NPT User Guide

On-Ramp Wireless Incorporated
10920 Via Frontera, Suite 200
San Diego, CA 92127
U.S.A.

NPT User Guide

010-0060-00 Rev. A

May 16, 2011

Contents

1 Introduction 1

2 Node Provisioning Tools 2

2.1 System Requirements 2
2.2 Setup and Configuration 3

3 eNode Software Upgrade Utility 4

3.1 Prerequisites 4
3.2 Performing an eNode Firmware Upgrade on an eHost 4
3.3 Performing an eNode Firmware Upgrade on a superHost 5

4 eNode Flash Configuration Utility 7

4.1 Programming an eNode Using an eHost 7
4.2 Programming an eNode Using a superHost 7

5 eNode Key Provisioning Utility 9

5.1 Setup and Configuration 9
5.2 Starting the eNode Key Provisioning Utility on an eHost 11
5.3 Starting the eNode Key Provisioning Utility on a superHost 13
5.4 Provisioning One or More eNodes 13

Appendix A Python Installation for a Windows-based Computer 16

A.1 Installing Python and Python Scripts 16
A.2 Identifying the Serial Port 25

Appendix B Creating RSA Keys 27

Appendix C Creating SSL Certificates 28

C.1 Generating a CA Certificate 28
C.2 Generating a Certificate Signing Request for the NPT Client 29

Appendix D Abbreviations and Terms 31

Figures

Figure 1. eNode on an eHost 5
Figure 2. superHost without a Cover Showing 16 eNodes with Antennas 6
Figure 3. Node Provisioning Architecture 9
Figure 4. eNode with Antenna Inserted onto an eHost 14
Figure 5. superHost Showing eNodes Alternating 180 Degrees from Slot to Slot 15

Revision History

Revision	Release Date	Change Description
A	May 17, 2011	Initial release.

1 Introduction

This document describes the setup, configuration, and use of a collection of utilities called Node Provisioning Tools (NPT) used for eNode provisioning.

NOTE: The intended audience for this document is test engineers, product test engineers, and product test technicians.

Complete provisioning of eNodes (sometimes referred to as nodes) involves three distinct utilities:

1. eNode Software Upgrade Utility (sw_upgrade.py)
   This utility is used for upgrading the eNode firmware.
2. eNode Flash Configuration Utility (config_node.py)
   This utility is used for programming eNode flash configuration parameters.
3. eNode Key Provisioning Utility (provision_node_keys.py)
   This utility is used for programming eNode Over-the-Air (OTA) security keys.

eNode provisioning must be completed in the order shown above. However, through the normal course of lab work or manufacturing, an upgrade may be needed after completely provisioning a set of nodes. In this case, it is not necessary to upgrade the flash configuration because it hasn't changed. It is acceptable to perform items 1 and 3 only, from the list above, in the order listed.

NOTE 1: For security reasons, upgrading the eNode firmware results in the deletion of its keys. After upgrading the eNode firmware, the eNode should also be re-provisioned with new keys.

NOTE 2: It is also recommended, for security reasons, that the eNode's firmware be programmed as part of the integration process, even if the eNode appears to have the current and proper firmware version. This is to overwrite any potentially malicious firmware image that purports to be a valid image.

To provision eNodes, an eHost or superHost must be connected to the NPT client. eNodes are inserted into a circuit board on the eHost or superHost through which the eNodes are provisioned using the node provisioning tools on the NPT client. An eHost allows the operator to provision one eNode at a time and is connected directly to the NPT client with a serial cable or a USB cable to a serial dongle. A superHost allows the operator to provision 16 eNodes at a time. It is connected by an Ethernet cable to a DHCP (Dynamic Host Configuration Protocol)-enabled router which is connected, by an Ethernet cable, to the NPT client. Multiple superHosts can be connected to an NPT client depending upon the number of ports available in the DHCP-enabled router.

NOTE: The use of eHost and superHost platforms for programming, configuring, and provisioning eNodes as described in this user guide are for informational and reference purposes only. While these devices can be used to provision a small number of eNodes, they were not designed for high volume production applications. For eNode production, it is recommended that a dedicated, high-volume production fixture be used.

2 Node Provisioning Tools

Node Provisioning Tools (NPT) are utilities that run on a client PC. The eNode Software Upgrade Utility (sw_upgrade.py) and the eNode Flash Configuration Utility (config_node.py) run on the NPT client without the need for any other server connectivity. These utilities simply require access to a serial port for eHost eNode configuration or Ethernet for superHost eNode configuration. The eNode Key Provisioning Utility (provision_node_keys.py) uses secure Ethernet protocols, such as Secure Socket Layer (SSL), to communicate with a Local Key Server (LKS) to provision an eNode with a gateway key, a code download (CDLD) key, and a node-specific root key.

For details regarding the LKS, eHost and superHost, refer to the following documents.

- *LKS User Guide*
- *eHost Product Specification*
- *superHost Product Specification*

2.1 System Requirements

The system requirements for an NPT system are as follows:

- A client running Windows® 7 operating system

  NOTE: Using a Cygwin environment on a Windows-based platform is not recommended for running the provisioning utility. However, it is useful for running incidental applications and utilities such as:

  - Untar / unzip (for extracting files)
  - Generating keys and certifications with OpenSSL

- Python® 2.6 including the module for PySerial (version 2.5). To install Python on a Windows-based computer, see Appendix A.

NOTE 1: If a USB-to-UART adapter is used, an adapter based on an FTDI2232D device is recommended. Internal testing discovered that adapters based on Prolific PL2303 devices experienced problems on platforms running Windows 7 operating systems and therefore are not recommended.

NOTE 2: With Windows 7, errors are occasionally observed in the serial port data. Reducing the serial baud rate tends to help reduce such errors but does not necessarily eliminate them.

2.2 Setup and Configuration

Provisioning an eNode involves three basic steps which must be performed, in order, by three standalone utilities as follows:

1. Programming the eNode firmware image using the eNode Software Upgrade Utility (sw_upgrade.py)

   NOTE 1: For security reasons, when a new firmware image is programmed into the eNode flash by any means other than an OTA software download, any existing keys within the eNode are deleted. Therefore, reprogramming the eNode's firmware also requires re-provisioning the keys.

   NOTE 2: It is also recommended, for security reasons, that the eNode's firmware be programmed as part of the integration process, even if the eNode appears to have the current and proper firmware version. This is to overwrite any potentially malicious firmware image that purports to be a valid image.

2. Programming the eNode configuration data using the eNode Flash Configuration Utility (config_node.py)
3. Programming the eNode keys using the eNode Key Provisioning Utility (provision_node_keys.py)

   NOTE: For security reasons, after the keys are programmed, the eNode's JTAG port is disabled and can only be re-enabled by completely clearing all flash data, including the factory calibration values.

These utilities are described in succeeding chapters. All of these utilities are contained in the 'provisioning_npt.tar.gz' file. Copy this file to a desired directory on the NPT client and untar/unzip it, which will place the extracted files in the subdirectory 'npt'. The subdirectory 'npt' will be created by the extraction process if it does not already exist.

NOTE: If Cygwin is installed on a Windows-based computer, then it is possible to use "Linux-like" commands at the Cygwin command prompt. For example, the 'provisioning_npt.tar.gz' file can be untarred/unzipped using the following command:

```
tar xzf provisioning_npt.tar.gz
```

Otherwise, use Windows-based compression applications such as WinZip or 7Zip to untar/unzip the file.

3 eNode Software Upgrade Utility

The eNode Software Upgrade Utility is a standalone Python script with the file name 'sw_upgrade.py' that can be used to upgrade the eNode firmware image through an eHost or superHost.

NOTE: The use of eHost and superHost platforms for programming, configuring, and provisioning eNodes as described in this user guide are for informational and reference purposes only. While these devices can be used to provision a small number of eNodes, they were not designed for high volume production applications. For eNode production, it is recommended that a dedicated, high-volume production fixture be used.

3.1 Prerequisites

- The eHost board must be connected to a serial port (or a USB-to-UART converter) on an NPT client.
- The superHost must be on the same network as the NPT client and requires DHCP.
- In order to perform an eNode firmware upgrade, the eNode must be in the "idle" RF state. The eNode Software Upgrade Utility (sw_upgrade.py) automatically places the eNode in the idle state.

3.2 Performing an eNode Firmware Upgrade on an eHost

When the eNode is in the idle state, the eNode is ready for receiving a new firmware binary image. This is done through an eHost with the following command from a DOS command terminal. Note that some filenames are user-defined.

```
sw_upgrade.py -d COM1 <enode_firmware.bin>
```

The following figure shows an eNode on an eHost.

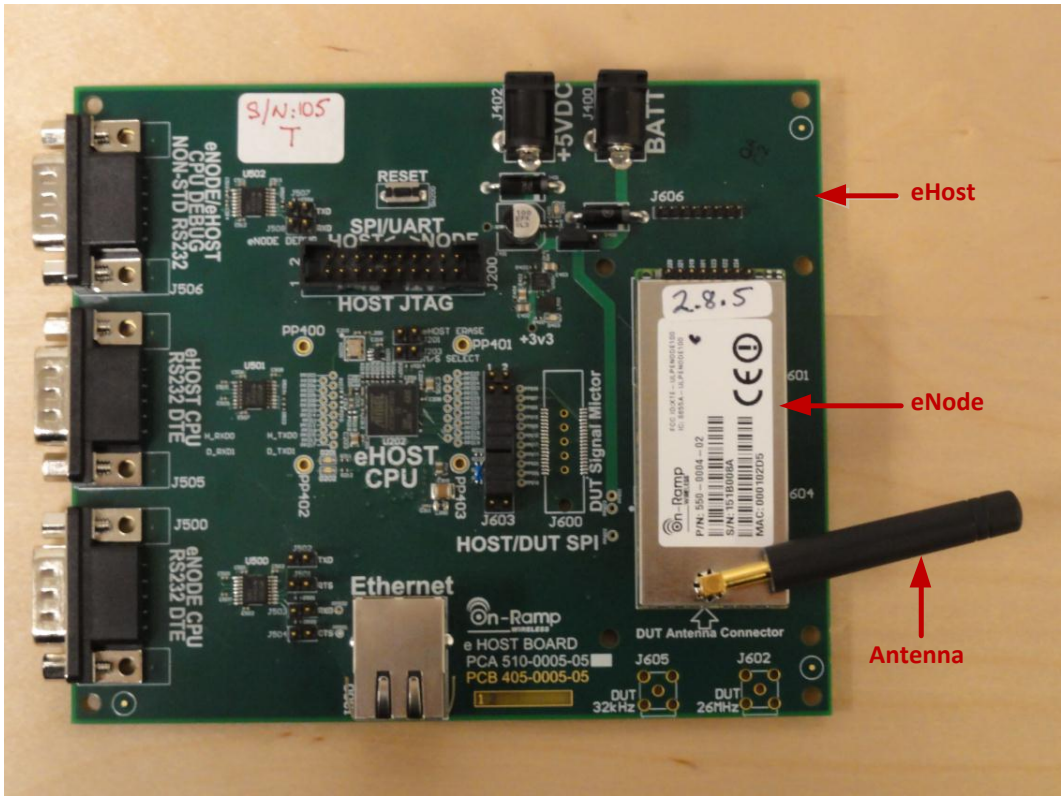


Figure 1. eNode on an eHost

NOTE: In order to perform a firmware upgrade with the eNode Software Upgrade Utility, the eNode must have an existing firmware revision of 4.4.0 or later. If you have a firmware revision of 4.4.0 or earlier, contact On-Ramp Wireless at support@onrampwireless.com

3.3 Performing an eNode Firmware Upgrade on a superHost

When the eNode is in the idle state, the eNode is ready for receiving a new firmware binary image. To upgrade an eNode on a superHost, type the following command. Note that some filenames are user-defined.

```
sw_upgrade.py -t <super_host_ip> -n <node_idx> <enode_firmware.bin>
```

Example: `sw_upgrade.py -t 192.168.1.2 -n 6 c:\users\admin\enode_r6.bin`

where:

- `super_host_ip` is the IP address of the superHost
- `node_idx` is the index of the node being programmed on the superHost

NOTE: This command upgrades the eNode at slot <node_idx> on the superHost at IP address <super_host_ip> . This command can be called repeatedly with different node indexes to program other nodes on the same superHost.

Additional usage syntax for the eNode Software Upgrade Utility can be obtained using the command:

```
sw_upgrade.py --help
```

The following figure shows a superHost and points out the numbering and positions of the eNodes.

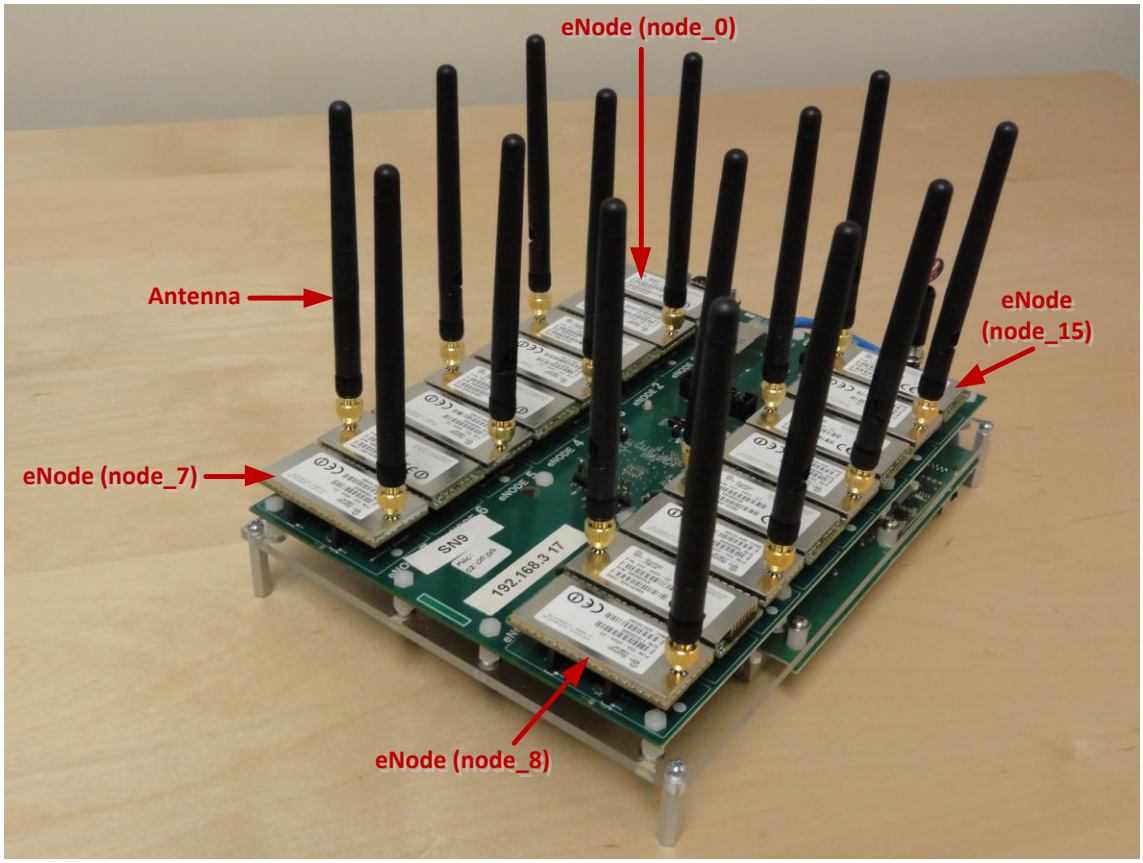


Figure 2. superHost without a Cover Showing 16 eNodes with Antennas

4 eNode Flash Configuration Utility

The eNode Flash Configuration Utility is a standalone Python script with the file name 'config_node.py' that programs the flash sector containing the eNode configuration parameters. The eNode Flash Configuration Utility reads the 'node_config.txt' text file which contains the desired parameter settings. Each parameter setting within the 'node_config.txt' file has a default value associated with it and contains self-documenting text describing the parameter and its allowable values. Most of the parameters configured here impact overall system performance of the devices as they operate the customer's ULP network.

NOTE: The use of eHost and superHost platforms for programming, configuring, and provisioning eNodes as described in this User Guide are for informational and reference purposes only. While these devices can be used to provision a small number of eNodes, they were not designed for high volume production applications. For eNode production, it is recommended that a dedicated, high-volume production fixture be used.

WARNING: The flash configuration parameter values are carefully selected by On-Ramp Wireless to optimize device and system performance. Certain configuration values or combination of values can result in degraded device or system performance, or in the worst case, result in device malfunction. These configuration parameters should not be modified.

4.1 Programming an eNode Using an eHost

The eNode can be programmed with the proper configuration parameters provided by On-Ramp Wireless, using an eHost and the following command at the DOS command prompt. Note that some filenames are user-defined.

```
config_node.py -c <node_config.txt> -d COM1
```

Example: `config_node.py -c c:\users\admin\device_network_cfg.txt -d COM1`

where:

- `-d` specifies the serial port connected to the eHost

4.2 Programming an eNode Using a superHost

To program the configuration parameters using a superHost, use the following command. Note that some filenames are user-defined.

```
config_node.py -i <super_host_ip> -n <node_idx> -c <node_config.txt>
```

Example: `config_node.py -i 192.168.1.2 -n 6 c:\users\admin\device_network_cfg.txt`

where:

- `super_host_ip` is the IP address of the superHost (filename is user-defined)
- `node_idx` is the index of the node being programmed on the superHost

Additional usage syntax can be obtained using the command:

```
config_node.py --help
```

5 eNode Key Provisioning Utility

The eNode Key Provisioning Utility (provision_node_keys.py) is the Node Provisioning Tool that retrieves the gateway key, the code download (CDLD) key, and the node-specific root key from the LKS through an SSL connection. It then programs the keys into the eNode through an eHost or superHost. Multiple NPT clients can be used simultaneously with a single LKS.

NOTE: The use of eHost and superHost platforms for programming, configuring, and provisioning eNodes as described in this User Guide are for informational and reference purposes only. While these devices can be used to provision a small number of eNodes, they were not designed for high volume production applications. For eNode production, it is recommended that a dedicated, high-volume production fixture be used.

WARNING: After an eNode is provisioned with keys, its JTAG interface is permanently disabled. The only way to re-enable JTAG is to completely clear all flash contents. This requires a factory recalibration of the eNode which must be done by On-Ramp Wireless. Contact On-Ramp Wireless at support@onrampwireless.com.

5.1 Setup and Configuration

The following diagram illustrates the hardware setup for eNode provisioning.

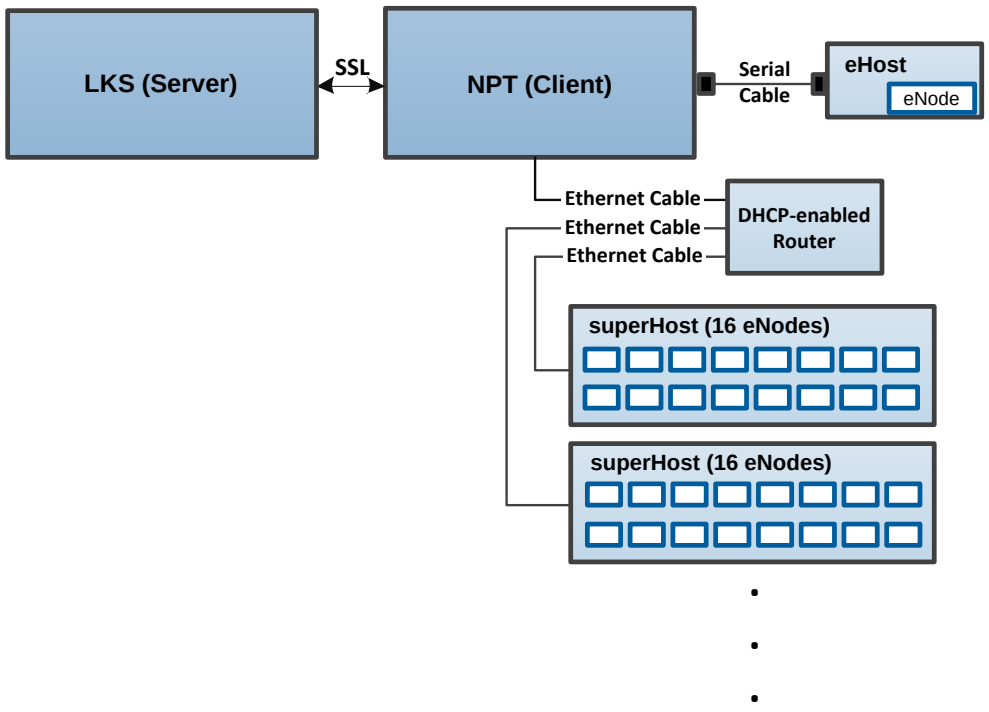


Figure 3. Node Provisioning Architecture

To setup and configure eNode key provisioning, follow the steps below.

1. In order for the NPT client to connect to the LKS through an SSL connection, an SSL certificate for the NPT client must be created, along with an RSA public/private key pair. A Certificate Authority (CA) must then sign the NPT client certificate. Refer to the following appendices for more details:
   - Appendix B: Creating RSA Keys
     Provides instructions on generating an RSA public/private key pair for the NPT client.
   - Appendix C: Creating SSL Certificates
     Provides details on how to create and sign security certificates.
2. Complete the following before proceeding:
   - Generate the NPT client keys
   - Generate the NPT client certificate signing request
   - Transfer the NPT client certificate signing request to a "security server" which is acting as the Certificate Authority
   - On the security server, sign the NPT client certificate signing request using the CA private key and CA certificate which will generate the NPT client certificate
3. After completing the items above, transfer the following items to a desired directory on the NPT client (typically the same directory as the eNode Key Provisioning Utility):
   - NPT client keys
   - NPT client certificate
   - CA certificate (but not the CA keys)
4. Be sure the LKS is up and running with the appropriate key database so that the eNodes can be programmed.
5. Make a note of the LKS's IP address or DNS name and which TCP port is being used for the LKS protocol link. This is assigned on a per-customer basis. For questions related to this, contact On-Ramp Wireless at support@onrampwireless.com.
6. Verify that the firewall, if any, is configured to allow LKS protocol traffic.
7. Connect an eHost to a UART serial port (or a USB-to-UART adapter) on the NPT client using a null cable. If using a superHost, connect an Ethernet cable to the superHost's Ethernet port (J300) and into a switch or router that is on a network with a DHCP server. The superHost displays its IP address through the J500 serial port.

   NOTE 1: The J500 serial port is configured to use a non-standard pinout (RTS and CTS used for other purposes), so a special DB9 cable should be used to connect only to:
   - pin 2 (superHost RXD),
   - pin 3 (superHost TXD), and
   - pin 5 (Ground).

NOTE 2: The J500 serial port on the superHost, by default, uses 115200 baud with character format 8N1, and no flow control.

For more details about the eHost and superHost, refer to the following documents:

- *superHost Product Specification*
- *eHost Product Specification*

5.2 Starting the eNode Key Provisioning Utility on an eHost

To start the eNode Key Provisioning Utility on an eHost, type a command similar to the following (all on one line). Note that some filenames are user-defined.

```
provision_node_keys.py -s <lks_ip_address> -p <lks_port_num> -d COM1 -c
<npt_cert.crt> -a <ca_cert.crt> -k <npt_key.priv.pem> -b <baud_rate> -B
<batch_num>
```

where:

- `lks_ip_addr` is the IP address or DNS hostname of the LKS
- `lks_port_num` is the TCP port number of the server for the LKS protocol
- `COM1` is the serial port (on a Windows-based platform) connected to the eHost
- `npt_cert.crt` is the signed SSL certificate for the NPT client (filename is user-defined)
- `ca_cert.crt` is the Certificate Authority's SSL certificate (filename is user-defined)
- `npt_key.priv.pem` is the unencrypted RSA private key for the NPT client in a PEM file format (filename is user-defined)
- `baud_rate` specifies the serial port baud
- `batch_num` is a user-determined, monotonically increasing, 32-bit integer number associated with a provisioning run

  The NPT client uses the batch number when requesting the node keys from the LKS. The LKS generates the requested node root keys and associates the batch number with the node and its keys in the database. This allows greater control over which keys are subsequently exported from the LKS to be securely shipped to the KMS.

  The purpose of the batch number is to facilitate tracking and logical grouping of node keys. For example, the batch number can be a sales order number so that all nodes produced and provisioned for that sales order can be grouped and tracked together. If the sales order number is not a true integer but contains alpha-numeric characters, then it must be mapped to and associated with an integer batch number.

NOTE: If batch numbers are not monotonically increasing, when keys are exported to the KMS by batch number, it is difficult to distinguish the keys that were created at an earlier date from those that were created at a later date.

Note that the batch number argument is initially treated as a string and converted to an integer value. String arguments starting with a '0' (for example, 05082011) are interpreted as an octal value and string arguments starting with '0x' (for example, 0x1238ef) are interpreted as a hexadecimal value.

Batch Number Examples

The following examples assume that the batch numbers are assigned in order by date from earliest to latest.

Correct: 100, 101, 102, etc.

Incorrect:

a100, a101, a102	Reason: Batch numbers contain a letter.
21A, 22A, 23A	Reason: Batch numbers contain a letter.
b42, b41, b40	Reason: Batch numbers contain a letter and are monotonically decreasing.
72, 51, 89	Reason: The batch numbers are not monotonically increasing.

NOTE: If verbose messaging is needed with the eNode Key Provisioning Utility (during initial server deployment, for example), it can be enabled by appending one or more "-v" options to the invocation script as shown below (all on one line):

```
provision_node_keys.py -s <lks_ip_address> -p
<lks_port_num> -d COM1 -c <npt_cert.crt> -a <ca_cert.crt>
-k <npt_key.priv.pem> -b <baud_rate> -B <batch_num> -v -v
-v
```

Verbosity Description

No "-v" options:	No messages displayed. Executes without displaying messages unless there is an error.
1 "-v" option:	Displays high level informational messages.
2 "-v" options:	Displays medium level informational messages.
3 "-v" options:	Displays all messages including low level/trivial informational messages.

NOTE: Future releases will support full logging capability. For now, output messages can be "tee'd" or redirected to an output file.

Example:

```
provision_node_keys.py -s 192.168.1.2 -p 4038 -d COM1 -c
npt_cert.crt -a ca_cert.crt -k npt_key.priv.pem -B 101 -v -v
>> log_file.txt
```

5.3 Starting the eNode Key Provisioning Utility on a superHost

To start the eNode Key Provisioning Utility on a superHost, type a command similar to the following (all on one line). Note that some filenames are user-defined.

```
provision_node_keys.py -s <lks_ip_address> -p <lks_port_num> -i
<super_host_ip> -n <node_idx> -c <npt_cert.crt> -a <ca_cert.crt> -k
<npt_key.priv.pem> -B <batch_num>
```

Example:

```
provision_node_keys.py -s 192.168.1.2 -p 4038 –i 192.168.2.27 -n 0 –c
npt_cert.crt -a ca_cert.crt -k npt_key.priv.pem -B 101 –v –v >>
log_file.txt
```

where:

- `lks_ip_addr` is the IP address or DNS hostname of the LKS
- `lks_port_num` is the TCP port number of the server for the LKS protocol
- `super_host_ip` is the IP address of the superHost
- `node_idx` is the index of the node being provisioned on the superHost
- `npt_cert.crt` is the signed SSL certificate for the NPT client (filename is user-defined)
- `ca_cert.crt` is the Certificate Authority's SSL certificate (filename is user-defined)
- `npt_key.priv.pem` is the unencrypted RSA private key for the NPT client in a PEM file format (filename is user-defined)
- `batch_num` is an arbitrary monotonically increasing integer number to associate with a provisioning run

  The NPT client uses the batch number when requesting the node keys from the LKS. The LKS generates the requested node root keys and associates the batch number with the node and its keys in the database. This allows greater control over which keys are subsequently exported from the LKS to be securely shipped to the KMS. The batch number can be something meaningful such as a sales order number, purchase order number, or a production run number, etc. Multiple nodes can be associated with one batch number.

5.4 Provisioning One or More eNodes

The eNode Key Provisioning Utility has been updated to clear the eNode's exception buffer as part of the provisioning process. The exception buffer is a section of flash reserved for storing error messages for later retrieval. The exception buffer is cleared by default during production when the keys are provisioned, that way any messages subsequently found in the buffer are known to have occurred after provisioning.

For additional usage syntax information, type:

```
provision_node_keys.py --help
```

To provision one or more eNodes using either an eHost or superHost, use the following steps.

CAUTION: It is important that the eNode ***not*** be inserted into or extracted from the eHost or superHost while the power is still on. This can cause physical damage to the eNode, eHost (or superHost), or both.

1. Verify that the eHost (or superHost) is powered off.
2. Insert an eNode into its mating connectors on an eHost or superHost. Be sure the eNode's MMCX RF antenna connector is on the appropriate side as indicated by the silkscreened text on the eHost or superHost board.

   eHost

   For an eHost, the RF antenna side of the eNode should be inserted into the eHost where it indicates "DUT Antenna Connector." See the following picture.

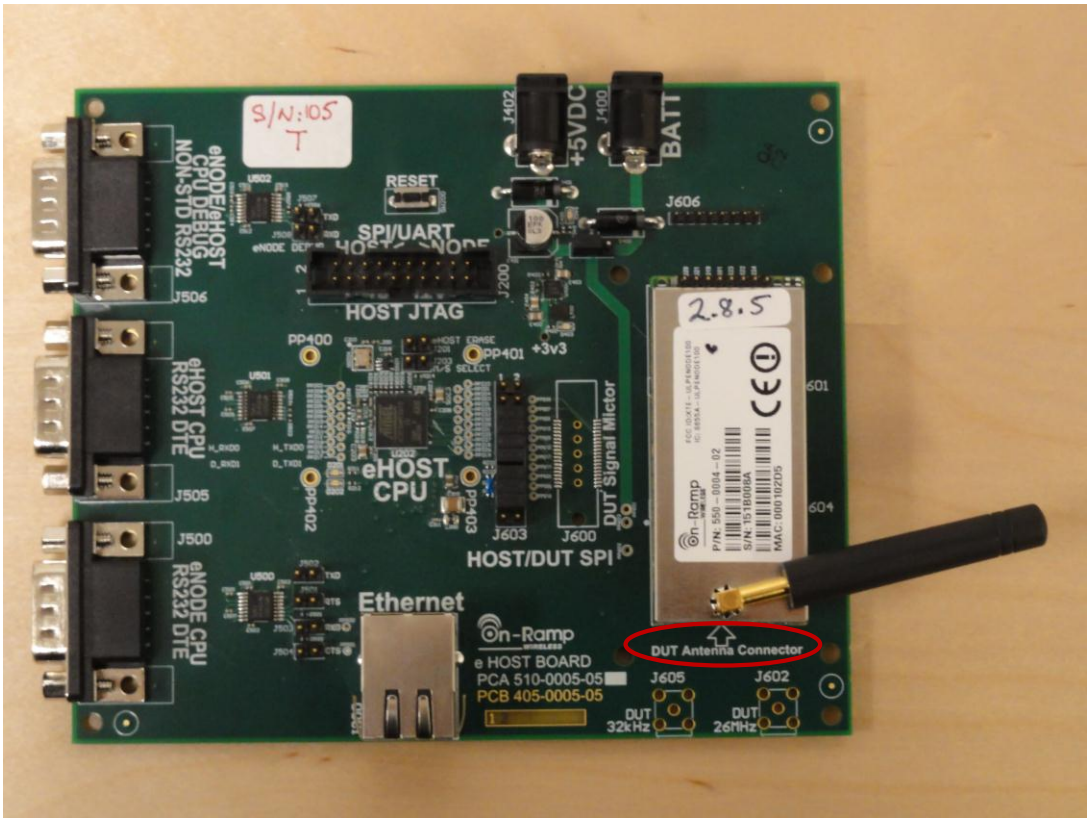


Figure 4. eNode with Antenna Inserted onto an eHost

superHost

In the case of the superHost, the orientation of the eNode alternates 180 degrees from slot to slot. ***Be careful to ensure that the eNodes are inserted correctly into the connectors or damage could occur.***

See the following picture.

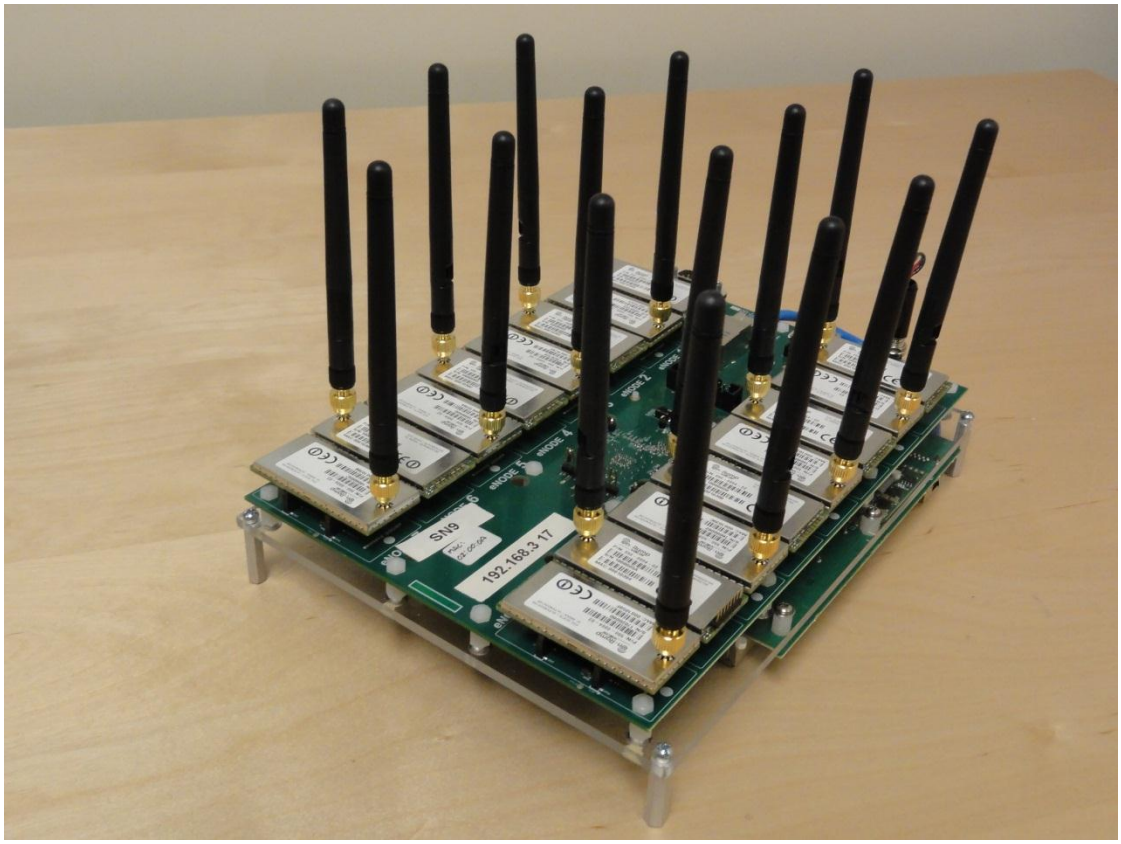


Figure 5. superHost Showing eNodes Alternating 180 Degrees from Slot to Slot

3. Turn on the power for the eHost (or superHost).
4. Run the eNode Key Provisioning Utility as described above. If an error message such as "TRANSPORT: ack failed" is displayed, power cycle the eHost (or superHost) and try running the eNode Key Provisioning Utility again.
5. If the key provisioning is successful, the message "Key provisioning successful" is displayed when the verbosity level of the eNode Key Provisioning Utility is set to a non-zero value. Otherwise, the message "Key provisioning FAILED" is displayed. If the default verbosity level is used (for example, no "-v" option), then the eNode Key Provisioning Utility executes without displaying any messages, unless an error occurs.
6. Turn off the power for the eHost (or superHost).

   WARNING: An eNode can be damaged by "hot swapping" it while the power is on.

7. Remove the provisioned eNode from the eHost (or superHost).
8. Repeat this process as required for additional eNodes.

Appendix A Python Installation for a Windows-based Computer

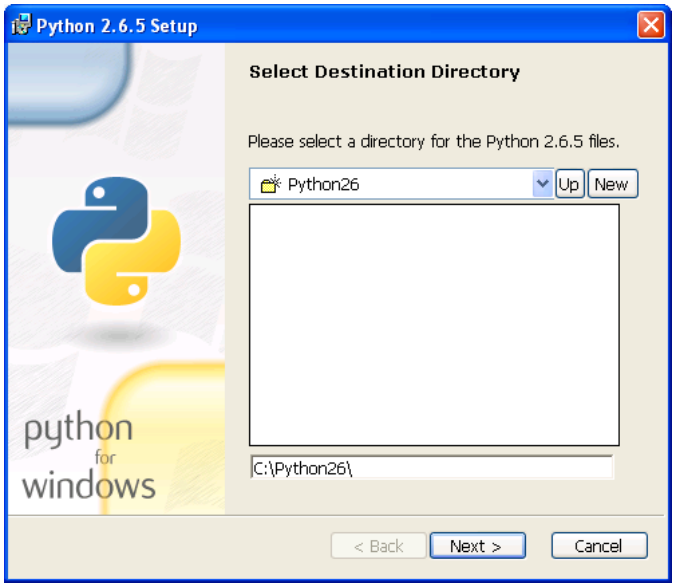
This procedure describes how to set up Python 2.6 on a computer running Windows 7 operating system.

NOTE: For computers with the Windows 7 operating system, select 'Run as administrator' for installation executables.

A.1 Installing Python and Python Scripts

To install python and python scripts, complete the following steps:

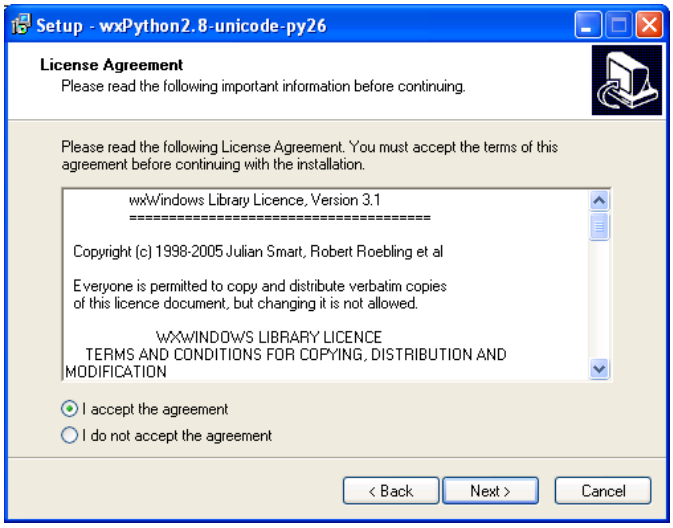
1. Create a temporary folder (for example, py26) in the local directory.
2. Copy all the files from the SDK/eHost/Python26 directory to the local py26 directory.
3. Double click on the **python-2.6.5** Windows-based installer package. To install the files to the local Python 26 folder on the hard drive, follow the installation prompts.


Python 2.6.5 Setup
Select Destination Directory
Please select a directory for the Python 2.6.5 files.
Python26
Up
New
C:\Python26\
python
for
windows
< Back
Next >
Cancel

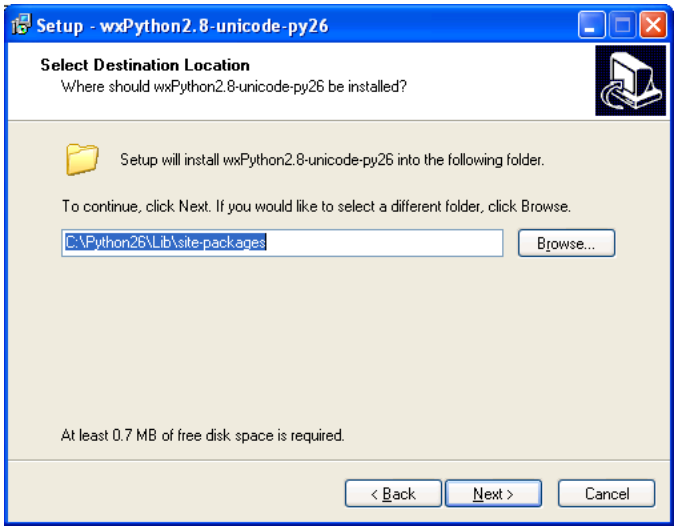

Python 2.6.5 Setup
Customize Python 2.6.5
Select the way you want features to be installed.
Click on the icons in the tree below to change the
way features will be installed.
Python
Register Extensions
Tcl/Tk
Documentation
Utility Scripts
Test suite
Python Interpreter and Libraries
This feature requires 21MB on your hard drive. It
has 5 of 5 subfeatures selected. The subfeatures
require 27MB on your hard drive.
python
for
windows
Disk Usage
Advanced
< Back
Next >
Cancel

NOTE: If there is an existing Python installation on the computer, remove it prior to installing the following files. Use the default installation parameters.

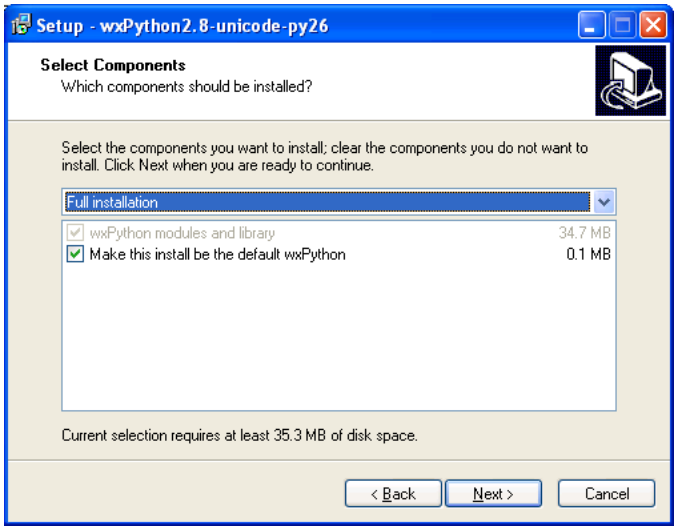
4. Double-click on wxPython2.8win-unicode-2.8.10.1-py26.
5. Click I accept the agreement, and click **Next**.



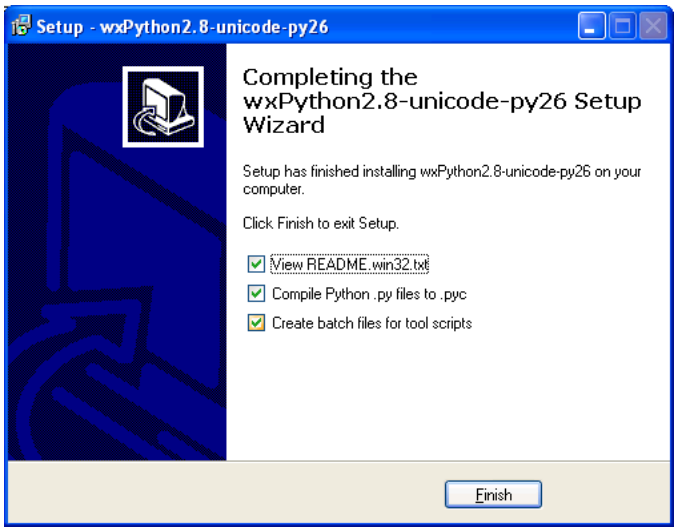
6. Select the Destination Location in which to install the Python script, or **Browse** to the location where the Python script will be installed, and click **Next**.



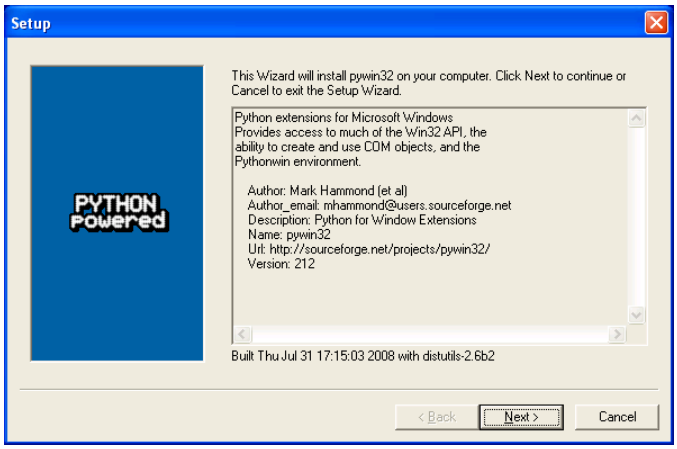
7. Select the components to install, and click **Next**.

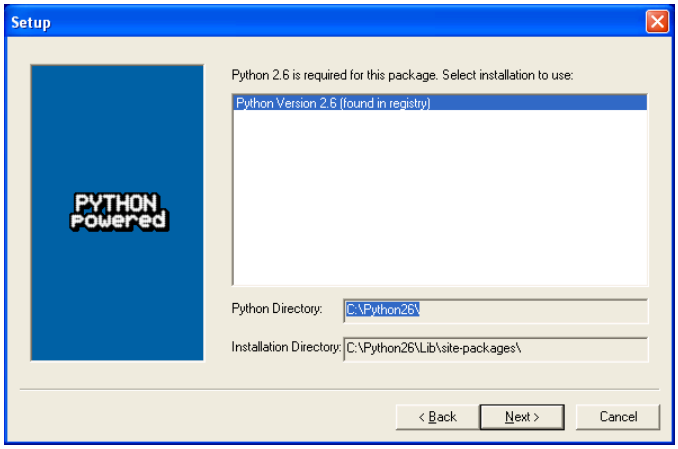


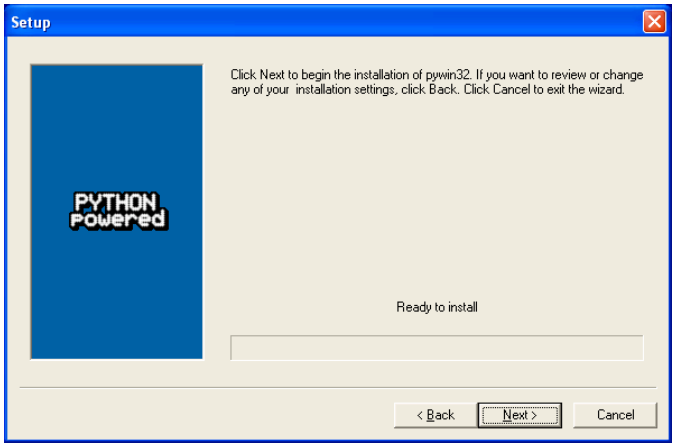
8. Click **Finish**. A Python window displays indicating that the packages are being integrated.

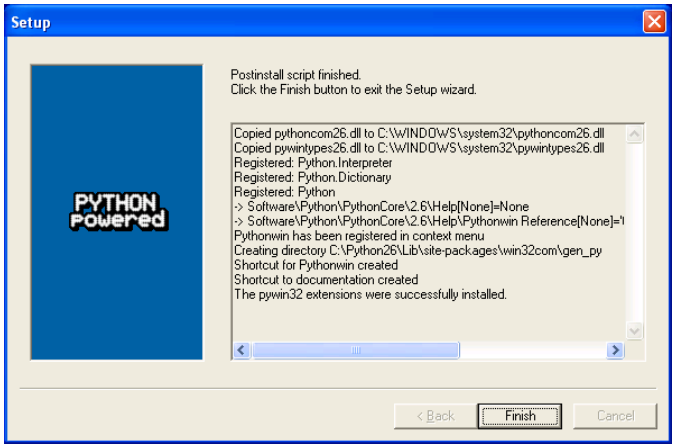


9. Double-click on the **pywin32-212.win32-py2.6** Windows-based installer package.
10. To install the files to the local Python 26 folder on the hard drive, follow the installation prompts.



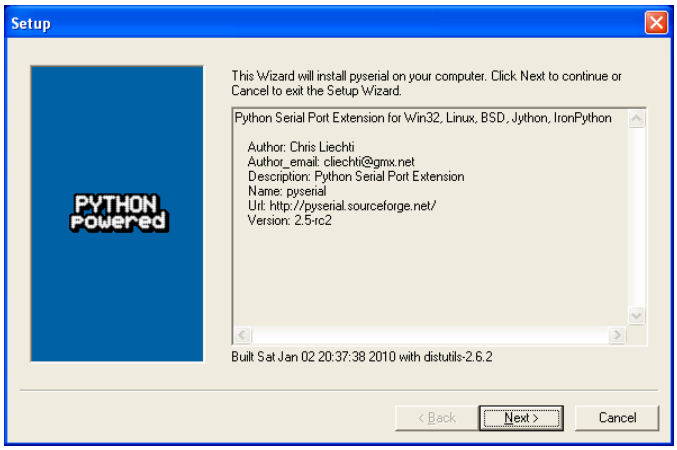


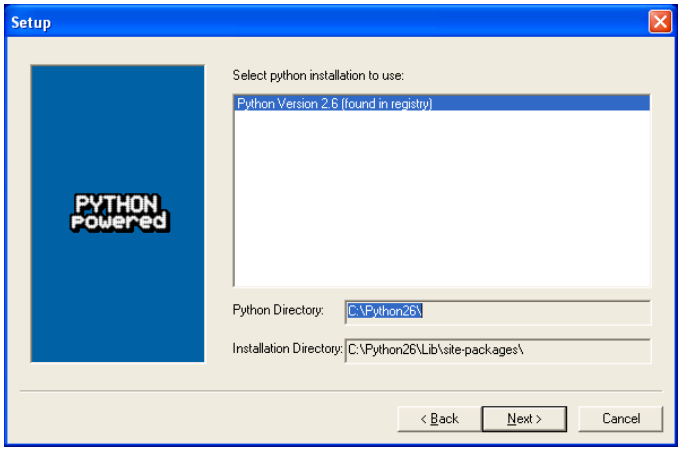


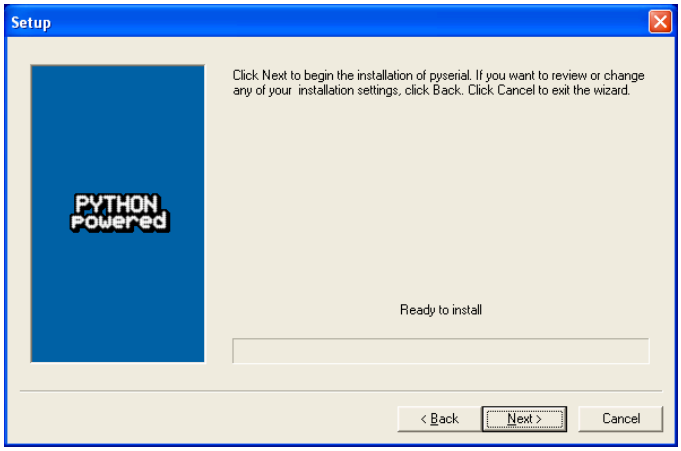


11. Double-click on the **pyserial-2.5.win32.exe** Windows-based installer package.

12. To install the files to the local Python 26 folder on the hard drive, follow the installation prompts.

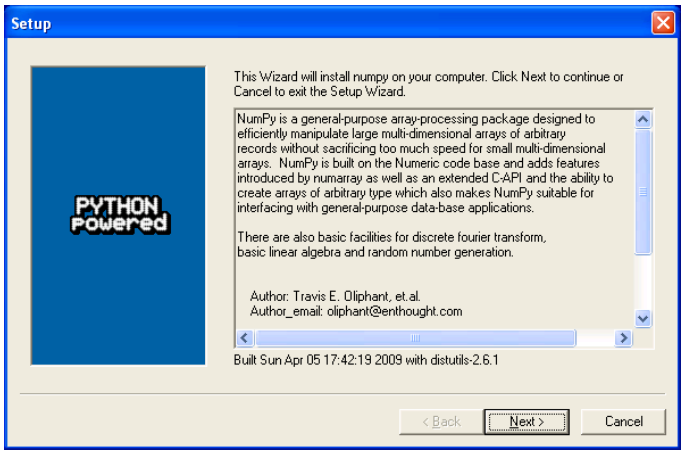


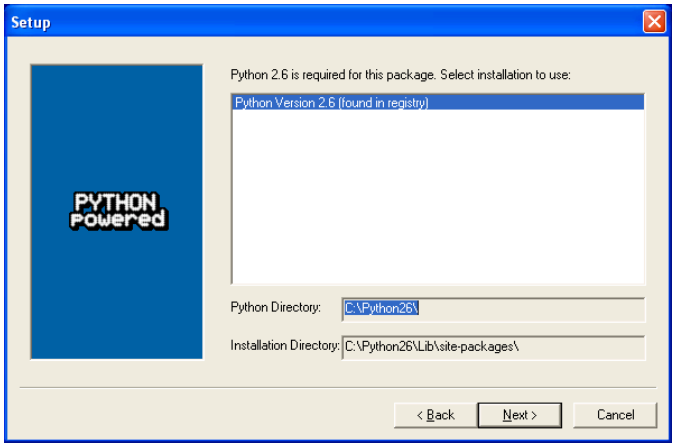


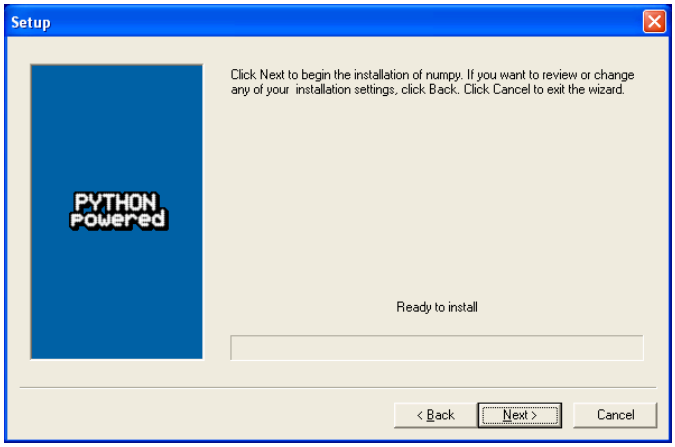


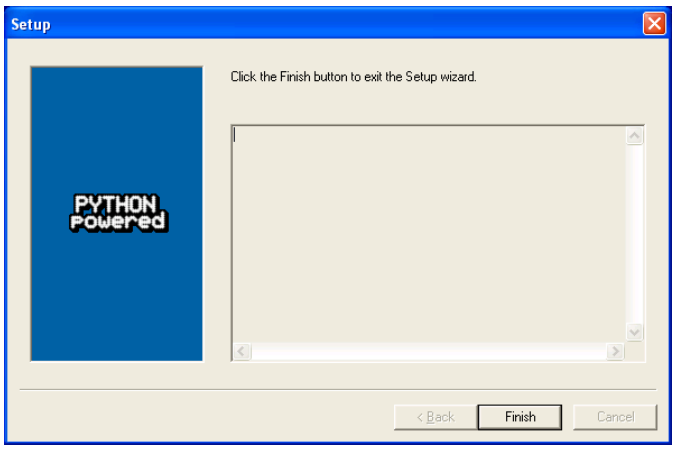
13. Double-click on the **numpy-1.4.1-win32-superpack-python2.6** Windows-based installer package.

14. To install the files to the local Python 26 folder on the hard drive, follow the installation prompts.

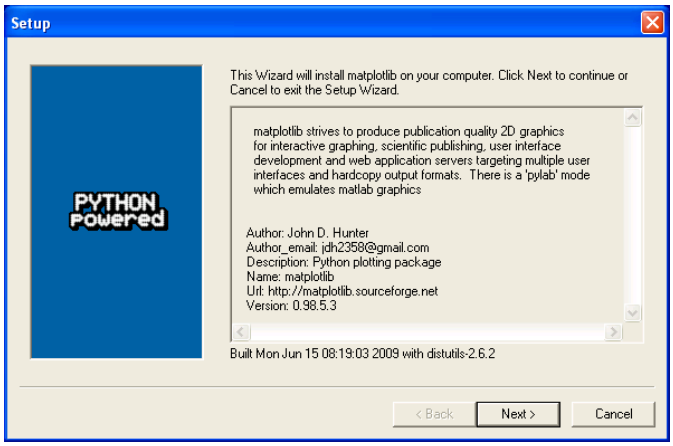


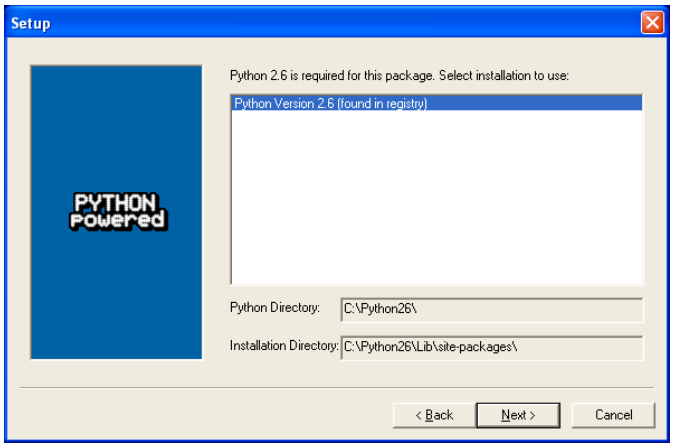


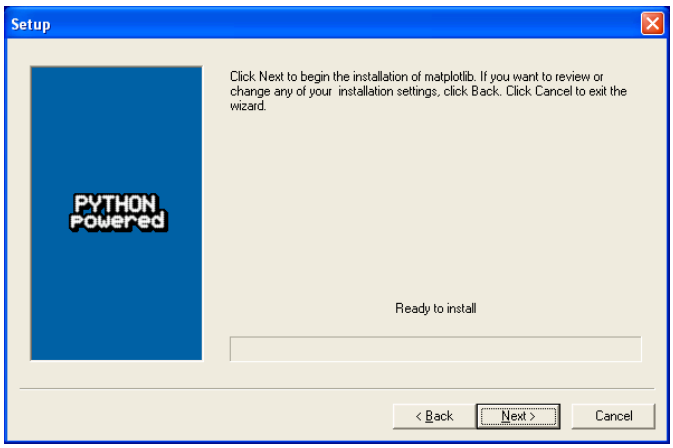




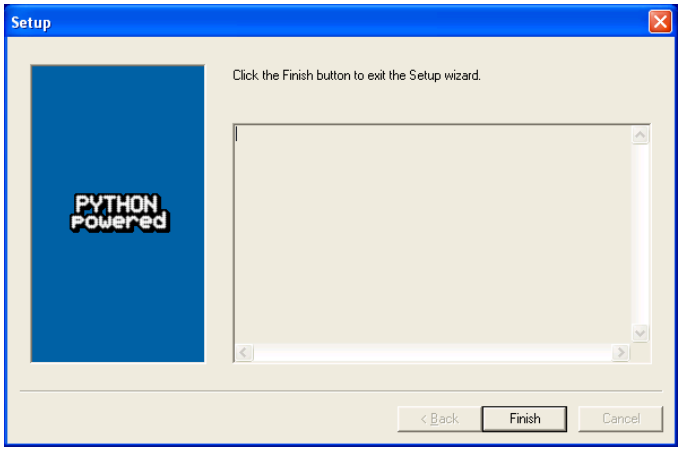
15. Double-click on the **matplotlib-0.98.5.3.win32-py2.6** Windows-based installer package.
16. To install the files to the local Python 26 folder on the hard drive, follow the installation prompts.







17. Click **Finish**.

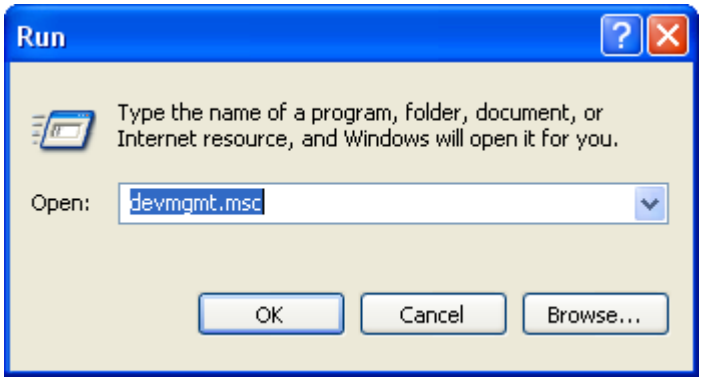


18. Double-click on the **PyVISA-1.3.win32.exe** Windows-based installer package. Follow all of the default installation instructions until complete.

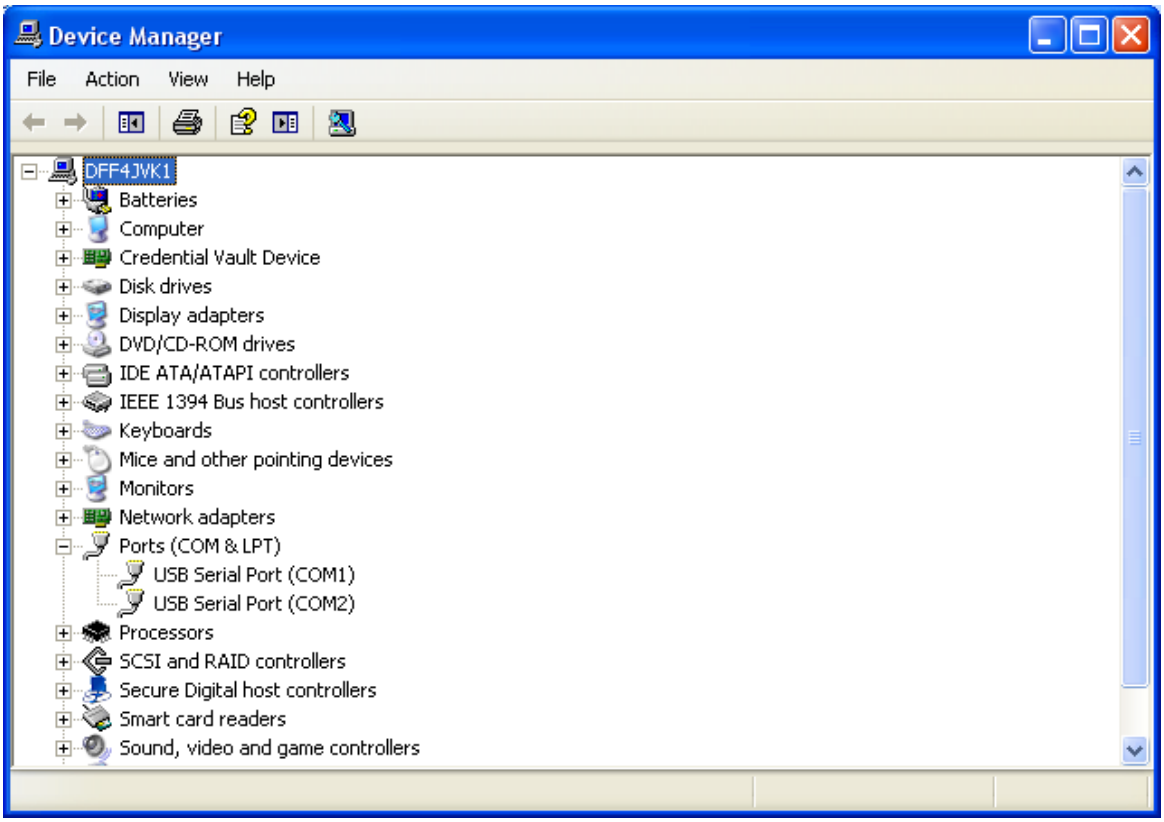
A.2 Identifying the Serial Port

When used with an eHost, the utilities connect from the NPT client to the eHost through a serial cable. Complete the following steps to identify the serial port that will be used with these utilities:

1. Connect the computer to the middle serial port (eHost CPU) on the eHost. Use a serial port on the computer or a USB cable to a serial dongle.
2. For the serial port ID, from the Start menu, launch a command shell (**Run →cmd**).
3. In the shell, type **devmgmt.msc**.



4. Click **OK**.
5. In the Device Manager window, expand **Ports**. The USB to Serial Adapter is located in **USB Serial Port (COM1)**.



6. Use the COM port identified above with the utilities. The COM port is typically specified with a –d option. For example, **config_node.py -c node_config.txt -d COM1**

Appendix B Creating RSA Keys

These instructions describe the steps necessary to create an RSA public/private key pair. RSA key pairs must be generated for secure communication between entities such as a Local Key Server or a Node Provisioning Tool client. These instructions are for Linux-based computers or Windows-based computers with a Cygwin environment. Note that RSA key generation does not need to be performed on the computer that will be using the keys.

NOTE 1: Creation of RSA keys can be performed on any computer and copied to another computer. However, care must be taken when transferring a private key. Generally, private keys should not be transferred off the target machine. However, it is acceptable to create a private key on a secure server for a target machine and then securely transfer the private key to the target machine.

NOTE 2: If Cygwin is installed on a Windows-based computer, then it is possible to use "Linux-like" commands at the Cygwin command prompt to generate the keys locally on the Windows computer.

1. Create a 2048-bit RSA private key using the following command. Note that the filename is user-defined.

   ```
   openssl genrsa -out <my_key.priv.pem> 2048
   ```

   NOTE: This command does not have the '-des3' option which creates an encrypted private key file. PyCrypto does not support encrypted private key files.

2. Create an RSA public key from the private key. Note that some filenames are user-defined.

   ```
   openssl rsa -in <my_key.priv.pem> -pubout > <my_key.pub.pem>
   ```

Be sure to follow these safeguards:

- ***Never distribute or disclose the private key. Only distribute the public key.***
- Encrypt messages using the intended recipient's public key. Decrypt received messages using the private key. The message should be encrypted by the sender using the public key.
- Sign transmitted messages using the private key. Verify/authenticate signatures using the sender's public key.

Appendix C Creating SSL Certificates

This appendix describes the steps necessary to create signed SSL Authentication Certificates. It is necessary to create a signed certificate for the Local Key Server (LKS) as well as for each client running the eNode Key Provisioning Utility. Note that certificate generation and signing does not need to be performed on the machine that will be using the signed certificate. These instructions are for Linux-based computers or Windows-based computers with a Cygwin environment.

NOTE 1: It is recommended that all certificate signing be performed on a secure server. Creation of signed SSL Authentication Certificates is typically done by the IT department on a separate and secure server. Under no circumstances should the CA private key be left on an unsecured machine (such as the NPT).

NOTE 2: If Cygwin is installed on a Windows-based computer, then it is possible to use "Linux-like" commands at the Cygwin command prompt.

The Certificate Authority (CA) is the entity that signs certificate requests to generate certificates. The CA can be an actual CA (such as VeriSign, etc.) or the LKS owner can act as its own CA. Unlike the LKS servers or NPT clients, which must have certificates with Common Name fields that match their respective IP addresses or hostnames, the CA does not have this restriction. Any secure computer can be used as the CA to sign certificates. However, the CA certificate should ***not*** use the same Common Name as any of the LKS server or NPT client certificates it is signing.

C.1 Generating a CA Certificate

On a secure server configured to act as a certificate authority, perform the following steps:

1. An RSA private/public key pair is needed to create a CA certificate. If an existing RSA key pair is not already available for the CA, follow the instructions in Appendix B: Creating RSA Keys to generate the keys.
2. Save the keys to an unencrypted PEM file format (for example, 'ca_key.priv.pem' and 'ca_key.pub.pem'). A 2048-bit key length should be sufficient.

NOTE: The CA private key must be kept secure at all times. It should never be copied to an unsecure computer. Ideally, certificate signing should be performed on a dedicated security server.

3. Generate the CA certificate using the following command. Note that some filenames are user-defined.

```
openssl req -new -x509 -days <365> -key <ca_key.priv.pem> -out
<ca_cert.crt>
```

4. You are prompted for information regarding the Certificate Authority. Enter the requested information as appropriate. Examples are provided in the following list.

NOTE: Be sure to use a different Common Name for the CA certificate than that used by any of the other certificates it signs.

```
Common Name = <IP Address or Qualified Domain Name>
```

Unlike the NPT certificate, the Common Name for the CA certificate ***does not*** need to be an actual IP address or fully qualified domain name of a particular computer. An arbitrary common name can be used for the CA certificate, such as 'cert_authority'.

- Country Name (2 letter code) [AU]:US
- State or Province Name (full name) [Some-State]:California
- Locality Name (for example, city) []:San Diego
- Organization Name (for example, company) [Internet Widgits Pty Ltd]:OnRamp Wireless
- Organizational Unit Name (for example, section) []:OnRamp
- Common Name (for example, YOUR name) []:certificate-authority.onramp.local
- Email Address []:support@onrampwireless.com

C.2 Generating a Certificate Signing Request for the NPT Client

1. Select an RSA key pair for the NPT client. If an existing key pair for the NPT client is not available, follow the instructions in Appendix B: Creating RSA Keys to generate a new set of keys.
2. Save the keys to an unencrypted PEM file format (for example, 'npt_key.priv.pem' and 'npt_key.pub.pem'). Be sure not to use a password protected private key file, otherwise, the password will have to be entered repeatedly.
3. Using the NPT client private key, generate a Certificate Signing Request (CSR) using OpenSSL at a Cygwin command prompt as follows. Note that some filenames are user-defined.

   ```
   openssl req -new -key <npt_key.priv.pem> -out <npt_cert.csr>
   ```

   NOTE: If Cygwin is not available on the Windows computer, then the certificate signing request (CSR) can be generated on another computer. However, this will also require transferring the NPT private key to the other computer in order to generate the CSR. Therefore, if the CSR cannot be generated on the NPT client itself, then it should only be done on another secure machine. The NPT private key should then be deleted from the remote secure machine.
4. You are prompted for several pieces of information. Respond as appropriate. Examples are provided in the following list.

   NOTE: It is important that the Common Name field be filled in with the fully qualified domain name (or the IP address) of the NPT client. For example, IP Address: 10.50.4.6 or FQDN: LKS_Server.onrampwireless.com.

```
Common Name = <IP Address or Qualified Domain Name>
```

It is also important that this Common Name be different from that of the Certificate Authority. It is not necessary to enter anything for the last two prompts (that is, challenge password and optional company name).

- Country Name (2 letter code) [AU]:US
- State or Province Name (full name) [Some-State]:California
- Locality Name (for example, city) []:San Diego
- Organization Name (for example, company) [Internet Widgits Pty Ltd]:OnRamp Wireless
- Organizational Unit Name (for example, section) []:OnRamp
- Common Name (for example, YOUR name) []:<domain name or IP address of NPT client>
- Email Address []:support@onrampwireless.com
- A challenge password []:<leave blank>
- An optional company name []:<leave blank>

5. After the CSR file has been created, transfer the CSR file to a secure server acting as a certificate authority that has the CA private key and certificate.
6. On the secure server, sign the NPT client's CSR using the CA's certificate and private key to generate the NPT client's certificate file (npt_cert.crt). Note that the following example should be typed all on one line and that some of the filenames are user-defined.

   Example:

   ```
   openssl x509 -req -days <365> -in <npt_cert.csr> -CA <ca_cert.crt> -
   CAkey <ca_key.priv.pem> -set_serial <0001> -out <npt_cert.crt>
   ```

   NOTE 1: The 365-day value was selected arbitrarily. Longer or shorter periods can be used.

   NOTE 2: The serial number 0001 was chosen arbitrarily. However, if the same serial number is used again, it can create issues with clients that have cached the server certificate information. As a precaution, it is recommended that the serial number be rolled every time a new server certificate is generated.
7. Transfer the CA certificate (but not the keys) and the NPT client certificate back to the target NPT client machine.
8. On the NPT client machine, copy the CA certificate, the NPT client certificate, and both NPT client keys to the appropriate location on the NPT client. Typically this is the same directory as the eNode Key Provisioning Utility.

Appendix D Abbreviations and Terms

Abbreviation/Term	Definition
AP	Access Point
CA	Certificate Authority
CDLD	Code Download
CSR	Certificate Signing Request
DHCP	Dynamic Host Configuration Protocol
DOS	Disk Operating System
DNS	Domain Name System
eNode	End point device. An eNode is commonly referred to as Node.
KMS	Key Manager Server
LKS	Local Key Server
Node	The generic term used interchangeably with eNode.
NPT	Node Provisioning Tools. A suite of utilities for setting up, configuring, and provisioning eNodes. The suite consists of: ■ eNode Software Upgrade Utility (sw_upgrade.py) This utility is used for upgrading the eNode firmware. ■ eNode Flash Configuration Utility (config_node.py) This utility is used for programming eNode flash configuration parameters. ■ eNode Key Provisioning Utility (provision_node_keys.py) This utility is used for programming eNode OTA security keys.
OTA	Over-the-Air
RSA	Rivest, Shamir, Adleman encryption algorithm
SSL	Secure Socket Layer
TCP	Transmission Control Protocol