Getting Started

Control Plane-Platform Development Kit 2.11

March 2004

Contents

Figures

Tables

Revision History

Revision	Description	Date	Author
2.11	Updated for Release 2.11	March 2004	Udaya Shankara/Amit Kaul
2.1	Updated for Release 2.1	December 2003	Udaya Shankara/Amit Kaul
2.0	Updated for Release 2.0	August 2003	Shelesh Bansal

Part 1: Introduction

1 Introduction

The purpose of the Getting Started document is to help set up a development environment through which you can write, build, and test the Control Plane sample application on top of CP-PDK. This document describes the steps needed to setup a system in different configurations supported by CP-PDK . Current release of CP-PDK supports VxWorks® or Red Hat 7.3 Linux® in Control Plane and VxWorks® or Montavista Linux® in the Forwarding Plane . CP-PDK is tested only for Intel® IXDP2400 Advanced Development Platform; hence the platform setup details are specific to this platform.

The document describes the system requirements, directory structure, and provides a brief description of the Conformance Test Framework, which is included with the 2.1 release of the CP-PDK.

Before reading this document, it is suggested to refer to IXA SDK Software Framework Documentation especially "Getting Started Guide". This document contains the details of underlying core-component and microblock infrastructure and steps to build and run these underlying components for IXDP2400 Advanced Development Platform.

1.1 System Requirements

You must have the following components in order to complete the system setup:

1. A Windows® system, with 2 NIC cards. One network card is dedicated to communicating with the Intel® IXDP2400 Advanced Development Platform from the Windows Host System. Other NIC card can be used to connect to the corporate network for any downloads or getting any VxWorks Tornado License. If you don't require any license or downloads from your corporate network, only one NIC card should be enough. If you want to do the remote CP-PDK testing, use two serial ports.
2. Tornado 2.2
3. An Intel® IXDP2400 Advanced Development Platform.
4. If you want to test a remoted Control Plane for the CP-PDK, you need the following:
   - One system with an Intel® Pentium® processor
   - A crossover cat5 cable, or a hub and three straight-through cat5 cables if you are doing the remoted version.
5. One serial cable
6. The Board Specific Package (BSP) for the Intel® IXDP2400 Advanced Development Platform.
7. Winzip or something similar.
8. www.cygwin.com/setup.exe
9. Python 2.2.
10. Start from a clean system. That is, a system with no previous installation of SDK/PDK. Remove any previous installation of the SDK/PDK from the machine on which the new installation is to be done.

1.2 CP-PDK Documentation

The ***CP-PDK Overview Page*** included in the Software Framework documentation directory lists all the documents included with the CP-PDK and provides a link to each document.

NPF-ratified APIs are being implemented. For ratified API specifications, please refer to the Network Processing Forum (NPF) at www.npforum.org.

1.3 Directory Structure

Throughout this document "<IX_SDK_DEV>" is referred as the directory where SDK is installed. CP-PDK is located in <IX_SDK_DEV>/src/cp_pdk/ directory.

The top level of the CP-PDK consists of the Control Plane, Forwarding Plane, Apps, Conformance Test and FP Module core/proxy component directories. The following sections list the files included in each of these top-level directories:

Table 1. CP-PDK Modules

Sections	Description
Control Plane	This directory contains the implementation of the Control Plane. The header files are found in src/cp_pdk/npf_api/
ForwardingPlane	This directory contains the standalone-forwarding plane binary.
FPTransportPlugin	This directory contains the Transport Plug-in implementation for the Forwarding Plane. It contains the files needed to link a test harness with the forwarding plane transport plug-in. The header files for this module are available in src/cp_pdk/inc/
CPTransportPlugin	This directory contains the Control Plane Transport Plug-in implementation. It contains the files needed to link a test harness with the Control Plane Transport Plug-in. The header files for this module are available in src/cp_pdk/inc/
Transport Plug-in Code Generator	This utility can be used to generate the encapsulation and de-capsulation code required for any data types that need to be sent between the Control Plane and the Forwarding Plane through the Transport Plug-in src/cp_pdk/TPCodeGen
Pil	This directory contains the PIL library, which is required by the control and forwarding plane binaries. PIL library is compiled and dynamically linked. This header files for this module are available in: src/cp_pdk/common/pil/inc/
ConformanceTester (cppui)	The Control Plane Platform User interface (cppui) tool is test environment developed to use with the CPPDK. This is a simple graphical user interface, which runs with Python2.2. This can also be used in the text mode to run generated test scripts. The user interface is completely generated and input to the generator is XML notation of the test cases. This tool is explained in **Conformance Test Framework - Control Plane Platform User Interface (CPPUI) Guide**

FPModule Core Components

There are two directories that contain the FP Module ingress and egress core components. These run with the rest of the IXA SDK components. The two directories are:

Table 2. CP-PDK Core Components

Directory	Description
fpmodule_core	This contains the FP Module ingress core component. It interacts with the rest of the IXA SDK core components and it must be compiled, when the CP-PDK is to be used with the IXA SDK.
fpmodule_proxy	This contains the FP Module egress **proxy** core component, designed for interfacing with the IXA SDK core components running on the egress side of the IXDP2400 platform.

1.4 Including the CP-PDK into the IXA SDK Build

CPPDK is integrated with following SDK pipelines:

1. 4Gb Ethernet IPv4 Forwarding Application **(src/applications/ipv4_forwarder/4gb_ethernet)**
2. OC-48 POS-IPV4 Forwarding Application - DiffServ for POS Application. **(src/applications/ipv4_diffserv/oc48_pos)**
3. ATM over OC12 Pipeline **(src/applications/ipv4_diffserv/4oc12_atm)**

Use **IXA SDK** BuildingBlocksAppsDesignGuide for more information on these pipelines.

1.4.1 VxWorks

Adding FP Module ingress core component into the IXA SDK

To add the FPModule ingress component into the SDK workspace:

1. Edit the **ix_sa_registry_data.xml** file in ingress directory for respective pipeline. For example, edit ***<IXA_SDK_DEV>/src/applications/ipv4_forwarder/4gb_ethernet/ingress/oc48_ethernet_ingress_config/ix_sa_registry_data.xml*** for **ipv4 4gb** pipeline.
2. Find the following section in the file:

```
<property name="SA_EXEC_ENGINES">
   <property name="SA_EE_01" type="uint32" value="1">
     <property name="CC_LIST" type="string" value=~IX_CC_STKDRV
IX_CC_IPV4 IX
_CC_CSIX_TX IX_CC_QM IX_CC_SCHEDULER IX_CC_ETH_RX~/>
      </property>
     </property>
```

3. Add **IX_CC_FPM** to the value field of the **CC_LIST** property.
4. Save the file.

Adding FP Module Egress Proxy core component into the IXA SDK

To add the FPModule egress proxy core component into the SDK workspace:

1. Edit the **ix_sa_registry_data.xml** file in egress directory for respective pipeline. For example, edit ***<IXA_SDK_DEV>/src/applications/ipv4_forwarder/4gb_ethernet/egress/oc48_ethernet_egress_config/ix_sa_registry_data.xml*** for 4GB Ethernet ipforwarder pipeline.

2. Find the following section in the file:

```
<property name="SA_EXEC_ENGINES">
   <property name="SA_EE_01" type="uint32" value="1">
     <property name="CC_LIST" type="string" value=~IX_CC_QM
IX_CC_SCHEDULER IX
_CC_ETH_TX IX_CC_CSIX_RX~/>
      </property>
     </property>
```

3. Add **IX_CC_FPM_PROXY** to the value field of the **CC_LIST** property.
4. Save the file.

Enabling remoted stacks (CP-PDK on a remoted Control Plane)

If you are running the control plane remotely (that is, not on XScale) then the **REMOTED_STACK_FLAG** needs to be set to 1.

Note: The **REMOTED_STACK_FLAG** must be enabled only on the Ingress side.

5. Edit the **ix_sa_registry_data.xml** file in the ingress directory for the respective pipeline. For example, edit ***<IXA_SDK_DEV>/src/applications/ipv4_forwarder/4gb_ethernet/ingress/oc48_ethernet_egress_config/ix_sa_registry_data.xml*** for 4GB Ethernet ipforwarder pipeline.
6. Find the following line in the file:

```
<property name="REMOTED_STACK_FLAG" type="uint32" value="0"/>
```

7. Change the value of the **REMOTED_STACK_FLAG** to **1**.
8. Save the file.

Compiling the FP Module ingress Core component with the IXA SDK

1. Open the workspace **<IXA_SDK_DEV>\src\workspace\ixa_sdk_2.2.wsp**.
2. Go to the Builds tab of the IXA SDK 2.2 workspace.
3. Expand the ingress build for chosen pipeline. For example, open **A_oc48_ethernet_ingress Build** for 4GB Ethernet build and select the build target.
4. Right-click and choose Properties.
5. Choose the Macros Tab:
    1. Create a new macro. In the Name dialog, type **PDK_PROJS.**
    2. In the Value field, type in **$(PDK_SUB_PROJS).**
    3. Click Add/Set, and click on the Apply, and OK buttons.
    4. In the drop-down for Macros, select **PDK_SUB_PROJS.**
        (1) Replace the value field with **src/cp_pdk/fpmodule_core**. The value field for **PDK_SUB_PROJS** should only show **src/cp_pdk/fpmodule_core** as the value.
        (2) For ATM Pipeline, add another line as **src/library/xscale/properties**
        (3) Click Add/Set, and click on the Apply, and OK buttons.
6. In the macros tab, select **GLOB_DEFINES** and

1. Add -**DINCLUDE_PDK**.
2. Add the following flags depending on your pipeline:
   (1) **-DIX_CC_ETH** for Ethernet based-pipeline.
   (2) **-DIX_CC_ATM** for ATM based-pipeline.
   (3) **-DIX_CC_QOS** for Diffserv services.

   For example, add both **IX_CC_ETH** and **IX_CC_QOS,** if the pipeline is Diffserv over Ethernet.

7. Save and reload the IXA SDK workspace. This does not add the project to the workspace but builds it into the image.

Open the **A_oc48_ethernet_ingress** Build & rebuild All. This builds the executable with the FPM ingress CC linked in.

Compiling the FP Module Egress Proxy Core component with the IXA SDK

1. Open the workspace **<IXA_SDK_DEV>\src\workspace\ixa_sdk_2.2.wsp**.
2. Go to the Builds tab of the IXA SDK 2.2 workspace.
3. Expand the egress build for chosen pipeline. For example, open **A_oc48_ethernet_egress Build** for 4GB Ethernet build and select the build target.
4. Right-click and choose Properties.
5. Choose the Macros Tab.
   1. Create a new macro. In the Name dialog, type **PDK_PROJS**.
   2. In the Value field, type **$(PDK_SUB_PROJS)**.
   3. Select Add/Set, and click on the Apply, and OK buttons.
   4. In the drop down for Macros, select **PDK_SUB_PROJS**.
      (1) Replace the value field with **src/cp_pdk/fpmodule_proxy**. The value field for the **PDK_SUB_PROJS** should show only **src/cp_pdk/fpmodule_proxy** as the value.
      (2) Select Add/Set, and click on the Apply, and OK buttons.
6. In the macros tab select **GLOB_DEFINES**
   1. Add -**DINCLUDE_PDK**.
   2. Add following flags depending on your pipeline.
      (1) **-DIX_CC_ETH** for Ethernet based pipeline
      (2) **-DIX_CC_ATM** for ATM based pipeline
      (3) **-DIX_CC_QOS** for Diffserv Services

      For example, add both **IX_CC_ETH** and **IX_CC_QOS**, if the pipeline is **Diffserv over Ethernet**.
7. Save and Reload the IXA SDK workspace. This does not add the project to the workspace but causes it to be built into the image.

Open the **A_oc48_ethernet_egress** Build & rebuild All. This builds the executable with the FPM egress CC linked in.

1.4.2 Linux

Linux in Forwarding Plane is supported only for 4Gb-IPv4-Forwarder Application Pipeline.

Adding FP Module ingress core component into the IXA SDK

To add the FPModule ingress component into the SDK workspace:

1. Edit the **ingress_internal_registry_data_linux.h** file in ingress, at the following location.

   <IXA_SDK_DEV>/src/applications/ipv4_forwarder/4gb_ethernet/ingress/oc48_ethernet_ingress_config/include/sa/internal for ipv4 4gb pipeline.

2. Find the following section in the file:

```
    {"/SystemApp/SA_EXEC_ENGINES", "SA_EE_01", 1, 1, 1, ""},
    /* List all CC IDs defined in bindings.h for being included in
this application (should be using the defined values) */
    {"/SystemApp/SA_EXEC_ENGINES/SA_EE_01", "CC_LIST", 1, 2, 0, "1
0 5 6 7 9 16"},
    {"/SystemApp", "MICROENGINES", 0, 0, 0, ""},
```

3. Add the Core Component ID for FPM core to the CC_LIST as shown above. The ID by default is 16 and can be found in the following file.

   <IXA_SDK_DEV>/src/applications/ipv4_forwarder/4gb_ethernet/include/bindings.h

   Tip : This can also be found in this file by searching for IX_CC_FPM string

4. Save the file.

Adding FP Module Egress Proxy core component into the IXA SDK

To add the FPModule egress proxy core component into the SDK workspace:

1. Edit the egress_internal_registry_data_linux.h file in ingress, at the following location.

   <IXA_SDK_DEV>/src/applications/ipv4_forwarder/4gb_ethernet/egress/oc48_ethernet_egress_config/include/sa/internal for ipv4 4gb pipeline.

2. Find the following section in the file:

```
   {"/SystemApp/SA_EXEC_ENGINES", "SA_EE_01", 1, 1, 1, ""},
    /* List all CC IDs defined in bindings.h for being included
in this application (should be using the defined values) */
    {"/SystemApp/SA_EXEC_ENGINES/SA_EE_01", "CC_LIST", 1, 2, 0,
"6  7  8  4 19"},
    {"/SystemApp", "MICROENGINES", 0, 0, 0, ""},
```

3. Add the Core Component ID for FPM Proxy to the CC_LIST as shown above. The ID by default is 19 and can be found in the following file.

   <IXA_SDK_DEV>/src/applications/ipv4_forwarder/4gb_ethernet/include/bindings.h

   Tip : This can also be found in this file by searching for IX_CC_FPM_PROXY string

4. Save the file.

Enabling remoted stacks (CP-PDK on a remoted Control Plane)

If you are running the control plane remotely (that is, not on XScale) then the **REMOTED_STACK_FLAG** needs to be set to 1.

Note: The **REMOTED_STACK_FLAG** must be enabled only on the Ingress side.

1. Edit the **ingress_internal_registry_data_linux.h** file in the ingress directory for the respective pipeline. For example, edit ***<IXA_SDK_DEV>/src/applications/ipv4_forwarder/4gb_ethernet/ingress/oc48_ethernet_ingress_config/include/sa/internal/ingress_internal_registry_data_linux.h*** for 4GB Ethernet ipforwarder pipeline.
2. Find the following line in the file:

   ```
   {"/System_Properties", "REMOTE_STACK_FLAG", 1, 1, 0, "1"},
   ```

3. Change the value of the **REMOTED_STACK_FLAG** to **1** as shown above.
4. Save the file.

Compiling the FP Module ingress Core component with the IXA SDK

1. Open the following make file:

   <IXA_SDK_DEV>/src/applications/ipv4_forwarder/4gb_ethernet/ingress/Makefile.linux_kernel

   Add the following directories to CC_DIRS:

   src/cp_pdk/fpmodule_core

   src/cp_pdk/ForwardingPlane/FPModule/Shim

2. Open the following make file:

   <IXA_SDK_DEV>/src/applications/ipv4_forwarder/4gb_ethernet/ingress/oc48_ethernet_ingress_config/Makefile.linux_kernel

   Add INCLUDE_PDK to BUILD_TYPE_FLAGS

3. Save the file.
4. Compile ingress by using the following command at *<IXA_SDK_DEV>/src/applications/ipv4_forwarder/4gb_ethernet/ingress*

   make –f Makefile.linux_kernel

Compiling the FP Module Egress Proxy Core component with the IXA SDK

1. Open the following make file:

   <IXA_SDK_DEV>/src/applications/ipv4_forwarder/4gb_ethernet/egress/Makefile.linux_kernel

   Add the following directories to CC_DIRS:

   src/cp_pdk/fpmodule_proxy

2. Open the following make file:

<IXA_SDK_DEV>/src/applications/ipv4_forwarder/4gb_ethernet/egress/oc48_ethernet_egress_config/Makefile.linux_kernel

Add INCLUDE_PDK to BUILD_TYPE_FLAGS

3. Save the file.
4. Compile egress by using the following command at *<IXA_SDK_DEV>/src/applications/ipv4_forwarder/4gb_ethernet/egress*

make –f Makefile.linux_kernel

Part 2: Setup and Installation

2 Setup and Installation

CP-PDK is tested with Intel® IXDP2400 Advanced Development Platform hardware.

Following are the various configurations in which CP-PDK can be run:

CP-PDK can run in two configurations:

- **Remote :** Control Plane and Forwarding Plane parts of CP-PDK reside in two different processors.
- **Colocated:** Control Plane and Forwarding Plane parts of CP-PDK are compiled together so both run in IXDP2400 XScale.
  - CP-PDK 2.0 in IXA SDK 3.1: VxWorks in Forwarding Plane (IXDP2400) only
  - CP-PDK 2.11 in IXA SDK 3.51: VxWorks and Montavista Linux in Forwarding Plane (IXDP2400).

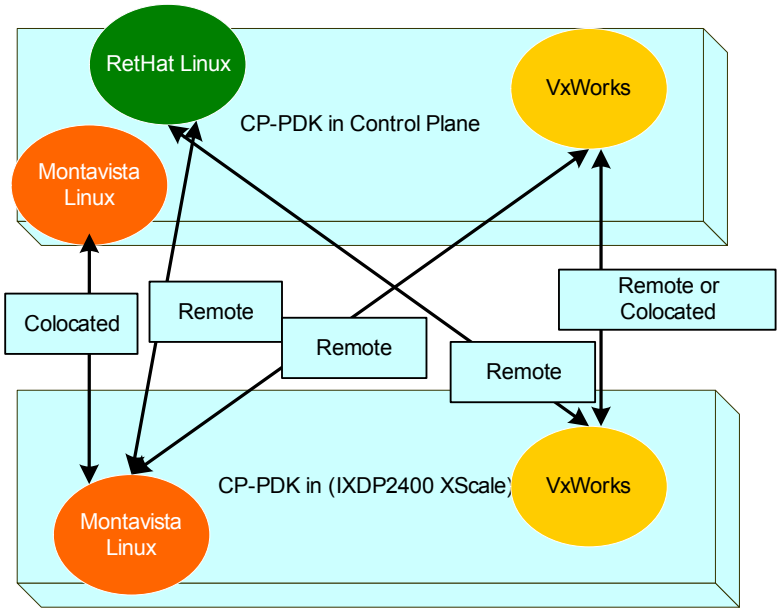

Figure 1. CP-PDK in different configurations

The following sections describe how to set up hardware, install, and execute the components of the CP-PDK and all its required tools.

2.1 Configuring an IXDP2400 System with VxWorks for CP-PDK

2.1.1 Get your Windows system up and running, possibly installing the OS.

2.1.2 Install Tornado 2.2

1. Though it is not necessary to uninstall any older versions of the Tornado, ensure that the Tornado 2.2 is installed in a different directory.
2. There are two stages to this install, which are explained below: Both need CD keys, which can be obtained from the Tornado installation kit.
    1. Install:
        - `Tornado 2.2/VxWorks 5.5 for SA/XScale`
        - Accept default values for all options on all pages of the install wizard except for the following:
            (1) On installation of the files, the **License Management Configuration** page is displayed.
            (2) Select **Configure to use a floating license** and click Next.
            (3) On the next page you are asked if the setup should contact Wind River.
            (4) Select No. Do not contact the Wind River web site. Click Next.
            (5) Click Exit.
    2. Install BSPs/Drivers for VxWorks 5.5 – SA/XScale by selecting all default values.
3. If you are performing a remoted VxWorks Control Plane testing (running control plane on a separate IA based VxWoks system), you must also install the Pentium version of Tornado. It consists of two CDs:
    (1) Tornado 2.2/VxWorks 5.5-IAx86
    (2) BSPsDrivers for VxWrks 5.5-IAx86

Follow the same installation procedure as the XScale versions but install in a different directory. The Tornado installer warning ensures this.

2.1.3 Install the BSP for the Intel® IXDP2400 Advanced Development Platform

1. Create a directory in your Tornado directory:
   `C:\Tornado2_2\target\config\ixdp2400_be`
2. Unzip the Intel® IXDP2400 Advanced Development Platform BSP into your new directory.

2.1.4 Install Cygwin

1. This step is optional, and is necessary only if you want to build the CP-PDK from the command line.

2. Run the setup.exe from www.cygwin.com.
3. Execute the installation from the Internet using Internet Explorer* settings for proxies.
4. Pick defaults (You can install it anywhere depending on your choice).
5. The Cygwin presents a screen for selecting programs to be installed.

   You must pick at least:

   make

   You probably require:

   A text editor such as vi or emacs.

2.1.5 Install Python

1. The **CPPUI GUI** requires Python 2.2 .
2. You can download the python from www.python.org
3. Add the Python.exe path in the PATH environment variable
   1. Right click on the **My computer** icon.
   2. Select Properties->Advanced
   3. Click on the Environment Variable.
   4. Select the PATH from the System variable.
   5. Add the python.exe path there.

2.1.6 Start Tornado

A **Create Project in New/Existing Workspace** dialog pops up. Select the **New** tab, which is the default, and select the **Create a bootable VxWorks image** list item. Then click the OK button.

1. Step 1 dialog:
   1. Set Name: to the required name of your choice.
   2. Set 'Location:' to …\proj\<project name>
   3. Enter the project description, which is optional.
   4. Add a new workspace in **…\proj\<workspace>.wsp** Click the Next button.
2. Step 2 dialog:
   1. Select **ABSP** radio button.
   2. Select **ixdp2400_be** from the BSP drop down list. If you are doing the remoted IA VxWorks Control Plane testing on the Pentium machine, then you must repeat all the steps mentioned above, except the following:
      a. Starting of the Pentium version of Tornado
      b. Selection of the correct Pentium BSP instead of the **ixdp2400_be**.
   3. Select gnube from the tool drop down list, in case of XScale based Tornado. Select gnu from the tool drop down list, in case of IA based Tornado.
   4. Click the Next button.

3. Step 3 dialog:
    1. Click the Finish button.
4. Step 4: Once the workspace window comes up, you must add a few pieces to the boot image. There is an error in the config, which causes two conflicting options to be turned on every time you change any other option. Ignore the errors till the completion of the next step.
    1. Click the VxWorks tab and select the following components. (To **include** the components, select the item, right click and, select '**include'** entry in the list):

       operating system components/IO system Component/DOSFS2 file system components.

       Select both 'CBIO Disk Partition Handler' and 'File System Backup and Archival'.

       Click the OK button.

       operating system components/POSIX components/POSIX scheduler

       operating system components/POSIX components/POSIX timers

       network components/networking protocols/network applications/DNS resolver
    2. **Exclude** the following components to resolve the error. (To exclude the components, select the item, right click and, select '**exclude'** entry in the list):

       network components/ basic network initialization components/bootline processing components/dhcp device address initialization/DHCP client timestamp removal

       network components/ basic network initialization components/bootline processing components/dhcp device address initialization/DHCP client timestamp setup
5. Step 5: You can edit the **config.h** in the IXDP2400 BSP (C:Tornado2_2\target\config\ixdp2400_be\config.h) to have VxWorks boot with your own default values.
    1. Change:

       ```
       #define IXDP2400_BAUD_RATE   57600
       ```

       To:

       ```
       #define IXDP2400_BAUD_RATE   38400
       ```

       (only if the baud rate in step 8.9 is 38400)
    2. Change:

       ```
       #define MASTER_BOOT_LINE "fei(0,0)host:VxWorks " \
       "h=10.3.31.225 e=10.3.31.33:FFFFFF00 u=target pw=target
       tn=ixdp2400E"
       ```

       To:

       ```
       #define MASTER_BOOT_LINE "fei(0,0)host:VxWorks " \
       "h=192.168.1.1 e=192.168.1.2 u=angel pw=angel
       tn=ixdp2400E"
       ```
6. Step 6: Select Build from the tool bar, and Rebuild all to rebuild the BSP.

   Now you need to create a new bootrom for the IXDP2400. The one on the IXDP2400 is for VxWorks 5.4, and it does not work with the 5.5 bootable image created using the above.
    1. Click on the Build/Build Boot ROM.

2. Select **ixdp2400_be** as the BSP.
3. Select bootrom.bin as the image to build.
4. Select gnube as the tool.

If you are performing a remoted VxWorks Control Plane testing on the Pentium machine, then you must repeat all the above steps, except the following:

i. Start of the Pentium version of Tornado

ii. Selection of the correct Pentium BSP instead of the **ixdp2400_be**.

2.1.7 Set up the IP address for the Ethernet card connected to the IXDP2400

7. Go to Start->Settings->Network and Dial-up connections.
8. Double-click Local Area connections.
9. Go to Properties-> Internet Protocol (TCP/IP) -> Properties -> <IP Address>.
10. Input the IP Address, for example, 192.168.1.1.

2.1.8 Set up the FTP server

1. Start Tornado's FTP server.
2. Click Security/Users.
3. Click new user.
    1. Enter a name and password for the new user. The examples in the steps explained below use **angel** for both the user name and password.
    2. Set the user's home directory to:

    C:\Tornado2_2\target\proj\<YOUR PROJECT>\default

2.1.9 Set up your Hardware

1. Connect to the Egress of the IXDP2400.
2. If doing the remoted version:
    1. Connect the IXDP2400 and your Pentium test system (IA VxWorks system on which the Control Plane is to be run) to the hub.
    2. Connect the adapter for which (of the Windows machine on which the Tornado is installed) you set the address earlier in step 6 of this section to the hub.
3. Otherwise:
    1. Connect the IXDP2400 to your Windows Development system with a crossover cable.
    2. On AI, connect your Egress and Ingress's Management Ethernet port and the host machine to a hub.
4. Connect the serial cable between the IXDP2400 and your Windows system.
5. Run Tornado's VxWorks Com shortcut, com1 or 2 depending on the one you have connected.

6. Turn on your IXDP2400.
7. It boots into redboot

Note: If there are garbage characters coming out of your VxWorks Hyper terminal, it indicates that your setting is incorrect. Disconnect the connection. Navigate to File→Properties, click 'Configure' button and set the bits per second to 38400 or 57600 and the flow control to 'none'. You can re-connect by clicking the call button on the tool bar.)

8. You may need to upgrade the Redboot image in order to support the new VxWorks bootrom.
    1. At the redboot prompt type:
    2. `Redboot> load –r –v –b 0x400000 filename –m y`
    3. Simultaneously, on the HyperTerminal menu, click on Transfer/Send File…
    4. Browse for the redboot.bin.
    5. Select the Ymodem protocol.
    6. Click Send.
    7. Write the new image to the flash file system.

```
Redboot> fis write -f 0xc4000000 -b 0x400000 -l 0x40000
Redboot> fis init
```

    8. Choose yes to overwrite the question.
    9. You can do the same thing on the slave/ingress side.
    10. Reset the system.
9. Upgrade the VxWorks bootrom
    1. At the redboot prompt type:

```
lo -v -b 0x200000 -r -m y bootrom
```

    2. Start sending the file from the HyperTerminal prompt, using the ymodem protocol again. The file you require should be:
       C:\Tornado2_2\target\config\ixdp2400_be\bootrom.bin.
    3. On completion of the download, enter the following:

```
fis create -b 0x200000 -l 0x100000 -f 0xc4040000 bootrom
```

    4. Choose Yes to all overwrite prompts.
10. To force it to boot into VxWorks type:

```
Redboot> go 0xc4048000
```

11. Stop VxWorks from auto booting by pressing space at the countdown prompt.
12. Configure to VxWorks to switch off your machine.

```
[VxWorks Boot]: c

'.' = clear field;  '-' = go to previous field;  ^D = quit

boot device          : fei0
processor number     : 0
host name            : host
file name            : VxWorks
inet on ethernet (e) : 192.168.1.2
```

```
inet on backplane (b):
host inet (h)          : 192.168.1.1
gateway inet (g)       :
user (u)               : angel
ftp password (pw) (blank = use rsh): angel
flags (f)              : 0x0
target name (tn)       : ixdp2400E
startup script (s)     :
other (o)              :
```

13. To load the kernel and reboot, type: `@`.
14. If the kernel is booted successfully, you can see the following messages in the console:

```
Attached TCP/IP interface to fei unit 0
Attaching interface lo0...done
Loading symbol table from host:VxWorks.sym ...done


 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
      ]]]]]]]]]]]  ]]]]     ]]]]]]]]]]       ]]              ]]]]         (R)
 ]     ]]]]]]]]]  ]]]]]]     ]]]]]]]]       ]]              ]]]]
 ]]     ]]]]]]]  ]]]]]]]]     ]]]]]] ]     ]]              ]]]]
 ]]]     ]]]]] ]    ]]]  ]     ]]]] ]]]   ]]]]]]]]]  ]]]] ]] ]]]]  ]]   ]]]]]
 ]]]]     ]]]  ]]    ]  ]]]     ]] ]]]]] ]]]]]]   ]] ]]]]]]] ]]]] ]]   ]]]]
 ]]]]]     ]  ]]]]     ]]]]]      ]]]]]]]] ]]]]   ]] ]]]]    ]]]]]]]    ]]]]
 ]]]]]]      ]]]]]     ]]]]]]    ]  ]]]]]  ]]]]   ]] ]]]]    ]]]]]]]]    ]]]]
 ]]]]]]]    ]]]]]  ]    ]]]]]]  ]    ]]]   ]]]]   ]] ]]]]    ]]]] ]]]]    ]]]]
 ]]]]]]]]  ]]]]]  ]]]    ]]]]]]]      ]     ]]]]]]]  ]]]]    ]]]]  ]]]] ]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]       Development System
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]       VxWorks version 5.5
 ]]]]]]]]]]]]]]]]]]]]]]]]]]       KERNEL: WIND version 2.6
 ]]]]]]]]]]]]]]]]]]]]]]]]]       Copyright Wind River Systems, Inc., 1984-2002

                               CPU: Intel IXDP 2400 BE (Master/Egress).
     Process
or #0.
                              Memory Size: 0x2000000.  BSP version 1.2.5b.
                             WDB Comm Type: WDB_COMM_END
                            WDB: Ready.

->
```

2.1.10 Configure a target server

You need to configure for the following two target servers:

- ingress
- egress.

1. Go back to the Tornado, click on the tools→target server→configure.
2. Click the New button.
3. Find the Target Server Name field and give a name for it.

4. Set the target IP address to be used. For example, 192.168.1.2 can be used as the egress management port IP address and 192.168.1.3 for the ingress management port respectively.
5. Set up various target server properties, such as:
    1. Under the Core File and Symbols, choose file, and then browse to the following boot image created earlier.
       C:\Tornado2_2\target\proj\<YOUR PROJECT>\default\VxWorks
    2. Under the Memory Cache Size, specify 10MB or more.
    3. Under the Console and Redirection, select the check box Redirect Target IO.
6. Click the Launch button.
7. You should get a little target in your systray without an exclamation mark. If there is an exclamation mark in it, then you must shut down the target server and check whether the steps explained above are followed correctly.
8. Choose your server in the pull down menu on the tool bar of the Tornado IDE.

   Click the launch shell button. If the shell does not come up, know that the above steps are followed correctly.

The steps mentioned above should be repeated twice to configure the target server for egress and ingress respectively.

2.2 Configuring CP-PDK Build Environment

This section explains the steps needed to setup the build environment of CP-PDK in respective operating systems VxWorks and Montavista Linux. Build Environment for CP-PDK is needed with respect to the Operating System being used in either Control Plane or Forwarding Plane. RedHat Linux Build environment involves a straightforward **RedHat 7.3** installation and hence it is not separately discussed in this section.

Important Note : CP-PDK must be compiled only after Egress and Ingress Core Components are compiled. This is because CP-PDK uses some header files that are available in *<IXA_SDK_DEV>/build/include. T*his directory is available only after the Core Components are built. Since CP-PDK runs on Ingress processor of IXDP2400, hence compilation sequence should be ***Egress Core Component → Ingress Core Component → CP-PDK***. This will enable relevant Ingress header files to be available for CP-PDK during compilation.

2.2.1 Configuring an IA32 VxWorks Build Environment

Follow this section VxWorks setup.

2.2.1.1 Create IA based VxWorks boot image

1. Install IA based Tornado. If the XScale version Tornado is already installed, then install the IA version in a separate directory. For example, if the XSCALE version is installed in c:\Tornado2.2 directory then create c:\Tornado2.2-IA for IA based Tornado. Follow the above-mentioned instructions.
2. Follow the instructions explained in Section 2.1.6 - Step 6 to create VxWorks boot image. In this type of configuration, there are a total of three target servers, which are mentioned below:
    1. ingress

2. egress
3. target server for IA VxWorks system (on which the control plane is running).

3. Change the following parameter:

   Network components/basic network initialization components/network buffer initialization. Then right-click and select Properties. Click on the params tab, select IP_MAX_UNITS and change its value to 20.

4. Edit the config.h to make the following changes:

   For example, if the boot image is built for pcpentium then goto Tornado2.2-IA\target\config\PcPentium\, then edit config.h and make the following changes in config.h file.

   1. Search for INCLUDE_PC_CONSOLE and, change #undef INCLUDE_PC_CONSOLE to #define INCLUDE_PC_CONSOLE
   2. Search for DEFAULT_BOOT_LINE
   3. In this case go to pcpentium and, enter the following:

      `#define DEFAULT_BOOT_LINE`

      `Change` **`fd0=0,0(0,0)host:/fd0/VxWorks.st h=90.0.0.3 e=90.0.0.50 u=target`** `to`

      `fei=0,0(0,0)host:VxWorks "h=”IP address of the Tornado-IA host” e=”IP address of the remoted IA boot machine” u=angel"`

   4. Save the config.h file

5. Rebuild all
6. Click on the Build, and then click on the Build Boot Rom….
7. Select the correct type of Pentium in the Select aBSP
8. Select bootrom.bin from the Select an Image to build option.
9. Select ‘gnu’ from ‘Select a tool’ option and, click Select→Key ok to build the bootrom.bin image.

2.2.1.2 Making the bootable disk

1. Go to `C:\Tornado2.2-IAhost\x86_win32\bin`.
2. Run **torVars.bat** to set the environment variables.
3. Go to C:\Tornado2.2-IA\target\config\PCPentium directory to create the bootable floppy. Insert formatted blank floppy and run the command `mkboot a: bootrom.bin`
4. Use this floppy to boot up an IA32 system with VxWorks
5. Press any key first, and then press the key **c** to modify the boot parameters:

```
boot device             : feis
unit number             : 0
Processor number        : 0
host name               : host
file name               : VxWorks
inet on ethernet (e)    : 192.168.1.2 “IP address of the IA
VxWorks machine on which control plane is to be run”
```

```
host inet (h)        : 192.168.1.1 “IP address of the IA
Tornado machine”
user (u)             : x86
ftp password (pw) (blank = use rsh): x86
flags (f)            : 0x0
```

2.2.2 Configuring Monta Vista Linux Build Environment

The Monta Vista Linux is installed on a host machine. The host machine must have RedHat Linux 7.3 installed. The compilation is done on the host machine. This directory is then mounted on the Master and the Slave. Follow this section for Linux setup.

2.2.2.1 Installing Monta Vista Linux

1. Install RedHat 7.3
2. Install Monta Vista Host CD (host-mvl3.0.0.iso)
3. Install Monta Vista Target CD (arm_xscale_be-mvl3.0.0.iso)
4. Install patch (ixp2400_arm_xscale_be-sb030227.img)
5. Before doing any application build, do the following:

   cd /opt/hardhat/devkit/lsp/intel-ixdp2400/linux-2.4.18_mvl30

   PATH=$PATH:/opt/hardhat/devkit/arm/xscale_be/bin

   make ixdp2400_config; make oldconfig; make dep

6. Export the following environment variables. If ixa_sdk_3.51 is installed in a different path, make the corresponding changes while setting IXA_SDK_DEV.:

```
    export IXA_SDK_DEV=/root/ixa_sdk_3.5
    export PATH=$PATH:/opt/hardhat/devkit/arm/xscale_be/bin
    export
IXP2XXX_TOOLCHAIN_ROOT=/opt/hardhat/devkit/arm/xscale_be

    export
IXP2400_TOOLCHAIN_ROOT=/opt/hardhat/devkit/arm/xscale_be
    export
IXP2400_KERNEL_SOURCE_ROOT=/opt/hardhat/devkit/lsp/intel-
ixdp2400-
            arm_xscale_be/linux-2.4.18_mvl30
    export CONFIG=XSCALE_BE
    export IXOS=LINUX
    export BUILD_TYPE=DEBUG
    export BUILD_MODE=HARDWARE
     export HARDWARE_TYPE=IXP2400
```

7. Install DHCP server package (dhcp-2.0p15-8.i386.rpm for Red Hat 7.3) on the Linux host if it is not already installed. To see if the package is already installed, issue the following command:

   rpm –qa | grep dhcp

   If the above mentioned package is missing, then using Red Hat 7.3 CD#2, reinstall it as:

rpm –I /mnt/cdrom/redhat/RPMS/dhcp-2.0p15-8.i386.rpm

8. Install tftp server package (tftp-server-0.28-2.i386.rpm) on the Linux host if it is not already installed. To see if the package is already installed, issue the following command:

   rpm –qa | grep tftp

   If the above mentioned package is missing, reinstall it as:

   rpm –I /mnt/cdrom/redhat/RPMS/tftp-server-0.28-2.i386.rpm

   If there is no /tftpboot directory create it by issuing the following command:

   mkdir /tftpboot

9. To export the filesystem, edit /etc/exports. Add the following lines:

   /opt/hardhat/devkit/arm/xscale_be/target *(rw, no_root_squash,no_all_squash)

   /opt/xscale_be_test/linux_kernel/xscale_be *(rw,no_root_squash, no_all_squash)

10. Restart the NFS by issuing the following command:

    /etc/rc.d/init.d/nfs restart

11. The exported directories can be verified by issuing the following command:

    exportfs

12. Copy the kernel image (zImage) to /tftpboot directory
13. Enable the tftp protocol by editing the file /etc/xinetd.dtftp. Change the line from disable=yes to disable=no
14. Restart the xinetd daemon by issuing the following command:

    /etc/rc.d/init.d/xinetd restart

15. Make sure that the Host PC address and the addresses for the Master and Slave Ethernet interfaces are on the same subnet.
16. Set the MAC addresses of the Master and Slave in /etc/dhcpd.conf file on the host as follows:

```
subnet 10.3.31.0 netmask 255.255.255.0 {
  }
subnet 10.10.10.0 netmask 255.255.255.0 {
  host master {
    hardware Ethernet 00:90:d7:00:11:1f;
    fixed-address 10.10.10.1;
    option root-path “/opt/hardhat/devkit/arm/xscale_be/target”;
  }
  host slave {
    hardware Ethernet 00:90:d7:00:11:20;
```

```
        fixed-address 10.10.10.2;

        option root-path "/opt/hardhat/devkit/arm/xscale_be/target";

    }

}
```

17. Start inetd by issuing the following command:
    `/etc/rc.d/init.d/xinetd restart`

2.3 Running CP-PDK Forwarding Plane (FP) with VxWorks

This section discusses the steps needed to run CP-PDK Forwarding Plane in VxWorks with different configurations for Control Plane.

2.3.1 Download and run the SDK

1. Select the correct target server
2. Download the SDK ingress and egress executable files to the slave and master respectively
3. Open the egress and ingress shells and start SDK as per the SDK Getting Started Guide. Use bladeId as 1 for running both Egress and Ingress binaries.

2.3.2 Setting Tornado Environment Variables

We need to setup few VxWorks Tornado environment variables in cygwin shell before compiling the images.

Step 1 : You need the IXROOT variable set.

export IXROOT=<IXA_SDK_DEV>

Step 2: Edit the **env_tor.sh** file present in <IXA_SDK_DEV>/src/cp_pdk to make any change in the path where Tornado is installed. By default, the path is given as:

export WIND_BASE=c:/Tornado2.2

Step 3: Every time you start a new cygwin shell, you need to set those variables in the order to enable compilation of CP-PDK. Hence to set the variable, enter the following:

source env_tor.sh

2.3.3 CO Located SETUP – VxWorks in CP

1. Configure the CP-PDK.

```
$ cd < IXA_SDK_DEV>/src/cp_pdk/
$ make config
./configure.sh
What OS is hosting this build? [] CYGWIN
Are the CP and FP to be colocated (YES/NO)? [] YES
```

```
What is the target OS? [] VXWORKS
What is the target architecture? [] IXP
Should stubs be used for IXASDK? [] NO
```

2. Build the CP-PDK by giving the following command:
    1. `$ cd cppui/build`
    2. `$ make`
3. Download the PIL.
    1. Go back to the Tornado IDE.
    2. Select the correct target server of the IXDP2400 in the ingress shell.
    3. Select Project → Download and browse to the following: <IXA_SDK_DEV>\src\cp_pdk\common\pil\lib\ixp_VxWorks\libpil.out.
4. Download the conformance tester (CPPUI).
    1. Click on download, browse to <IXA_SDK_DEV>\src\cp_pdk\cppui\build\cppui
5. Run the CPPDK.
    1. Run the Control Plane/Forwarding Plane
        (1) Go back to the shell that you had launched earlier. In this case, go to ingress shell
        (2) Run the CPPDK by entering cppui.
    2. Run GUI in windows machine
        (1) Start the cygwin Shell
        (2) Start the GUI by entering
        Python <IXA_SDK_DEV>\src\cp_pdk\cppui\ui\cppui.py
        (3) Connect the GUI with the backend

        Press the connect button on the GUI and type the correct IP address of the Control Plane machine.

2.3.4 Remote Setup - RedHat LINUX in CP

3. Configure the CP-PDK.

    $ cd <IXA_SDK_DEV>/src/cp_pdk

    $ make config

    ./configure.sh

    What OS is hosting this build? [] CYGWIN

    Are the CP and FP to be co-located (YES/NO)? [] NO

    What is the target OS of the Contol Plane? []VXWORKS

    What is the target architecture of the Control Plane? []IXP

    What is the target OS of the Forwarding Plane? [] VXWORKS

What is the target architecture of the Forwarding Plane? [] IXP

Should stubs be used for IXASDK? [] NO

4. Build the CP-PDK for the Forwarding Plane by giving the following commands:

$ cd ForwardingPlane

$ make

This generates the ForwardingPlane.out file in <IXA_SDK_DEV>/src/cp_pdk/ForwardingPlane/bin/ixp_VxWorks directory and libpil.out file in <IXA_SDK_DEV>/src/cp_pdk/common/pil/lib/ixp_VxWorks directory.

6. Install the Control Plane of the CP-PDK.

1. Copy /ftp the CPPDK file onto your Linux system from the <IXA_SDK_DEV>/src/cp_pdk directory.
2. It is assumed that the cp_pdk is copied in //home/pdk//pdksrc
3. Configure the CP-PDK.

```
$ cd /home/pdksrc/cp_pdk
$ make config
./configure.sh
What OS is hosting this build? [] CYGWIN
Are the CP and FP to be colocated (YES/NO)? [] NO
What is the target OS of the Contol Plane? [] LINUX
What is the target architecture of the Contol Plane? [] IA
What is the target OS of the Forwarding Plane? []LINUX
What is the target architecture of the Forwarding Plane?
[]IA
Should stubs be used for IXASDK? [] NO
```

4. Build the CP-PDK for the Control Plane by entering the following command:

$ cd cppui/build

$ make

This generates the cppui executable file in the /home/pdk//pdksrc/cp_pdk/cppui/build/ directory. In addition to this, it also creates the shared library for the PIL (libpil.so and libpil.a files). Set the environment variable **LD_LIBRARY_PATH** to /home/pdk/pdksrc/cp_pdk/common/pil/lib/ia_linux.

7. Build the kernel module

If the control plane is to be run on Linux, then the following steps should be followed for building and inserting the kernel module into Linux kernel:

1. $ cd /home/pdk/pdksrc/cp_pdk/ControlPlane/CoreBlocks/PacketHandler/src
2. $ make

This builds the kernel module vip.o (virtual IP module).

Insert the kernel module into Linux kernel using:

$ Insmod vip.o

8. **Download the Forwarding Plane Executable to the Intel® IXDP2400 Advanced Development Platform.**
    1. Download PIL.
        (1) Go back to the Tornado IDE.
        (2) Select the correct Target Server of the IXDP2400 in the ingress shell.
        (3) Select Project → Download and, browse to <IXA_SDK_DEV>\src\cp_pdk\common\pil\lib\ixp_VxWorks\libpil.out.
    2. Download the forwarding plane.
        (1) a. Click on download and, browse to <IXA_SDK_DEV>\src\cp_pdk\ ForwardingPlane\bin\ixp_VxWorks\ForwardingPlane.out
9. **You are now ready to run the Conformance Test Framework on the Linux system.**
    1. Run the Control Plane on the Linux system by entering 'cppui'
    2. Run the GUI
        (1) Start the cygwin Shell
        (2) Start the GUI by entering the following command: **Python <IXA_SDK_DEV>\src\cp_pdk\cppui\ui\cppui.py**
        (3) Connect the GUI with the backend

        Press the connect button on the GUI and type the correct IP address of Control Plane Linux machine.
    3. Start the Forwarding Plane
        (1) Spawn a target Shell on Ingress Target.
        (2) Run the Forwarding Plane by entering

        ```
        fp_main("<feid>", "Ipaddr>","1")
        ```

        Wherein feid is the Ingress BladeId, and Ipaddr is the IPaddress of Control Plane machine.

2.3.5 Remote Setup : VxWorks in CP

1. Setup tornado variables
    1. Edit env_tor.sh file to edit the environment variable WIND_BASE

        export WIND_BASE=c:/Tornado2.2_XS (Path for IA Tornado)
    2. Source env_tor.sh

2. Install the CP-PDK
    1. Configure the CP-PDK.

```
$ cd <IXA_SDK_DEV>/src/cp_pdk
$ make config
./configure.sh
What OS is hosting this build? [] CYGWIN
Are the CP and FP to be colocated (YES/NO)? [] NO
What is the target OS of the Contol Plane? [] VXWORKS
What is the target architecture of the Contol Plane? [] IA
What is the target OS of the Forwarding Plane? [] VXWORKS
What is the target architecture of the Forwarding Plane?
[] IA
Should stubs be used for IXASDK? [] NO
```

2. Build the CP-PDK by entering the following command:

   $ cd cppui/build

   $ make

3. Download the Forwarding Plane Executable to the Intel® IXDP2400 Advanced Development Platform.

1. Download the PIL.
   (1) Go back to the Tornado IDE.
   (2) Select the correct Target Server of the IXDP2400 in the ingress shell.
   (3) Select Project → Download and, browse to <IXA_SDK_DEV>\src\cp_pdk\common\pil\lib\ixp_VxWorks\libpil.out.
2. Download the forwarding plane.
   (1) Click on Download and, browse to <IXA_SDK_DEV>\src\cp_pdk\ ForwardingPlane\bin\ixp_VxWorks\ForwardingPlane.out

4. Download the Control Plane Executable to the Intel® IA32 Platform.

1. Create a target server to the IA32 target machine.
2. Spawn a target Shell.
3. Download the PIL.
   (1) Go back to the Tornado IDE.
   (2) Select the correct target server of the IXDP2400 in the ingress shell.
   (3) Select Project → Download and, browse to <IXA_SDK_DEV>\src\cp_pdk\common\pil\lib\ixp_VxWorks \libpil.out.
4. Download the conformance tester (cppui).

   Click on Download and, browse to <IXA_SDK_DEV>\src\cp_pdk\cppui\build\cppui.

5. You are now ready to run the CPPDK.

1. Run the Control Plane on the IA32
   (1) Spawn a target Shell on IA32.
   (2) Run the Control Plane by entering cppui
2. Run the GUI

(1) Start the cygwin Shell On Tornado Host Machine

(2) Start the GUI by entering

```
Python <IXA_SDK_DEV>\src\cp_pdk\cppui\ui\cppui.py
```

(3) Connect the GUI with the backend.

Press the connect button on the GUI and type the correct IP address of Control Plane VxWorks machine.

3. Start the Forwarding Plane

(1) Spawn a target Shell on Ingress Target.

(2) Run the Forwarding Plane by entering

```
fp_main("<feid>", "Ipaddr>","1")
```

Wherein feid is the Ingress BladeId, and Ipaddr is the IPaddress of Control Plane machine.

2.4 Running CP-PDK Forwarding Plane (FP) with Montavista Linux

This section discusses the steps needed to run CP-PDK Forwarding Plane in Montavista Linux with other Control-Plane configurations.

2.4.1 Run the SDK

1. Issue the following command on egress:

   load –r –b 0x1c208000 zImage

   go 0x1c208000

   Linux must boot up and display the Linux login prompt. Login as “root” without any password.

2. Repeat the above step for Ingress too.
3. Mount the required directory on the target by issuing the following command:

   Mount –o vers=2 10.10.10.10:/opt/xscale_be_test/linux_kernel/xscale_be /mnt

4. Create the required devices at the egress by issuing the following commands:

   mknod –m 766 /dev/SaUtil c 254 0

   mknod -m 766 /dev/L2Config c 252 0

   The first one is needed for the system application and the second one is needed for running the L2 config utility.

5. Create the required devices at the ingress by issuing the following commands:

```
mknod –m 766 /dev/SaUtil c 254 0
mknod –m 766 /dev/RConfig c 253 0
```

The first one is needed for the system application and the second one is needed for running the Route config utility.

6. Insert kernel modules at the egress by issuing the following command:

   /mnt/startegress.sh

   Run SDK at the egress by issuing the following command:

   /mnt/sa start 1

7. Insert the kernel modules at the ingress by issuing the following command:

   /mnt/startingress.sh

   Run the SDK at the ingress by issuing the following command:

   /mnt/sa start 1

2.4.2 CO Located Setup : Montavista Linux in CP

1. Configure the CP-PDK.

```
$ cd < IXA_SDK_DEV>/src/cp_pdk/
$ make config
./configure.sh
What OS is hosting this build? [] MV_LINUX
Are the CP and FP to be colocated (YES/NO)? [] YES
What is the target OS? [] MV_LINUX
What is the target architecture? [] IXP
Should stubs be used for the Forwarding Plane? [] NO
Should stubs be used for IXASDK? [] NO
```

   Set the IXROOT variable. This can be done by adding the following line to the .bashrc file.

   export IXROOT=<IXA_SDK_DEV>

2. Build the CP-PDK by giving the following command:

```
$ cd cppui/build
$ make
```

3. Edit the startegress.sh file as follows:

   After the line “insmod Ethernet_tx_cc.o” add the following line:

   insmod fpm_proxy_cc.o

4. Edit the stopegress.sh file as follows:

   At the beginning of the file, add the following line:

   rmmod fpm_proxy_cc

5. Edit the startingress.sh file as follows:

   After the line “insmod stkdrv_cc.o” add the following line:

insmod fpm_core_cc.o

After the line “insmod mac_config_util.o” add the following line:

insmod shim_module.o

6. Edit the stopingress.sh file as follows:

   At the beginning of the file, add the following line:

   rmmod shim_module

   After the line “rmmod oc48_ethernet_ingress_config” add the following line:

   rmmod fpm_core_cc

7. Create the required device at the ingress by issuing the following commands:

```
mknod –m 766 /dev/Shim c 230 0
```

8. Follow steps mentioned in Section 2.4.1
9. Run the CPPDK.
   1. Run the Control Plane/Forwarding Plane
      (1) Telnet to the ingress through an xterm
      (2) Run the CPPDK by entering cppui.
   2. Run GUI
      (1) Start the cygwin Shell on the Windows machine
      (2) Start the GUI by entering
      Python <IXA_SDK_DEV>\src\cp_pdk\cppui\ui\cppui.py
      (3) Connect the GUI with the backend

      Press the connect button on the GUI and type the correct IP address of the Control Plane (ingress) machine.

2.4.3 Remote Setup : RedHat Linux in CP

1. Follow step 1 in Section 2.4.2
2. Build the CP-PDK for the Forwarding Plane by giving the following commands:

   $ cd ForwardingPlane

   $ make

   This generates the ForwardingPlane file in
   <IXA_SDK_DEV>/src/cp_pdk/ForwardingPlane/bin/ixp_linux directory
3. Go to the following directory:

   <IXA_SDK_DEV>/src/cp_pdk/ForwardingPlane/FPModule/PacketHandlerManager/src and issue the following command:

   make –f Makefile.linux_kernel

The above command will generate pkthndlr_linux_kernel_module.o. Copy this file to the following directory:

/opt/xscale_be_test/linux_kernel/xscale_be/ixp2400/debug

4. Edit the startegress.sh file as follows:

   After the line “insmod Ethernet_tx_cc.o” add the following line:

   insmod fpm_proxy_cc.o

5. Edit the stopegress.sh file as follows:

   At the beginning of the file, add the following line:

   rmmod fpm_proxy_cc

6. Edit the startingress.sh file as follows:

   After the line “insmod stkdrv_cc.o” add the following line:

   insmod fpm_core_cc.o

   After the line “insmod mac_config_util.o” add the following line:

   insmod pkthndlr_linux_kernel_module.o

   insmod shim_module.o

7. Edit the stopingress.sh file as follows:

   At the beginning of the file, add the following line:

   rmmod shim_module

   rmmod pkthndlr_linux_kernel_module

   After the line “rmmod oc48_ethernet_ingress_config” add the following line:

   rmmod fpm_core_cc

8. Create the required device at the ingress by issuing the following commands:
   ```
   mknod –m 766 /dev/Shim c 230 0
   ```
9. Follow the steps mentioned in Section 2.4.1
10. Install the control plane of the CPPDK as explained in Section 2.3.4.
11. Run the Control Plane on the Linux system by entering ‘cppui’
12. Run the GUI on a Windows machine
    1. Start the cygwin Shell
    2. Start the GUI by entering the following command:
       Python <IXA_SDK_DEV>\src\cp_pdk\cppui\ui\cppui.py
    3. Connect the GUI with the backend

       Press the connect button on the GUI and type the correct IP address of Control Plane Linux machine.

13. Open an xterm and telnet to the ingress. Start the Forwarding Plane by entering the following:
    ```
    ForwardingPlane <feid>, “Ipaddr>”,1
    ```

Wherein feid is the Ingress BladeId, and Ipaddr is the IPaddress of Control Plane machine.

2.4.4 Remote Setup: VxWorks in CP

1. Set up the control plane of the CPPDK as explained in Remote Setup : VxWorks in CP.
2. Configure the CP-PDK as follows for the Forwarding Plane.

```
$ cd < IXA_SDK_DEV>/src/cp_pdk/
$ make config
./configure.sh

What OS is hosting this build? [] MV_LINUX
Are the CP and FP to be colocated (YES/NO)? [] NO
What is the target OS of the Contol Plane? [] MV_LINUX
What is the target architecture of the Contol Plane? [] IXP
What is the target OS of the Forwarding Plane? [] MV_LINUX
What is the target architecture of the Forwarding Plane? [] IXP
Should stubs be used for IXASDK? [] NO
```

   Set the IXROOT variable. This can be done by adding the following line to the .bashrc file.

   export IXROOT=<IXA_SDK_DEV>

3. Follow steps from 2 to 9 in Remote Setup - RedHat LINUX in CP.
4. Open an xterm and telnet to the ingress. Start the Forwarding Plane by entering the following:

```
ForwardingPlane <feid>, "Ipaddr>",1
```

   Wherein feid is the Ingress BladeId, and Ipaddr is the IPaddress of Control Plane machine.

Part 3: Conformance Test Framework

3 Conformance Test Framework

CPPDK uses CPPUI as a conformance test framework graphical tool. This chapter provides an overview of the CPPUI tool and its interaction with the CP-PDK.

The CPPUI tool consists of the GUI (user interface) and the backend (CPPUI Server and CPPDK) as shown in the following figure.

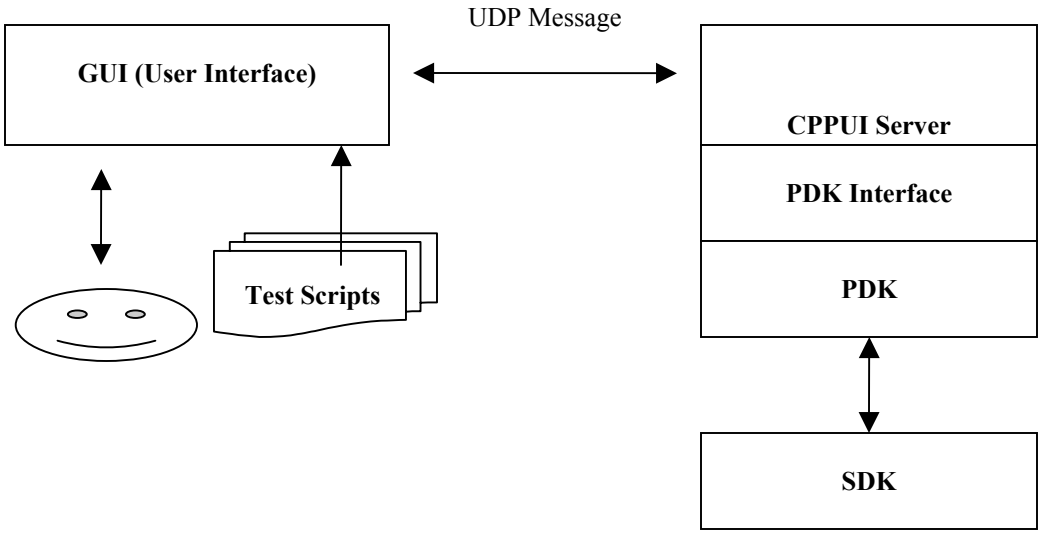

Figure 2. Overview of CPPUI

UDP message communication is used between the GUI and the backend. For each user input, new UDP message is encoded and sent to CP-PDK. In the CP-PDK, this message is decoded and the corresponding API is invoked. The result is encoded and sent back to the UI in the UDP message. In the UI, the response is decoded and displayed to user.

The CPPUI consists of a set of modules that are statically linked with the APIs in the CP-PDK. The CPPUI exercises the APIs with a range of correct and incorrect arguments and looks for a match between the expected and actual return values.

If the expected and actual return values from the APIs match for the different test cases, it can be presumed that the implementation complies with API specifications. For a complete description of the cppui, refer to the ***Control Plane Platform User Interface (cppui) User Guide*** included with this release of the CP-PDK.

To start the CPPUI GUI, run the following command from the cppui/ui/ directory:

```
python cppui.py
```

Make sure that the python installation directory path is a part of the system PATH variable.

3.1 Packet Testing example using CPPUI

This section describes an example of how to do packet testing in **IPv4-4GBEthernet pipeline**.

Step 1: Make sure that the status of physical port link used to do packet testing (ping) is down by pulling out the network cable.

Step 2: Depending on your network topology, make sure you make changes in IP Address and PortId of the helper script cp_pdk/cppui/ui/ts/modules/ipv4/helpscripts/Interface/ipv4_create_bind_if_lan.ts.

This helper script by default creates an IPv4 Interface and Sets *IP Address of 10.10.40.2* and on *PortId 0*.

Step 3: The reference CPPUI Test Script to execute in CPPUI tool for basic packet testing is **ipv4_setup_lan_disc_tables.ts** , that is present in **cp_pdk/cppui/ui/ts/modules/ipv4/testscripts/common** directory. There is also a **readme.txt** file which details out the various helper-scripts that this script includes. This script invokes all the requisite Interface Mgmt, IPv4 APIs helper scripts including the one mentioned in above step.

Step 4: After executing all the previous steps successfully, (check the history log of CPPUI), you may add NextHop Entry with e.g. IP Address of Packet Generator (10.10.40.1) and then add the Prefix Entry with the same IP Address (10.10.40.1). This will enable the ARP packets being sent and response received.

Step 5: Only, when the script is run, plug-in the cable so that physical port link is up and check if Virtual Interfaces created in Control Plane has proper address assigned.

Step 6: Start pumping packets from Packet Generator (10.10.40.1) with destination as IPv4 Interface configured using CP-PDK (10.10.40.2)

.

Part 4: Frequently Asked Questions

4 Frequently Asked Questions

1. **Why does CP-PDK compilation fail?**

   Make sure that you build core components in the order of Egress CC → Ingress CC before you compile CP-PDK. CP-PDK uses certain header files that are copied only during Ingress CC build.

2. **Why does the Fpmodule_core and fpmodule_proxy not compile?**

   Verify below mentioned Flags in the macros tab, **GLOB_DEFINES**

   1. **-DINCLUDE_PDK.**
   2. And following flags depending on your pipeline:
      (1) **-DIX_CC_ETH** for Ethernet based-pipeline.
      (2) **-DIX_CC_ATM** for ATM based-pipeline.
      (3) **-DIX_CC_QOS** for Diffserv services.

   For example, add both **IX_CC_ETH** and **IX_CC_QOS,** if the pipeline is Diffserv over Ethernet.

3. **Why do compilation errors occur in Linux, even when the cp_pdk code has just been copied from the CD?**

   The shell scripts contain ^M chars and they require cleanup by running dos2unix command. Go to the main dir of cp_pdk (e.g. /<root_dir>/cp_pdk) and issue commands

   dos2unix configure.sh

   dos2unix .bpfinc.sh

4. **How to include option component in the build?**

   Include your component in the OptionalComponents.make file in the cp_pdk directory.

5. **On running the cppui, why does the error message "not able to load libpil.so" show up?**

   Make sure that the library path is properly exported by:

   ```
   export
   LD_LIBRARY_PATH=/<root_dir>//cp_pdk/common/pil/lib/ia_linux/
   ```

   in the linux environment and the libpil.out is loaded in the VxWorks.

6. **Why cant the cppui tool connect onto a socket?**

   Probably someone else is using the port. Hence, change your port setting by export `CPPDK_PORT=8000`

7. **How to enable logger for the respective component?**

   Following flags can be added in the makefile to enable various loggers.

   CFLAGS += -DCPPUI_WAIT_RESP "Wait for async response from FE"

   CFLAGS += -DEXIT_ON_DISCONNECT "Normal shutdown on Disconnect from UI"

CFLAGS += -DPRINT_LOG	"Print log messages on stdout, otherwise writes log into "cppui.log"
CFLAGS += -DIF_LOG	"Interface Management Log"
CFLAGS += -DDS_LOG	"DiffServ Log"
CFLAGS += -DIPV4_LOG	"IPV4 Log"
CFLAGS += -DATM_LOG	"ATM Log"
CFLAGS += -DMPLS_LOG	"MPLS Log"

8. **In the remote CP in linux, configuration virtual interface is not created - Why?**

   We must set attribute and PORT UP Event for a particular port to get the virtual interface created. If the virtual interface is not yet created, then make sure that the vip module is inserted in the kernel. The lsmod command should show the vip as kernel module. If not, insert vip.o to kernel by entering the `insmod` command. You have to restart the CP and FP after inserting the vip.o in the kernel.

9. **How to view/delete virtual interfaces created by cp-pdk?**

   The **ifconfig** command shows the created virtual interfaces. If the virtual interfaces are not deleted, you should delete it every time during the shutdown of the control plane by entering the following command:

   ```
   iptunnel del vi1-0(virtual interface)
   ```

   Make sure that before you start the CPPUI, `lsmod` should show 0 users to vip.

10. **Instead of using GUI as the Conformance test Environment, can we have a text-based version instead?**

    Yes. Refer to the ***Control Plane Platform User Interface (cppui) User Guide*** included with this release of the CP-PDK.