaded kernel modules.

Step 1 Run **lspce**.

```
# lspci
```

You should see a listing of PCI devices similar to the system output shown below:

```
00:00.0 Host bridge: Intel Corporation Device 0958
00:14.0 SD Host controller: Intel Corporation Device 08a7 (rev 10)
00:14.1 Serial controller: Intel Corporation Device 0936 (rev 10)
00:14.2 USB controller: Intel Corporation Device 0939 (rev 10)
00:14.3 USB controller: Intel Corporation Device 0939 (rev 10)
...
```

Step 2 Run **lsmod** | **sort**.

```
# lsmod | sort
```

Sorting the output of **lsmod** shows a list of kernel modules:

```
Module                  Size  Used by
ad7298                 12765  0
af_packet              30766  0
...
```

Postrequisites

Compare your results with the listed results to ensure that the system is performing as expected.

Related Links

System Configuration on page 13
Confirm that all hardware, software, and networking configuration is properly set up, that IDP XT detects hardware devices, and that the proper IDP XT processes and services are started.

Configuring Software

Confirm the configuration of IDP XT processes and services.

Confirm that IDP processes and services have been properly configured. Run the **ps** command with **-e** and **-f** options to show the full status of all processes.

Verify the status of the system's services. Run the **service** command with the **-status-all** option.

Step 1 Run the **ps** command.

```
# ps -ef
```

You should see a list of active processes similar to the system output shown below:

```
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Jul10 ?        00:00:07 init [3]
root         2     0  0 Jul10 ?        00:00:00 [kthreadd]
root         3     2  0 Jul10 ?        00:00:11 [ksoftirqd/0]
root         4     2  0 Jul10 ?        00:00:00 [kworker/0:0]
root         5     2  0 Jul10 ?        00:00:00 [kworker/u:0]
...
```

Step 2 Run the **services** command.

```
# services --status-all
```

The **services** command generates a list of available services similar to that shown below:

```
[ ? ]  acpid
[ + ]  appweb
[ + ]  arptables
[ + ]  atd
...
```

Postrequisites

Compare your results with the listed results to ensure that the system is performing as expected.

Related Links

System Configuration on page 13
Confirm that all hardware, software, and networking configuration is properly set up, that IDP XT detects hardware devices, and that the proper IDP XT processes and services are started.

Configuring Networking

Confirm that IDP XT networking is properly configured.

Confirm that the system's networking configuration is correct. Run the **ifconfig** command with the **-a** option to show the current configuration of all network interfaces on the system.

Step 1 Run **ifconfig**.

```
# ifconfig -a
```

The **ifconfig** command generates information about all network interfaces on the system:

```
eth0      Link encap:Ethernet  HWaddr aa:bb:cc:dd:ee:ff
          inet addr:128.224.140.205  Bcast:128.224.141.255  Mask:255.255.254.0
...
```

Related Links

System Configuration on page 13
Confirm that all hardware, software, and networking configuration is properly set up, that IDP XT detects hardware devices, and that the proper IDP XT processes and services are started.

4

Security

Confirm that IDP XT security features are operational.

Security is a primary design consideration for connected devices within an IoT system. Intelligent Device Platform includes features to implement a robust security plan for a wide array of IoT systems. Security features include access control of critical system resources and digital signature validation for trusted software. This section guides you through the steps needed to validate the security features of IDP XT.

Performing a Secure Boot on Quark

Performing a secure boot involves configuring and building a platform project, burning the flash image into the flash on the board, deploying the signed images to the board, and booting the board.

This example was developed on a board that was running in *secure open* mode.

Step 1 Configure and build a platform project.

For more information, see *Building Platform Projects for Quark Boards*.

Use the following configure command:

```
$ $WIND_LINUX_CONFIGURE --enable-board=intel-quark --enable-addons=wr-idp \
--enable-kernel=standard --enable-rootfs=glibc-idp \
--with-template=feature/secure-boot \
--enable-parallel-pkgbuilds=4 --enable-jobs=4
```

Step 2 Build the project.

```
$ make fs
```

Step 3 Confirm that the flash image and the SRM signed rootfs tar file were generated.

Step 4 Update the target's Flash firmware.

For more information, see: *Updating Flash Firmware for Quark Boards*

Step 5 Deploy the SRM signed rootfs on the USB boot disk.

Follow the instructions for deploying the image and rootfs and rebooting the board in:

- *Deploying Quark Boards Using a vfat-Formatted USB Drive*
- *Deploying Quark Boards Using a Script*

Related Links

Security on page 17
Confirm that IDP XT security features are operational.

Updating Flash Firmware for Quark Boards

Performing a Secure Boot on Advantech UTX-3115 Using UEFI

The Intel Baytrail device provides a secure boot policy that allows only the signed bootloader (**grub.efi**) to run on the UEFI.

Step 1 Configure and build a platform project.

Follow the configure and build steps in *Building Advantech UTX-3115 Boards*. Use the following configure command:

```
$ $WIND_LINUX_CONFIGURE --enable-board=intel-atom-baytrail --enable-addons=wr-idp \
--enable-kernel=standard --enable-rootfs=glibc-idp \
--enable-parallel-pkgbuilds=4 --enable-jobs=4
```

Step 2 Sign the rootfs file with SST.

For more information, see *Signing the rootfs File*

Step 3 Burn AMI BIOS **A115X013X32.bin** onto an Intel Advantech UTX-3115 board using the SF100 DediProg device.

For more information, see: *Updating BIOS Images on Advantech UTX-3115 Boards Using an SF-100 Programmer*

Step 4 Deploy the signed rootfs on U-Disk.

For more information see:

- *Deploying Advantech UTX-3115 Boards Using a Script*
- *Deploying Advantech UTX-3115 Boards Manually*

Step 5 Insert the U-Disk into the Advantech UTX-3115 board.

Step 6 Configure the BIOS and reboot the board.

Select the following menu items:

a) **Advanced** > **CSM Configuration** > **Video** > **UEFI only** > **F4** > **Reboot**
b) **Advanced** > **CSM Configuration** > **CSM Support** > **[Disabled]** > **F4** > **Reboot**
c) **Security** > **Secure Boot menu** > **Secure Boot** > **[Disabled]**
d) **Security** > **Secure Boot menu** > **Secure Boot** > **Key Management** > **Delete All**
e) **Secure Boot Variables** > **YES** > **F4** > **Reboot**

Step 7 Enter the EFI shell console.

Press **F7** and select **UEFI: Built-in EFI shell**.

Step 8 Enter the U-Disk VFAT partition.

Select **fs0: enter U-Disk VFAT partition**.

Step 9 Run the signed **grub.efi** file **BOOTIA32.efi**.

```
# cd EFI\BOOT
# ./grub.efi
Platform is in Setup Mode
KEK LEN: 1068
Created KEK Cert
DB LEN: 2727
Created db Cert
PK LEN: 1086
```

Step 10 Configure the BIOS again and reboot the board.

a) Press **F7** and select **Enter Setup**.
b) **Security** > **Secure Boot menu** > **Secure Boot** > **[Enable]** > **F4** > **Reboot**

Step 11 Enter the U-Disk VFAT partition.

Select **fs0: enter U-Disk VFAT partition**.

Step 12 Run the signed **grub.efi** file **BOOTIA32.efi**.

The **BOOTIA32.efi** boots the kernel successfully.

Step 13 Confirm that the secure boot policy is working correctly on the board.

a) Copy an unsigned **grub.efi** file to **/EFI/BOOT** in the VFAT partition on the U-Disk.
b) Insert the U-Disk into the Advantech UTX-3115 board.
c) Press **F7** and select **UEFI: Built-in EFI shell**.
d) Run the unsigned **grub.efi** file **UNSIGNED_BOOTIA32.efi**.

The following message appears:

```
fs0:\EFI\BOOT> UNSIGNED_BOOTIA32.efi
Error reported: Access Denied
```

This is the correct result when the secure boot policy is working correctly.

Related Links

Security on page 17

Confirm that IDP XT security features are operational.

Using the Tamper-Proof File System

The tamper-proof file system is part of the Integrity Measurement Architecture (IMA); including this capability on an embedded device provides device owners with strict control over the software deployed on the device.

The purpose of application integrity measurement is to assure that the run results of text-based scripts can be trusted when the system invokes them with a *controlled approach*.

NOTE: For compiled executable files and text-based plain scripts, the tamper proof capability always prevents them from running if they cannot provide a verified signature. Text-based plain scripts are bash, perl, or python scripts that are invoked from an absolute or relative path.

Examples of controlled invocation:

```
$ ./certain-script.sh
$ /root/certain-perl-script.pl
```

However, when these scripts are executed directly by the interpreter, the tamper proof capability does not prevent them from running; running from the interpreter is not a *controlled approach*.

Examples of uncontrolled invocation:

```
$ bash ./certain-script.sh
$ perl /root/certain-perl-script.pl
```

By default, when you enable the tamper-proof file system using **--enable-addons=wr-idp** and boot the device, the tamper-proof file system capability is included by default. To activate the tamper-proof file system, use the tar file named ***-dist-srm.tar.bz2**, for example:

***projDir*/export/images/wrlinux-image-glibc-idp-intel-quark-dist-srm.tar.bz2**

Step 1 Build a platform project and boot your board in the standard way.

For more information see:

- *Building Platform Projects for Quark Boards*
- *Building Advantech UTX-3115 Boards*

The following configure command uses the intel-quark BSP as an example:

```
$ $WIND_LINUX_CONFIGURE --enable-rootfs=glibc-idp --enable-addons=wr-idp \
--enable-kernel=standard --enable-board=intel-quark
```

Building this configuration creates two tar files:

wrlinux-image-glibc-idp-intel-quark.tar.bz2

This image contains all the SRM capabilities, but they are not enabled by default.

wrlinux-image-glibc-idp-intel-quark-dist-srm.tar.bz2

This image contains all the SRM capabilities. They are enabled by default with default keys. Use this image to demonstrate SRM capabilities. You can find the default keys at:

***projDir*/layers/wr-idp/wr-srm/files/keys/owner-cert.pem**
***projDir*/layers/wr-idp/wr-srm/files/keys/owner-private.pem**
***projDir*/layers/wr-idp/wr-srm/files/keys/vendor-cert.pem**
***projDir*/layers/wr-idp/wr-srm/files/keys/vendor-private.pem**

NOTE: Use the SST commands to sign either set of images with custom keys. For more information, see *Key Management for Vendors*.

Step 2 Verify that an unauthorized application cannot run on the system.

```
# ls
examples
# cp `which ls` ./ls-copied
# ls
examples  ls-copied
# ./ls-copied
-sh: ./ls-copied: Permission denied
# echo $?
126
```

The exit status value **126** indicates that the command did not run successfully. In this case, the RSA signature for **ls-copied** was not found on the system.

Step 3 Modify a script or executable and verify that it will not run.

This example makes arbitrary modifications to **imtools** and then tries to run it.

```
# imtools -h
Usage:imtool      --verifycert  <CA Certificate>
                  --listcert
                  --removecert  <CA Certificate>
                  --verifyrpm   <RPM Package>
# vi /usr/bin/imtools
<Make some modifications to the script, for example, by changing some help text, and
save the file>

# imtools -h
-sh: /usr/bin/imtools: /bin/sh: bad interpreter: Permission denied
```

The script cannot run because the RSA signature for **im-tools** does not match the one stored on the system.

Step 4 Verify that an authorized application can run successfully.

The **ls** command has an RSA signature and has not been modified.

```
# ls
examples  ls-copied
# echo $?
0
```

The exit status value **0** indicates that the command ran successfully. In this case, the RSA signature for **ls** matched the one stored on the system.

Step 5 (Optional) If you need to disable the tamper-proof file system, build the project with the **--without-layer=wr-srm** option in configure command.

```
$ $WIND_LINUX_CONFIGURE --enable-rootfs=glibc-idp --enable-addons=wr-idp \
--without-layer=wr-srm --enable-kernel=standard --enable-board=intel-quark
```

Related Links

Security on page 17
Confirm that IDP XT security features are operational.

Installing McAfee Embedded Control

In order to use McAfee Embedded Control with IDP XT, you must include it in your Wind River Linux platform project.

- Build a platform project and boot your board in the standard way.

  You must include at least the following option and template in your configure command:

  Option: **--enable-addons=wr-idp**
  Layer: **--with-layer=wr-mcafee**

  For more information see:

  - *Building Platform Projects for Quark Boards*
  - *Building Advantech UTX-3115 Boards*
  - *McAfee Embedded Control Users Guide*

  The following example uses the intel-quark BSP:

```
$ $WIND_LINUX_CONFIGURE --enable-board=intel-quark --enable-kernel=standard \
--enable-rootfs=glibc-idp --enable-addons=wr-idp --with-layer=wr-mcafee \
--enable-parallel-pkgbuilds=4 --enable-jobs=4
```

- Run the **sadmin status** command.

```
# sadmin status
McAfee Solidifier:             Disabled
McAfee Solidifier on reboot:     Disabled
ePO Managed:          No
Local CLI access:     Recovered
[fstype] [status] [driver status] [volume]
* ext3 Unsolidified Unattached /
```

Related Links

Security on page 17
Confirm that IDP XT security features are operational.

Verifying Grsecurity

Confirm that Grsecurity is operational.

Grsecurity provides an added layer of security to your project.

Create a new directory and change directory to that new directory. For example, create a directory named **~/WindRiver/workspace/my-grsec** .

Step 1 Run the configure command from the host.

```
$ /home/wruser/WindRiver/wrlinux-5/wrlinux/configure \
--enable-board=intel-quark|atom-baytrail \
--enable-kernel=standard \
--enable-rootfs=glibc-idp \
--enable-addons=wr-idp \
--enable-jobs=4 \
--enable-parallel-pkgbuilds=4
```

Step 2 Run the **make** command.

```
# make
```

Create and deploy USB stick as instructed in the *Deployment* section. Boot the target from the newly created USB device.

Step 3 Run the **gradm** utility program.

```
# gradm -S
The RBAC is currently disabled
```

The result shown above indicates that Grsecurity is functioning.

Related Links

Security on page 17
Confirm that IDP XT security features are operational.

5

Connectivity

Run commands to test IDP connectivity.

Connectivity options include both local connectivity and IoT system or cloud connectivity.

This section guides you through the steps needed to validate the connectivity features of IDP XT.

MQTT Server

Confirm that the mosquito server is operational.

```
# export PATH=$PATH:/root/examples/mqtt-client/
# mqtt_test.lua -d localhost
```

The client will subscribe to **test/2** and start publishing.

Open new window on the host, start new ssh session to the device.

```
# export PATH=$PATH:/root/examples/mqtt-client/
# mqtt_publish.lua -d -t test/2 -m "quit"
```

The client in first window will receive quit message on test/2 and will exit.

6

Manageability - Web Server

Confirm that the **Webif** management interface is operational.

Device management requires facilities for updating and configuring the gateway along with providing device status. This section guides you through the steps needed to validate the management features of IDP XT.

Running Webif Procedure

Use the following steps to configure Webif:

Step 1 Modify the configure command to include the Webif component.

Use the following configure command:

```
$ WIND_LINUX_CONFIGURE --enable-board=intel-quark --enable-kernel=standard \
--enable-rootfs=glibc-idp --enable-addons=wr-idp \
--enable-jobs=4 --enable-parallel-pkgbuilds=4
```

It is unnecessary to include **feature/webif** in the configure line to include in the Webif component; Webif is included automatically when you use **--enable-rootfs=glibc-idp**.

Complete the remaining steps for secure boot to confirm the boot is working correctly.

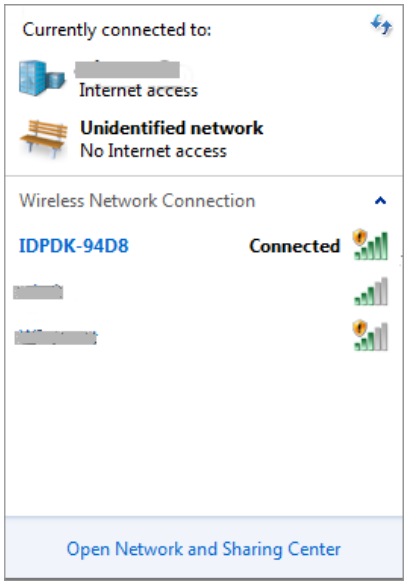
Optionally, you can perform a secure or verified boot on the IDP XT target using a modified configure command.

For board-specific information, see:

Performing a Secure Boot on Cross Hill and Clanton Hill Boards
Performing a Secure Boot on Advantech UTX-3115 Using UEFI
Performing a Verified Boot

NOTE: Wind River recommends that you perform a secure boot of your IDP XT target before using Webif to configure your IDP device. The instructions that follow assume that you will be performing a secure boot.

Step 2 Configure your host/target connection.

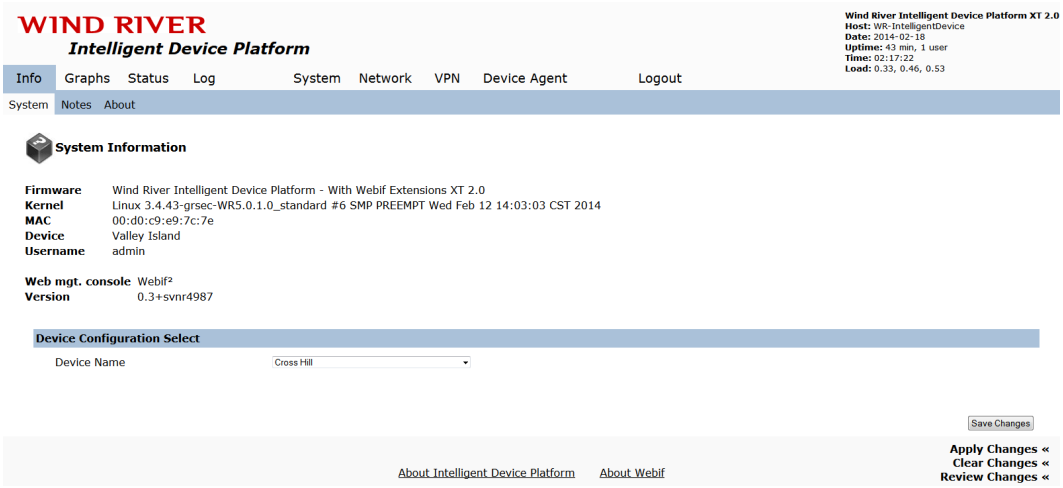
Connection Type	Procedure
If your IDP XT target has a WiFi module:	On your host machine, search for the Wireless network (**IDPDK-XXXX**) started by the IDP target and connect to it. The default password is **windriveridp** and is stored in the **/etc/config/wireless** file on the IDP XT target. 
If your IDP XT target does not have a WiFi module:	Connect the IDP XT target to your host machine using an Ethernet cable. Find out the IP address assigned to **eth0** of the IDP XT target. Both systems should be able to ping to each other.

Step 3 Start the Webif interface in a Web browser on your host.

Connection Type	Procedure
If your IDP XT target has a WiFi module:	Type **https://192.168.1.1** and press **ENTER**.
If your IDP XT target does not have a WiFi module:	Type **https://*ipAddrOfEth0*** and press **ENTER**.

Step 4 At the prompt, enter **admin** for both the username and password, then click **OK**.

The Webif interface displays the System Information page.



Step 5 To change the default configuration setting, see *Saving Changes in the Webif Interface*.

Step 6 To add a new Webif page in the interface, see *Adding a New Webif Page*.

You can also start, create, and join a ZigBee network from the Webif interface. For more information, see:

- *Starting a Zigbee Network*

Related Links

Manageability - Web Server on page 27
Confirm that the **Webif** management interface is operational.

7

Application Development

LUA Example 32

The following tests indicate that IDP is functioning as expected for application development activities.

The gateway is designed to support popular application environments for IoT software. This section will guide you through the steps needed to validate the management features of IDP XT.

Open JDK

Confirm that the Java VM is operational by running the following command:

```
# java -cacao -version
java version "1.6.0_27"   IcedTea Runtime Environment (IcedTea6 1.12.5) (6b27-1.12.5)
CACAO (build 1.6.0+r68fe50ac34ec,compiled mode)
```

SQLite

Confirm that SQLite DBMS is operational.

```
# sqlite3
 SQLite version 3.7.10 2012-01-16
        13:28:40   Enter ".help" for
        instructions   Enter SQL statements terminated
        with a ";"
```

Now that you have verified that SQLite is working you can quit the program by running the following command:

```
sqlite> .quit
```

Python Interpreter

Confirm that Python interpreter is operational

```
# python
  Python 2.7.2 (default, May  7 2014, 06:33:20)   [GCC 4.6.3] on linux2   Type "help",
"copyright",
        "credits" or "license" for more information.
>>>import sys
>>> print (sys.version)
```

Use the following command to escape from the Python shell:

```
>>> Ctrl+D
```

LUA Example

Confirm that Lua interpreter is operational.

Use the following steps for Lua testing:

Step 1 Transfer the **.lua** files to your IDP XT target.

```
$ scp *.lua root@<IP-Address-of-IDP-Target>
```

Step 2 Execute the Lua example files.

The following examples show the commands and the system output for the various example files.

```
root@WR-IntelligentDevice:~# lua hello.lua
Hello world, from Lua 5.1!
root@WR-IntelligentDevice:~# luac hello.lua
root@WR-IntelligentDevice:~# lua luac.out
Hello world, from Lua 5.1!
root@WR-IntelligentDevice:~# lua echo.lua WindRiver IDP 2.1
0       echo.lua
1       WindRiver
2       IDP
3       2.1
root@WR-IntelligentDevice:~# lua sort.lua
original
        Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec
after quicksort
        Apr,Aug,Dec,Feb,Jan,Jul,Jun,Mar,May,Nov,Oct,Sep
after reverse selection sort
        Sep,Oct,Nov,May,Mar,Jun,Jul,Jan,Feb,Dec,Aug,Apr
after quicksort again
        Apr,Aug,Dec,Feb,Jan,Jul,Jun,Mar,May,Nov,Oct,Sep
root@WR-IntelligentDevice:~# lua life.lua
----------------------------------------
----------------------------------------
--OO--------------------------O---------
-OO-------------------------O-O---------
---O-------------------------O---------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
---O------------------------------------
--O-O-----------------------------------
--O-O-----------------------------------
---O------------------------------------
----------------------------------------
-------OO-------------------------------
-------OO-------------------------------
----------------------------------------
Life - generation 2000
```

Related Links

Application Development on page 31
The following tests indicate that IDP is functioning as expected for application development activities.