Conformance Test Framework (Control Plane Platform User Interface)

User's Guide

Control Plane-Platform Development Kit 2.11

March 2004

Contents

Figures

Tables

Part 1: Introduction

1 Introduction

The Control Plane Platform User interface (CPPUI) tool is a test environment developed for CP-PDK. The CPPUI is a simple graphical user interface that runs with Python 2.2.2. You can use this tool in text mode to run generated test scripts. The user interface is generated completely and input to the generator is XML notation of the test cases.

1.1 Purpose

This document specifies the usage of the CPPUI tool with the PDK.

1.2 Scope

The terms Graphical User Interface (GUI) and User Interface (UI) are used to specify test engine in this document and they do not relate to its general meaning of graphic mode and text mode. The design of the user interface, tool generator, interaction of the tool with the PDK, and input to the generator are not covered in this document.

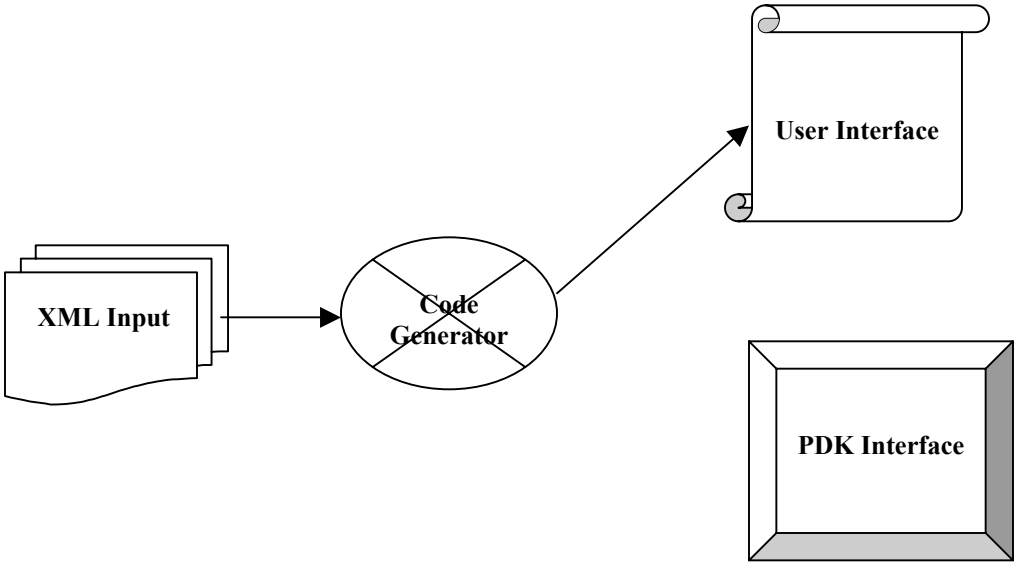

Figure 1: High Level Overview of Code Generator

The code generator reads input from the XML file and generates the following:

- Python user interface
- PDK interface (C language code)

The PDK interface should be compiled with the PDK. You can use the UI to configure or test the PDK. Design of the code generator, details of the UI code, and the PDK interface are not covered in this document.

Part 2: Components of CPPUI

2 Components of CPPUI

This chapter provides an overview of the CPPUI tool and its interaction with the PDK.

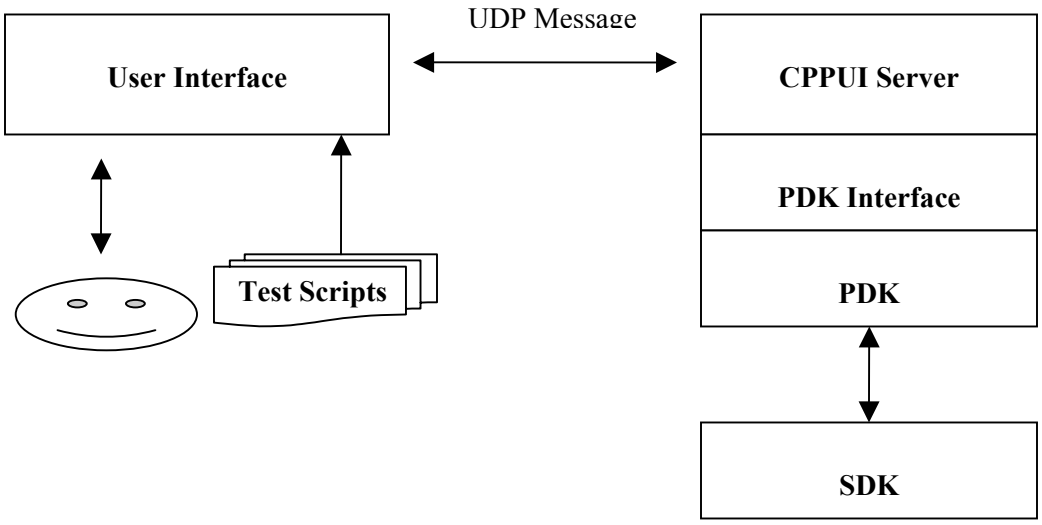

Figure 2: Interaction of the CPPUI with the PDK

The UDP message communication is used between the UI and the CPPUI server. A new UDP message is encoded and sent to the PDK for each user input. . This message is decoded in the PDK, and the corresponding API is invoked.
The result is encoded and sent back to the UI in the UDP message. The response is then decoded and displayed to the user.

User Interface Components

You can give input to the user interface through dialogs and test scripts. The CPPUI understands the test scripts that are generated through the tool. You can provide input to the UI through different menu options displayed in the dialogs.

The UI provides the following options:

- Connect/disconnect the CPPUI server
- Load test scripts
- Enable /disable interaction mode and logging
- View log files
- Traverse PDK responses in increasing/decreasing order

The PDK response is displayed in the following two windows:

- PDK command responses
- PDK events

Figure 3 illustrates a high-level overview of the UI Components.

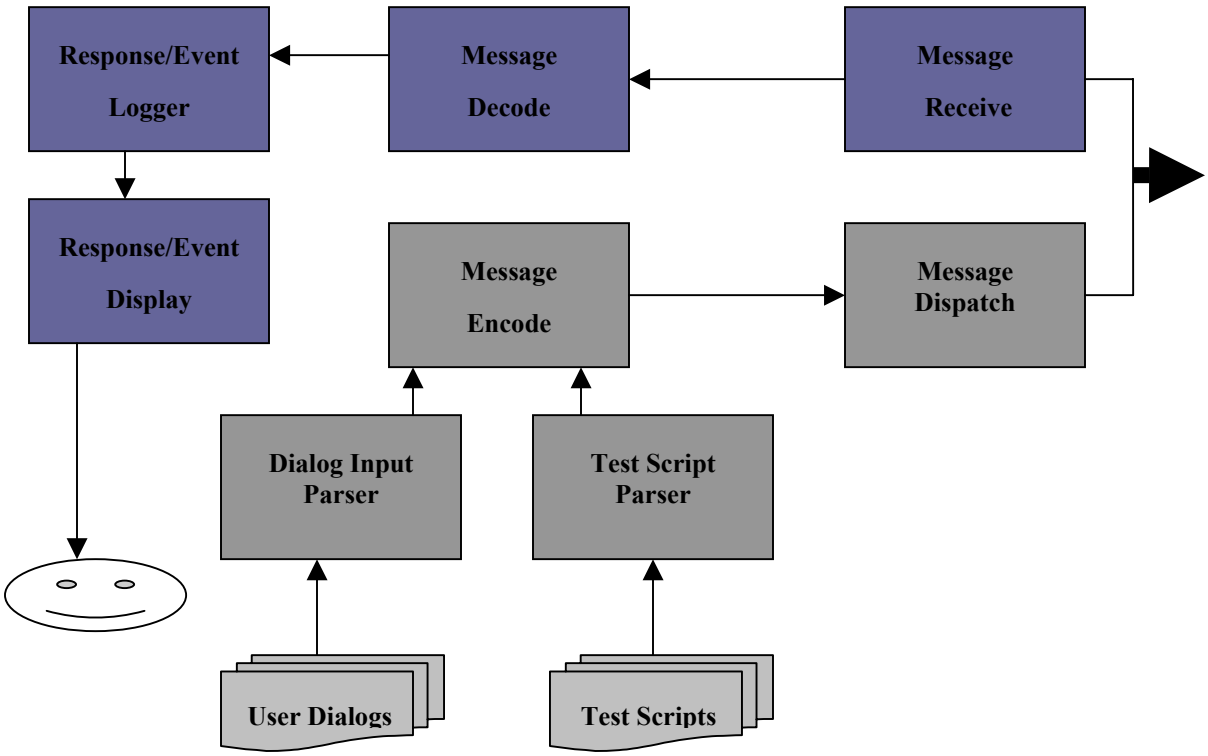

Figure 3: High-level Overview of UI components

2.1 Server Components

The server runs on the UDP port and accepts messages from the UI. New state information is created on receipt of every message. This information is active till the response is sent or the message timeout.

If the message is valid, the PDK API is invoked after decoding. The PDK response is encoded and sent back to the UI. The PDK events are encoded and sent to all the connected UIs through the event interface of the server.

Figure 4 displays a high-level overview of the server components.

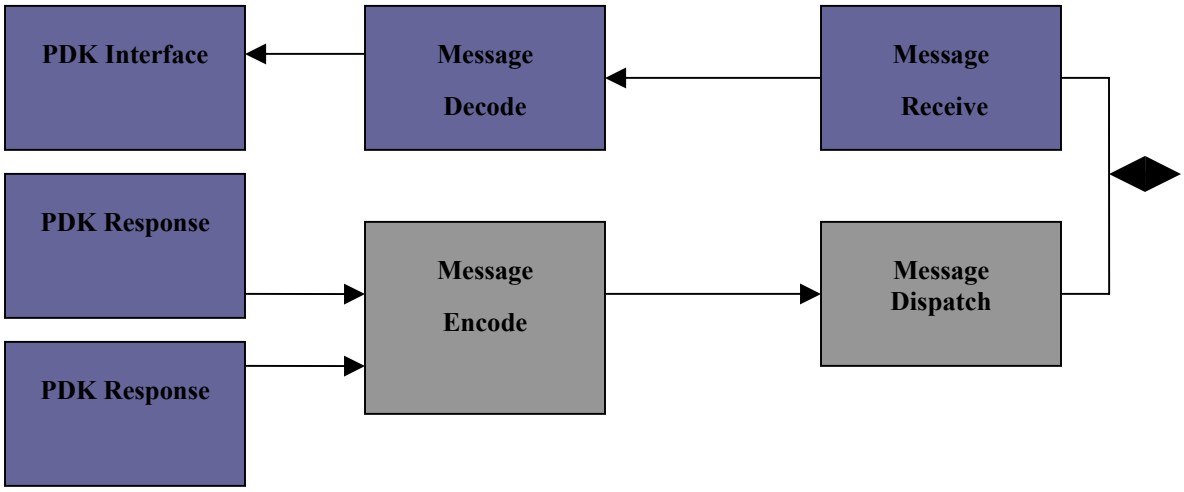

Figure 4: High-level Overview of Server Components

Part 3: Graphic Mode

3 Graphic Mode

3.1 Startup

Run python **cppui.py** to start the CPPUI. Ensure that the Python installation directory path is a part of the system PATH variable. You can view the GUI window as shown in Figure 5

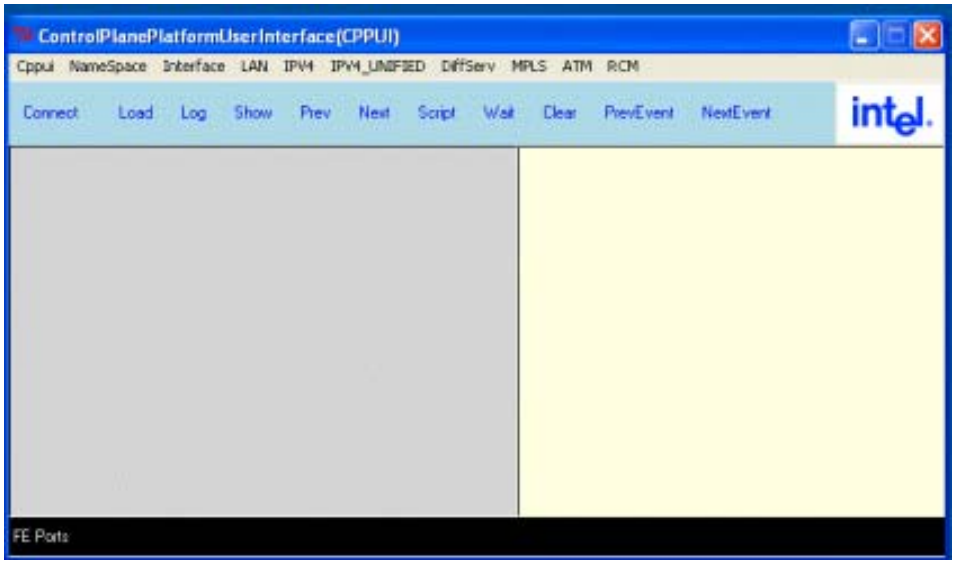

Figure 5: Graphical User Interface Window

3.2 Connect / Disconnect

3.2.1 Connect

1. Click Connect command to connect the CPPUI tool.
2. You can view the CP-PDK connect dialog box. At this instance, the Connect command changes to Disconnect.
3. You must enter the IP address and the port information of the server and click OK button. Click Cancel to cancel the connection. . The IP address indicates the IP address of the machine, in which the server is running and the port is the server UDP port.
4. If there is no connect success response, you can only view the log files and cannot perform any other operations.

3.2.2 Disconnect

1. Click Disconnect command to disconnect the CPPUI tool from the server.
2. The Disconnect dialog box is displayed. At this instance, the Disconnect command changes to Connect.
3. Click Yes to disconnect else No.

3.3 Load

1. Click Load command to execute the test script.
2. You can view the File Input dialog box.
3. Select the test script to execute.
4. All the PDK responses are shown in the response window on selecting the test script. .

3.4 Log

1. Click **Log** command to enable/disable the log.
2. You can view a new File Input dialog box if the log is disabled. ,
3. You can either enter a new file name or select an existing one. If you select an existing file, its old contents are lost.

3.5 Show

1. If you have enabled the log and click **Show** command, you can view the enabled log file.
2. If the log is disabled, you can view the File Input dialog box. Select the log file to view.

3.6 Prev

Click **Prev** command to view the previous PDK responses. .

3.7 Next

Click **Next** command to view the next PDK response.

3.8 Script

1. The test script file should be enabled to generate any of the test cases. You can enable the test script only when the UI is not connected to the server.
2. Click **Script** command to enable the test script file. You can view a new File Input dialog box. You can either specify a new test script file or select an existing one.
3. If you select an existing file the new test case is appended to the existing ones.
4. Disable the test script file after generating the test case. Else the connection to the server fails.

3.9 Wait

1. Click **Wait** command to enable/disable confirmation before executing each command.
2. If you enable the wait mode, confirmation is obtained for each input.

3.10 Clear

1. Click **Clear** command to clear the response history and event history.
2. All the counters (command sequence and event) are reset to zero on click of the Clear command. You can view all the cleared history in the log file if you have enabled the log.

3.11 PrevEvent

Click **PrevEvent** command to view the previous PDK event**.**

3.12 NextEvent

Click **NextEvent** command to view the next PDK event.

Part 4: Operation in Text Mode

4 Operation in Text Mode

The CPPUI tool starts the text mode engine when you invoke the cppui.py with any argument. The server host and the port are set to localhost and 7890 respectively by default. You can change the host and the port by invoking it with the following argument:

```
-host <server address> -port <server UDP port>
```

No test script is supported in the text mode operation. The UI sends connect request message to the server on start of the text mode, and the commands as shown in Figure 6 is displayed.

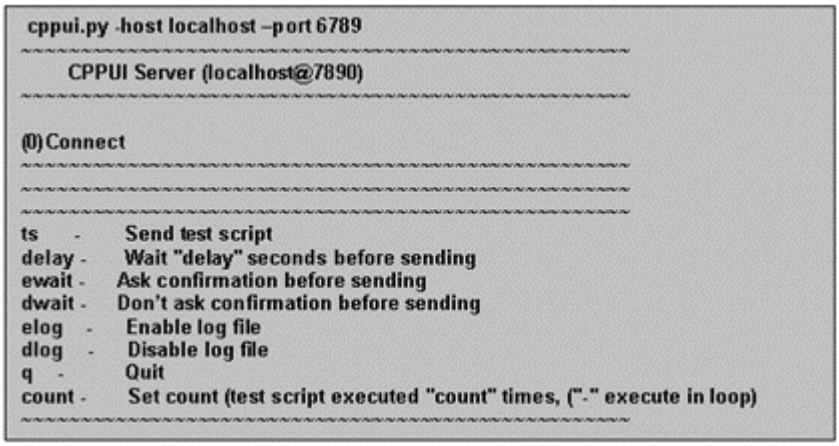

Figure 6: Available Commands in the Text Mode

You can enter one of the options displayed in **Test: prompt.** All the confirmations are displayed with Y/ N options. If you do not specify any value in the confirmation, Y (Yes) is taken as the default value. You can specify the test script file without relative path or absolute path if the test script is within the **ts** directory. You must specify the test script file using absolute or relative path, if the file is not in the ts directory.

If you select the q (quit) option, disconnect request is sent to the server and the program exits. If you set the delay, each command in the test script file is executed with the delay specified in seconds. The delay is effective only when the wait mode is not enabled.

Part 5: CPPUI Server

5 CPPUI Server

The CPPUI server receives messages from the User Interface (UI) and invokes PDK API. The PDK response is sent back to the user interface. The UDP communication channel is used to communicate between the UI and the server.

5.1 Build

1. The CPPUI source files are compiled and linked with the PDK in order to build a server application. You can use the following compilation flags:
    - CPPUI_WAIT_RESP
    - EXIT_ON_DISCONNECT
2. The CPPUI_WAIT_RESP flag is enabled by default.
3. If you compile the source files with CPPUI_WAIT_RESP flag after the invocation of NPF asynchronous API, the source files wait for ASYNC_RESP_WAIT_TIME response.
4. The ASYNC_RESP_WAIT_TIME is 5 seconds by default. The ASYNC_RESP_WAIT_TIME flag is a constant defined in include file cppui.h.
5. If you compile the source files with the EXIT_ON_DISCONNECT flag on display of CPPUI _DISCONNECT_REQ message, the PDK shutdown is called and the application exits normally
6. The EXIT_ON_DISCONNECT flag is not enabled by default, and this is the only way you can shutdown the server application through UI.

5.2 Running the application

1. When you build an application an executable **cppui** is created by default.
2. The server application uses the UDP port 7890 to receive the UI messages.
3. You can change these port values by setting the CPPDK_PORT environment variable.

Part 6: CPPUI Test Environment

6 CPPUI Test Environment

6.1 File Organization

All the files in the CPPUI tool fall under the directory cp_pdk/cppui, which has the following sub directories:

Table 1. Sub-directories of the cp_pdk/cppui

Directory	Description
build	The build directory has **make** file to build a server application and it has the following two subdirectories: 1. src – This contains C source files 2. inc – This contains the header files
ui	The ui directory contains Python user interface file - **cppui.py** and it has the following three sub directories: 1. ts - This directory contains all the test scripts and it can have sub directories. The test script directory can include other test scripts files by **`–include <test_script_other>`**. The file mentioned in the **`test_script_other`** can be anywhere within the **ts** directory. The CPPUI test tool scans for all the files under the ts directory 2. log - This is the default directory used by the CPPUI tool to place the log files 3. .gif - This directory contains the image files that are used by the GUI. If needed, you can move the ui directory to any other directory or machine. But for better functioning, it is advised not to move directories inside the **ui**.
Gen	This directory holds the code generator and the XML files. **gen.py** is the Python generator and **parse** is the script file to invoke the generator. Input to the generator is the text file and each line of this file is: `XML file name; input to the generator` Any line preceded by # is considered as a comment line and they are not considered in the generation process. In parse script, main.xml is the default generator input file. You can add new files / delete files from the **main.xml,** but ensure that the **common.xml** and **enums.xml** are in the beginning of the file. It has the following subdirectory: • xml - This directory contains the XML files which are input to the generator

6.2 Test Script

6.2.1 Format

You can input values to the user interface through dialogs or test scripts. The GUI generates these test scripts. Each line in the test script file is a test command. General format of the test command is shown below:

```
[('Command name', 'Command id'), ('Param1 Name', 'Param1 value'),
.....]
```

Sample test script files are shown in Figure 7.

FILE: qos.ts

```
--include FE_1_NAMESPACE.ts

[('DiffServ_Create_Classifier', '100'), ('NoOfDpe', '1'), ('IfHandle', 'LAN:0'), ('Direction', '1'), ('DpeType', '1'), ('ClassifierType', '1'),
('Precedence', '1'), ('DestAddress', '10.0.0.1'), ('DestNetpLen', '2'), ('SrcAddress', '10.0.0.2'), ('SrcNetpLen', '2'), ('Dscp', '1'), ('Protocol', '0'),
('DestL4Port', '80'), ('SrcL4Port', '98')]

 [('DiffServ_Dpe_Add_Associate', '104'), ('no', '1'), ('DpeHandle', 'CLASSIFIER: 0'), ('PrevDpeHandle', 'CLASSIFIER:0')]#
[('DiffServ_Dpe_Query', '105'), ('no', '1'), ('DpeHandle', 'CLASSIFIER: 0')]
```

FILE: FE_1_NAMESPACE.ts

```
[('NameSpace_Create', '1'), ('NodePath', '/NPFSys/0/Router/0/IfPort'), ('NodeType', '0'), ('NodeDataType', '9'), ('NodeInst', '0')]

[('NameSpace_Create', '1'), ('NodePath', '/NPFSys/0/Router/0/IfPort/0'), ('NodeType', '1'), ('NodeDataType', '9'), ('NodeInst', '0')]

[('NameSpace_Create', '1'), ('NodePath', '/NPFSys/0/Router/0/IfPort/1'), ('NodeType', '1'), ('NodeDataType', '9'), ('NodeInst', '0')]

[('NameSpace_Create', '1'), ('NodePath', '/NPFSys/0/Router/0/IfPort/2'), ('NodeType', '1'), ('NodeDataType', '9'), ('NodeInst', '0')]

[('NameSpace_Create', '1'), ('NodePath', '/NPFSys/0/Router/0/IfPort/3'), ('NodeType', '1'), ('NodeDataType', '9'), ('NodeInst', '0')]

[('NameSpace_Create', '1'), ('NodePath', '/NPFSys/0/Router/0/IfPort/4'), ('NodeType', '1'), ('NodeDataType', '9'), ('NodeInst', '0')]

[('NameSpace_Create', '1'), ('NodePath', '/NPFSys/0/Router/0/IfPort/5'), ('NodeType', '1'), ('NodeDataType', '9'), ('NodeInst', '0')]
```

Figure 7: Sample Test Script

Note: The test script file qos.ts includes another test script file FE_1_NAMESPACE.ts and it has three test commands. When you use the qos.ts in the UI, you should note that the command sequence is similar to the commands in the file FE_1_NAMESPACE.ts followed by the commands in the file qos.ts.

The red colored text within the first parenthesis in Figure 7 denotes the command name and the command id. The remaining text within each parenthesis identifies the parameter name and the value. It is better not to change the value field in the first parenthesis as it can result in unknown command from the server or wrong values passed to the wrong PDK API and finally server crash.

6.2.2 Generation

To generate the test script, perform the following steps:

1. Start the GUI application (python cppui.py)
2. Enable the test script file
3. Select the command from the desired menu
4. Enter the required values in the input dialog

5. Click OK
6. You can view a new confirmation dialog box. Click Yes to continue and No to cancel.

6.3 Enumerated Values

If any input parameter is enumerated type, you can view the Select button in the input dialog box. You can view the results containing all enumerated values of the first parameter in a drop down window on click of the select button as shown in Figure 8.

The parameter name, for which enumerations are shown, is highlighted with black background color. You can select the desired value by double-clicking on the value. Click **v** button to move to the next enumerated parameter. . The drop down window disappears on coverage of all the enumerated parameters.

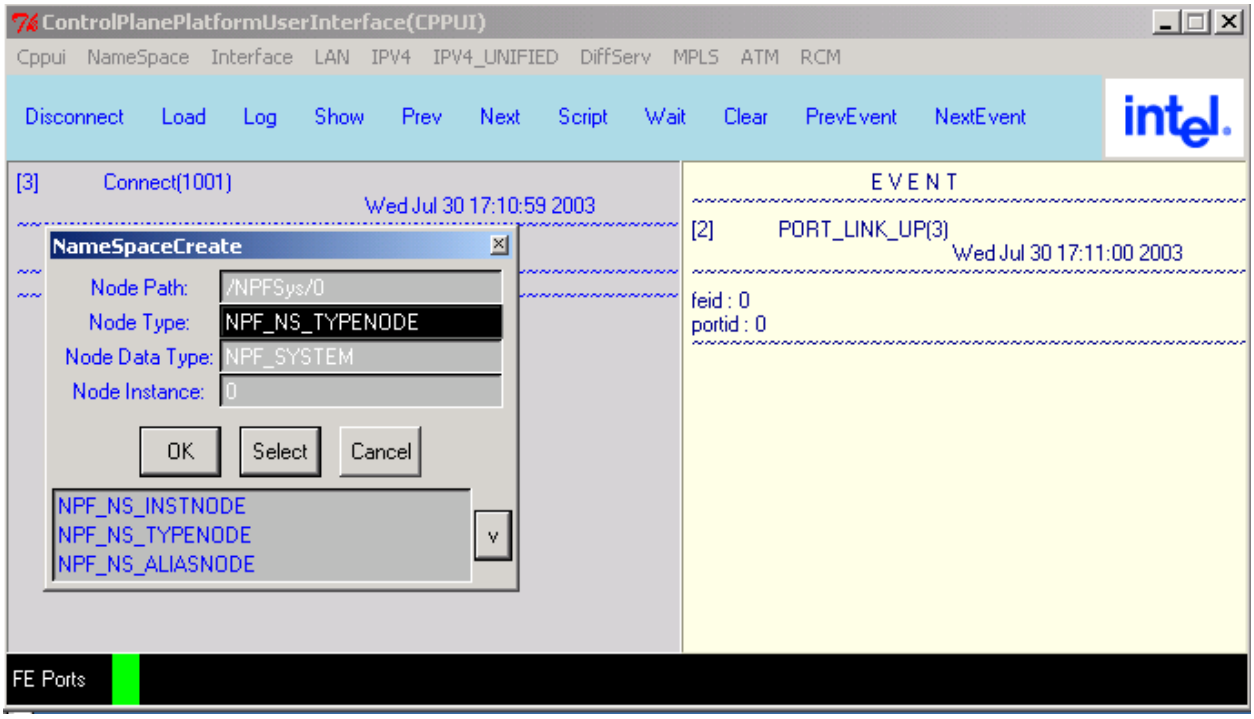

Figure 8: Selection of Enum values