SmartMesh WirelessHART Tools Guide

Table of Contents

1 About This Guide

1.1 Related Documents

The following documents are available for the SmartMesh WirelessHART network:

Getting Started with a Starter Kit

- SmartMesh WirelessHART Easy Start Guide - walks you through basic installation and a few tests to make sure your network is working
- SmartMesh WirelessHART Tools Guide - the Installation section contains instructions for the installing the serial drivers and example programs used in the Easy Start Guide and other tutorials.

User Guide

- SmartMesh WirelessHART User's Guide - describes network concepts, and discusses how to drive mote and manager APIs to perform specific tasks, e.g. to send data or collect statistics. This document provides context for the API guides.

Interfaces for Interaction with a Device

- SmartMesh WirelessHART Manager CLI Guide - used for human interaction with a Manager (e.g. during development of a client, or for troubleshooting). This document covers connecting to the CLI and its command set.
- SmartMesh WirelessHART Manager API Guide - used for programmatic interaction with a manager. This document covers connecting to the API and its command set.
- SmartMesh WirelessHART Mote CLI Guide - used for human interaction with a mote (e.g. during development of a sensor applicaition, or for troubleshooting). This document covers connecting to the CLI and its command set.
- SmartMesh WirelessHART Mote API Guide - used for programmatic interaction with a mote. This document covers connecting to the API and its command set.

Software Development Tools

- SmartMesh WirelessHART Tools Guide - describes the various evaluation and development support tools included in the SmartMesh SDK including tools for exercising mote and manager APIs and visualizing the network.

Application Notes

- SmartMesh WirelessHART Application Notes - app notes covering a wide range of topics specific to SmartMesh WirelessHART networks and topics that apply to SmartMesh networks in general.

Documents Useful When Starting a New Design

- The Datasheet for the LTC5800-WHM SoC, or one of the castellated modules based on it, or the backwards compatible LTP5900 22-pin module.
- The Datasheet for the LTP5903-WHR embedded manager.
- A Hardware Integration Guide for the mote SoC or castellated module, or the 22-pin module - this discusses best practices for integrating the SoC or module into your design.
- A Hardware Integration Guide for the embedded manager - this discusses best practices for integrating the embedded manager into your design.
- A Board Specific Integration Guide - For SoC motes and Managers. Discusses how to set default IO configuration and crystal calibration information via a "fuse table".
- Hardware Integration Application Notes - contains an SoC design checklist, antenna selection guide, etc.
- The ESP Programmer Guide - a guide to the DC9010 Programmer Board and ESP software used to program firmware on a device.
- ESP software - used to program firmware images onto a mote or module.
- Fuse Table software - used to construct the fuse table as discussed in the Board Specific Integration Guide.

Other Useful Documents

- A glossary of wireless networking terms used in SmartMesh documentation can be found in the SmartMesh WirelessHART User's Guide.
- A list of Frequently Asked Questions

1.2 Conventions Used

The following conventions are used in this document:

`Computer type` indicates information that you enter, such as specifying a URL.

Bold type indicates buttons, fields, menu commands, and device states and modes.

Italic type is used to introduce a new term, and to refer to APIs and their parameters.

Tips provide useful information about the product.

Informational text provides additional information for background and context

Notes provide more detailed information about concepts.

Warning! Warnings advise you about actions that may cause loss of data, physical harm to the hardware or your person.

```
code blocks display examples of code
```

1.3 Revision History

Revision	Date	Description
1	07/16/2012	Initial release
2	08/28/2012	Added Admin Toolset Guide
3	03/18/2013	Numerous small changes
4	10/28/2014	Minor changes

2 Introduction

This document covers the installation and use of various tools available to interact with a SmartMesh WirelessHART network. The relationship of these software tools is shown in figure 1. At the lowest level are the FTDI drivers, which allow your computer to interact with the API and CLI of the Mote over a USB-to-serial link. Next, the user can interact with the CLI of the Mote and Manager via a serial terminal client. Several tools make use of the Manager XML API:

- The Admin Toolset (which resides on the Manager) allows a user to visualize and configure the network using a web-browser.
- The SmartMesh SDK is a Python-based set of tools used to demonstrate various aspects of the Mote and Manger APIs.

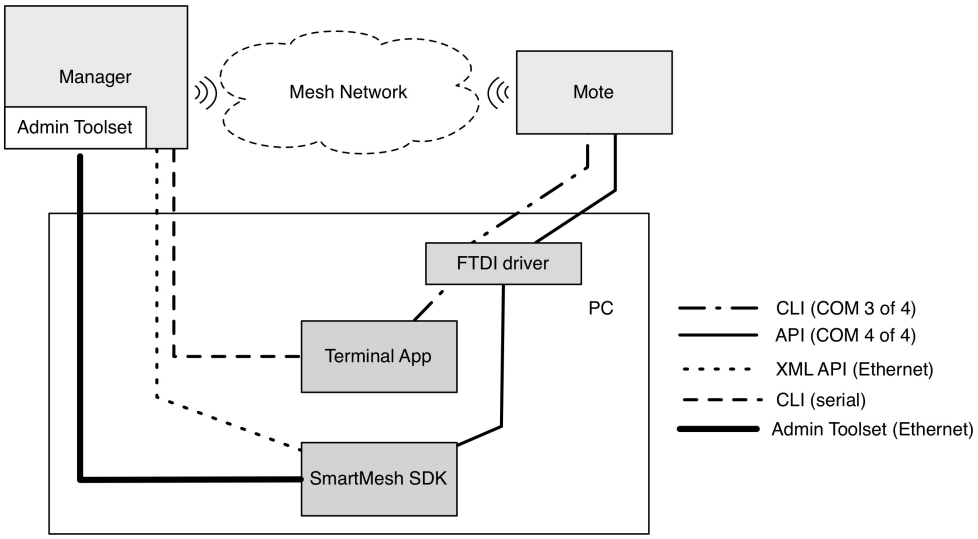


Figure 1 - software tools for interacting with a SmartMesh WirelessHART network.

3 Installation

3.1 Before You Begin

3.1.1 What You Need

The following hardware elements are required:

1	Computer running Microsoft Windows, with one available USB port
1	Ethernet cable
1	DB9 serial cable (if your computer does not have a DB9 port, you will need to use a USB-serial adapter)
1	SmartMesh WirelessHART Manager
1	DC9006
1	DC9003A-C programmed to function as a SmartMesh WirelessHART Mote
1	USB cable to connect your computer to the DC9006

The following software elements are required:

name	description	installation guide
SmartMesh SDK	Software Development Kit used to interact with the device's Application Programming Interface (API).	SmartMesh WirelessHART Tools Guide

3.2 Setup

3.2.1 System Overview

Figure 1 illustrates the components of an eval/dev kit. The kit contains a manager and 5 motes. In the Interacting With a Network section, you will join a mote to the manager, and send messages between them.

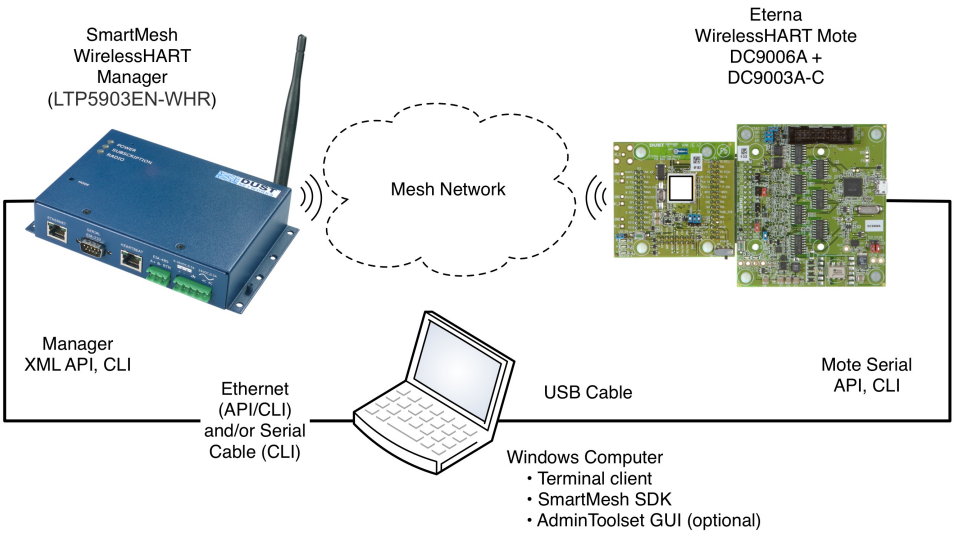


Figure 1 - Eval kit components

Before you can send data, you will need to install several pieces of software and verify that the software can connect to your mote and manager by following these steps:

- Step 1: Prepare the Hardware
- Step 2: Install software on the PC
  - a: FTDI Serial Drivers for the DC9006A board
  - b: SmartMesh SDK and Python

You will then move to the Interacting With a Network section and:

1. Use APIExplorer to establish a PC to Manager (LTP5903CEN-WHR) connection
2. Use APIExplorer to stablish a PC to Mote (DC9003A-C) connection
3. Have the Mote join the Manager over the air
4. Send messages between the Manager and Mote and vice versa

3.2.2 Step 1 - Prepare the Hardware

Manager

By default, LTP5903CEN-WHR manager ships with a static address (192.168.99.100) on the Ethernet port - this is likely not a suitable IP adress for your local network.

1. Connect a DB9 serial cable between your computer and the LTP5903CEN-WHR serial port (marked "serial 2").
2. Connect an Ethernet cable between the manager and your local network.
3. Power on the manager.
4. Launch a terminal client (Hyperterminal is built into Windows - you can install another if you prefer) - the settings are 115200 baud, 8 data bits, No parity, 1 stop bit, no flow control. Access the Linux login prompt by entering the following username and password:

Username: `dust`
Password: `dust`

In order to use your manager with a DHCP server-assigned address, use the `ifswitch-to-dhcp` command:

```
dust@manager:~$ sudo ifswitch-to-dhcp
Switching interface to DHCP... eth0 down interfaces modified setting ethernet options: speed=100
duplex=full
ADDRCONF(NETDEV_UP): eth0: link is not ready
udhcpc (v1.13.2) started
Sending discover...
Sending discover...
Sending discover...
No lease, forking to background
eth0 up Done!
```

Now note the IP address assigned to eth0 using `ifconfig` - this is the address you will use to connect to the manager with the various SmartMesh SDK python tools.

```
dust@manager:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:17:0D:80:10:5B
          inet addr:172.16.1.109  Bcast:172.16.1.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:1 dropped:0 overruns:0 frame:0
          TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:90 (90.0 B)
          Interrupt:21 Base address:0x4000
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1294160 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1294160 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:55701502 (53.1 MiB)  TX bytes:55701502 (53.1 MiB)
```

In order to use your manager with a static IP address (not typical), use the `ifswitch-to-static` command, You will need to have been provided with the IP address, gateway address, and netmask by your network administrator or IT guy.

```
dust@manager:~$ ifswitch-to-static
usage: ifswitch-to-static <ip address> [gateway address] [netmask] [dns server]
dust@manager:~$
dust@manager:~$ sudo ifswitch-to-static 172.16.1.103
Switching interface to static IP allocation... ifdown: interface eth0 not configured
eth0 down interfaces modified setting ethernet options: speed=100 duplex=full
ADDRCONF(NETDEV_UP): eth0: link is not ready
eth0 up Done!
```

Mote

In order to communicate with the DC9003A-C mote, you will need to connect a DC9006 interface board to each using the board-to-board connector, as shown in the figure:

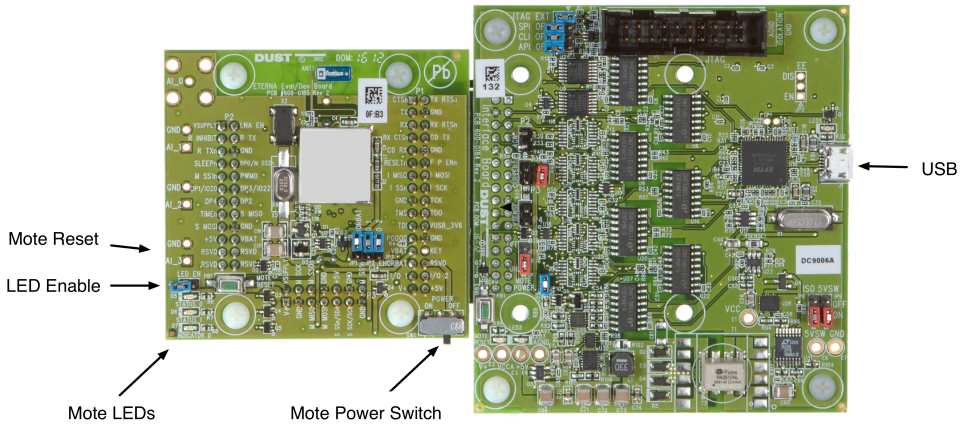


Figure 1 - DC9003 board (left) connected to DC9006 board (right)

- Turn the slide switch marked "Power" on the DC9003 board to ON.
- Connect a micro-USB cable that shipped with your kit to each DC9006. Do not connect them to your computer until you have installed the serial drivers.

> ⚠ When connected to a DC9006A board and a computer, the Mote may appear to be operating in spite of the power switch being off. The 4 COM ports will appear but you will not be able to communicate with the mote reliably. Make sure that the power switch on the mote is set to on to ensure proper operation.

3.2.3 Step 2a - Installing FTDI Serial Drivers

Installing FTDI Serial Drivers

Driver installation has two steps:

- Download FTDI driver software
- Connect a mote (DC9006 + DC9003A-C) and run through driver setup

Devices communicate with your computer using a serial connection via USB. When you connect the device to your computer, you should be asked to install a driver for it. Because the device uses a serial chipset from Future Technology Devices International (FTDI) which is found in many different devices, it is possible that you already have a version of the FTDI drivers installed on your machine.

If you don't have the drivers installed, download the version appropriate for your operating system from http://www.ftdichip.com/Drivers/VCP.htm. We recommend you download the driver files on your computer's desktop.

Once you have installed the driver, use the same USB port each time you reconnect the device to the computer. If you connect to a different USB port, you will need to repeat the following procedure for that port.

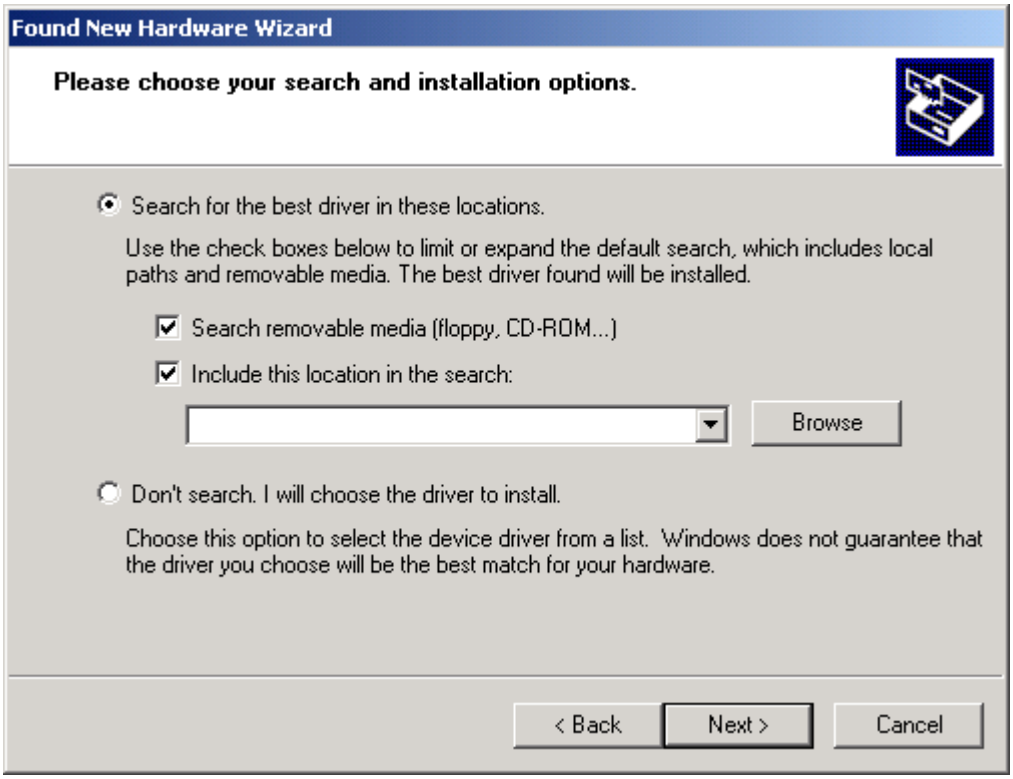
Windows Driver Installation

On Windows, follow the steps below to finalize the installation.

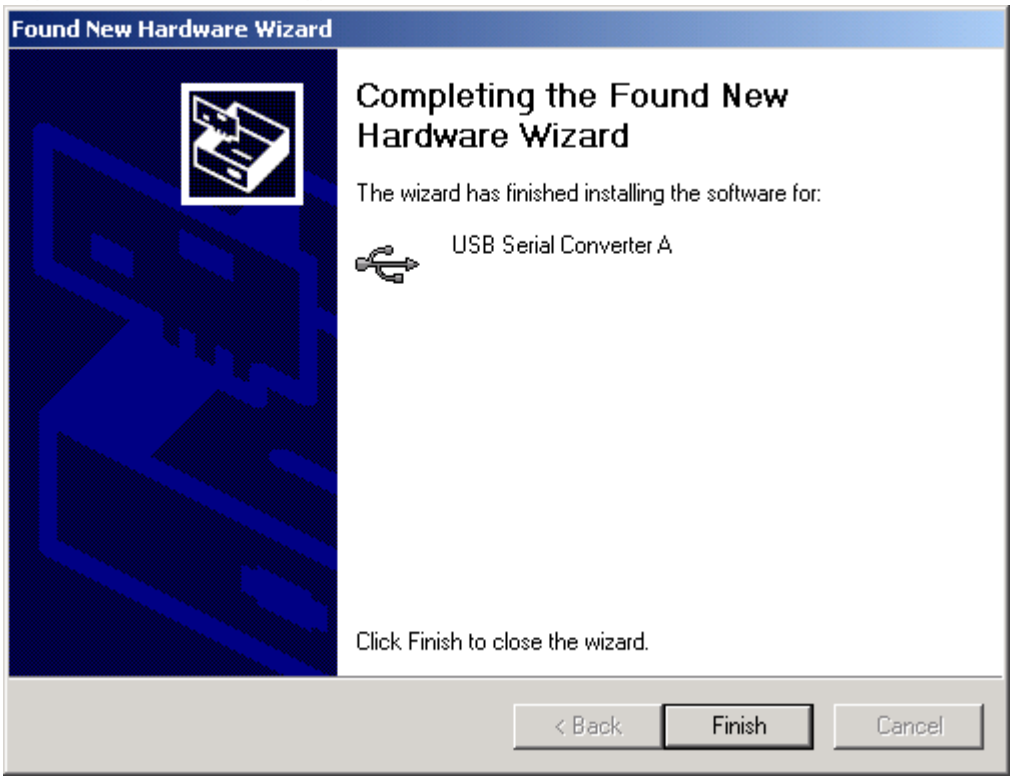
1. Connect the USB cable between the device (manager or mote) and your computer. If the Found New Hardware Wizard appears, go to step 2.
   If the Found New Hardware Wizard does not appear, do the following:
   1. Ensure that the port is functional, and that the device is connected correctly. If the Wizard still does not appear, open the Windows Device Manager to see how Windows has recognized the device.
   2. If a device named “Dust Interface Board” is listed as an unknown device (yellow icon), right-click the device and select **Update Driver**. This displays the Found New Hardware Wizard.
   3. Go to step 2.

2. In the Wizard, click the option to **Install from a list or specific location** and click **Next.**

3. Select the box to **Include this location in the search**. Then, use the **Browse** button to navigate to your desktop, and click **Next**.

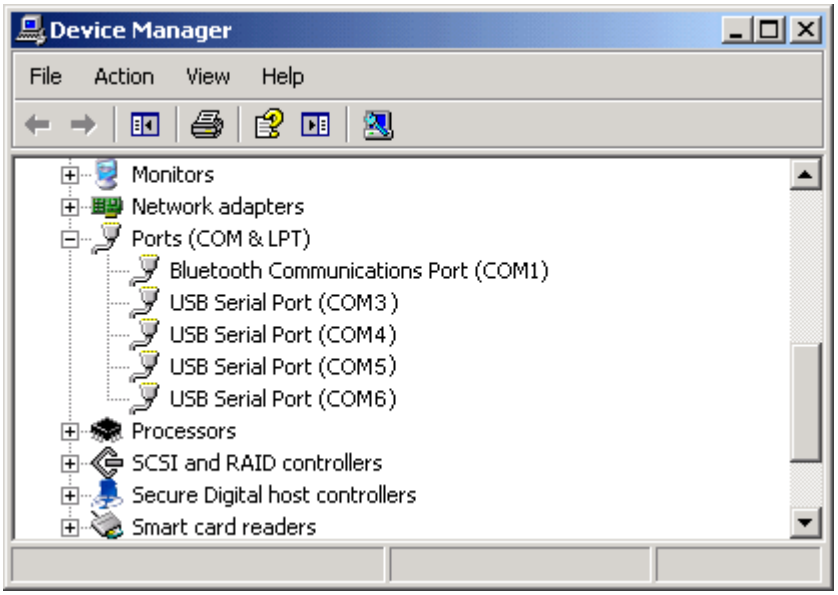


4. After the Wizard installs the software, click **Finish**.
5. When the Found New Hardware Wizard reappears, repeat steps 2 through 4 to continue the installation. Repeat these steps each time the Wizard appears.

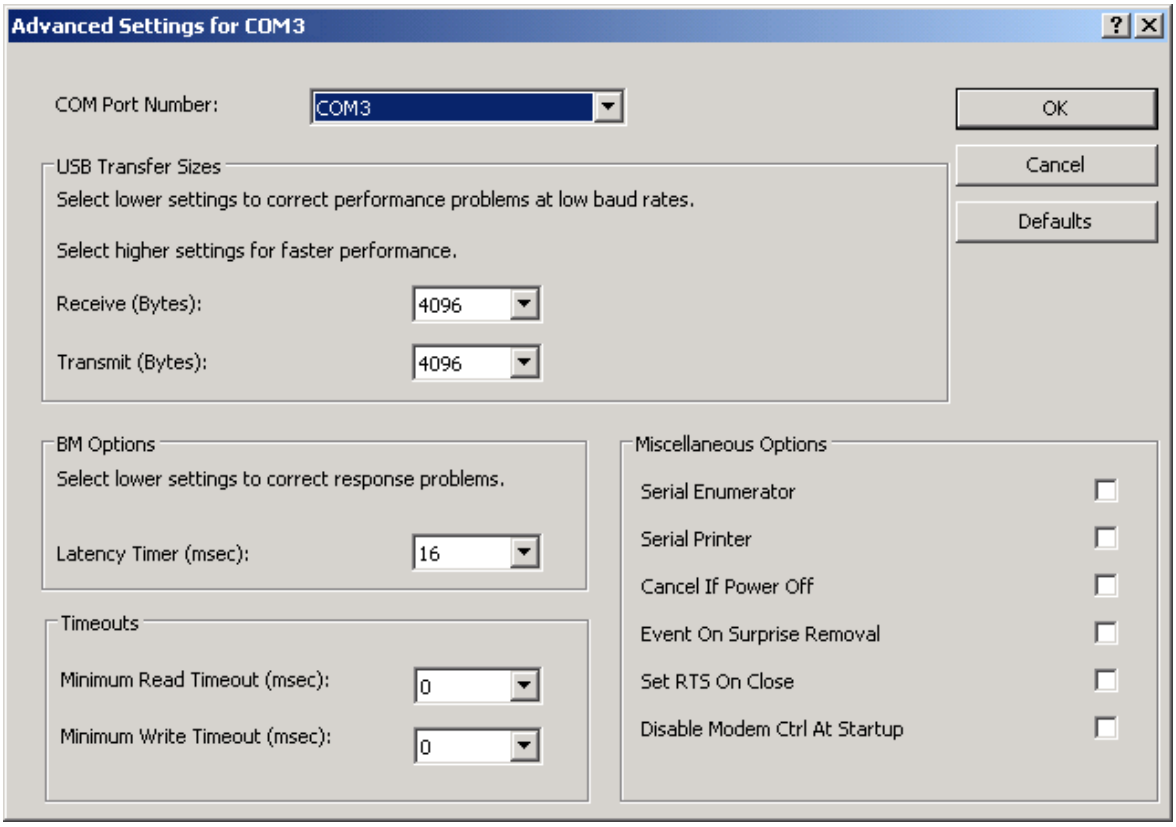


> Because of the way Windows works, you may be prompted to go through the Wizard up to eight times to complete the installation and mapping of the USB port. The manager will install a total of four virtual serial ports, along with the USB devices to control them.

6. When the installation and mapping of the USB ports is complete, open the Device Manager to find out the COM port numbers that have been assigned to the virtual serial ports.
    1. Choose the **Control Panel** from the Start menu.
    2. Open the `System` folder.
    3. Click the **Hardware** tab and click Device Manager.
    4. Open **Ports** to see the COM ports.
       You should see four new COM ports in the Device Manager.
    5. Make a note of the four COM port numbers.



7. Configure the following **Advanced Settings** for each of the four new COM ports:
    1. Right-click a COM port and click **Properties**.
    2. Click the **Port Settings** tab, and then click **Advanced**.
    3. Deselect the **Serial Enumerator** option, and click **OK**.
    4. Click **OK** to return to the Device Manager.
    5. Repeat this step for each of the four new COM ports. When you are finished, close the Device Manager.



> From now on, mark the physical USB port you use for the mote and always use this port for plugging in the mote. Doing this will ensure the COM port assignment will be preserved.

Function of Serial Ports

After installing the DC9006 (the SmartMesh WirelessHART Mote), four serial ports are created on your computer. You can interact with it over two different serial ports:

- The Command Line Interface (CLI) serial port. Use a third-party serial terminal software to connect to it and type in the commands detailed in the SmartMesh WirelessHART Mote CLI Guide
- The Application Programming Interface (API) serial port. Use SmartMeshSDK-based applications to connect to it and exercise the commands detailed in the SmartMesh WirelessHART Mote API Guide

The table below indicates the mapping of the different serial ports, and their settings. For example, suppose the installation created ports COM17, COM18, COM19 and COM20. In the table below, the third port is COM19 and the fourth is COM20.

device	serial port number	usage	baudrate	data bits	parity	stop bits
SmartMesh WirelessHART Mote	third*	CLI	9600	8	N	1
	fourth*	API	115200**	8**	N**	1**

*: refers to the serial ports created by the FTDI drivers.

**: default values.

We recommend that you write down the number of the API and CLI ports of the SmartMesh WirelessHART Mote connected to your computer. We will refer to those numbers throughout the evaluation guide.

It has been observed in some installations under Windows 7 that the serial ports do not enumerate in order, and the CLI and API ports may not be the 3rd and 4th ports, respectively. If this occurs, you will need to test each port using APIExplorer (in the SmartMesh SDK) to find the API port, and use a terminal program to find the CLI port.

Terminal Client

Hyperterminal is the default serial client on Windows XP, and it can be used to communicate with the manager and mote CLI. You can install another terminal client if you prefer or if one was not included in your Windows installation.

3.2.4 Step 2b - Installing the SmartMesh SDK

Overview

The SmartMesh SDK is a Python-based set of tools used to demonstrate various aspects of the Mote and Manger APIs. Basic installation allows you to run all the sample applications as Windows executables. Advanced installation instructions for Python are intended for developers who will modify the example applications.

Installation

Download the SmartMeshSDK

Find the download in the Dust Networks area of the Linear Design Tools site. The SmartMeshSDK download link is in the "Software Utilities" section. This will download the latest version of SmartMeshSDK.

The complete SmartMeshSDK is contained in the file `SmartMeshSDK-full-X.X.X.X.zip`.

- Download latest rev of SmartMeshSDK zip file `SmartMeshSDK-full-X.X.X.X.zip`.
- Unzip the file. A folder by the same name will be created with 4 sub folders `-/api`, `/doc`, `/src` and `/win`.
    - Standalone applications are found in `/win`.
- No other installation is required. You may move the `SmartMeshSDK-X.X.X.X/` folder anywhere convenient.

Detailed instructions on the contents and use of the SmartMesh SDK can be found in the SmartMesh IP Tools Guide or the SmartMesh WirelessHART Tools Guide.

Advanced Instructions for Developers

Installation for Running from Source Code

To run the SmartMeshSDK sample applications directly from source code you need to install Python and some additional libraries. This is only necessary if you wish to modify the behavior of the sample apps. This process is also required for running the applications on systems that cannot directly run Windows executables.

Python installation instructions are different for 32-bit and 64-bit versions of Windows. Please follow the appropriate set of instructions below.

Installation on Windows XP or 32-bit Windows 7

Install Python 2.7

1. Download the latest Python 2.7 Windows Installer from http://python.org/download/.
   At the time of writing, the latest Windows Installer was `python-2.7.2.msi`.
2. Double click on the installer and follow the installation steps using all default settings.

Install PySerial

PySerial adds serial port support to Python.

1. Download PySerial for Python 2.7 from http://sourceforge.net/projects/pyserial/files/pyserial/.
   At the time of writing, the latest version was pyserial-2.5.

   > ⚠ Make sure to install PySerial for Python 2.7, e.g. `pyserial-2.5.win32.exe`.
   > Do not install PySerial for Python 3.0, e.g. `pyserial-py3k-2.5.win32.exe`.

2. Double click on the installer and follow the installation steps using all default settings.

Install PyWin32

This step is optional. It's not necessary unless you want to run the Serial Mux Configurator from source. The Serial Mux Configurator can be run from its executable form.

PyWin32 adds Windows-specific support to Python. While most applications in the SmartMeshSDK are platform independent, some like the Serial Mux Configurator have platform-specific requirements. PyWin32 provides different installers for various Python versions

1. Download the pywin32 installer. On the pywin32 downloads page (http://sourceforge.net/projects/pywin32/files/pywin32/), find the appropriate pywin32 installer based on your Python version (2.7) and whether you are running 32-bit or 64-bit Python. At the time of this writing, the latest 32-bit build for Python 2.7 was pywin32 build 217.
2. Double click on the installer and follow the installation steps using all default settings.

Installation on 64-bit Windows 7

Unless you know that you need to run the 64-bit version of Python and PySerial for some other purpose, you should run the 32-bit version. Follow the instructions above.

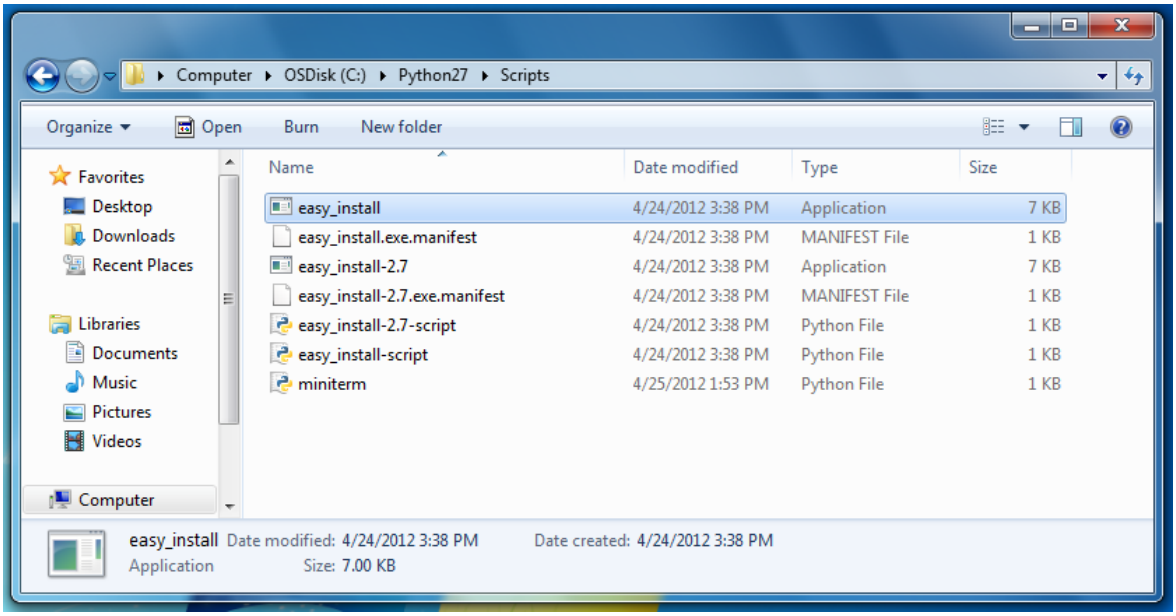
Install Python 2.7

1. Download the latest Python 2.7 Windows X86-64 Installer from http://python.org/download/.
   At the time of writing, the latest Windows Installer was `python-2.7.2.amd64.msi`.
2. Double click on the installer and follow the installation steps using all default settings.

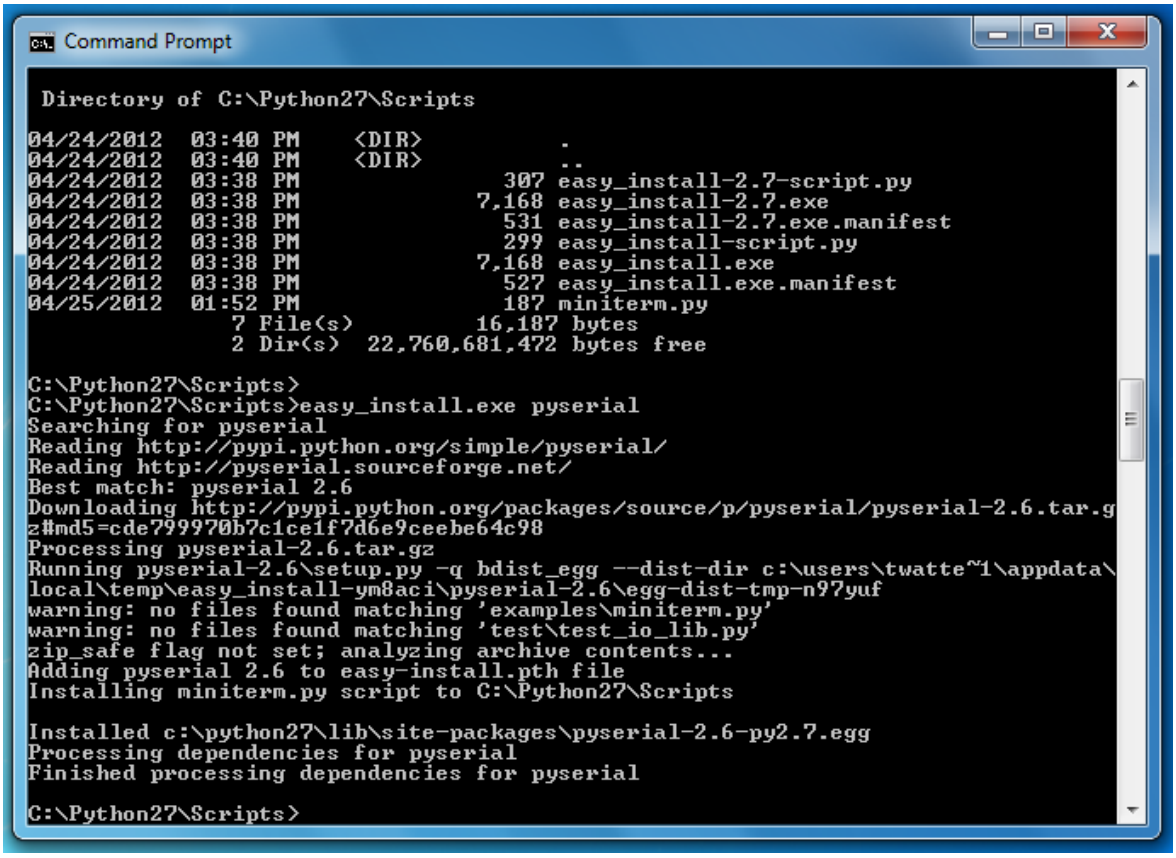
Install PySerial

Installing Pyserial is done differently for 64-bit Windows 7. It is a two step process:

1. Install setuptools from http://pypi.python.org/pypi/setuptools. At the time of writing, setuptools 0.6c11 is the latest version.
   1. Go to the Installation section for Windows and download `ez_setup.py` from the link in the instructions there.
   2. Run the `ez_setup.py` file that you downloaded, by double-clicking on the script icon. This should result in the creation of a folder named Scripts in your Python folder.



2. Install PySerial using the `easy_install.exe` script from the step above. The `easy_install.exe` script should be in your Python folder, e.g., C:\Python27\Scripts. (This folder might be a different place if you specified a different location when you installed Python.)
    1. Open a Windows Command Prompt. Navigate to the C:\Python27\Scripts folder. (How to get to the command prompt: http://www.sevenforums.com/tutorials/947-command-prompt.html)
    2. Run `easy_install.exe` and supply "pyserial" as a command line argument, e.g. `easy_install.exe pyserial`



Install PyWin32

This step is optional. It's not necessary unless you want to run the Serial Mux Configurator from source. The Serial Mux Configurator can be run from its executable form.

PyWin32 adds Windows-specific support to Python. While most applications in the SmartMeshSDK are platform independent, some like the Serial Mux Configurator have platform-specific requirements. PyWin32 provides different installers for various Python versions

1. Download the pywin32 installer. On the pywin32 downloads page (http://sourceforge.net/projects/pywin32/files/pywin32/), find the appropriate pywin32 installer based on your Python version (2.7) and whether you are running 32-bit or 64-bit Python. At the time of this writing, the latest 64-bit build for Python 2.7 was pywin32 build 217.
2. Run the installer.

Installation on Linux (Ubuntu)

Install Python 2.7

Python 2.7 is part of the standard Ubuntu distribution. If python is not present, you can install it with the Ubuntu package manager.

```
$ sudo apt-get install python2.7
```

Install PySerial

PySerial can be installed with the Ubuntu package manager.

```
$ sudo apt-get install python-serial
```

Installation on Macintosh OS X

Install Python 2.7

Python 2.7 is part of the standard OS distribution. If python is not present, you can install use a package manager like macports to install it

```
$ sudo port install python27
```

Install PySerial

PySerial can be obtained here. Follow the installation instructions in the /documentation/pyserial.rst file.

Test your Installation

1. Navigate to the `/src/` directory, then to the `bin/InstallTest` directory
2. Double click on `InstallTest.py`

3. If your installation is complete, a Python command window will open with the following text

```
Installation test script - Dust Networks SmartMeshSDK


Step 1. Python version
You are running Python 2.7.1
PASS

Step 2. PySerial installation
PASS

Press Enter to exit.
```

4. Make sure all tests contain the word PASS.

3.3 Troubleshooting

3.3.1 Linux FTDI Driver Installation

This section provides troubleshooting tips for some common Linux distributions.

Not all components of the SmartMesh SDK have been tested with Linux. Proceed at your own risk.

Ubuntu FTDI Driver Installation

On recent Ubuntu releases (12.04 and later), the FTDI drivers are included in the standard release. The kernel should load the FTDI drivers when the device is connected.

```
$ dmesg | grep FTDI
ftdi_sio 1-1:1.0: FTDI USB Serial Device converter detected
usb 1-1: FTDI USB Serial Device converter now attached to ttyUSB0
ftdi_sio 1-1:1.1: FTDI USB Serial Device converter detected
usb 1-1: FTDI USB Serial Device converter now attached to ttyUSB1
ftdi_sio 1-1:1.2: FTDI USB Serial Device converter detected
usb 1-1: FTDI USB Serial Device converter now attached to ttyUSB2
ftdi_sio 1-1:1.3: FTDI USB Serial Device converter detected
usb 1-1: FTDI USB Serial Device converter now attached to ttyUSB3
```

Based on the output above, the serial port to use to communicate with the device's API is `/dev/ttyUSB3`.

In order to access this device as an unprivileged user (not root), you will need to fix the permissions. The following command updates the permissions on `/dev/ttyUSB2` and `/dev/ttyUSB3`:

```
$ sudo chmod 666 /dev/ttyUSB[23]
```

3.3.2 Macintosh OS X FTDI Driver Installation

⚠ Not all components of the SmartMesh SDK have been tested with OS X. Proceed at your own risk.

OS X FTDI Driver Installation

On OS X 10.5, 10.6 or 10.7, install the FTDI driver from the disk image available from http://www.ftdichip.com/Drivers/VCP.htm (refer to the OS X Installation Guide).

After the drivers are installed, plug in the USB cable. The device name will be needed as input to the tools that connect to the serial port.

To determine the device name, enter the following command in Terminal.app:

```
$ ls /dev/*usbserial*
```

There should be several entries in the `/dev` directory with the format:

- `/dev/cu.usbserial-xxxxxxxx`
- `/dev/tty.usbserial-xxxxxxxx`

where `xxxxxxxx` is either the device's serial number or a location string that depends on which USB port your device is connected to. The last character (A, B, C or D) indicates the serial port. The port ending with C is the CLI port and port ending with D is the API port.

The `screen` program (provided with OS X) can be used as a serial terminal to connect to the CLI port.

```
$ screen /dev/tty.usbserial-01234567C 9600

> help
...
```

3.3.3 Not Getting Notifications

With the SmartMesh WirelessHART Manager, notifications are suppressed by default unless they are subscribed to. The *subscribe* API is documented in the SmartMesh WirelessHART Manager API Guide and can be tested using APIExplorer in the SmartMesh SDK.

3.3.4 Changing Network ID

This is only required if you operate your network in the same radio space as other SmartMesh WirelessHART networks.

The Network ID is the 16-bit identifier of your network, and defaults to 1229. Follow the steps below to change Network ID:

Manager

- Connect to the Manager CLI (via serial or SSH) - log into the Linux prompt, and launch nwconsole.
- Issue the following commands (here we'll set the Network ID to 100):

```
> set network networkId=100
netName:                myNet
networkId:              100
...
```

Mote

- Connect the device to your computer over USB
- Open the CLI port of that device using serial terminal
- Switch on your device

```
> mset netid 100
netid = 100

> reset

HART Mote 1.0.0-104
```

- Repeat for all your devices

Note that a reset of each device is required for the new Network ID to take effect.

3.3.5 Master/Slave

Mode Behavior

Motes have two modes that control joining and command termination behavior:

- **Master**, in which the mote runs an application that terminates commands and controls joining. By default all the motes in Starter kits are configured for **master** mode. The API is disabled in **master** mode.
- **Slave**, in which the mote expect a serially connected device to terminate commands and control join - by default the mote does not join a network on its own. The API is enabled in **slave** mode, and the device expects a serially attached application such as APIExplorer to connect to it. If *autojoin* is enabled via *SetParameter* (SmartMesh IP only), a **slave** mote will join the network without requiring a serial application to issue a *join* command.

The mode can be set through the CLI `set` command, and persists through reset (*i.e.* it is non-volatile).

LEDs

For motes (DC9003) in **master** mode, the STATUS_0 LED will begin blinking immediately upon power up, as the mote will start searching automatically. When the mote has joined, STATUS_0 and STATUS_1 LEDs will both be illuminated. In **slave** mode, no LEDs may light - this should not be mistaken for a dead battery.

LEDs of a DC9003 board will only light if the LED_EN jumper is shorted. Master mode LED support available in SmartMesh WirelessHART mote version >= 1.1.2.

Switching To Slave Mode

By default, motes in starter kits (DC9000 and DC9007) and are configured for **master** mode. To read the current configuration, connect the mote to a computer via a USB cable and use the `get` mote CLI command. To configure the mote for **slave** mode, use the `set` mote CLI command:

Use the `get mode` command to see the current mode:

```
> get mode
master
```

Use the `set mode` command to switch to **slave** mode:

```
> set mode slave
> reset
```

⚠ You must reset the mote for the mode change to take effect. Once set, the mode persists through reset.

Switching To Master Mode

To read the current configuration, connect the mote to a computer via a USB cable and use the `get mode` CLI command. To configure the mote for **master** mode, use the `set mode` CLI command.

Use the `get mode` command to see the current mode:

```
> get mode
slave
```

Use the `set mode` command to set the mote to **master** mode:

```
> set mode master
> reset
```

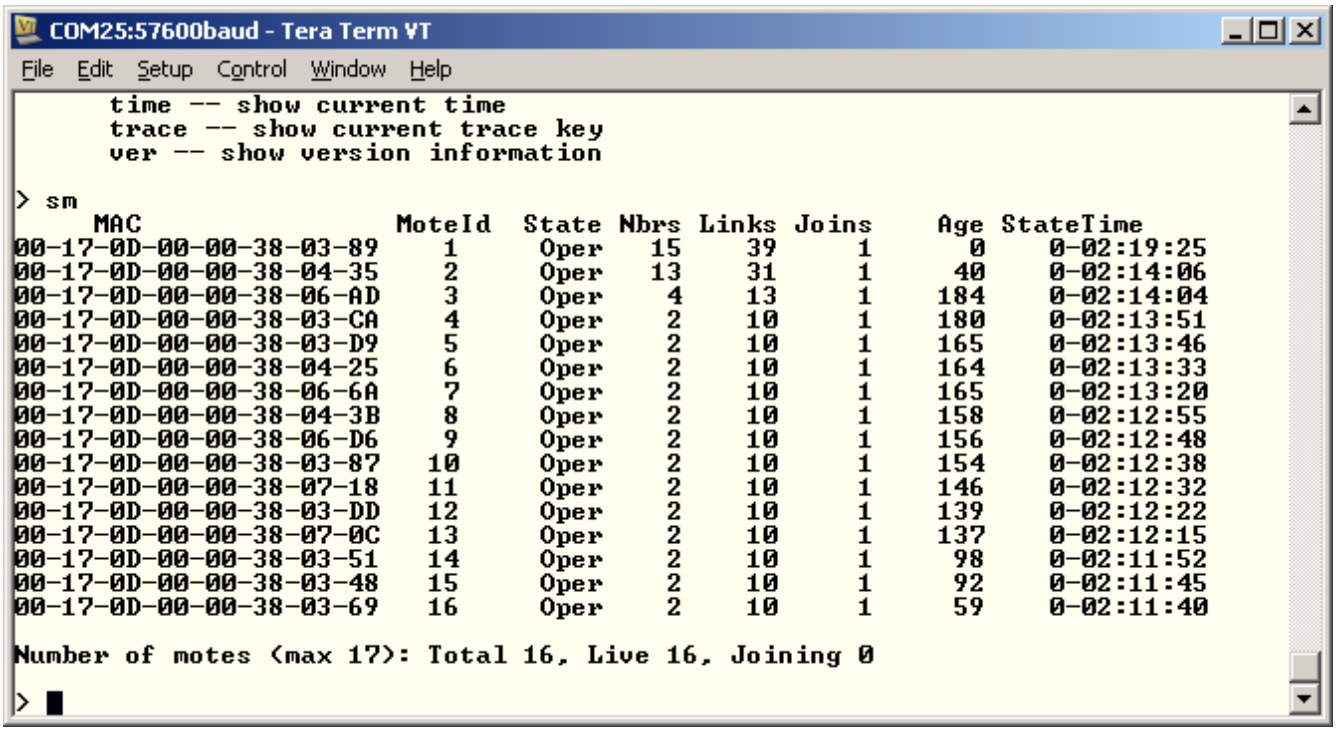
⚠ You must reset the mote for the `set mode` command to take effect. Once set, the mode persists through reset.

4 Serial Terminal Client

This page lists third-party Serial Terminal Clients which you can use to interact with a device over its command line interface (CLI). See the respective CLI guide for details on a particular mote or manager.

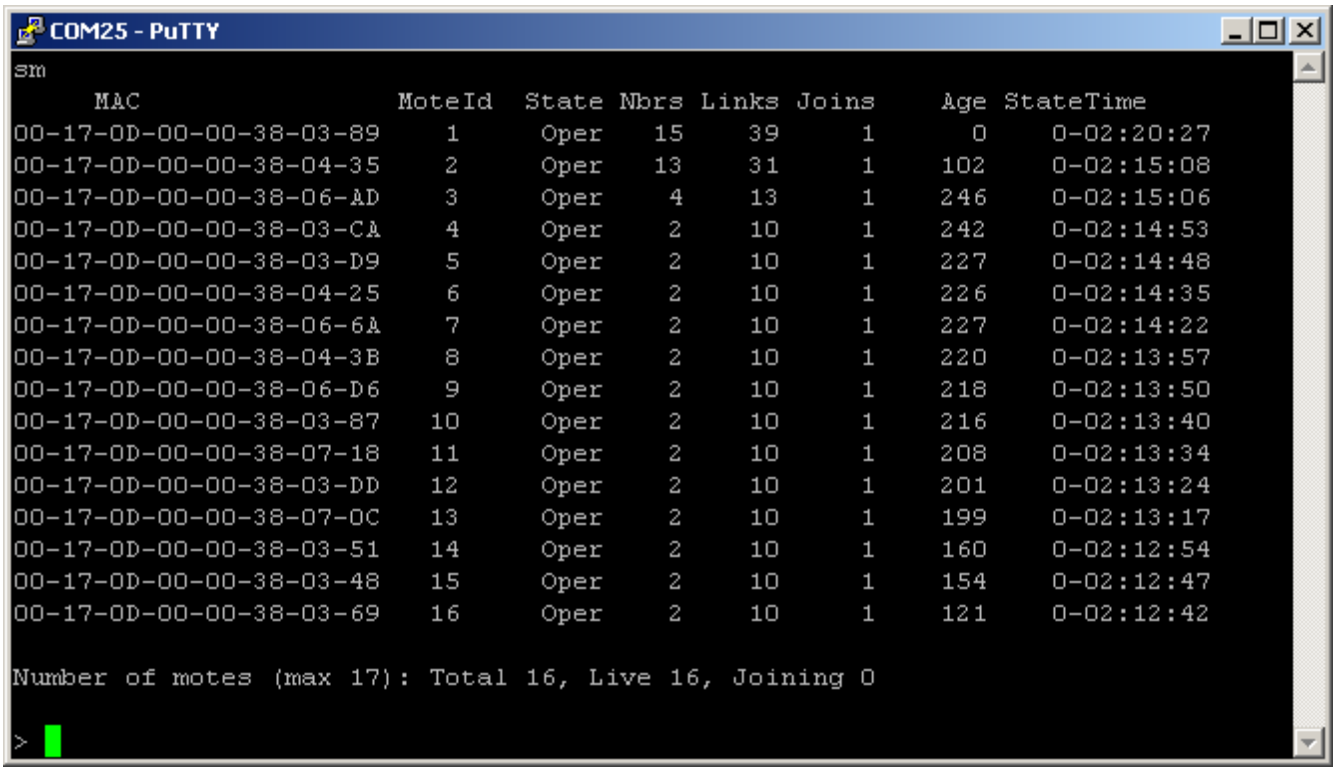
4.1 TeraTerm

- Supported platform: Windows
- Download from http://ttssh2.sourceforge.jp/index.html.en



4.2 PuTTY

- Supported platform: Windows
- Download from http://www.chiark.greenend.org.uk/~sgtatham/putty/



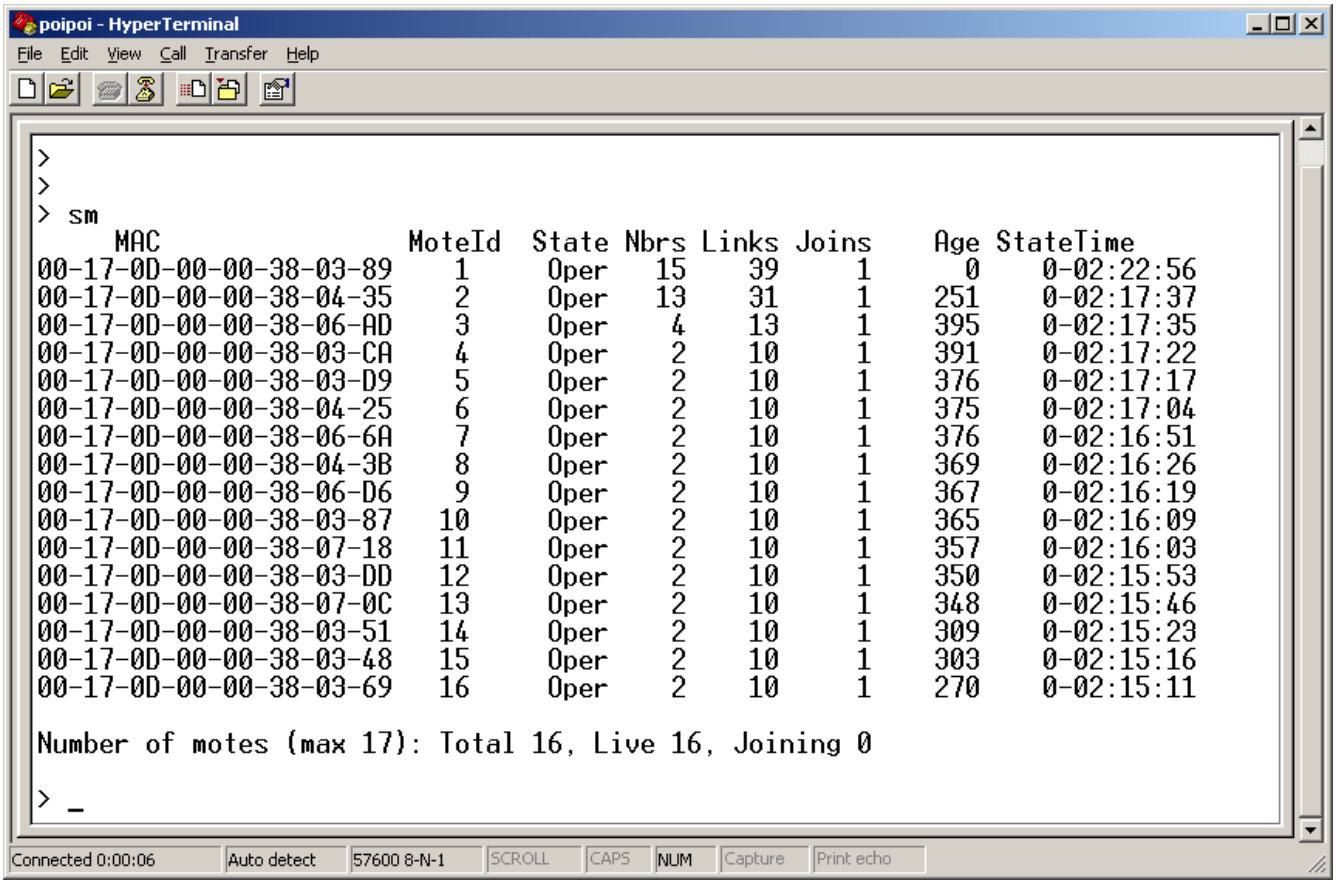
4.3 minicom

- Supported platforms: Linux, Unix
- Pre-installed in most distributions. Available through package managers such as fink or macports on OS X.
- Source: http://alioth.debian.org/projects/minicom/

4.4 Microsoft Windows HyperTerminal

- Supported platform: Windows XP
- Pre-installed in Windows XP

⚠ Not installed in Windows Vista or Windows 7


poipoi - HyperTerminal
File Edit View Call Transfer Help
> sm
MAC MoteId State Nbrs Links Joins Age StateTime
00-17-0D-00-00-38-03-89 1 Oper 15 39 1 0 0-02:22:56
00-17-0D-00-00-38-04-35 2 Oper 13 31 1 251 0-02:17:37
00-17-0D-00-00-38-06-AD 3 Oper 4 13 1 395 0-02:17:35
00-17-0D-00-00-38-03-CA 4 Oper 2 10 1 391 0-02:17:22
00-17-0D-00-00-38-03-D9 5 Oper 2 10 1 376 0-02:17:17
00-17-0D-00-00-38-04-25 6 Oper 2 10 1 375 0-02:17:04
00-17-0D-00-00-38-06-6A 7 Oper 2 10 1 376 0-02:16:51
00-17-0D-00-00-38-04-3B 8 Oper 2 10 1 369 0-02:16:26
00-17-0D-00-00-38-06-D6 9 Oper 2 10 1 367 0-02:16:19
00-17-0D-00-00-38-03-87 10 Oper 2 10 1 365 0-02:16:09
00-17-0D-00-00-38-07-18 11 Oper 2 10 1 357 0-02:16:03
00-17-0D-00-00-38-03-DD 12 Oper 2 10 1 350 0-02:15:53
00-17-0D-00-00-38-07-0C 13 Oper 2 10 1 348 0-02:15:46
00-17-0D-00-00-38-03-51 14 Oper 2 10 1 309 0-02:15:23
00-17-0D-00-00-38-03-48 15 Oper 2 10 1 303 0-02:15:16
00-17-0D-00-00-38-03-69 16 Oper 2 10 1 270 0-02:15:11
Number of motes (max 17): Total 16, Live 16, Joining 0
> _
Connected 0:00:06
Auto detect
57600 8-N-1
SCROLL
CAPS
NUM
Capture
Print echo

5 Admin Toolset

5.1 Introduction

The SmartMesh Admin Toolset is a web-based administrative utility built into the SmartMesh WirelessHART Manager that lets you perform network configuration and management tasks remotely.

Using Admin Toolset, you can examine the network and mote status, view open alarms, configure network security, change system and interface settings, set the manager's clock, and update system and mote software. You can also reboot the manager hardware and system software and execute other commands. Admin Toolset is supported for Windows Internet Explorer 7+, Safari 5.0+, and Firefox 10.0+. To run the Topology Viewer, Admin Toolset requires that your computer have the Java Runtime Environment (JRE) version 6 (or later) installed.

This chapter provides instructions on how to connect to the manager and access Admin Toolset.

5.1.1 Setting Up a Connection to the Manager

If your computer is not connected to the manager, follow these instructions to set up the connection. The setup procedure depends on how the manager is configured:

- Set up a DIRECT connection if the manager is configured with the factory default static IP address, 192.168.99.100.
- Set up a LAN connection if the manager is configured with an IP address on your LAN (for example, a DHCP-assigned address, or a static IP address assigned by your network administrator).

> ⚠ Note: If you do not know whether the manager is configured with the factory default static IP address or an IP address on your LAN, try first to connect to the manager via the serial CLI as described in the SmartMesh WirelessHART Manager CLI Guide.

To set up a direct connection to the manager:

- Connect your computer directly to the manager using an Ethernet cross-over cable.
- Go to Control Panel and set the Network Connection for your computer to a static IP address.

> ⚠ Note: For example, you could set computer IP address to 192.168.99.101. By default, the manager's static IP address is 192.168.99.100, with a subnet mask of 255.255.255.0.

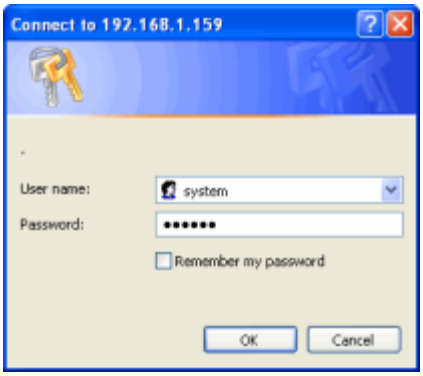
To set up a LAN connection to the manager:

- Using a straight-through Ethernet cable, connect your computer to the LAN that contains the manager.
- Go to Control Panel and set the Network Connection for your computer as specified by the LAN administrator.
- You must know the IP address of the manager to connect. This can be obtained via the manager CLI over serial (see WirelessHART Manager CLI Guide) and issuing the `ifconfig` command - the manager's IP address ("inet addr:") is listed under the eth0 interface.

5.1.2 Accessing Admin Toolset

To access Admin Toolset your computer must be connected to the manager (see the previous procedure). You also need to know the manager IP address.

- Connect your computer to the manager as described above.
- Open the Web browser.
- Enter the manager IP address in the Address field, e.g. `https://192.168.99.100`. Note that Admin Toolset uses a secure connection, and thus the https address.
- If a security alert displays indicating there may be a problem with the site security certificate, take the action required to proceed to the website. The manager uses a self-signed certificate that will cause most browsers to warn you when you connect.
- Log on by entering the username and password for Admin Toolset. The default username is `system` and password is `system`.



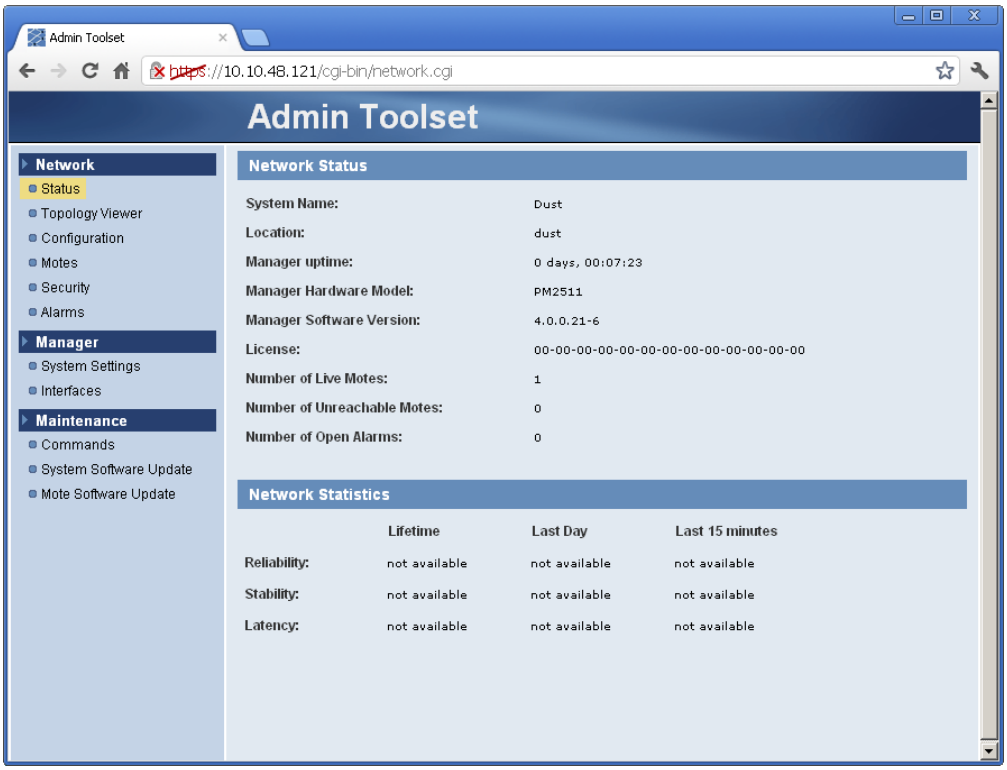
If you have the manager switched to DHCP addressing, you will need to obtain the IP address of the manager to use above. This can be done by logging into the manager CLI port and using the `ifconfig` command to read the address. See SmartMesh IP Manager CLI Guide for details on logging into the CLI.

5.1.3 Admin Toolset Screens

This chapter describes each Admin Toolset screen. Note that information shown in the screen example may differ from what appears on your screen and will depend upon your system configuration. Screens are broken into three categories: **Network** (applies to the network as a whole), **Manager** (settings specific to the manager itself), and **Maintenance** (password maintenance, diagnostic commands, and software upgrades).

Status Information

Click the **Status** link in the leftmost tab to display network status and statistics.



Field	Description
Network Status	
System Name	User-defined name for the system (string)
Location	User-defined location of the system (string)
Manager Uptime	Amount of time since the manager software was last started
Manager Hardware Model	The manager hardware model number (may not reflect marketing part number)

Manager Software Version	Software version running on the manager (major, minor, revision, build)
Number of Live Motes	Number of motes that are connected and reporting to the network (including the access point mote).
Number of Unreachable Motes	Number of motes that are not currently reporting to the network (**Lost** state in CLI), but were in the network at some time in the past.
Number of Open Alarms	Number of alarms (events that may impact network performance) that are currently unresolved.
Network Statistics	
Reliability	The percentage of data packets generated by motes (or accepted via *send* API) that the manager actually received. One hundred percent reliability means that every data packet was received. The reported values are network averages. The manager calculates data reliability by dividing the number of packets it received by the sum of number of packets received and packets lost: % Reliability = 100 * Unique Packets Received / Total packets generated
Stability	Path stability measures mote-to-mote transmissions from the perspective of the sender. It is the average percentage of packets that were acknowledged by the MAC layer recipient. The manager calculates path stability by dividing the number of acknowledged packets by the total number of packets transmitted: % Path Stability = 100 * Acknowledged packets/ Total packets transmitted
Latency	The average time (in milliseconds) required for a data packet to travel from the originating mote to the manager. Latency varies across the network. The value represents the average network latency. The manager calculates data latency for each packet by subtracting the time the packet was received at the manager from the packet timestamp in the network layer header, which indicates when the packet was generated or accepted by the mote. Data Latency = Time Packet Received – Packet Timestamp

Viewing Network Topology

Click the **Topology Viewer** link to display the network topology. The Topology Viewer requires Java to be installed on your computer and enabled in your browser. Use the control icons on the toolbar to expand or contract the topology, or zoom to fit the screen. Use the settings icon on the toolbar to change general display settings, such as the screen auto-refresh rate and the appearance of mote labels and paths.

To show more information about a mote or path, hold the pointer over it. Clicking selects a mote or path. By right-clicking a selected mote, you choose to show all paths from the mote to the manager. By right-clicking a selected path, you can show path stability by color: green indicates path stability over 70%, orange is a path stability between 30% and 70%, and red is a path stability below 30%.

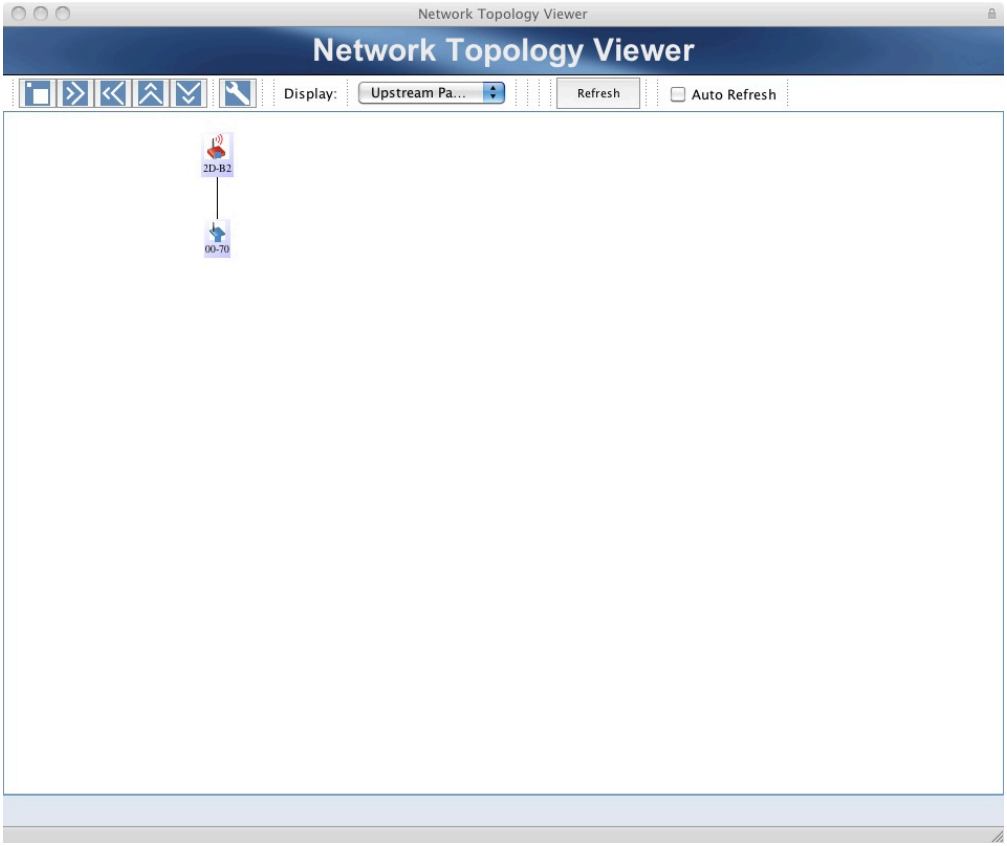
You can move motes and paths by dragging them. For paths, click the path to highlight it and drag points along the path.

Control Icons

- The leftmost icon is the zoom icon - it fits the topology on the screen
- The next four icons are controls to expand or contract the topology display
- The wrench is the settings icon, which changes display settings

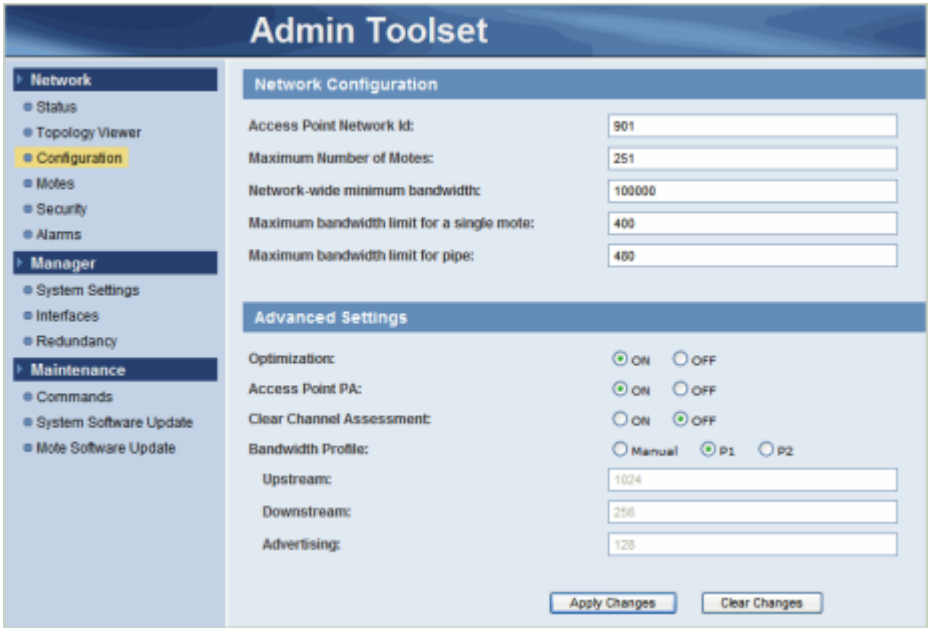
Other controls

- User can select to display upstream or downstream paths
- User can manually refresh via button, or select the auto refresh option to have the display update automatically
- Right-click a selected mote to display pop-up menu
- Hold the pointer over mote or path to display details
- Drag a points on a selected path to move a segment



Configuration Settings

Click the **Configuration** link to change network configuration settings.



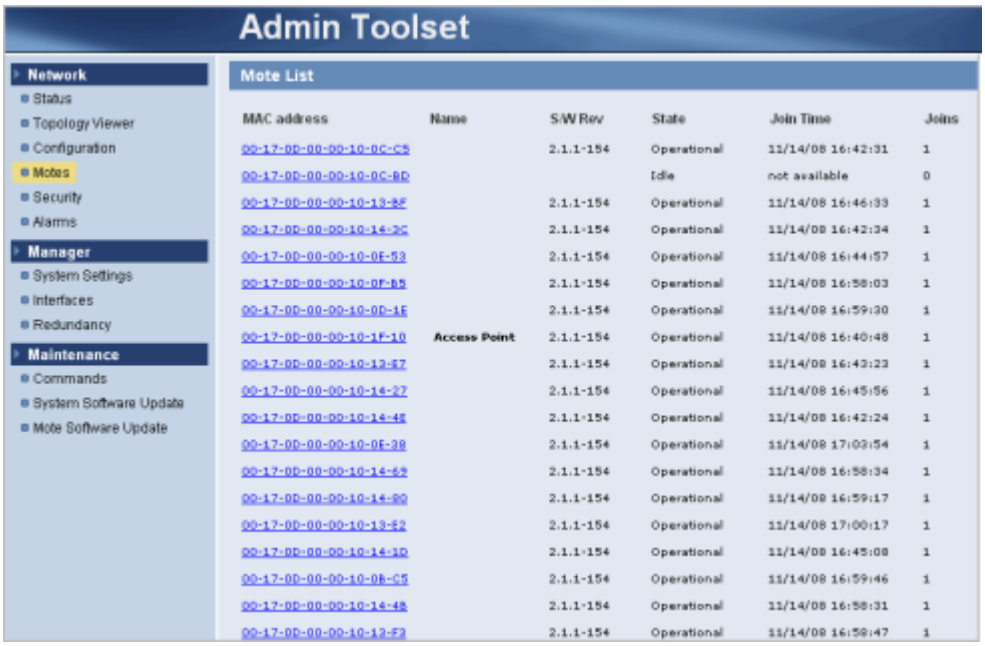
Field	Description
Network Configuration	
Access Point Network Id	The Network ID of the access point mote. If changed, the new access point Network ID will not take effect until you restart the manager software (see the "Commands" section).
Maximum Number of Motes	The maximum number of motes allowed in the network. This excludes access point motes, but includes all other motes in any state.
Network-wide Minimum Bandwidth	The base bandwidth (in msec/packet) that the manager should allocate for each device. Note that this applies only to manager-controlled bandwidth. It does not include bandwidth requested through mote service requests and cannot be used to allocate bandwidth for services.
Maximum Bandwidth Limit for Single Mote	The total usable non-pipe bandwidth allocated to a mote. Includes bandwidth allocated from: • Network-wide minimum bandwidth setting • Mote service requests

Maximum Bandwidth Limit for Pipe	Bandwidth limit (msec/packet) on all pipes—whether requested through the manager-API or created as a result of service requests. This is most useful for regulating service requests from field devices.
Advanced Settings	
Optimization	Turns on (or off) network optimization. When optimization is on, algorithms are enabled on the manager to continuously optimize network links to improve network performance. Because optimization is required for long-term network stability and performance, it is strongly recommended that optimization always remain on.
Access Point PA	Turns on (or off) the RF power amplifier on the access point mote. The setting takes effect after the manager is restarted. Refer to the product datasheet for specifications of RF power at each setting.
Clear Channel Assesment	Turns Clear Channel Assesment (CCA) on or off. When CCA is on, motes listen before transmitting, aborting if another signal is heard; when CCA is off, motes do not listen before transmitting. It is recommended that CCA remain off.
Bandwidth Profile	The bandwidth profile determines the frame size (number of timeslots) for upstream, downstream, and advertising traffic. There are three profiles available—a normal profile (P1), a low-power profile (P2), and a manual profile, which allows you to set the number of timeslots for upstream, downstream, and advertising traffic. The new bandwidth profile setting takes effect the next time the network reforms after the manager is reset.
Upstream	Upstream Number of timeslots for upstream traffic (from mote to manager).
Downstream	Number of timeslots for downstream traffic (from manager to mote).
Advertising	Number of timeslots for advertising traffic.

Changing the number of Upstream, Downstream, or Advertising timeslots can affect the channel hopping behavior of your network. Do not change these values without consulting an Applications Engineer.

Mote Information

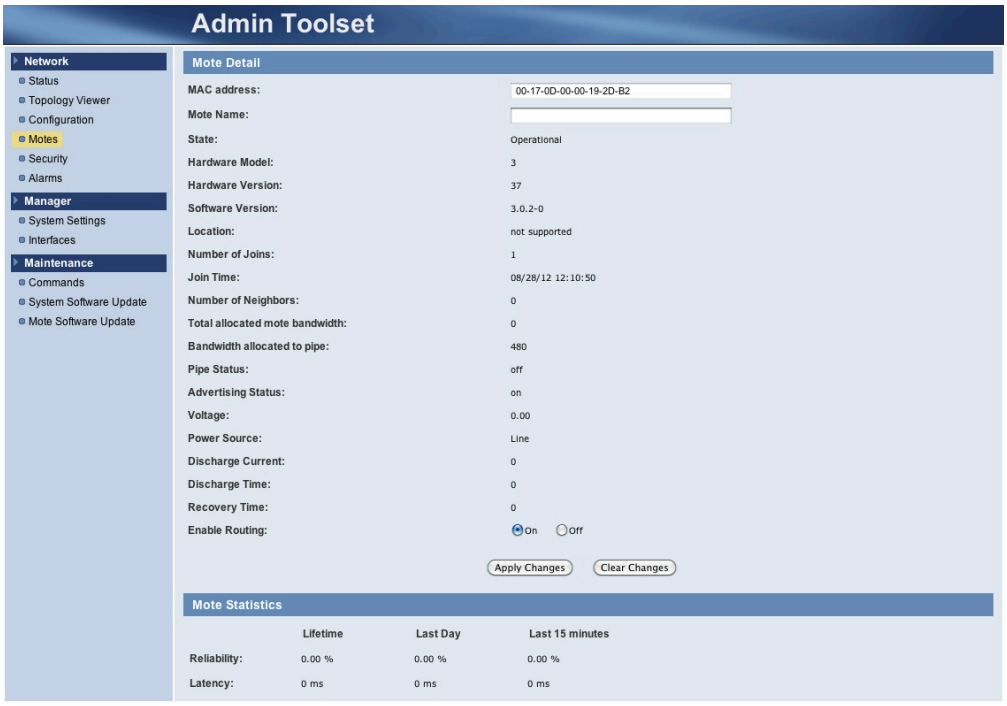
Click the **Motes** link to display information for all network motes. For details about a specific mote, click its MAC address.



Field	Description
MAC address	MAC address (EUI-64) of the mote.
Name	User-defined name (string) of the mote. The mote name can be assigned using the manager API.
S/W Rev	Software version running on the mote.
State	**Idle**: Mote has not been part of the network since the manager started. **Lost**: Mote is not currently part of the network. **Negotiating1**: Mote is in the process of joining the network. **Negotiating2**: Mote is in the process of joining the network. **Connected**: Mote is connected to the network. **Operational**: Mote is operational. **Disconnected**: Mote may be physically removed from network without affecting network.
Join Time	Date and time when the mote joined the network (in network time or UTC if set).
Joins	Number of times the mote has joined the network since the manager was last restarted.

Mote Details

The following mote status information appears when you click on the mote **MAC address** in the **Motes** screen.

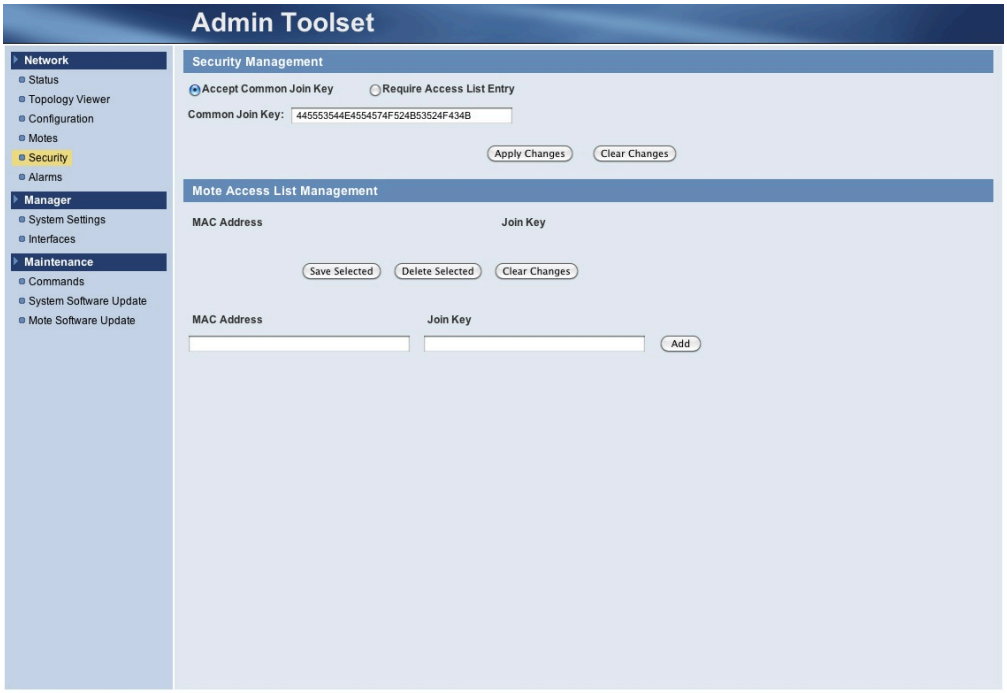


Field	Description
MAC Address	MAC address (EUI-64) of the mote.
Mote Name	Allows you to assign a name (string) to the mote.
State	Indicates the mote state (as described in Mote List above)
Hardware Model	Model number (1 byte).
Hardware Version	Hardware version (1 byte).
Software Version	Software version (Major, Minor, Revision, Build)
Location	Will display "not supported"
Number of Joins	Number of times mote has joined the network since the manager was last reset.
Join Time	Timestamp of mote join (in network time or UTC if set).

Number of Neighbors	Number of potential and actual neighbors associated with this mote.
Total Allocated Mote Bandwidth	Currently allocated bandwidth (in ms/packet) from the mote to the manager.
Bandwidth Allocated to Pipe	Currently allocated bandwidth (in ms/packet) for the pipe.
Pipe Status	Indicates whether there is an active pipe between the mote and the manager.
Advertising Status	Indicates whether advertising is turned on, off, or pending.
Voltage	Voltage at mote supply pin.
Power Source	Shows whether the mote is line powered, battery powered, or rechargeable (power scavenging).
Discharge Current	Maximum allowable discharge current in µA.
Recovery Time	Time (in seconds) required for the mote to recover from a power drain.
Enable Routing	Sets whether the mote may be used for routing.
Mote Statistics	Shows reliability and latency statistics for this mote. Reliability is the percentage of generated/accepted at serial interface packets from this mote that the manager actually received. One hundred percent reliability means that every packet was received. Latency is the average time (in ms) required for a data packet to travel from this mote to the manager.

Security Settings

Click the **Security** link to set the network security mode. The manager can be configured to accept any mote who matches the common join key. Or alternatively, to accept a mote only if the mote and its join key are contained in the manager's Access Control List (ACL). Motes in the ACL may (preferably) have unique join keys. To add a mote, enter the mote MAC address and join key in the **Mote Access List Management** area, and click **Add**. To delete a mote from the list, select the mote and click **Delete Selected**. To edit the join key of a mote in the ACL, select the mote, edit the key, and click **Save Selected**.

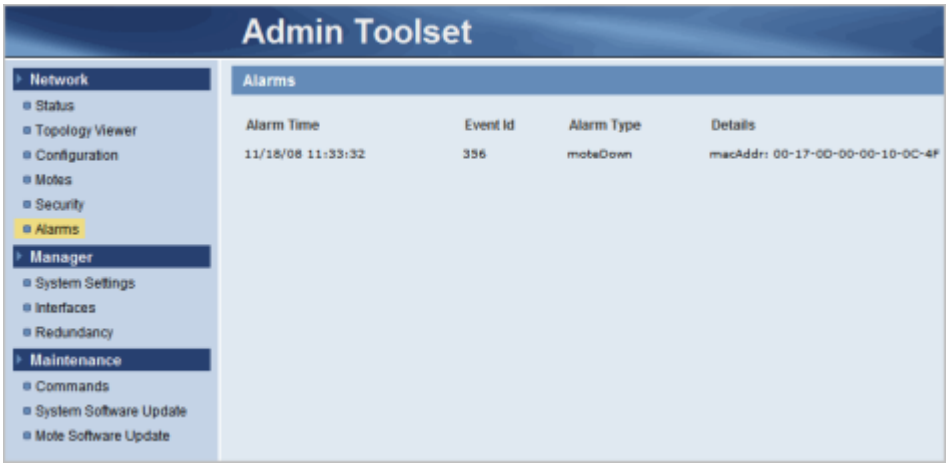


Field	Description
Security Management	
Accept Common Join Key	When this option is selected, motes that share the common join key are allowed to join the network.
Require Access List Entry	When this option is selected, only motes whose MAC address and join key are on the mote access list are allowed to join the network.
Common Join Key	The common join key used if "Accept Common Join Key" is selected.
Mote Access List Management	
MAC Address	MAC address (EUI-64) of the mote you are adding to the mote access list.
Join Key	User-defined join key for the mote you are adding to the mote access list. Preferably unique.

Alarm Information

Click the **Alarms** link to display events that may impact network performance. The manager posts an alarm when there is a problem with a mote or when overall network performance does not meet the Service Level Agreement (SLA) thresholds that have been set for reliability, path stability, and data latency. The manager removes the alarm when the alarm condition is corrected.

> ⚠ Note: To set SLA thresholds, use the manager API. Refer to the SmartMesh WirelessHART Manager API Guide for details.



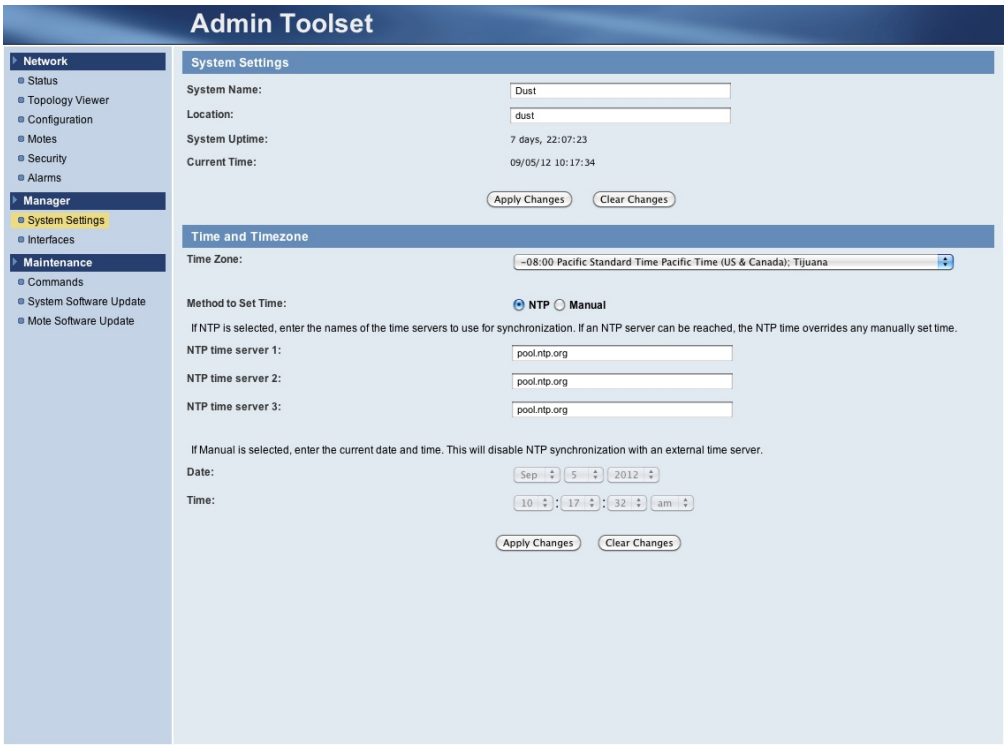
Field	Description
Alarm Time	Time the alarm occurred (in network time or UTC if set).
Event ID	Number identifying the alarm event.
Alarm Type	Type of alarm (see "Alarm Types" below).
Details	Alarm details (e.g. mote involved).

Alarm Types

Alarm Type	Description
Alarm Close	An alarm condition has been corrected.
Mote Down	A mote is no longer responding to messages from the manager.
Low Reliability	The network reliability dropped below the SLA threshold.
High Latency	The network latency increased above the SLA threshold.
Low Stability	The network stability dropped below the SLA threshold.
Mote Bandwidth	The bandwidth available to a mote dropped below a preset value.
Maximum Number of Motes Reached	The maximum number of motes for this network was reached. The maximum number of motes can be set using the Configuration screen.

System Settings

Click the **System Settings** link to set the system name, location, and clock.

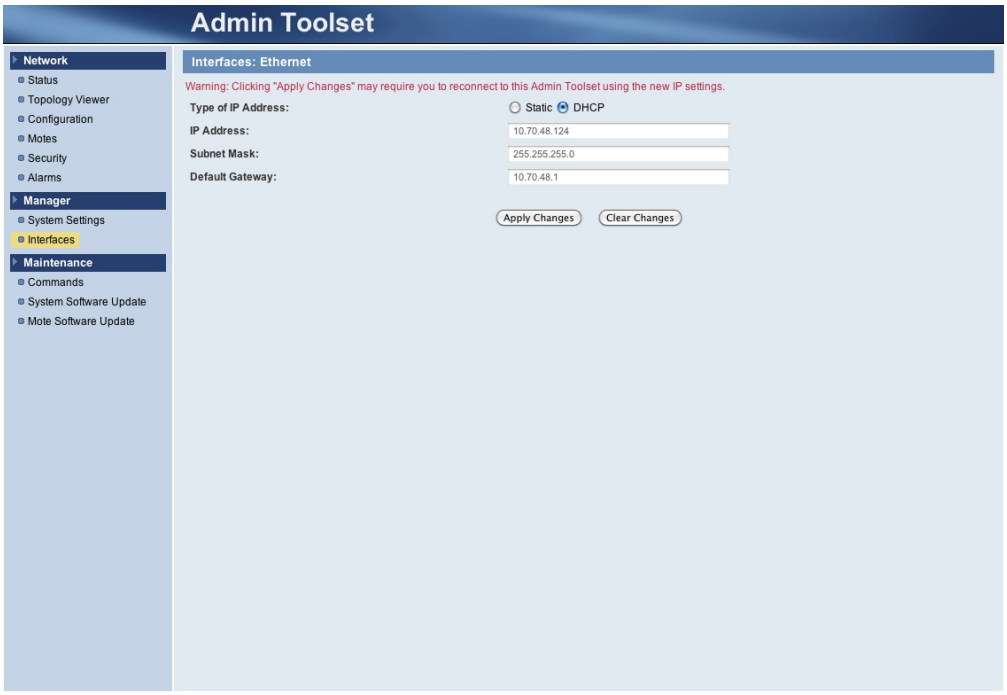


Field	Description
System Settings	
System Name	The user-defined name (string) for the system.
Location	The user-defined location (string) of the system.
System Uptime	The amount of time since the manager was last rebooted/power cycled.
Current Time	The current time according to the manager's clock.
Time and Time Zone	

Time Zone	Sets the time zone that applies to the manager's clock. Use the menu to select a different time zone for the manager.
Method to Set Time	Sets how the manager acquires the current time. Choose **NTP** if you want the manager to get the time from a Network Time Protocol (NTP) server you specify. Choose **Manual** to set the current time manually. Note that the manager could drift several seconds/day at elevated/reduced temperatures if using Manual time.
NTP Time Server	Allows you to specify up to three NTP servers that the manager can use to set its clock. If you want to use fewer servers, delete the value(s) from the NTP server fields.
Date	If you are setting the manager's clock manually, enter the current date.
Time	If you are setting the manager's clock manually, enter the current time.

Interface Settings

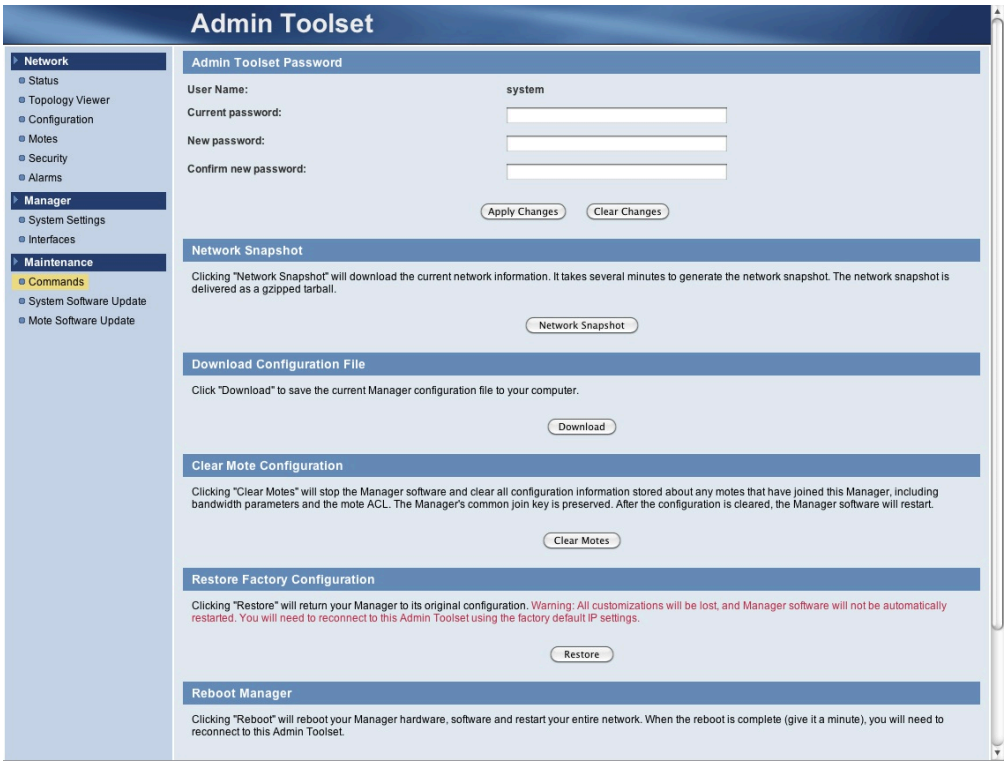
Click the **Interfaces** link to change the manager's interface settings.



Field	Description
Interfaces: Ethernet	Settings for the manager's Ethernet port.
Type of IP Address	Sets the manager's IP address as static (unchanging, assigned by a network administrator) or DHCP (LAN-assigned).
IP Address	Sets a static IP address for the manager. Only enabled if Type of IP Address = Static.
Subnet Mask	Sets the subnet mask for a static IP address.
Default Gateway	Sets the default gateway (router) for a static IP address.

Commands

Click the **Commands** link to change passwords and execute administrative and troubleshooting commands. After executing a command, status information displays at the top of the page. You may need to scroll to see the status information.

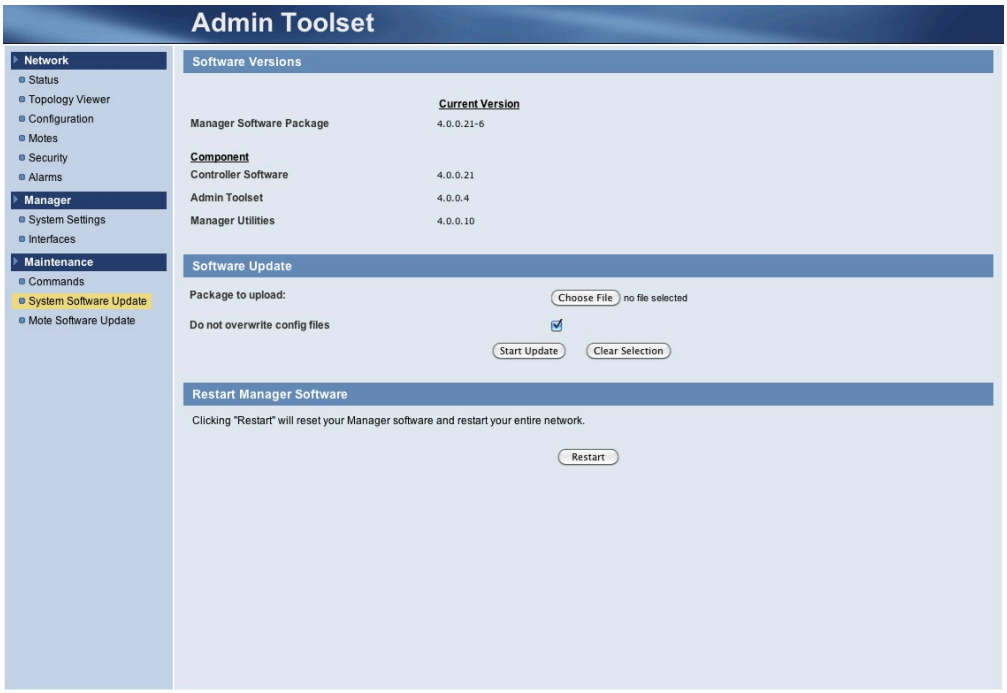


Field	Description
Admin Toolset Password	Allows you to change the password for logging on to the manager's Admin Toolset utility.
User Name	Always "`system`".
Current Password	The password to be changed. The default password is `system`.
New Password	Any nonzero length string can be used. There is no "policy" applied to ensure the password is strong.
Confirm new password	Must match the entered new password.
Network Snapshot	Saves a snapshot of current and historical network information to your computer to be used for network troubleshooting. It takes several minutes for the manager to build the network snapshot file for download.

Download Configuration File	Saves the manager's current configuration information to a file on your computer.
Clear Mote Configuration	Deletes mote information stored in the manager. All motes disconnect from the network and rejoin again.
Restore Factory Configuration	Restores the manager's factory configuration settings. All customizations to the manager's configuration will be lost.
Reboot Manager	Restarts the manager's hardware and software. All motes disconnect from the network and rejoin again. You will be disconnected from Admin Toolset and will need to log in again.
Restart Manager Software (off screen in figure)	Restarts the manager's software and the network. All motes disconnect from the network and rejoin again.

System Software Update

Click the **System Software Update** link to update software on the manager.



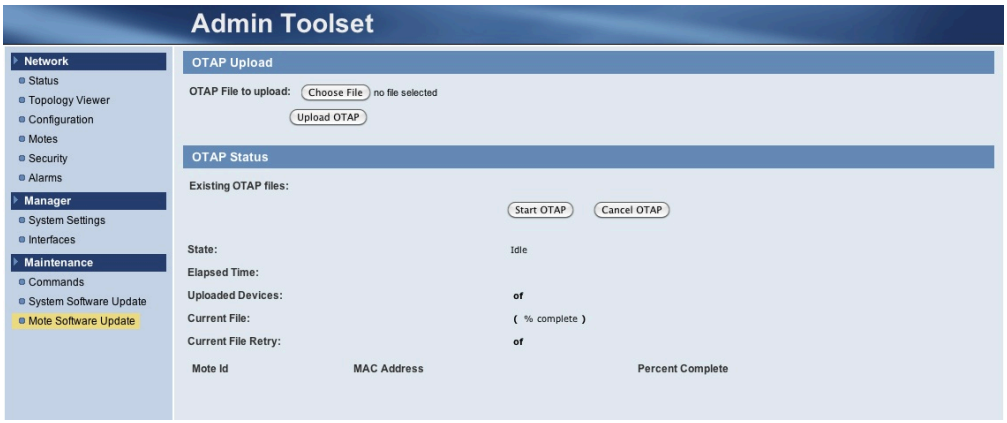
Field	Description
Software Versions	
Manager Software Package	The Current Version column indicates the bundle version of the software that is currently running on the manager.
Controller Software	The version of the core manager software.
Admin Toolset	The version of the web-based interface you are currently using.
Manager Utilities	Additional utilities for managing time, capturing diagnostics, etc.
Software Update	
Package to Upload	Choose file button brings up dialog to locate software package (.ipkg file).

Do not overwrite config files	Check this option to retain configuration information for the software package you are updating. For Manager Utilities, this includes the manager's time zone and clock settings. For Manager Software, this includes system and network settings, including the Network ID, security mode, redundancy settings, and access control list.
Restart Manager Software	Use the **Restart** button to reset the manager software and restart the network after the update is finished.

Overwriting the configuration file, resets the manager's Network ID and join key to the factory default settings. You will need to re-configure these settings on the manager in order for the network to reform. Before performing the update, download the current configuration file (see "Commands") or make a note of the Network ID, common join key, redundancy settings, and access control list.

Mote Software Update

Click the **Mote Software Update** link to update mote software on the manager and all network motes. The mote software update is contained in an OTAP (Over-The-Air-Programming) package. The first step is to upload the OTAP package (ipkg) to the manager. Once this is done, you can start the OTAP process which sends the update out to all network motes. The progress of the OTAP is shown in the **OTAP Status** area when you refresh the Browser window.



Field	Description
OTAP Upload	
OTAP File to Upload	Use the **Choose File** button to locate the OTAP package (ipkg). Click the **Upload OTAP** button to upload the package to the manager. If you need to re-upload an OTAP package that you have previously uploaded, you first need to restart the manager to remove the old OTAP package.
OTAP Status	
Existing OTAP Files	A list of the files contained in the OTAP package in the OTAP directory on the manager. Click **Start OTAP** to start sending the mote software update to all network motes. Click **Cancel OTAP** to cancel the OTAP process if it has not already progressed to the "commit" state.

State	Indicates the status of the OTAP process: None - The OTAP process has been cancelled. Initializing - OTAP handshakes are being performed in preparation for uploading files to the motes. Upload - OTAP files are being uploaded to the motes. Commit - The upload is complete and motes are being reset in order to activate the new software. Cancelling an OTAP at this stage will result in motes running different software versions. Completed - OTAP is completed.
Elapsed Time	Elapsed time since the OTAP started.
Updated Devices	Number of motes (out of the total number) that have confirmed receipt of OTAP files.
Current File	File that is currently being sent via OTAP, and percentage sent
Current File Retry	Number of times that the current file has been retransmitted (out of the total number of possible retransmissions).
Mote ID	Mote ID of the mote that is currently being updated.
MAC Address	MAC address (EUI-64) of the mote that is currently being updated.
Percent Complete	Percentage of the current file that has been updated on the mote indicated by Mote ID and MAC Address

6 SmartMesh WirelessHART SDK

For more information

This section is meant to give you an **plain-English general overview** of the SmartMesh SDK.

The **technical documentation** is provided in the `/doc/` folder of your installation, as an Doxygen-based description of all functions, parameters and variables.

6.1 About SmartMeshSDK

The SmartMesh SDK is a Python package which simplifies the integration of a SmartMesh IP or SmartMesh WirelessHART network into your application. It implements the Application Programming Interface (API) of the device it is connected to. A set of sample applications are included in the SmartMesh SDK, allowing a programmer to quickly understand the API and use it as part of a larger system.

6.2 SmartMeshSDK Features

- **Complete API definitions**. Supports the full API of the SmartMesh IP Manager, SmartMesh IP Mote, SmartMesh WirelessHART Manager, SmartMesh WirelessHART Mote. No need to copy-paste command definitions.
- **Low-level connectors**. Enables your application to connect to all Dust Networks devices, over all transport media, including serial, XML-RPC and SerialMux. No need to develop low-level modules.
- **dustUI visuals library**. A set of standard GUI elements with a common look-and-feel which can easily be customized.
- **Full source code**. Delve into the inner-workings of the SmartMeshSDK.
- **Example applications**. Both GUI based, and script based. Provided in source code and binary formats.
- **Fully documented**. High level hands-on introduction guide, as well as Doxygen-based source code documentation.
- **Portable**. Does not require anything beyond a standard Python installation. Runs on any platform which supports Python, including Microsoft Windows, Linux and MacOS.
- **Extensible**. Is designed to be integrated inside a larger application.

6.3 Document Organization

The remainder of this document is organized as follows, and it meant to be followed in order:

- Folder Contents gives you a first walk-through of the contents of the `SmartMeshSDK-full-X.X.X.X.zip` zip file you've just installed.
- Before detailing the internals of the SmartMesh SDK, example applications gives you an overview of the sample applications shipped with the SmartMesh SDK, both in binary and source code formats.

- Architecture will be your first look into the SmartMesh SDK source code, detailing the major packages and objects.
- The library of Graphical User Interface objects is detailed in dustUI Library.
- SmartMeshSDK Library covers the core components of the SmartMesh SDK, including the API definition and connector objects.

6.4 Folder Contents

Unzipping the `SmartMeshSDK-full-X.X.X.X.zip` file creates the following directory structure:

- `SmartMeshSDK-X.X.X.X/` with the following sub-directories:
  - `/src/` contains the source code of the SmartMeshSDK, and its sample applications. Note that, thanks to the interpreted nature of Python, you can run the applications directly from their source files.
  - `/win/` contains pre-compiled versions of the sample applications generated using the py2exe utility. This allows you to run the applications on a Windows computer without having to install Python.
  - `/doc/` contains HTML-based documentation of the SmartMeshSDK generated using Doxygen.
  - `/api/` contains C header files and sample code

6.5 Example Applications

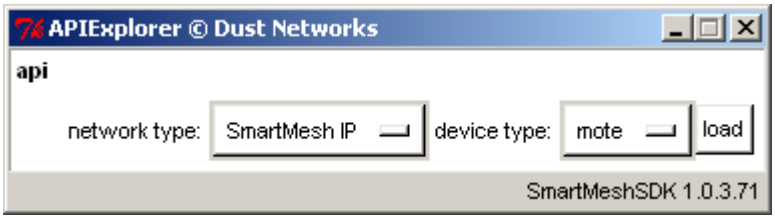
6.5.1 Running An Application

You can run an example application in two different ways:

From its pre-compiled Windows executable

The `/win/` directory contains all the sample applications, compiled into a Windows executable by the py2exe utility.

For example, double-clicking on `/win/APIExplorer.exe` starts the APIExplorer application

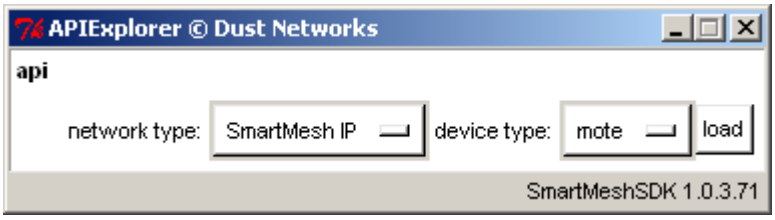


This is the recommended way of running the application if you do **not** wish to modify the behavior of the application.

From its Source File

Because Python is an interpreted programming language, simply double-click on an application's script to start it.

For example, double-clicking on `/src/bin/APIExplorer/APIExplorer.py` starts the APIExplorer application



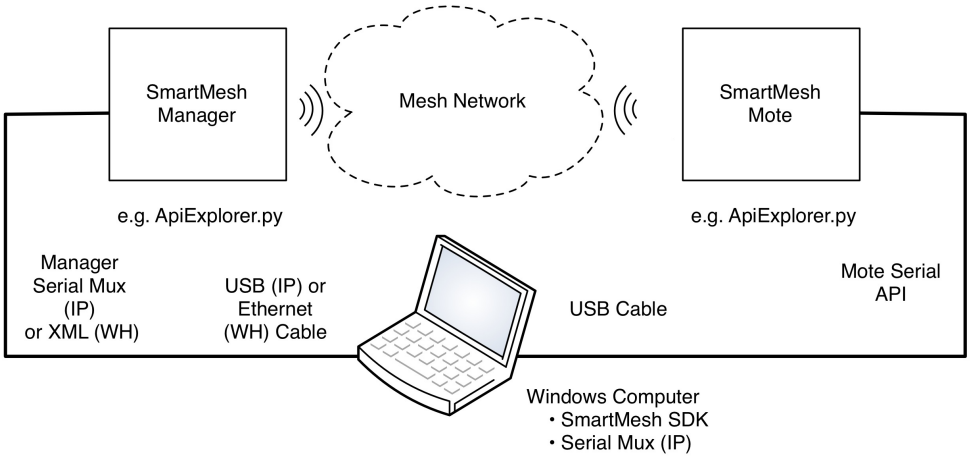
This is the recommended way of running the application if you are modifying the behavior of the application by editing its source code.

6.5.2 Color Coding

Some application enable you to enter data into field. The following color coding is used to indicate the outcome.

color	meaning
green	The value entered is valid and could be used.
yellow	The value entered is not valid. Please check the type of characters entered (e.g. whether all characters are numbers when entering an integer) and length.
orange	The value entered was accepted by the application, but triggered a warning. One example is when trying to connect to a device which is not powered on.
red	The value entered was accepted by the application, but triggered an error. One example is when trying to open an non-existing serial port.

6.5.3 Overview



- APIExplorer: APIExplorer is a graphical user interface that allows you to interact with the Application Programming Interface (API) of all SmartMesh devices.
- InstallTest: InstallTest is a simple console (non-graphical) application to test correct installation of Python components needed to run the SmartMesh SDK examples from source.
- PkGen: PkGen connects to a either a SmartMesh IP Manager or SmartMesh WirelessHART Manager, allowing you to send commands to the the packet generation application on the corresponding motes running in **master** mode.
- SimpleHartMote: SimpleHartMote is a console-based (non graphical) application which shows how to interact with the API of a SmartMesh WirelessHART mote programmatically.
- TempMonitor: TempMonitor connects to a either a SmartMesh IP Manager or SmartMesh WirelessHART Manager, allowing you to send commands to the the temperature sampling application on the corresponding motes running in **master** mode.

6.5.4 APIExplorer

Introduction

APIExplorer is a graphical user interface that allows you to interact with the Application Programming Interface (API) of all SmartMesh devices.

It can connect to:

- The SmartMesh IP Manager
- The SmartMesh WirelessHART Manager
- The SmartMesh IP Mote running in **slave** mode
- The SmartMesh WirelessHART Mote running in **slave** mode

Mote modes are discussed in the SmartMesh IP User's Guide and the SmartMesh WirelessHART User's Guide

Running

You can run the API Explorer application:

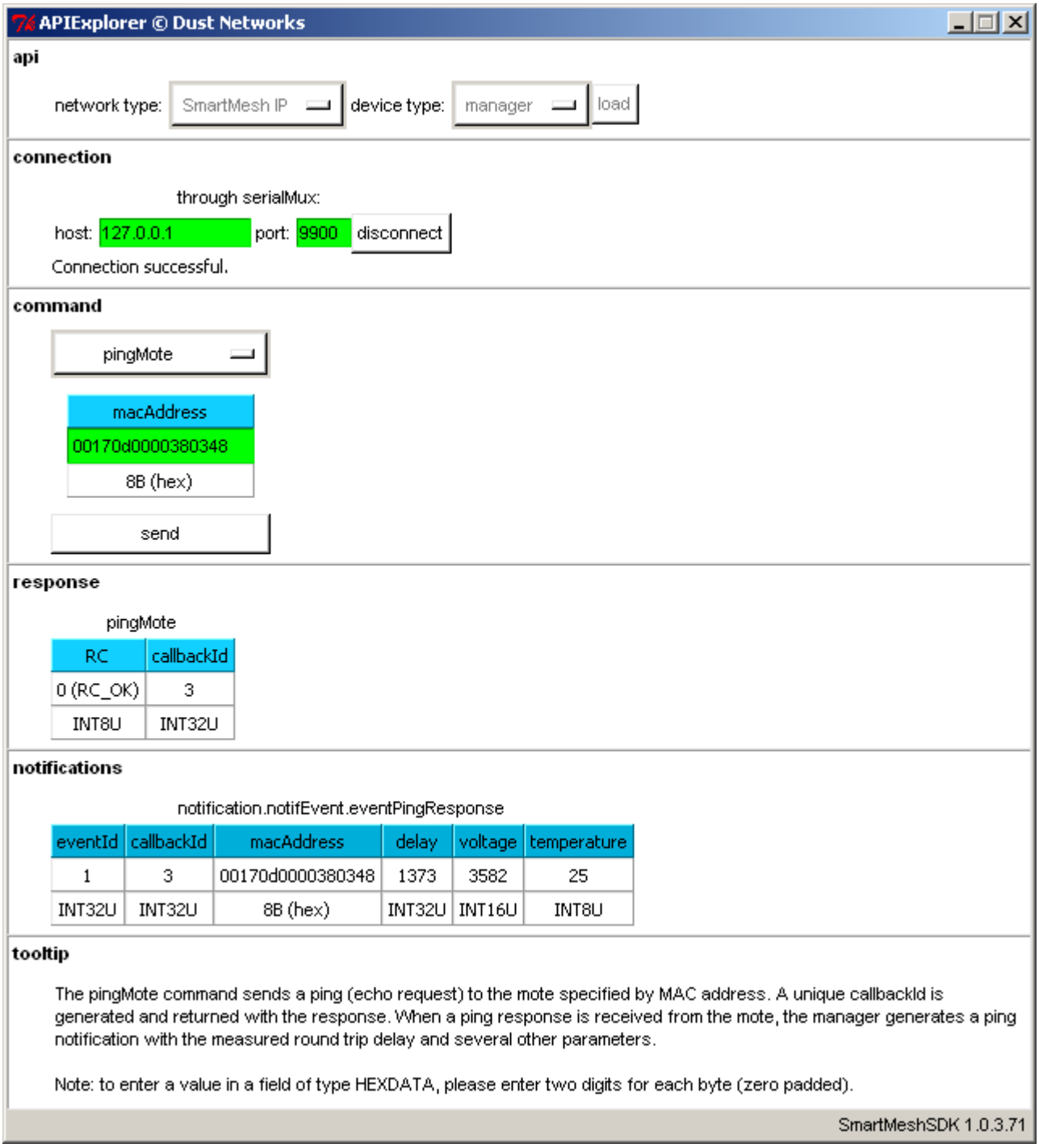
- by double-clicking on the Windows executable at `/win/ APIExplorer.exe`
- by double-clicking on its source files at `/src/ bin/APIExplorer/APIExplorer.py` (may require additional steps on non-windows OSes)

Description

API Explorer is a GUI based application to interact with the following devices' API interface:

- SmartMesh IP Manager
  - connected over serial
  - connected over Serial Mux
- SmartMesh IP Mote
  - connected over serial - mote must be in **slave** mode for the API to be enabled
- SmartMesh WirelessHART Manager
  - connected over XML-RPC
- SmartMesh WirelessHART Mote
  - connected over serial - mote must be in **slave** mode for the API to be enabled

An example of the APIExplorer window is shown below, after connecting to a SmartMesh IP Manager. Frame contents vary among devices as the manner of connection and available commands differs.


APIExplorer © Dust Networks
api
network type: SmartMesh IP
device type: manager
load
connection
through serialMux:
host: 127.0.0.1
port: 9900
disconnect
Connection successful.
command
pingMote
macAddress
00170d0000380348
8B (hex)
send
response
pingMote
RC
callbackId
0 (RC_OK)
3
INT8U
INT32U
notifications
notification.notifEvent.eventPingResponse
eventId
callbackId
macAddress
delay
voltage
temperature
1
3
00170d0000380348
1373
3582
25
INT32U
INT32U
8B (hex)
INT32U
INT16U
INT8U
tooltip
The pingMote command sends a ping (echo request) to the mote specified by MAC address. A unique callbackId is generated and returned with the response. When a ping response is received from the mote, the manager generates a ping notification with the measured round trip delay and several other parameters.
Note: to enter a value in a field of type HEXDATA, please enter two digits for each byte (zero padded).
SmartMeshSDK 1.0.3.71

The interface consists of 6 frames:

- Use the **api** frame to select the type if network and device you wish to connect to. Pressing the `load` button configures the corresponding API definition, and shows the other frames.
- Specify how to connect to your device in the **connection** frame. Depending on the type of network and device selected in the **api** frame, you may be offered multiple connection options.
- Use the **command** frame to select the command you want to send to the device. All commands in the device's API are available. All command parameters must be filled in - in general it is a good idea to use the *get* form of any command before using the *set* form so as to familiarize yourself with the commands arguments . Pressing the **send** button will send the command to your device using the connector selected in the connection frame.
- Reponses to the commands you send appear in the **response** frame.
- Notifications published by the device appear in the **notifications** frame.

Some devices, such as the SmartMesh IP Manager require you to subscribe to notifications (using the *subscribe* command) before you can receiving them.

- When selecting a command, its description appears in the **tooltip** frame.

Walk-throughs that use APIExplorer

The API Explorer is used in the following tutorials:

- For SmartMesh IP:
  - Interacting with the Manager
  - Interacting with a Mote
  - Log HDLC Frames
  - Upstream Communication
  - Downstream Communication
- For SmartMesh WirelessHART :
  - Interacting with the Manager
  - Interacting with a Mote

6.5.5 InstallTest

Introduction

InstallTest is a simple console (non-graphical) application to test correct installation of Python components needed to run the SmartMesh SDK examples from source.

Running

You can run the InstallTest application:

- by double-clicking on the Windows executable at `/win/InstallTest.exe`
- by double-clicking on its source files at `/src/bin/InstallTest/InstallTest.py`

Since the source version relies on python being installed correctly to run, you cannot use it to verify python installation, but it can be used to verify pyserial installation.

Description

InstallTest is a small command window application to verify correct installation of the python pieces required to run the SmartMesh SDK from source. When launched, a command window will open with the following text:

```
Installation test script - Dust Networks SmartMeshSDK v1.0.3.72
Testing installation of Python...
PASS!
You are running Python 2.7.3
on platform: Windows-XP-5.1.2600-SP3, x86
Testing installation of PySerial...
PASS!
You have PySerial 2.6

Testing installation of PyWin32...
FAIL!
Note: PyWin32 is only required to run the MuxConfig application.
You need to install PyWin32:
 - information http://sourceforge.net/projects/pywin32/
 - download and install the latest release for your Python version from http://s
ourceforge.net/projects/pywin32/files/pywin32/
Press Enter to exit.
```

6.5.6 PkGen

Introduction

PkGen connects to a either a SmartMesh IP Manager or SmartMesh WirelessHART Manager, allowing you to send commands to the the packet generation application on the corresponding motes running in **master** mode.

Mote modes are discussed in the SmartMesh IP User's Guide and the SmartMesh WirelessHART User's Guide

Running PkGen

You can run the PkGen application:

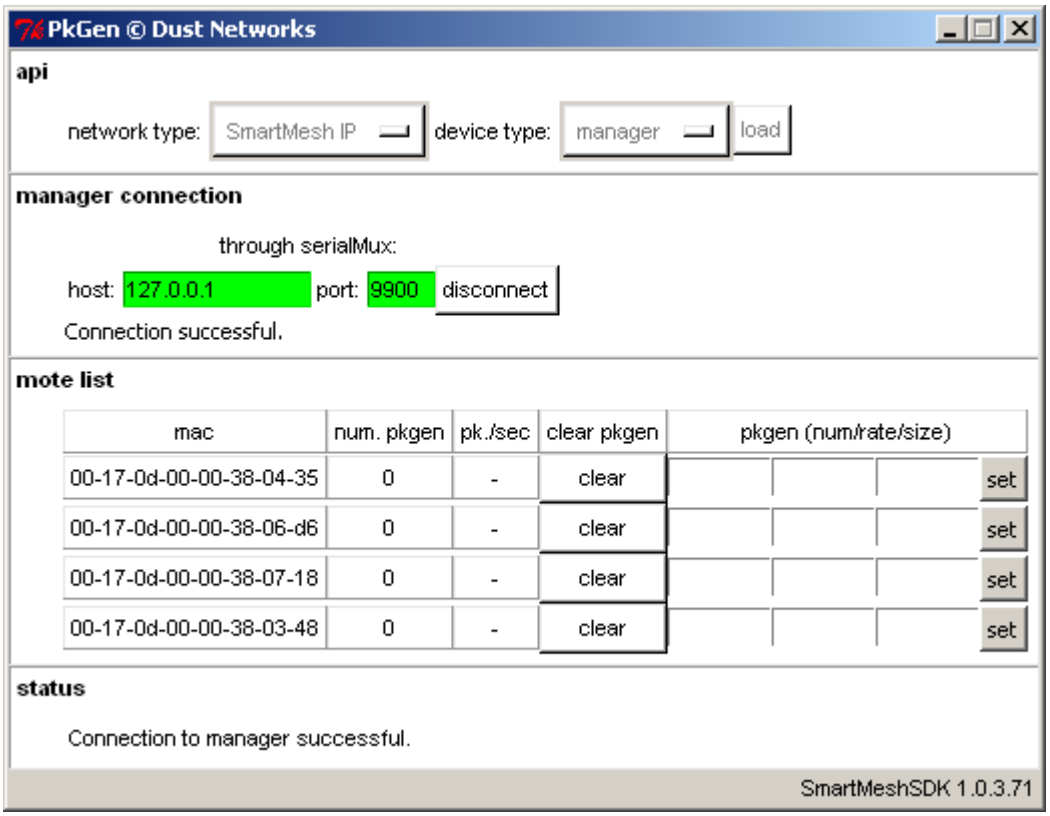
- by double-clicking on the Windows executable at `/win/ PkGen.exe`
- by double-clicking on its source files at `/src/ bin/PkGen/PkGen.py` (may require additional steps on non-Windows OSes)

Description

The PkGen application consists of four frames:

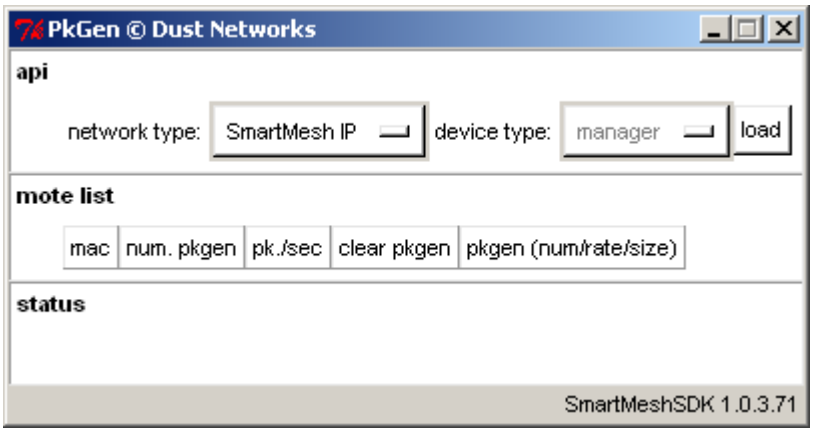
- The **api** frame allows you to specify whether you are connecting to a connect a SmartMesh IP or SmartMesh WirelessHART network. Since you are connecting to a manager, the "device type" selector is disabled.
- The **manager connection** frame allows you to specify how to connect to the manager.
- Once connected, the **mote list** contains the list of motes current in the network.
- The **status** frame displays general status information.

An example of the PkGen application window is shown below, after the user has connected to a SmartMesh IP Manager. Aside from the API and connection tabs, the application appears the same for SmartMesh WirelessHART Managers.

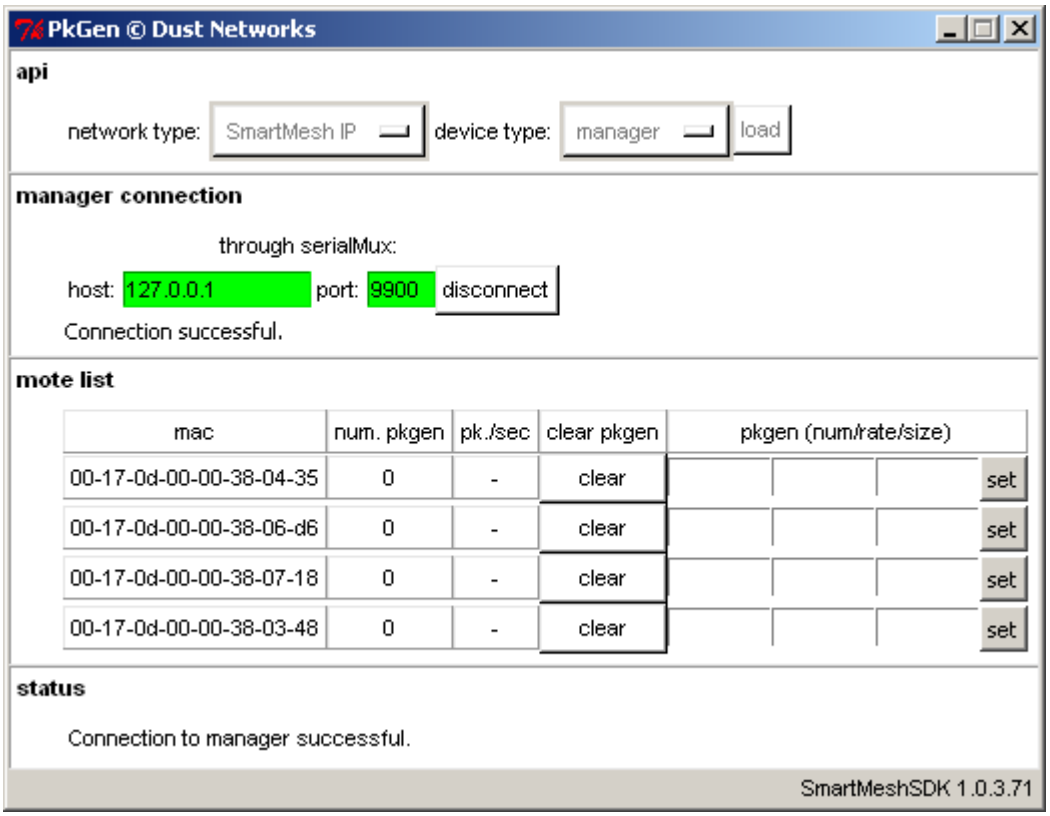


Steps

- Start the PkGen application, the following Window opens.



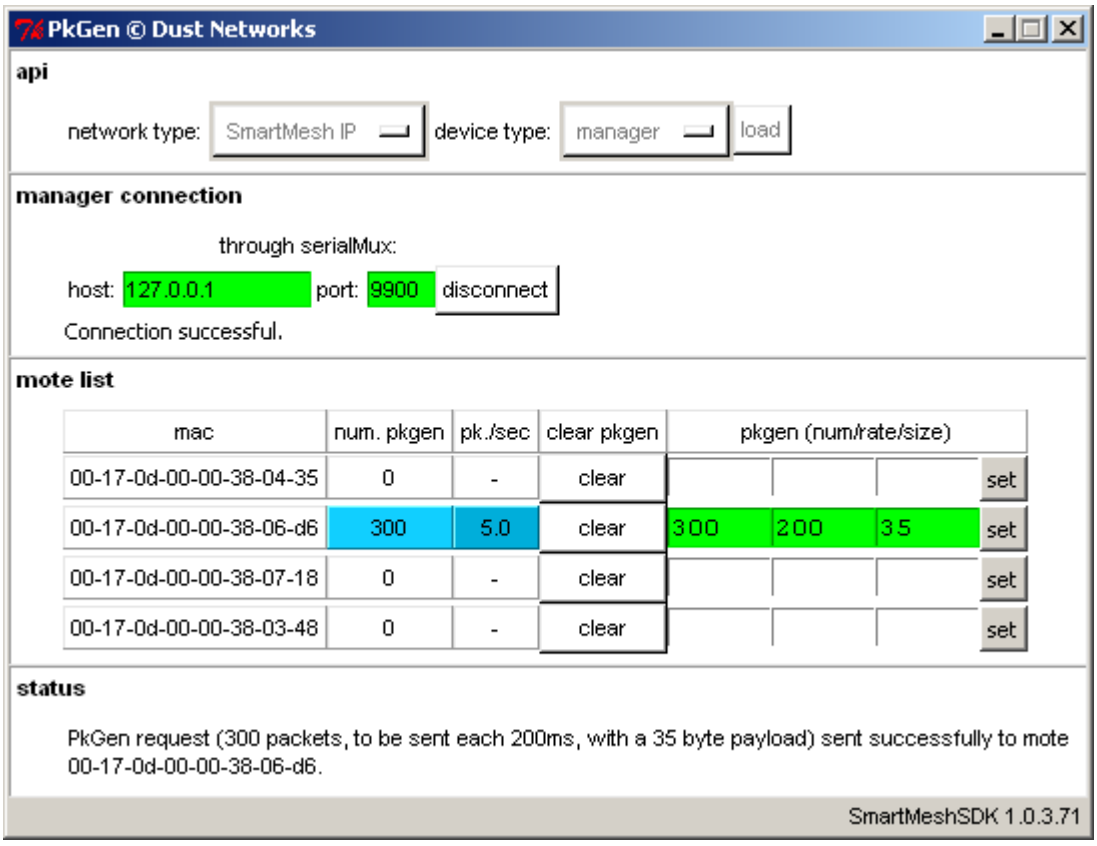
- Select the type of network you will connect to; in our case SmartMesh IP.
- Select the way to connect to the manager, in our case, using the SerialMux. After connecting, the application retrieves the list of motes currently in **Operational** state in the network.



The application does not display the access point or motes which are not in operational mode.

The application will list all motes in **Operational** mode, regardless of whether they are operating in **master** of **slave** mode. You can only interact with motes that you know are running in master mode.

- For the mote of you choice enter the configuration you wish to send to the PkGen application running on the mote:
  - `num` is the number of packets you want the mote to generate.
  - `rate` is the amount of time between two consecutive packets, in ms.
  - `size` is the size of the packet payload, in bytes.
- The configuration below has mote `00-17-0d-00-00-38-06-d6` send 300 packets, one every 200 ms, each carrying 35 bytes of the packet payload. Note that this rate may not be supportable for all devices for a given network size and topology.



- A number of counters are there, for your convenience:
  - *num* - the number of packets generated by the mote which have been received. Note that this only counts packets generated by the mote's PkGen application, i.e. if it is generating different types of data, it will not be accounted for here.
  - *pk./sec* - an approximate rate counter, in packets received by the application, per second.
- The **clear** button allows you to clear the counters.

6.5.7 SimpleHartMote

Introduction

SimpleHartMote is a console-based (non graphical) application which shows how to interact with the API of a SmartMesh WirelessHART mote programmatically.

Running

This sample application is a script:

- `SimpleHartMote` which exercises the API of the SmartMesh WirelessHART Mote.

You can run this script in two ways:

- by double-clicking on the corresponding Windows executable at `/win/`
- by double-clicking on its source files at `/src/bin/Simple/`

Description

This script take you step by step through exploring the capabilities of the SmartMesh SDK. One example output is:

```
Simple Application which interacts with the HART mote - (c) Dust Networks

================== Step 1. API exploration ========================
=====
Load the API definition of the HART mote
done.
=====
List all the defined command IDs:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 16, 17, 19, 20]
=====
List all the defined command names:
['setParameter', 'getParameter', 'setNVParameter', 'getNVParameter', 'send', 'jo
in', 'disconnect', 'reset', 'lowPowerSleep', 'hartPayload', 'testRadioTx', 'test
RadioRx', 'clearNV', 'search', 'testRadioTxExt', 'testRadioRxExt']
=====
Get the command name of command ID 4:
getNVParameter
=====
Get the command ID of command name 'getNVParameter':
4
=====
List the subcommand of command 'getNVParameter':
['macAddress', 'networkId', 'txPower', 'powerInfo', 'ttl', 'HARTantennaGain', 'O
TAPlockout', 'hrCounterMode']
```

```
=====
Get a description of the getNVParameter.powerInfo command:
The getNVParameter<powerInfo> command returns the power supply information store
d in mote's persistent storage.
=====
List the name of the fields in the getNVParameter.powerInfo request:
[]
=====
List the name of the fields in the getNVParameter.powerInfo response:
['powerSource', 'dischargeCur', 'dischargeTime', 'recoverTime']
=====
List the name of the fields in the testRadioTx response:
['RC']
=====
Print the format of the testRadioTx 'RC' response field:
int
=====
Print the valid options of the testRadioTx 'RC' response field:
[0, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
=====
Print the format of the getNVParameter.powerInfo 'powerSource' response field:
int
=====
Print the length of the getNVParameter.powerInfo 'powerSource' response field:
1
=====
Print the valid options of the getNVParameter.powerInfo 'powerSource' response f
ield:
[0, 1, 2]
=====
Print the description of each valid options of the getNVParameter.powerInfo 'pow
erSource' response field:
['Line', 'Battery', 'Rechargeable/Scavenging']
Script ended. Press Enter to exit.
```

6.5.8 TempMonitor

Introduction

TempMonitor connects to a either a SmartMesh IP Manager or SmartMesh WirelessHART Manager, allowing you to send commands to the the temperature sampling application on the corresponding motes running in **master** mode.

Mote modes are discussed in the SmartMesh IP User's Guide and the SmartMesh WirelessHART User's Guide.

Running TempMonitor

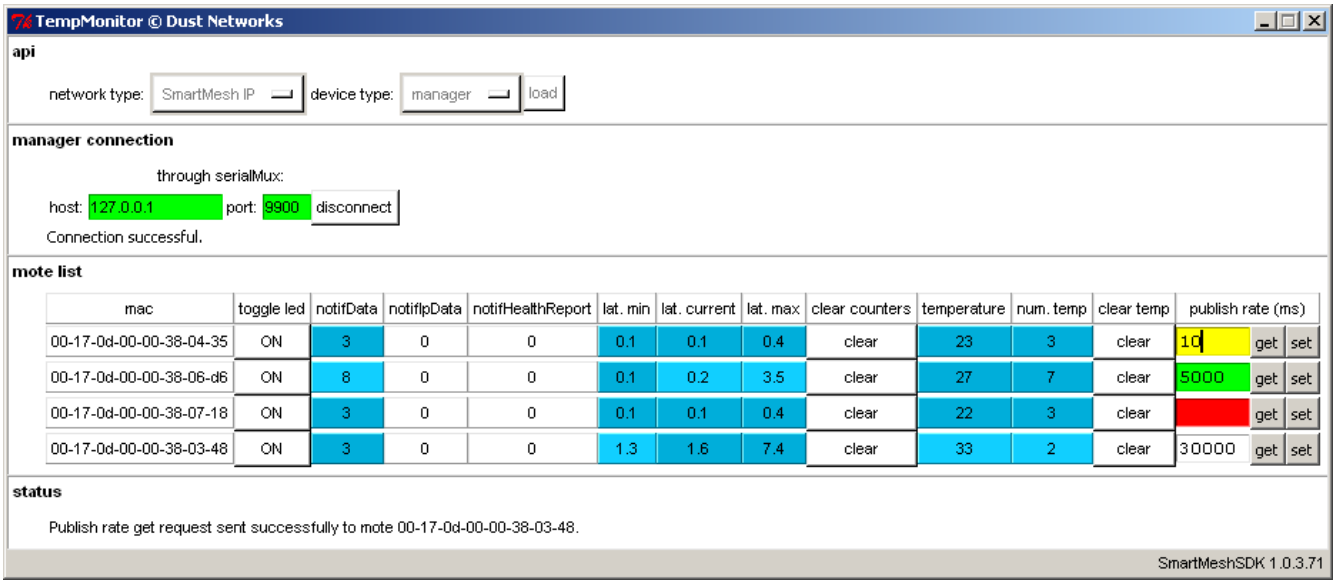
TempMonitor Connects to the Manager and interacts with motes running in **master** mode.

You can run the TempMonitor application:

- by double-clicking on the Windows executable at `/win/ TempMonitor.exe`
- by double-clicking on its source files at `/src/ bin/TempMonitor/TempMonitor.py`

Description

An example of the TempMonitor application window is shown below, after the user has connected to a SmartMesh IP Manager. Aside from the API and connection tabs, the application appears the same for SmartMesh WirelessHART Managers.



Description

TempMonitor is a GUI based application which connects to the manager, lists the motes in the network, and enables you to interact with the subset of them running in **master** mode.In particular, you can:

- Toggle the INDICATOR_0 LED of a mote (supported by SmartMesh IP motes only - clicking the **ON** button in the toggle led field for a SmartMesh WirelessHART mote will change the button text to "N.A.")
- Monitor the number of data packets received (*notifData*) from each mote
- Monitor the number of health reports received (*notifHr*) from each mote
- Monitor packet latency information
- Retrieve temperature and set the temperature publish rate on any mote

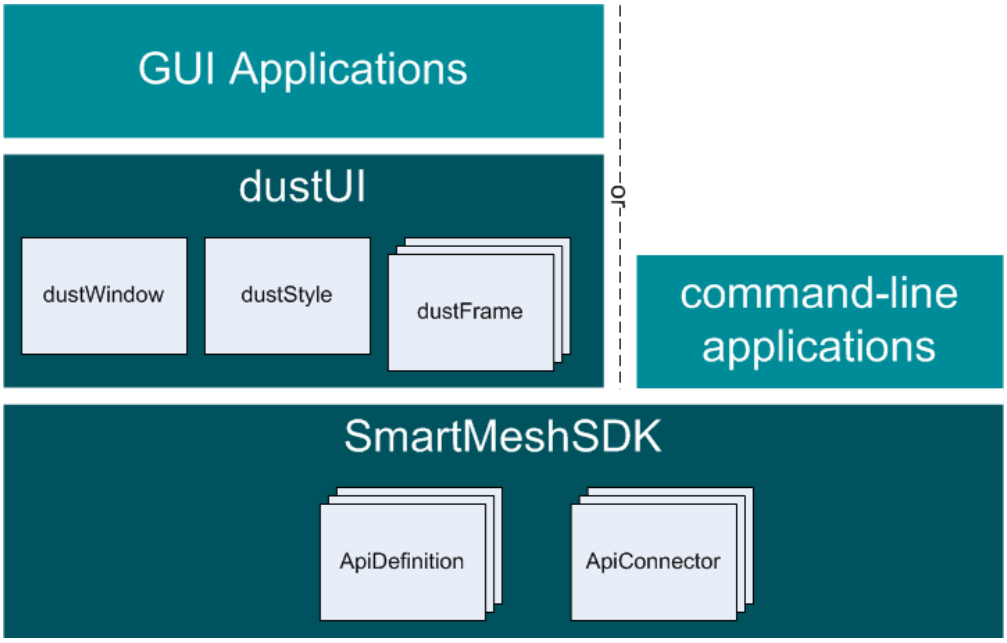
Use this application to:

- Generate occasional downstream traffic by toggling the LED on the motes (supported by SmartMesh IP motes only)
- Generate continuous upstream traffic by setting temperature publish rate

Internally, TempMonitor interacts with a small application on the motes that uses a Dust developed application protocol called OAP. This mote-resident application is only enabled when the mote is in **master** mode.

6.6 Architecture

6.6.1 Overview

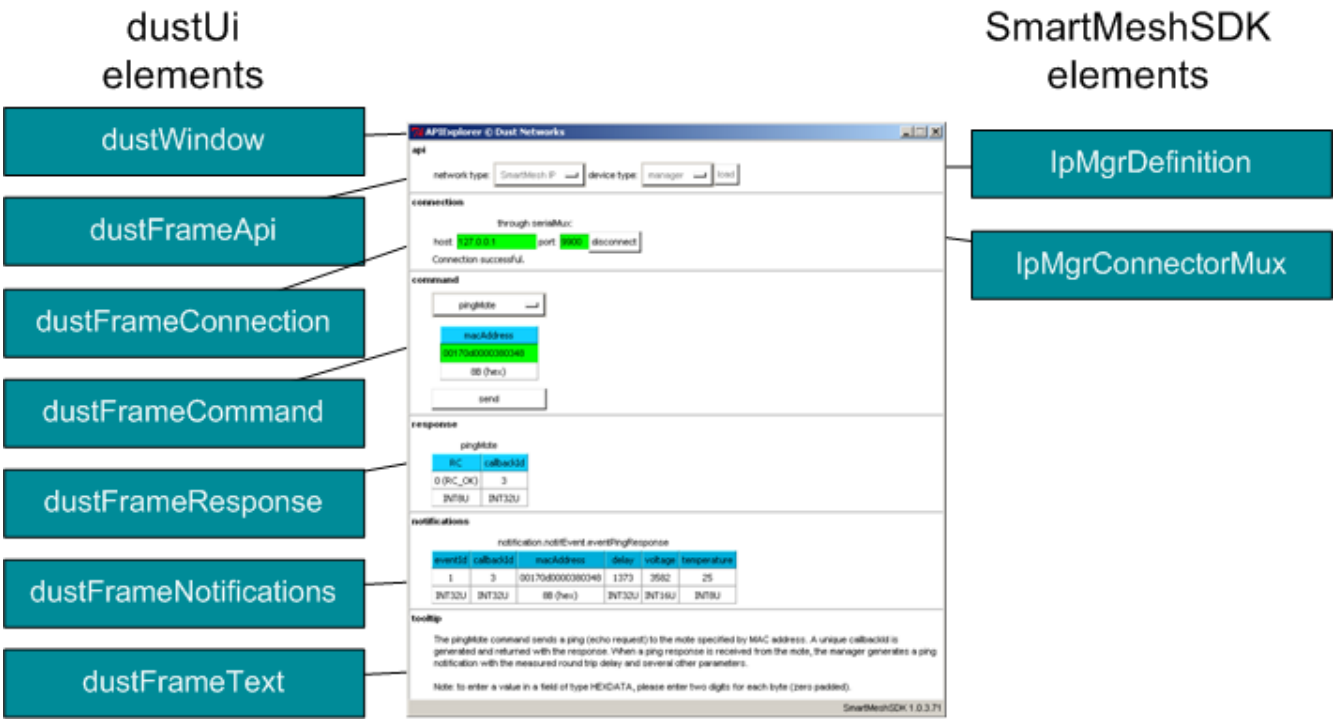


The SmartMesh SDK consists of three layers:

- The **SmartMeshSDK** layer provides a collection of ApiDefinitions and ApiConnectors:
  - an **ApiDefinition** defines the commands, responses and notification of a given API, as well as the functions to manipulate those.
  - an **ApiConnector** is used to connect to a physical device, over some transport mechanism.
- The **dustUI**layer is a library of visual elements to help build GUI applications. It consists of:
  - the **dustWindow** representing the main window of you application
  - **dustStyle**, a common stylesheet used through the dustUI library
  - a collection of **dustFrame** GUI elements
- This allows for two types of applications:
  - a **command line application** sitting directly on top the SmartMeshSDK layer
  - a **GUI application** that composes a window with of multiple dustFrame elements and interconnects them

6.6.2 An Example: Structure of the APIExplorer Application

The figure below shows the internal structure of the APIExplorer application.



The main application manages the different frames. Running the APIExplorer applications consists of the following steps:

1. The user selects the API definition in the **api** frame. This definition (here `IpMgrDefinition`) is returned to the main application, which passes it to the other frames.
2. The user selects the API connection in the **connection** frame. This connect (here `IpMgrConnectorMux`) is returned to the main application, which passes it to the other frames.
3. The **command** frame builds the drop-down menus of commands dynamically by exploring the API definition loaded.
4. The **command** frame uses the connector created to send and receive commands to the device.
5. Responses are received by the main application, and sent to the **response** frame for display.
6. Notifications are received by the main application and stored in global variables. The **notifications** frame periodically polls these variables and displays their contents.
7. The main application displays information in the **tooltip** frame whenever useful.

> ⚠ The dustFrame elements never call each other. It's the main application's role to pass information around among frames.

6.7 dustUI Library

6.7.1 Overview

The dustUI library is a collection of GUI (Graphical User Interface) elements which can be assembled to form an application. It is based on Tkinter, Python's de-facto standard GUI package. The modules of the dustUI library are in the `/src/SmartMeshSDK/dustUI/` folder of your SmartMesh SDK installation.

6.7.2 Module Description

Although the modules described below are meant to be included in applications, you can also **run them as standalone applications** by double-clicking on their source file. This starts a dummy application which shows what the GUI element looks like.

dustStyle.py

This module sets the look-and-feel of all dustUI GUI elements. Customizing the looks of your application is as simple as editing this file.

dustWindow.py

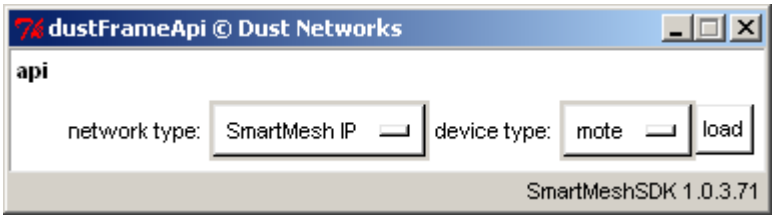
The Window of each application, which includes:

- the Dust Networks logo and copyright sign
- the name of the application at the top
- the version of the SmartMesh SDK displayed at the bottom

dustFrame.py

This is the parent class for all dustFrames, and is meant to be inherited.

dustFrameApi.py

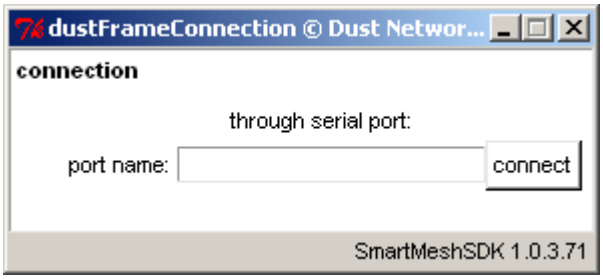


Specify which API definition to load.

dustFrameCommand.py

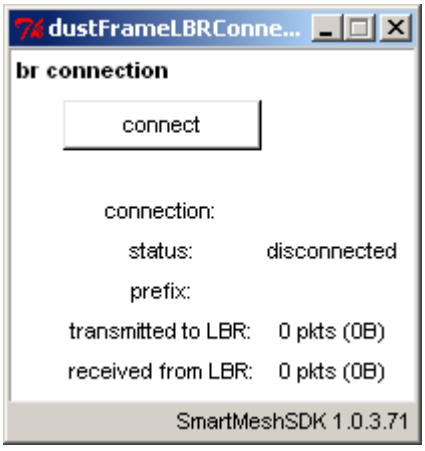
Browse the available commands and send them.

dustFrameConnection.py



Select the type of connector and connect to the device.

dustFrameLbrConnection.py



Connect to the LBR.

dustFrameFields.py

A parent class.

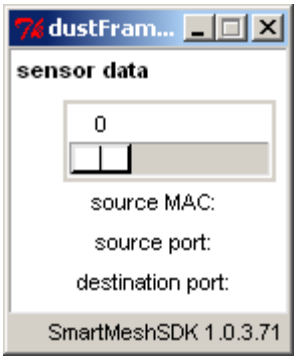
dustFrameResponse.py

Display the fields contained in a response.

dustFrameNotifications.py

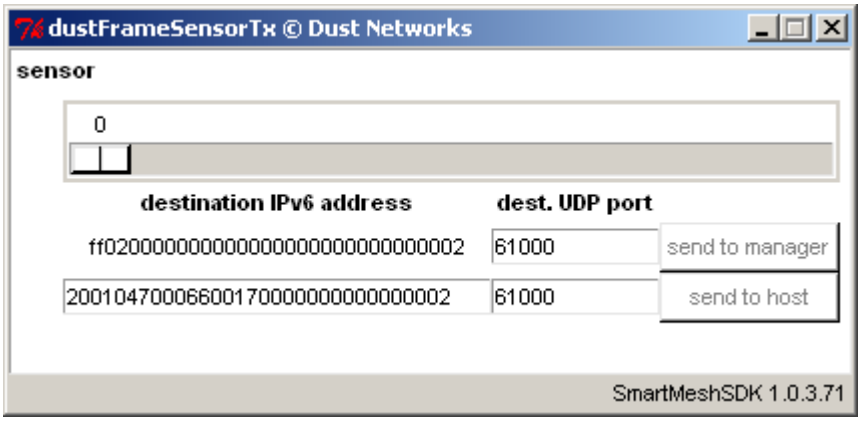
Display the fields contained in a notification.

dustFrameSensorData.py



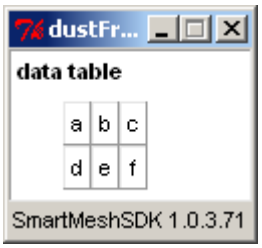
Display sensor data.

dustFrameSensorTx.py



Send sensor data.

dustFrameTable.py



Display a list of lists as a 2D table.

dustFrameTooltip.py

Display arbitrary text.

6.8 SmartMeshSDK Library

6.8.1 Overview

A collection of modules to connect to all Dust Networks devices and implement all related Application Programming Interfaces (APIs).

The SmartMeshSDK lives in the `/src/SmartMeshSDK/` directory of your SmartMesh SDK installation.

It consists of two sets of modules:

- API definitions, which define the API of a given device.
- API connectors, which allow to physically connect to a device.

6.8.2 Module Description

API Definitions

All API definitions live in the `/src/SmartMeshSDK/ApiDefinition/` directory.

`ApiDefinition.py` is the parent class of all API definitions, from which the following modules inherit.

module name	API definition for
IpMgrDefinition.py	SmartMesh IP Manager
`IpMoteDefinition.py`	SmartMesh IP Mote
`HartMgrDefinition.py`	SmartMesh WirelessHART Manager
`HartMoteDefinition.py`	SmartMesh WirelessHART Mote

API Connectors

`ApiConnector.py` is the parent class of all API connectors, from which the following modules inherit.

module name	connects to	connects over
`IpMgrConnectorMux`	SmartMesh IP Manager	SerialMux
`IpMgrConnectorSerial`	SmartMesh IP Manager	serial
`IpMoteConnector`	SmartMesh IP Mote	serial
`HartMgrConnector`	SmartMesh WirelessHART Manager	XML-RPC
`HartMoteConnector`	SmartMesh WirelessHART Mote	serial

7 Interacting With a Network

7.1 Introduction

This walkthrough is a step-by-step tutorial to familiarize yourself with a SmartMesh WirelessHART network. It guides you from switching on the SmartMesh WirelessHART Manager and SmartMesh WirelessHART Motes for the first time, to sending data. No prior knowledge about wireless sensor networking or Dust Networks products is required.

- In A First Network, you will switch on your network with default settings and use Admin Toolset to watch the network form.
- In Interacting with the Manager, you will log into the SmartMesh WirelessHART Manager over an SSH session and use the APIExplorer application to extract information about the SmartMesh WirelessHART Manager and the SmartMesh WirelessHART Motes connected to it.
- In Interacting with a Mote, you will use the APIExplorer application to control a SmartMesh WirelessHART Mote, mimicking the behavior of an external micro-controller. You will have that mote join a network and send data to the SmartMesh WirelessHART Manager.

7.2 A First Network

7.2.1 Overview

In this tutorial, you will use the SmartMesh WirelessHART Manager's Admin Toolset web interface to list the motes in your network. It assumes that you have previously followed the installation steps outlined earlier in this document.

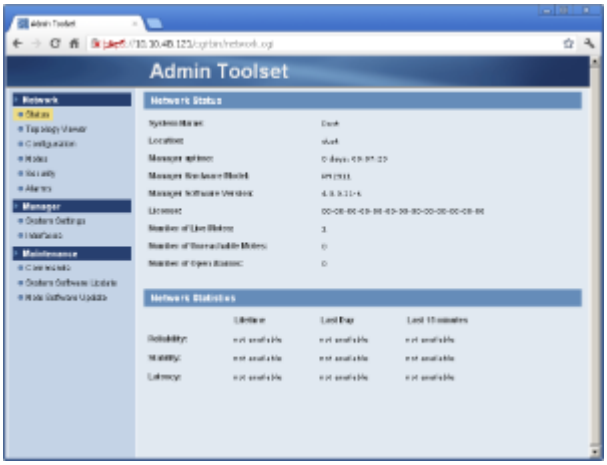
7.2.2 What You Need

- A SmartMesh WirelessHART Starter kit and an Ethernet cable, a USB-micro cable, and a DB9 serial cable.
- A serial terminal client to configure mote mode if changed from default
- A supported web browser

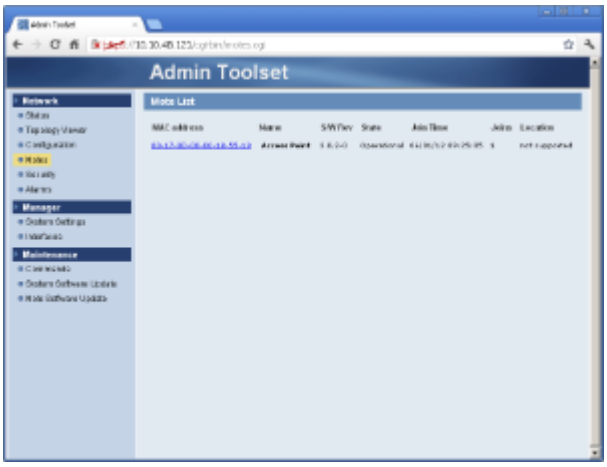
7.2.3 Steps

1. Switch off all the motes.
2. Connect the SmartMesh WirelessHART Manager to your network and obtain its IP address (see the Connecting to the Manager section of the SmartMesh WirelessHART User's Guide if you don't already know it.

3. On your computer, use your Web browser to navigate to the IP address of your SmartMesh WirelessHART Manager, i.e. enter `https://<manager IP address>/` in your browser's address bar. Use the following credentials to log in:
    - *Username*: `system`
    - *Password*: `system`
4. The following page opens:



5. Follow the **Motes** link to see the list of motes in the network. This list currently only contains the Access Point.

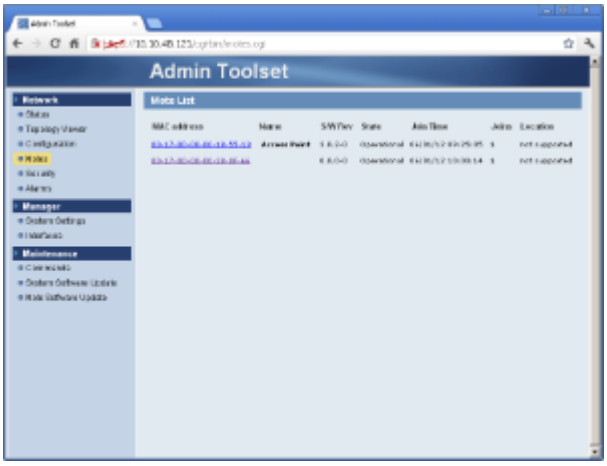


If you previously switched a mote to slave mode to use with APIExplorer or TempMonitor, return it to **master** mode:

```
> set mode master
> reset
```

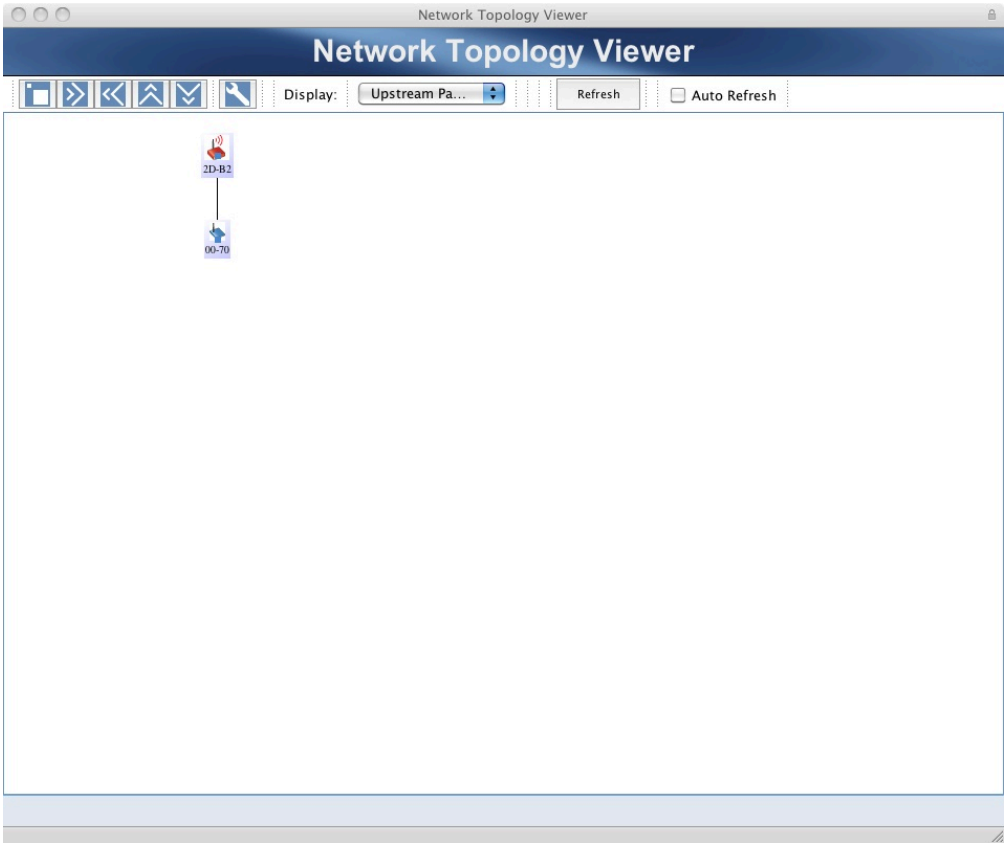
⚠ Motes in the starter kit (DC9007) ship in **master** mode. Mote modes and how to switch between them are discussed in Troubleshooting section of this guide and also in the SmartMesh WirelessHART User's Guide.

6. After resetting the mote, it will join the network, after which you will see it appear in the **Mote List** in Admin Toolset (you will need to refresh the page to see newly joined motes).



7. Turn on all your motes and watch them join

8. You can also use the Topology Viewer application to see the mesh. Note this may require enabling Java in your browser.



You are now ready to evaluate the performance of a SmartMesh WirelessHART network - see the application note How to Evaluate Network and Device Performance for recommendations.

Common Problems

Can't connect to Admin Toolset

- Is the SmartMesh WirelessHART Manager switched on?
- Did you use the correct IP address of the Manager, and use https?

No motes appear in AdminToolset

- Is the SmartMesh WirelessHART Mote switched on?
- Is the mote in **master** mode? See the Troubleshooting section of this guide, or the SmartMesh IP User's Guide for details on mote modes.

- Is the mote configured to the same Network ID as the Manager? See the Troubleshooting section of this guide for instructions on verifying/changing the Network ID.

7.3 Interacting with the Manager

7.3.1 Overview

In this tutorial, you will interact with the SmartMesh WirelessHART Manager (LTP5903CEN-WHR) over CLI using a Terminal Client, and via API, using the APIExplorer application.

Interacting with Manager CLI

The LTP5903CEN-WHR has two ways to communicate with its CLI:

- The serial port, to interact on the console using serial terminal software.
- The Ethernet port, to interact via SSH on a suitable terminal program, such as PuTTY.

Here we will connect to the manager CLI via serial port:

1. Connect your serial terminal client to the CLI port of the SmartMesh WirelessHART Manager (settings 115200 baud, 8 data bits, No parity, 1 stop bit, no flow control).
2. Log into Linux by entering the following username and password:
   - *Username*: `dust`
     Password: `dust`
3. This will bring you to the Linux prompt. At the Linux prompt ($), enter the following command:

Username: `dust`

Password: `dust`

```
dust@manager:~$ nwconsole
```

4. Enter the manager CLI user name and password. The default user name and password is:

Username: `admin`

Password: `admin`

5. Type `help` to get the list of available commands.

```
SmartMesh Manager 4.0.0.21
Manager UpTime: 0 days, 06:15:19


> help
help <command>
Commands:
   get
   set
   delete
   exec
   help
   logout
   su
   onechan
   deleteSessions
   ping
   otap
   show
   sm
   trace
```

6. Use the `sm` command (for "show motes") to obtain the list of connected motes. When the manager first starts, only the internal Access Point (mote 1) will be connected. When motes join the network, they will appear in this list.

```
> sm
Current time: 07/17/12 16:46:00 ASN: 2275333
Elapsed time: 0 days, 06:19:20
      MAC                 MoteId  Age Jn      UpTime   Fr Nbrs Links State
00-17-0D-00-00-19-2D-B2   ap  1        1      06:19:13  6    1    51 Oper
00-17-0D-00-00-30-00-70       2    11  1      06:10:12  2    1    32 Oper
```

7. Use the `show mote` command to get information about a specific mote.

```
> show mote 2
00-17-0D-00-00-30-00-70     2  23 Oper SW: 1.0.0-104 HW: 0 (*)
   Location is supported
   Upstream hops: 1, latency: 1.585, TTL: 127
   SourceRoute: Dist(Des):  1.2(0) Prim: 1,2 Sec:
   Power Source: battery
   Advertisement Period: 20.000
   Bandwidth:
      output planned / current       :   1.0000 /   1.0000
      global service / delta         :   1.0000 /   0.0000
      local service goal / current   :   1.0000 /   1.0000
      guaranteed for services / child:   1.0000 /   0.0000
      free                           :   0.0000
      Planned BW per parent: #1 -  1.0000
      Mote services BW:   1.0000
         Problem/Fix Motes 65535/65535 status 0 cur ID 65535
         Default    BW:   1.0000
   Neighbors: 1 (max 32). Links: 32 (max 200)
   Links per second: 3.417969 (limit: 21.329132)
   Frame: 0. Neighbors: 1. Parents: 1. Links rx:0, tx:31.
      Broadcast links
          0.  1. 0:  tdb
      -> #1 T Links 30/30/30 RSSI: -75 Q: 0.86
   Frame: 1. Neighbors: 1. Parents: 1. Links rx:1, tx:0.
      Broadcast links
      -> #1   Links 1/1/1 RSSI: -75 Q: 0.86
```

8. Use the `show stat short 0` command to get current statistics about your network.

```
> show stat short 0
It is now .................. 07/17/12 16:48:02.
This interval started at ... 07/17/12 16:30:00.
There are 26 valid 15 minute intervals stored.
NOTE: you may be expecting packets that are still in transit!
--------------------------------NETWORK STATS---------------------------------
PkArr  PkLost  PkTx(Fail/ Mic/ Seq)  PkRx  Relia.  Latency   Stability
    3       0    35(   6/   0/   0)    33    100%    1.8s      82.86%
--------------------------------MOTE STATS-----------------------------------
 Id Rx    Lost  Tx    Rx    Fwd   Drop  Dup   Ltncy Jn Hop avQ mxQ me ne Chg  T
  2     3     0     3     0     0     0     0   1.8  0   1   0   2  0  0 267 23
--------------------------------PATH STATS-----------------------------------
MoteA MoteB ABPower BAPower  ABTx(Fail)  ABRx  BATx(Fail)  BARx   Stab.
   1     2    -75     -47       0(   0)     0    35(   6)    33  82.86%
```

Once the network has formed, there will be information about each mote and the network as a whole.

9. When you're done, use `logout` to log out.

```
> logout
```

Tip #1

For more details on the manager CLI and interacting with the manager, refer to:

- SmartMesh WirelessHART User's Guide
- SmartMesh WirelessHART Manager CLI Guide

Tip #2

After obtaining the manager's IP address, you can also connect to the manager's CLI via SSH. Try it!

Interacting with API using APIExplorer

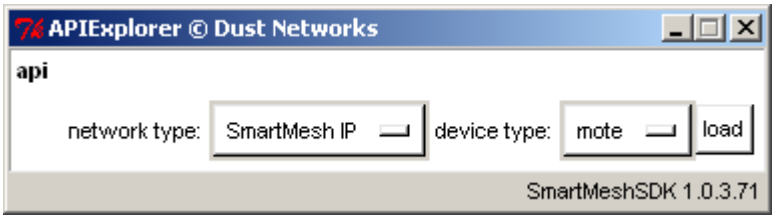
A Word on the SmartMesh SDK

The SmartMesh SDK is a Python package which simplifies the integration of a SmartMesh IP or SmartMesh WirelessHART network into your application. It implements the Application Programming Interface (API) of the device it is connected to. A set of sample applications are included in the SmartMesh SDK, allowing a programmer to quickly understand the API and use it as part of a larger system.

In this section, you will connect to the SmartMesh WirelessHART Manager over Ethernet and interact with its API using the SmartMesh SDK.

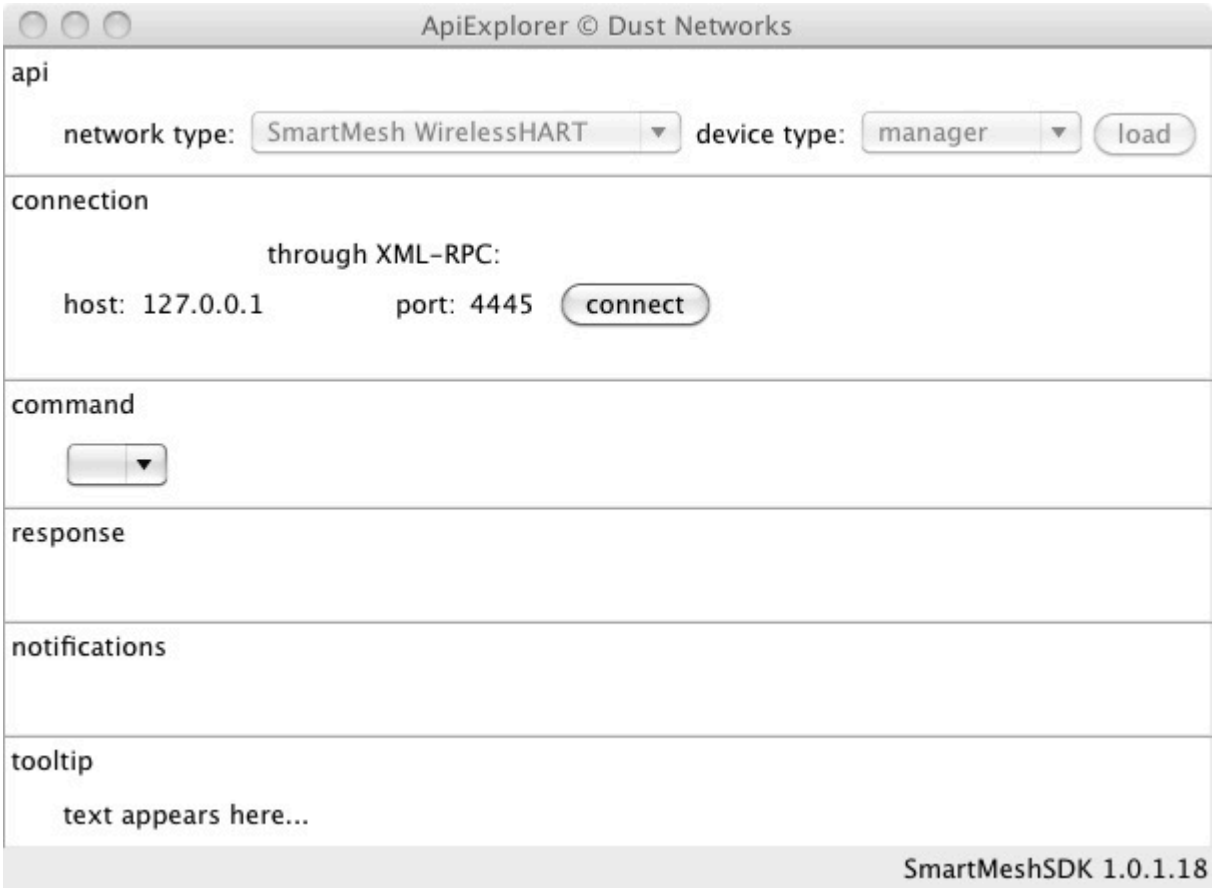
Connect to the Manager

1. Make sure your SmartMesh WirelessHART Manager is connected to your computer.
2. In the `SmartMeshSDK` directory, double click on the `bin/APIExplorer/APIExplorer.py` Python script. This opens the APIExplorer's window.

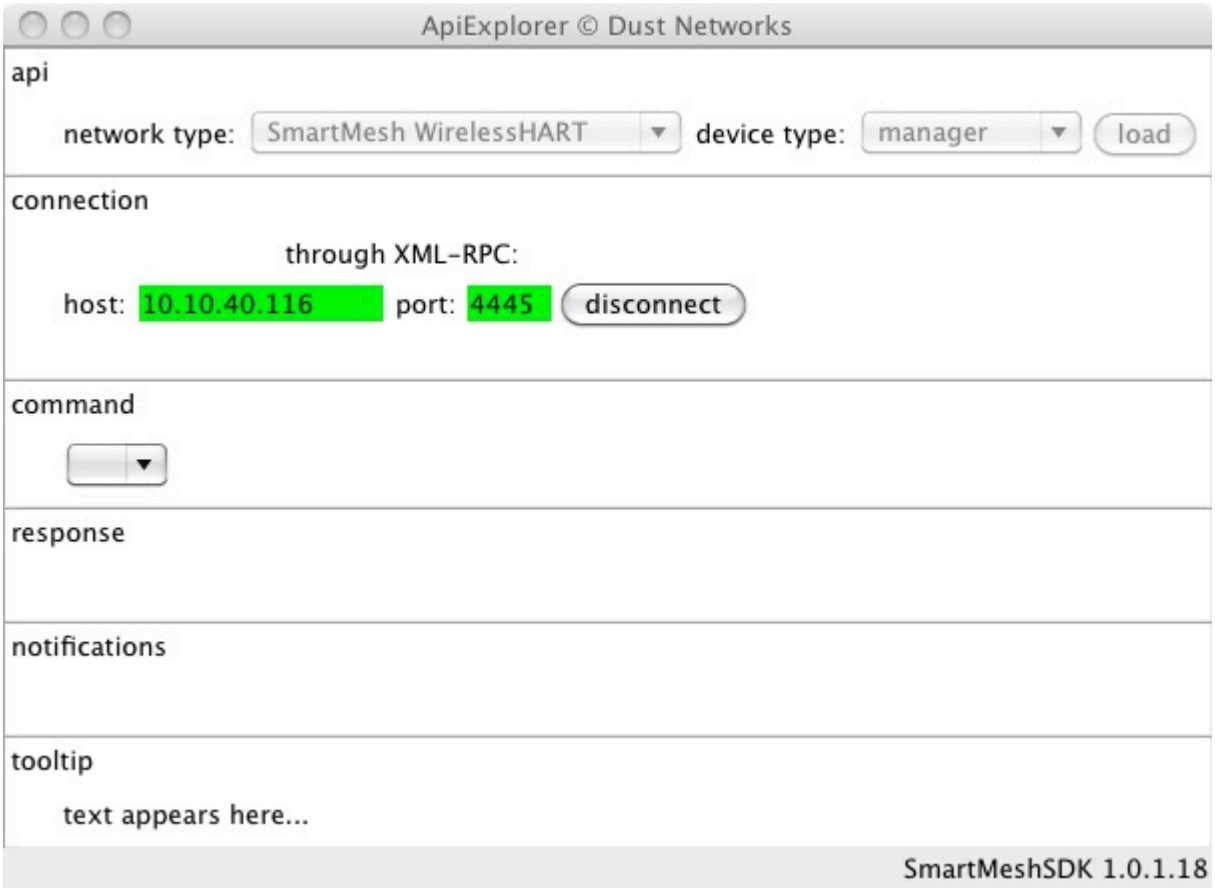


3. Tell the application you want to connect to a SmartMesh WirelessHART Manager by selected the following:
   - *network type*: `SmartMeshWirelessHART`
   - *device type*: `manager`

4. Click the **load** button.

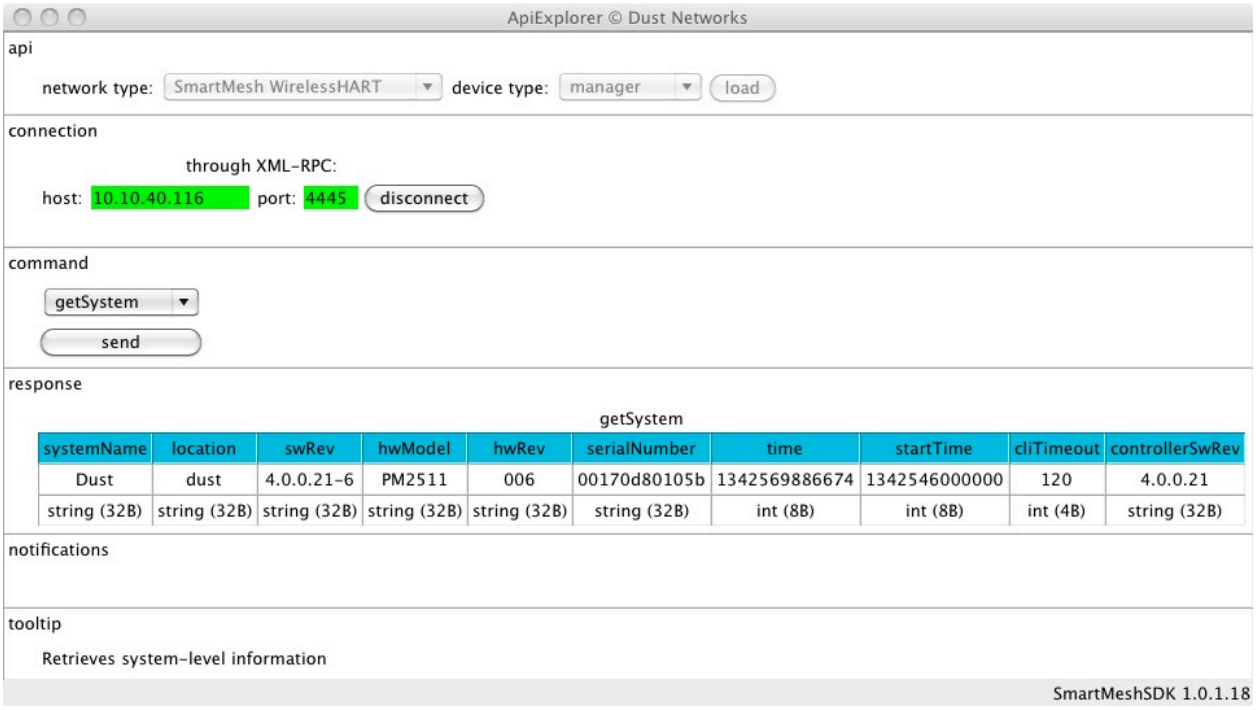


5. In the **connection** frame, you are presented with two options to connect to the SmartMesh WirelessHART Manager; we will connect through XML-RPC using the IP address you configured in Setup for the host address.
6. Click **connect**. The fields turn green indicating the connection is successful.


ApiExplorer © Dust Networks
api
network type:
SmartMesh WirelessHART
device type:
manager
load
connection
through XML-RPC:
host:
10.10.40.116
port:
4445
disconnect
command
response
notifications
tooltip
text appears here...
SmartMeshSDK 1.0.1.18

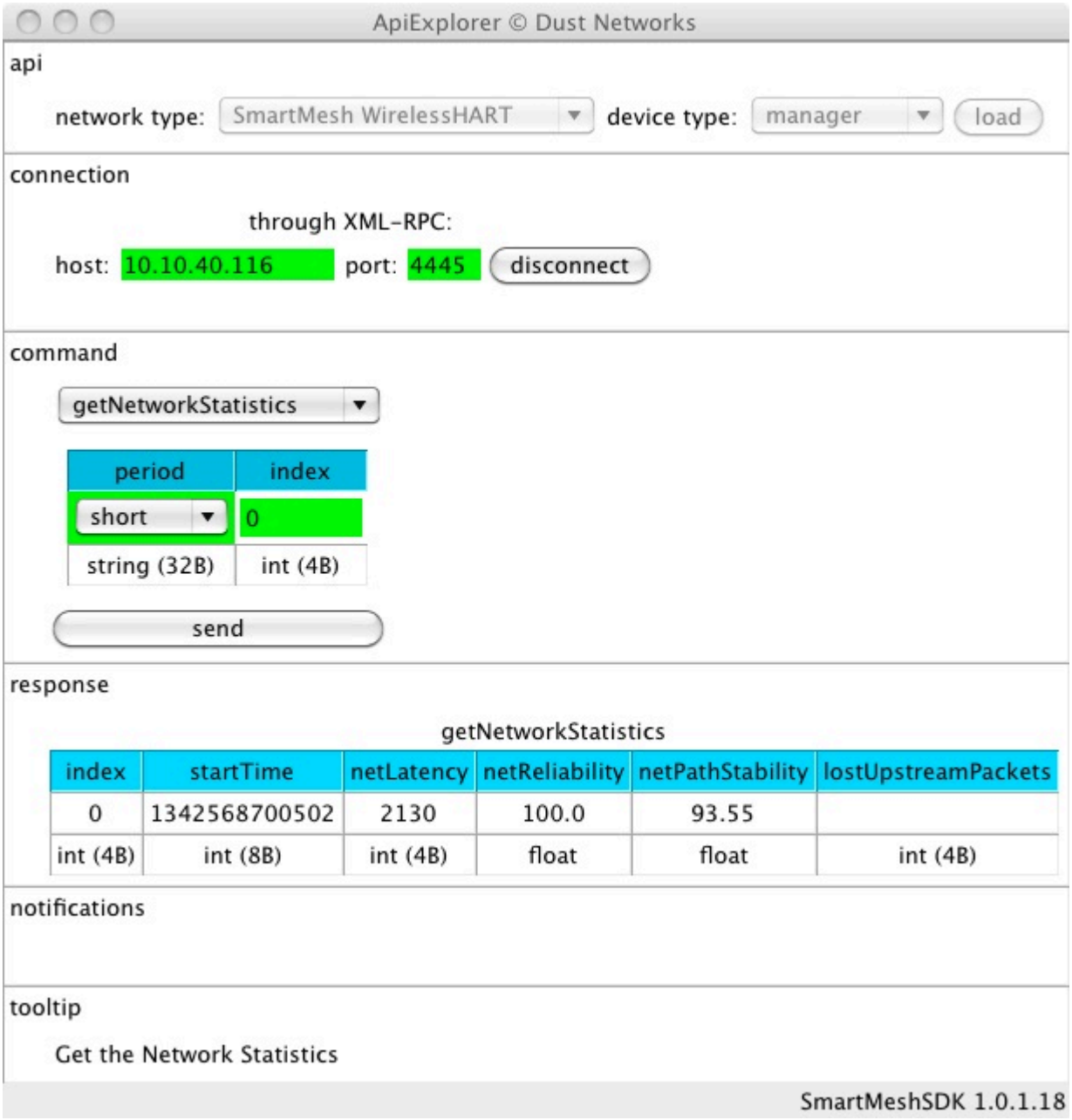
Obtain Information about the Manager and the Network

1. The drop-down menu in the **command** frame lists all the commands defined in the SmartMesh WirelessHART Manager API Guide. Select *getSystem* and press **send**. The response prints in the **response** frame, and contains the name, value and format for each field.



2. The image above indicates:
   - The Ethernet MAC address of the SmartMesh WirelessHART Manager is `00170d80105b`
   - Information about the hardware and software of the SmartMesh WirelessHART Manager

1. In a similar way, issue a *getNetworkStatistics* command to obtain information about the mesh network connected to the manager (this is the same info we obtained via manager CLI earlier).

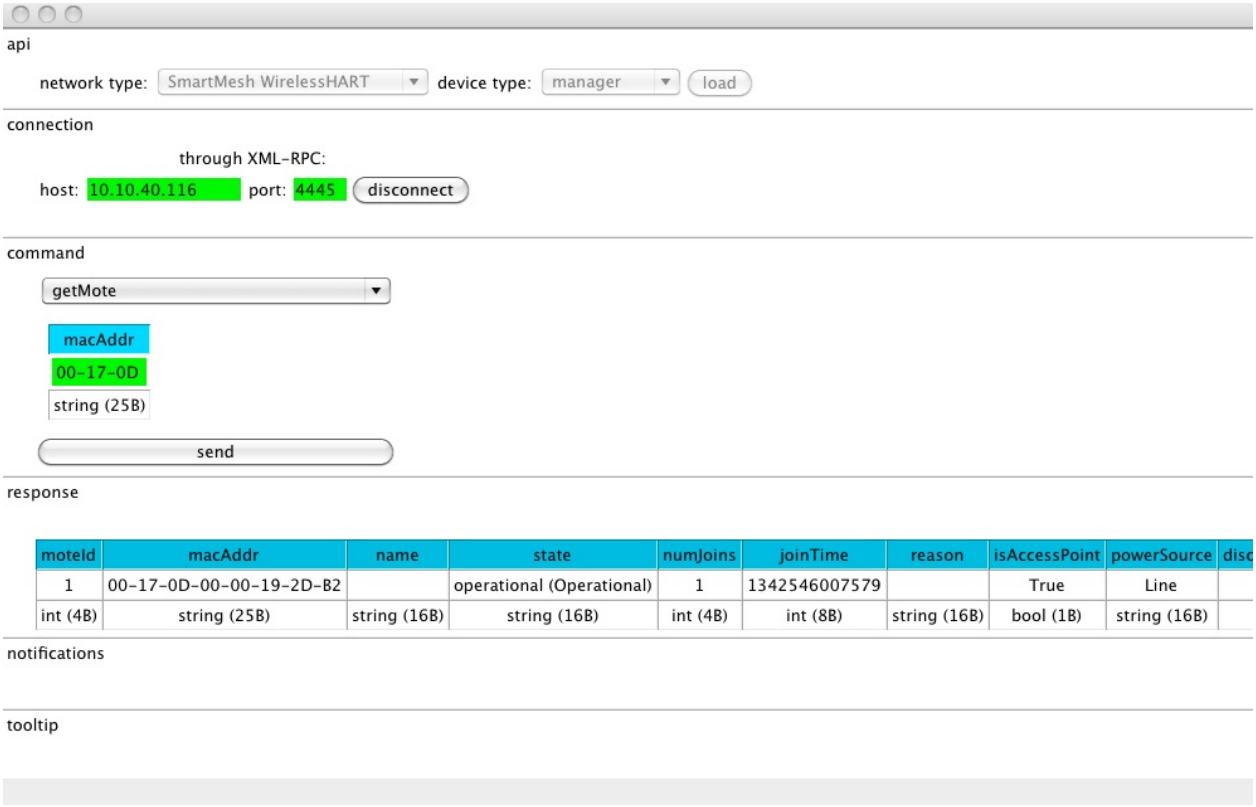


2. The image above indicates that:
   - Latency for the period was ~2 s
   - Reliability was 100%
   - Stability was 93% (up from the earlier manager CLI measurement)

Consult the SmartMesh WirelessHART Manager API Guide for details about all commands and fields.

Obtain Information about a Mote

1. Select the *getMote* command and enter the 8-byte (16-character) address (EUI-64) of the manager's access point (mote 1) in your network - you can get this through the manager CLI via the *show mote 1* command.
2. After pressing *send*, the manager returns information about the first device in the network (screencap cropped for readability) - note the field *isAccessPoint* is True.



3. Make note of the value in the **mac** field, which contains the MAC address of the Access Point, in our case `00170d0000192DB2`.

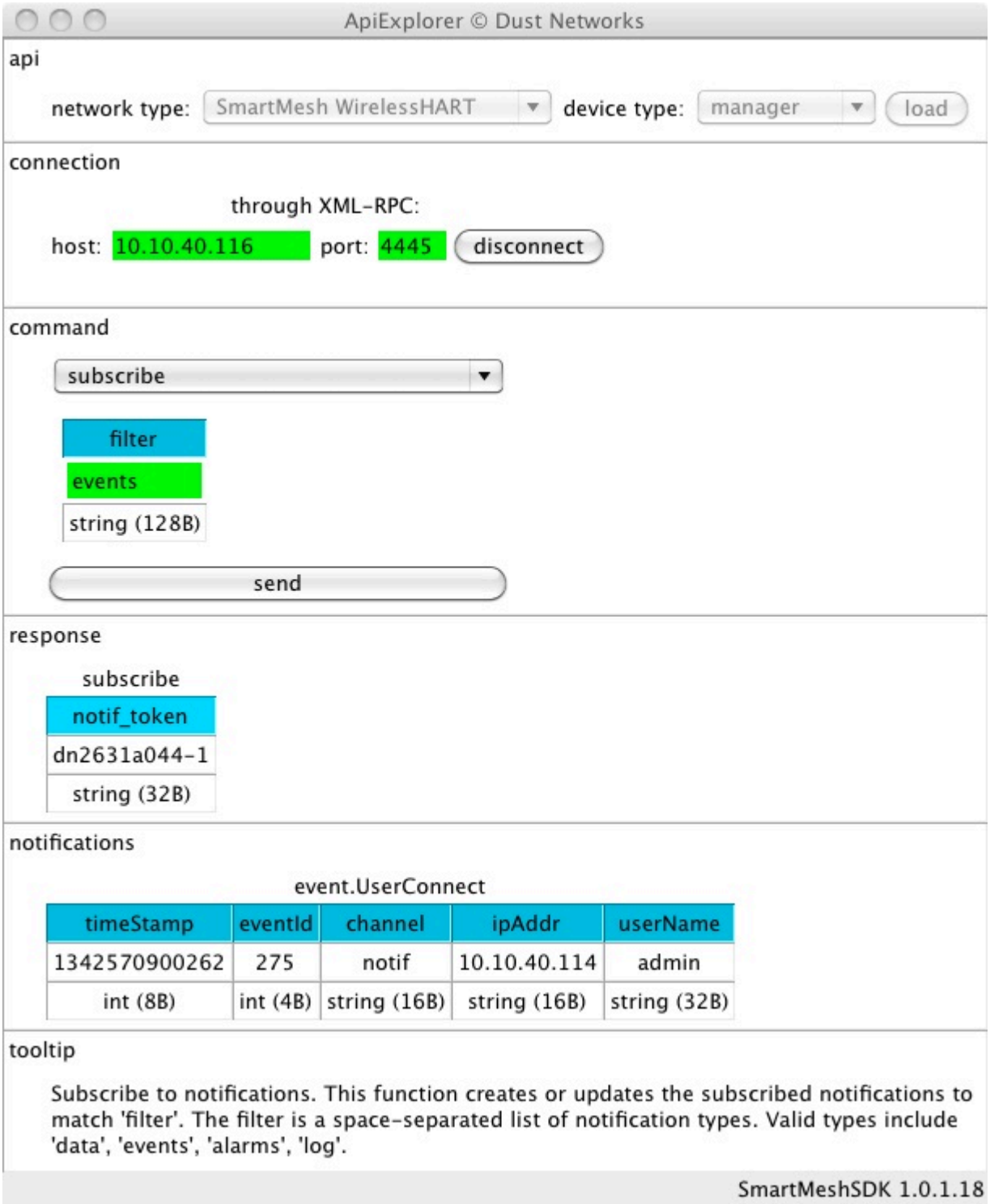
Subscribe to Notifications

In the sections above, you have sent *commands* to the device, which has answered immediately with *responses*. When an event happens, the device can also send you *notifications* immediately, without waiting for you to ask for it. There are a number of different types of notifications, as detailed in the SmartMesh WirelessHART Manager API Guide.

By default, the manager will not send you any notifications; you need to use the *subscribe* command to specify which types of notifications you'd like to receive.

1. Select *subscribe* and set the filter to **events**:
   Press **send** to subscribe to event notifications.

2. If you have a network running, you will see notifications appear in the **notifications** frame.



For more information

For more details on the API and interacting with the manager, refer to:

- SmartMesh WirelessHART User's Guide
- SmartMesh WirelessHART Manager API Guide

7.3.2 Common Problems

I get no output over the Manager CLI

- Is the device switched on?
- Have you connected your serial terminal to the manager's CLI port?
- Have you configured your serial terminal to the correct setting?

The serial setting for the manager CLI (DB-9, labeled "serial 2") is:

115200 baud, 8 data bits, No parity, 1 stop bit, no flow control

7.4 Interacting with a Mote

7.4.1 Overview

In this step, you will interact with the SmartMesh WirelessHART Mote (DC9003A-C+ DC9006) over the mote's CLI using a Terminal Client, and via API, using the APIExplorer application.

The SmartMesh WirelessHART Mote has two serial ports:

- the CLI port to interact directly over a serial terminal
- the API port to interact using the SmartMesh SDK

Interacting with the Mote CLI

1. Open your serial terminal client on the CLI port of the SmartMesh WirelessHART Mote (settings 9600 baud, 8 data bits, No parity, 1 stop bit, no flow control)

2. Type `help` to get the list of available commands

```
> help
help <command>
Commands:
   mstacks
   mtrace
   mtracesv
   mset
   mget
   minfo
   mseti
   mgeti
   mclearnv
   reset
   set
   get
   radiotest
   trace
   loc
   info
```

3. Use the `minfo` commands to get network-related information about your SmartMesh WirelessHART Mote.

```
> minfo
HART stack ver 1.0.0 #1
state:       Search
mac:         00:17:0d:00:00:30:00:70
moteid:      0
netid:       293
blSwVer:     9
UTC time:    54:890000
reset st:    0x100
```

4. By default, a mote joins automatically when it boots (this is **master** mode). Use the command `reset` to force a mote to reset. After a few minutes (if there is a SmartMesh WirelessHART Manager running), the mote's CLI indicates the joining steps. While the SmartMesh WirelessHART Mote is joining, you can use the `minfo` command to see its state evolve.

```
> minfo
HART stack ver 1.0.0 #1
state:       Oper
mac:         00:17:0d:00:00:30:00:70
moteid:      2
netid:       293
blSwVer:     9
UTC time:    1342572899:226000
reset st:    0x100
```

For more information

For more details on the mote's CLI and interacting with a Mote, refer to:

- SmartMesh WirelessHART User's Guide
- SmartMesh WirelessHART Mote CLI Guide

Interacting with API using APIExplorer

SmartMesh SDK

The SmartMesh SDK is a Python package which simplifies the integration of a SmartMesh WirelessHART Mote into your sensor/actuator device. It implements the Application Programming Interface (API) calls that an OEM microprocessor would normally exercise over the serial UART interface on the mote. A set of sample applications are included in the SmartMesh SDK, allowing a programmer to quickly understand the API and use it as part of a larger system.

In this section, you will connect to the SmartMesh WirelessHART Mote over a serial port (a COM port in Windows) and interact with its API using the SmartMesh SDK.

Connect to the Device

1. Make sure your SmartMesh WirelessHART Mote is connected to your computer.

2. Make sure your SmartMesh WirelessHART Mote is operating in **slave** mode by issuing the following command on the mote CLI:

```
> set mode slave
> reset
```

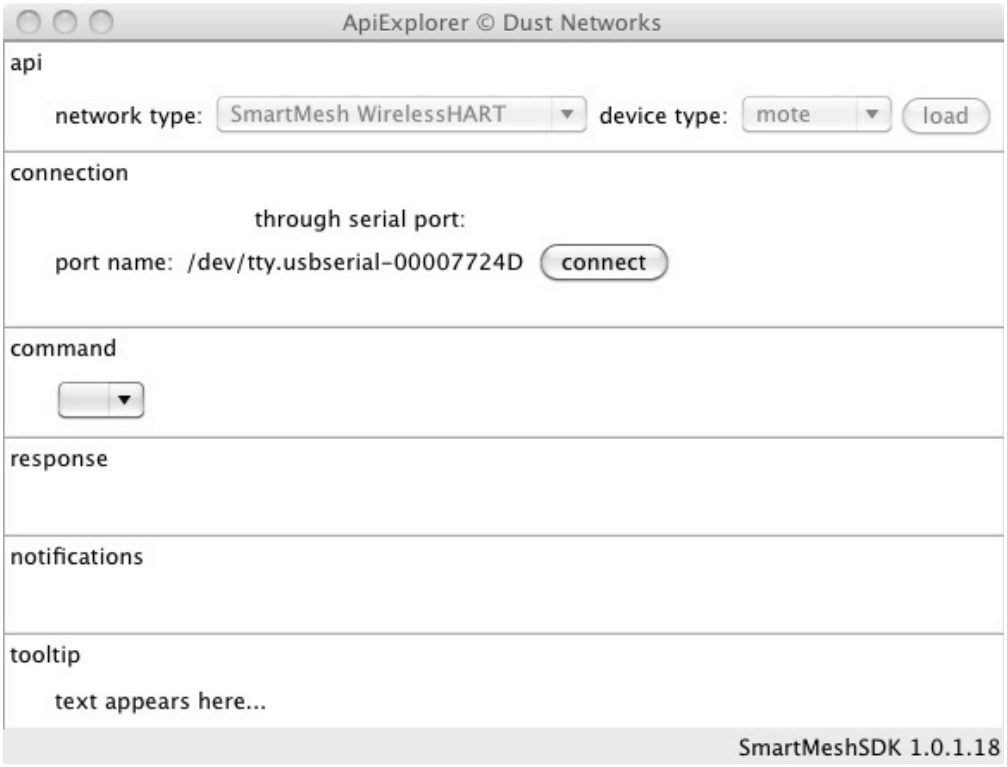
> ⚠ Motes in the starter kit (DC9007) ship in **master** mode. Mote modes and how to switch between them are discussed in the Troubleshooting section of this guide and also in the SmartMesh WirelessHART User's Guide

3. In the `SmartMeshSDK` directory, double click on the `bin/APIExplorer/APIExplorer.py` Python script. This opens the APIExplorer's window.

1.

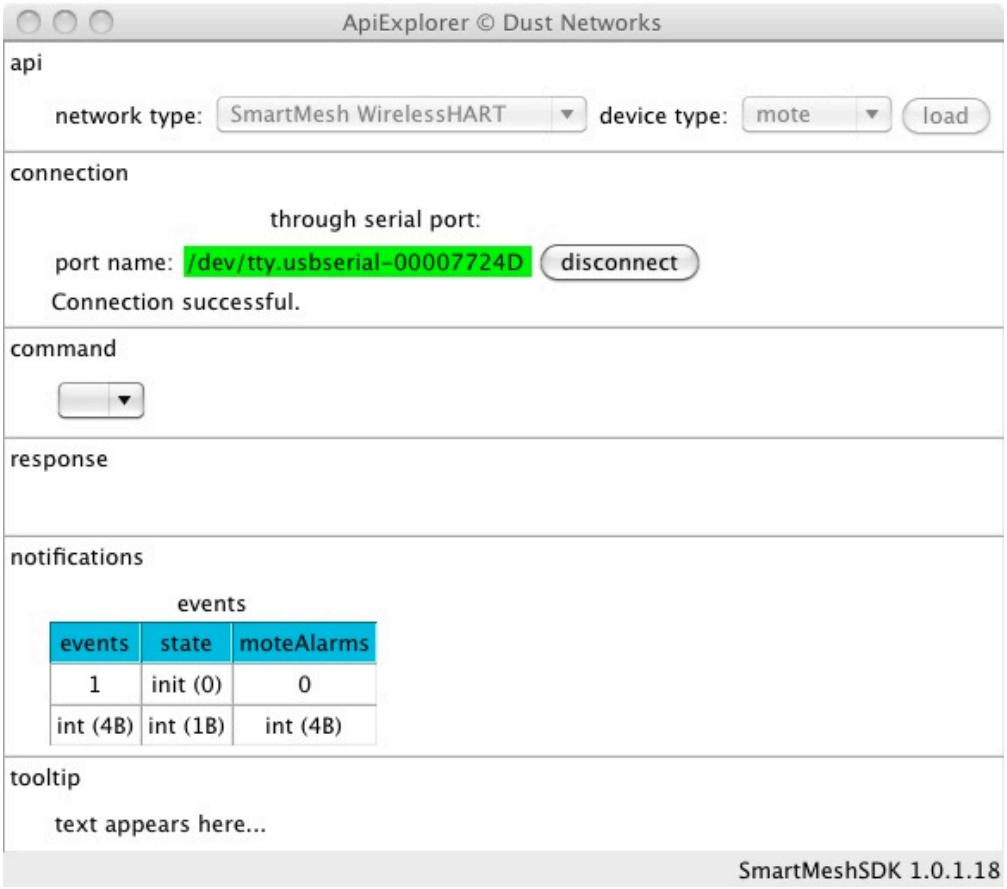
2. Tell the application you want to connect to a SmartMesh WirelessHART Mote by selected the following:
   - *network type*: `SmartMeshWirelessHART`
   - *device type*: `mote`

3. Click the **load** button. It will display a window similar to the following:



4. In the **connection** frame, enter the following:
   - *port name*: your SmartMesh WirelessHART Mote's API port number (note that this will be a COMx port on windows. This example is a Unix usb-to-serial port)

5. Click **connect**. The fields turn green indicating the connection is successful.



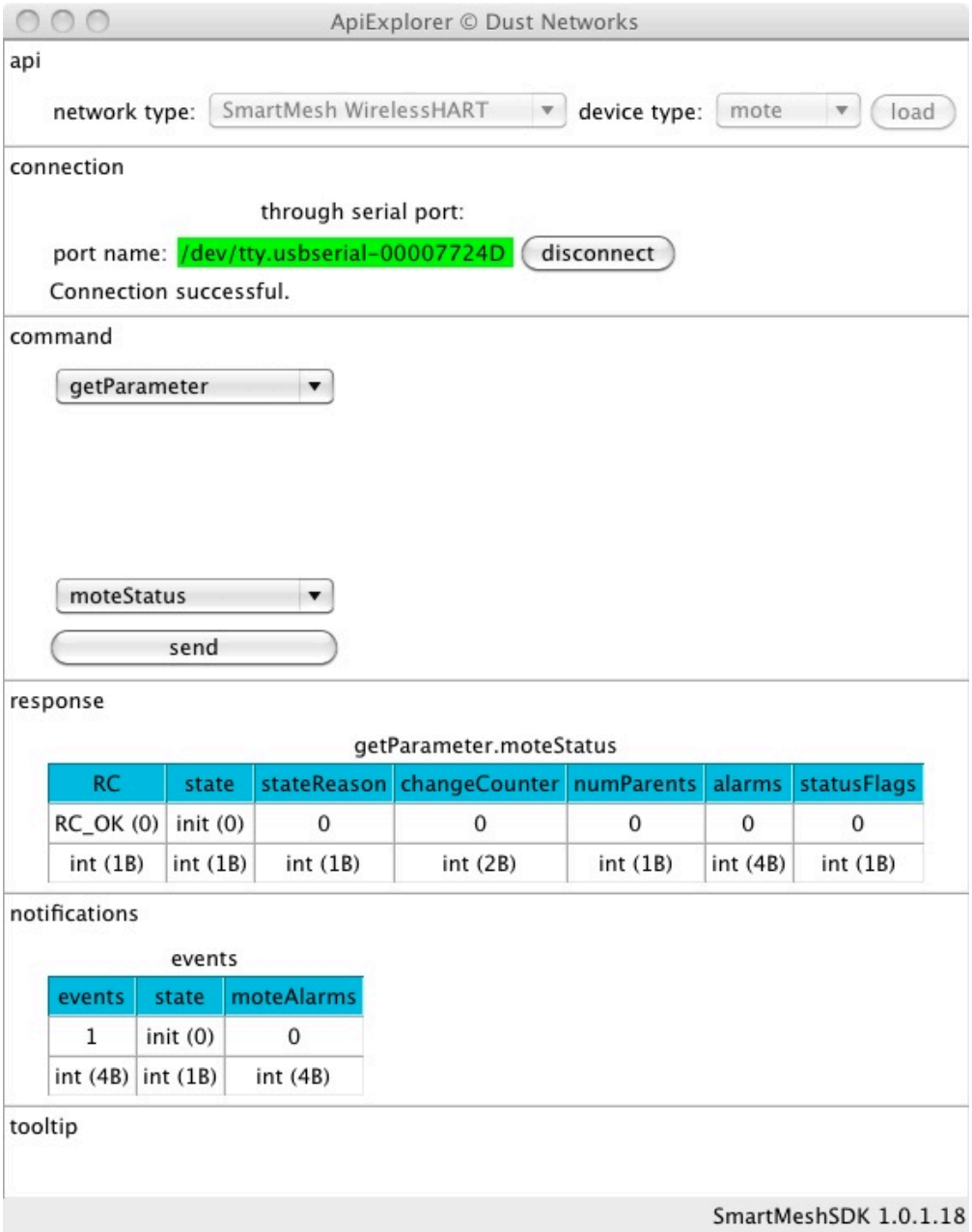
Getting Notifications from the Mote

Unlike for the SmartMesh WirelessHART Manager, you do not need to subscribe to receive notifications when using the SmartMesh WirelessHART Mote.

When a SmartMesh WirelessHART Mote boots, it sends a boot event notification periodically until it is acknowledged by an external device listening on the API serial port, e.g. the APIExplorer application. The the mote boot event (event 0x01) is displayed in the screen shot above.

Obtain Information About the Mote

1. The drop-down menu in the **command** frame lists all the commands defined in the SmartMesh WirelessHART Mote API GuideIP Mote Serial API Guide. Select *getParameter,* then *moteStatus* and press **send**. The response prints in the **response** frame, and contains the name, value and format for each field.



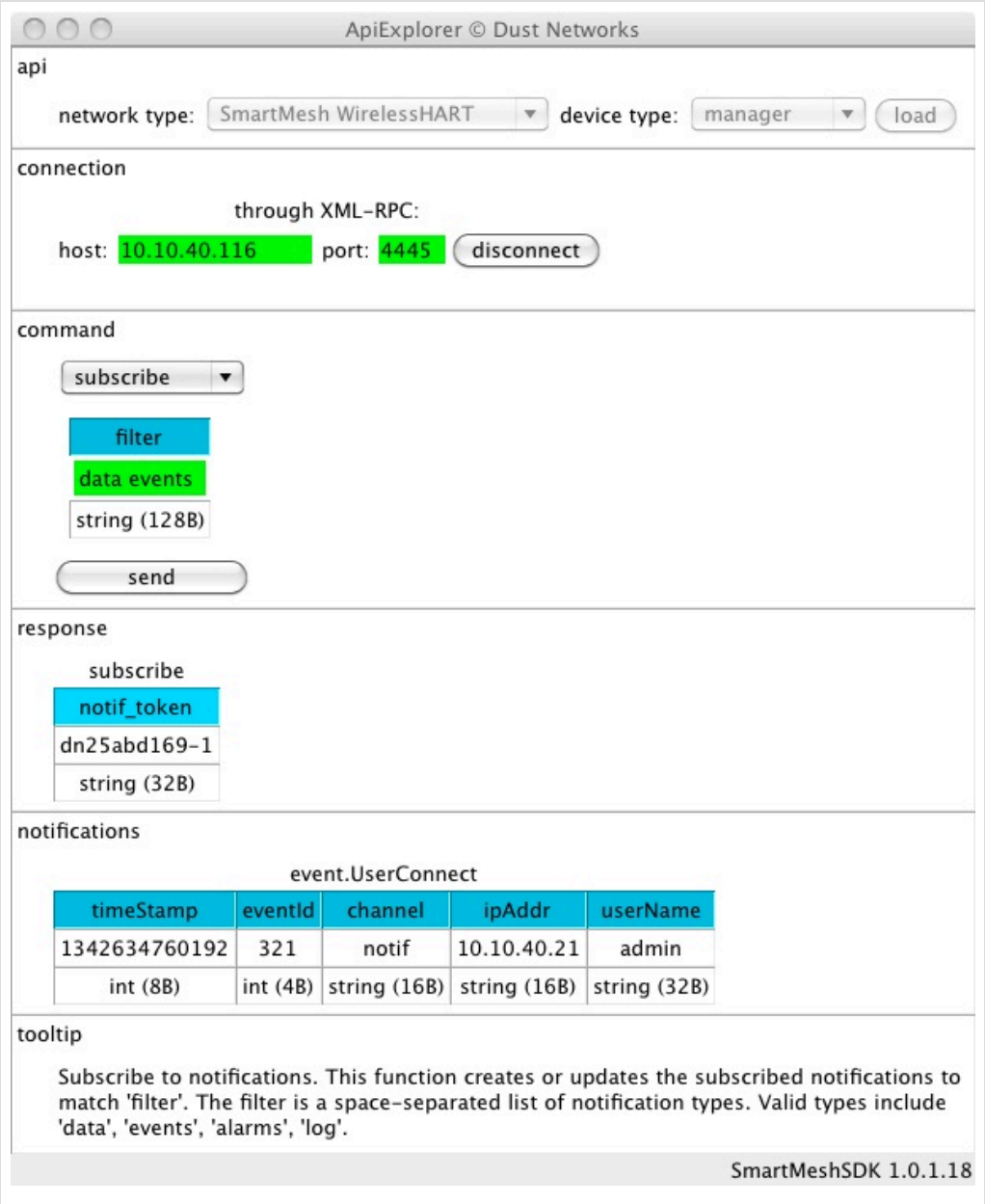
2. The image above shows the response to *getParameter.motestatus*, and that the mote is in the **Init** state, i.e. it is not trying to join a network.
3. Similarly, issue a *getNvParameter.networkId* command to verify that your SmartMesh WirelessHART Mote is configured with the correct Network ID.

Have the Mote Join the Network

1. Start a first APIExplorer application, and connect it to the SmartMesh WirelessHART Manager.
2. If it's not already done, start a second APIExplorer application, and connect it to the SmartMesh WirelessHART Mote.

3. For easier reading, we recommend that you display the two APIExplorer windows side-by-side.
   Reset the SmartMesh WirelessHART Mote by calling its *reset* command. After a few seconds, you should receive a *boot* (event 0x01) event notification at the SmartMesh WirelessHART Mote.

4. At the SmartMesh WirelessHART Manager, *subscribe* to data and event notifications. If you have a network running, you may receive notifications from time to time.



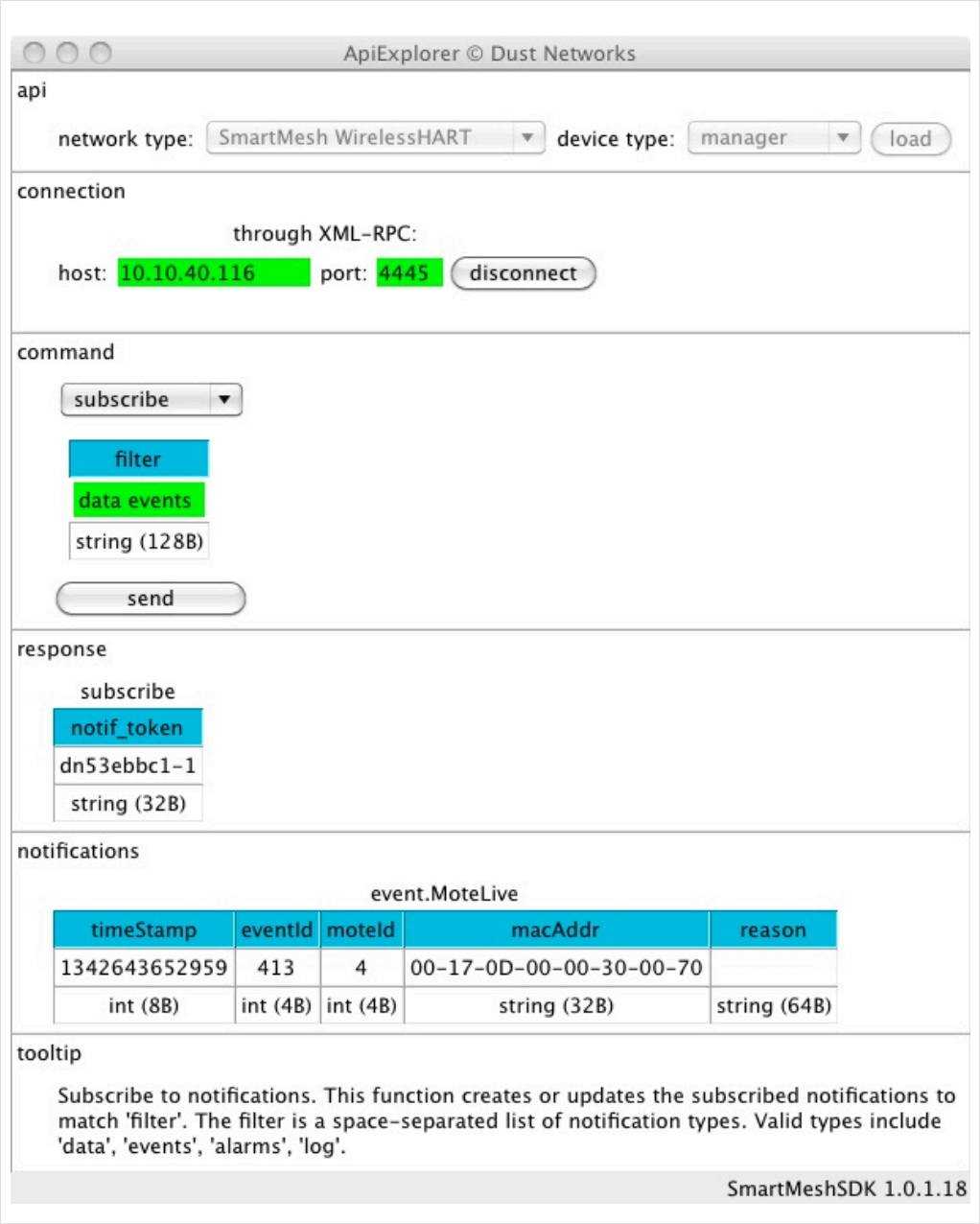
At the SmartMesh WirelessHART Mote, call the *join* command. This causes it to begin searching for the network. As it moves through the join process, you will see the following notifications on both the SmartMesh WirelessHART Mote and the SmartMesh WirelessHART Manager (they may come too quickly to see!):

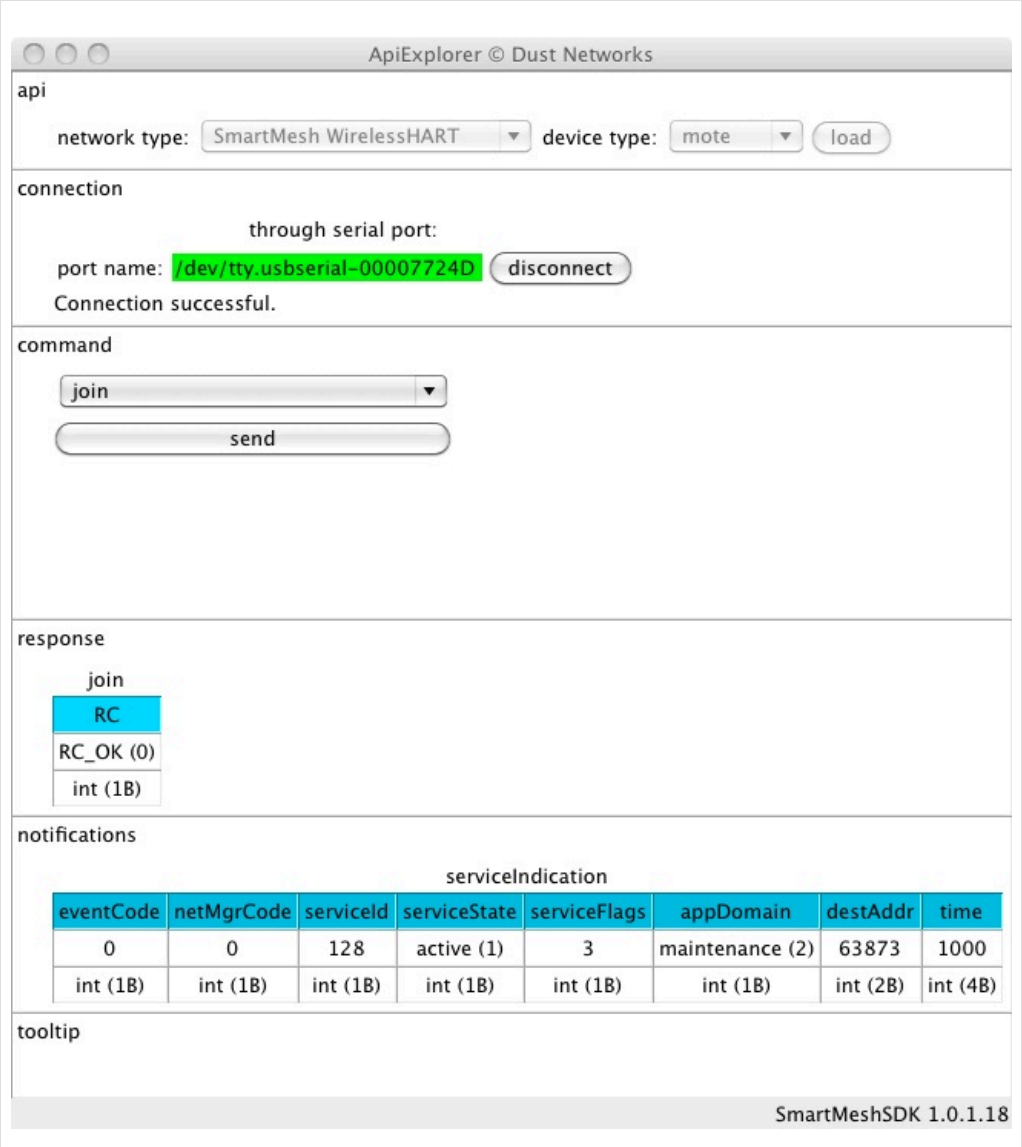
5.

at the SmartMesh WirelessHART Manager		at the SmartMesh WirelessHART Mote	
notification	**explanation**	**notification**	**explanation**
`netMoteJoin`	received a join request		
`netPathCreate`	created a path for the new SmartMesh WirelessHART Mote		
`netPathActivate`	added the path to the AP		
		`Operational`	the SmartMesh WirelessHART Mote is part of the network.
`netMoteLive`	the path was installed correctly, the SmartMesh WirelessHART Mote is considered operational		
		serviceIndication	the SmartMesh WirelessHART Mote has received its base bandwidth (a "maintenance" service)

You can also monitor the mote's status by issuing the *getParameter.moteStatus* mote API command.

After the SmartMesh WirelessHART Mote has joined:


ApiExplorer © Dust Networks
api
network type: SmartMesh WirelessHART
device type: manager
load
connection
through XML-RPC:
host: 10.10.40.116
port: 4445
disconnect
command
subscribe
filter
data events
string (128B)
send
response
subscribe
notif_token
dn53ebbc1-1
string (32B)
notifications
event.MoteLive
timeStamp
eventId
moteId
macAddr
reason
1342643652959
413
4
00-17-0D-00-00-30-00-70
int (8B)
int (4B)
int (4B)
string (32B)
string (64B)
tooltip
Subscribe to notifications. This function creates or updates the subscribed notifications to match 'filter'. The filter is a space-separated list of notification types. Valid types include 'data', 'events', 'alarms', 'log'.
SmartMeshSDK 1.0.1.18


ApiExplorer © Dust Networks
api
network type:
SmartMesh WirelessHART
device type:
mote
load
connection
through serial port:
port name:
/dev/tty.usbserial-00007724D
disconnect
Connection successful.
command
join
send
response
join
RC
RC_OK (0)
int (1B)
notifications
serviceIndication
eventCode
netMgrCode
serviceId
serviceState
serviceFlags
appDomain
destAddr
time
0
0
128
active (1)
3
maintenance (2)
63873
1000
int (1B)
int (1B)
int (1B)
int (1B)
int (1B)
int (1B)
int (2B)
int (4B)
tooltip
SmartMeshSDK 1.0.1.18

Have the Mote Request a Service

In this section, you will have the SmartMesh WirelessHART Mote ask for bandwidth to the SmartMesh WirelessHART Manager.

1. At the SmartMesh WirelessHART Mote, issue a *setParameter.service* command to request bandwidth to send one packet every 10 s to the SmartMesh WirelessHART Manager:
    - **serviceId** = `1` (services 0-127 are available for the mote to define)
    - **serviceRequestFlags** = `1` (source)
    - **appDomain** = publish
    - **destAddr** = `f981` (the gateway host address)
    - **time** is `10000` (in milliseconds)

2. After pressing *send*, the mote sends the request and marks the *serviceState* as "requested." The mote will then periodically repoll the manager for the status of the service. The SmartMesh WirelessHART Manager receives the request, installs the request bandwidth and signals the requesting SmartMesh WirelessHART Mote that the bandwidth has been installed. This results in a *serviceIndication* event notification at the SmartMesh WirelessHART Mote and the mote marking the service as active.

You can verify the allocated bandwidth by using the *getConfig /motes* command (called *getMote* in APIExplorer) at the SmartMesh WirelessHART Manager - the **allocatedPkPeriod** field shows the total service level for the mote. The *getParameter.service* command at the SmartMesh WirelessHART Mote also shows the status of the service - this API is useful for applications that may miss a notification. Note that the manager may lay in more or less bandwidth than you have asked for.

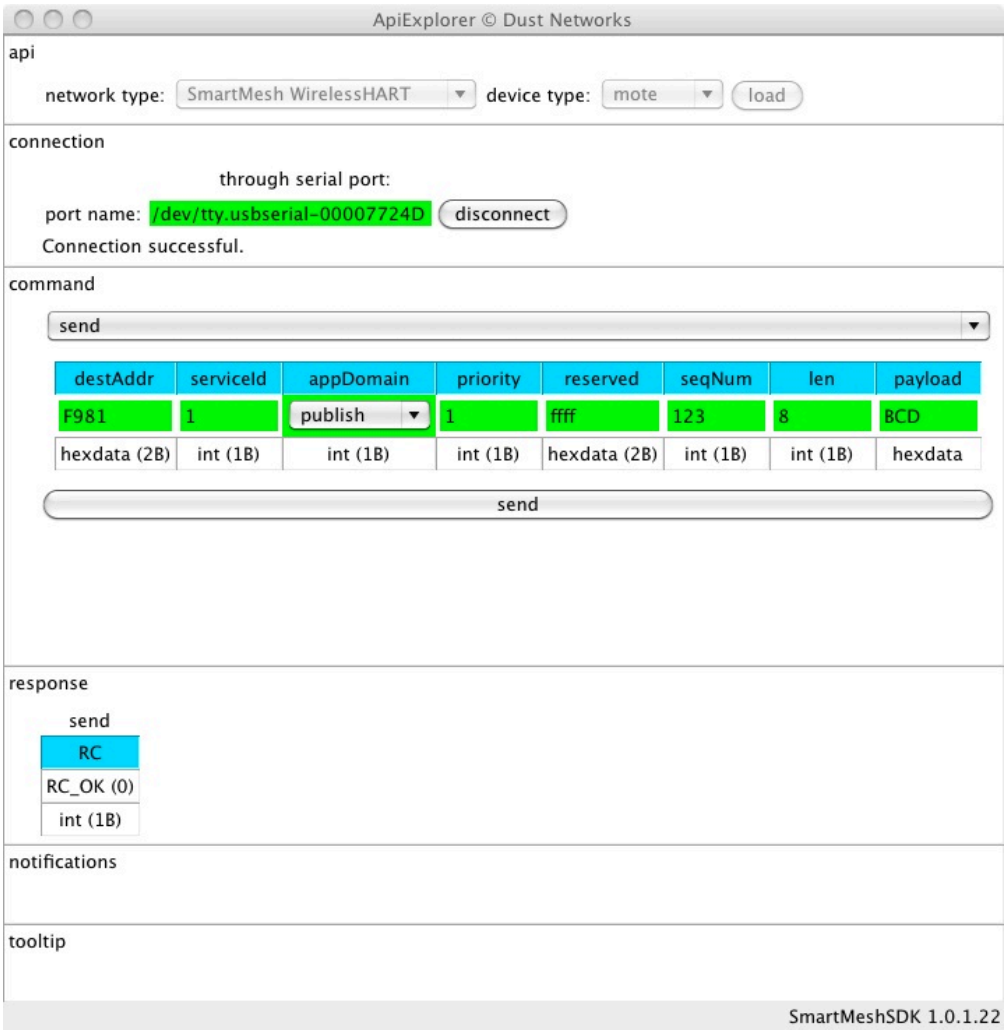
The bandwidth allocated is expressed in period between transmission, i.e. a small number indicates a larger bandwidth. The SmartMesh WirelessHART Manager uses a safety margin when allocating bandwidth, so the period really allocated is smaller that the one requested.

The APIExplorer command menu lists all *getConfig()* and *setConfig()* commands by a shorter nickname - e.g. *getConfig /Network/ChannelBlackList* is called *getBlackist*, and *getConfig /motes/Statistics/avgLatency* is called *getLatency*.

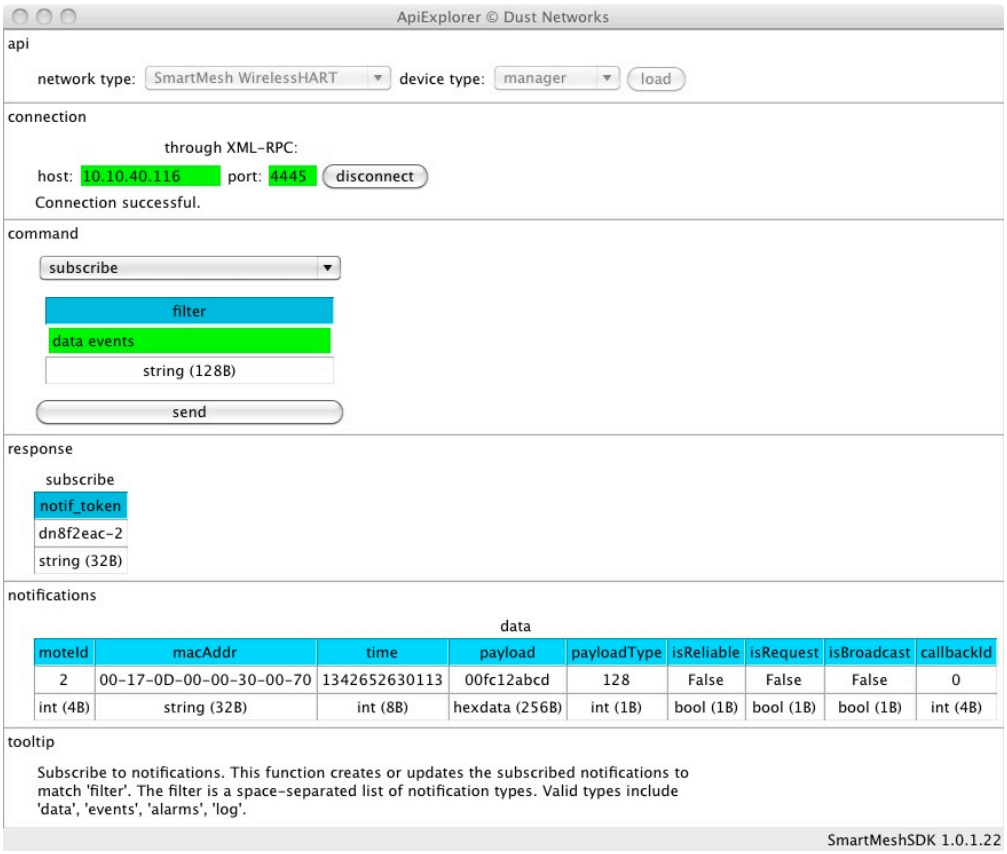
Have the Mote Send Data to the Manager

In this section, the SmartMesh WirelessHART Mote sends data to the SmartMesh WirelessHART Manager.

1. At the SmartMesh WirelessHART Mote, use the *send*command to create an send a packet to the gateway.
    - **destAddr** = `F981`
    - **serviceId** = `1`
    - **appDomain** = `publish`
    - **priority** = `1`
    - **reserved** = `FFFF`
    - **seqNum** = `123`
    - **len** = `4` (bytes)
    - **payload** = `00FC12ABCD`



2. At the SmartMesh WirelessHART Manager, the reception of that data packet will trigger a *Data* notification. You can verify that the packet at the SmartMesh WirelessHART Manager corresponds to the packet sent at the SmartMesh WirelessHART Mote



Ping a Mote

In the previous section we sent data from a mote to a manager. Now we will send a command from the manager to a mote. To *ping* a mote consists of sending it a wireless request which could travel multiple hops before reaching the mote. When the mote receives the command, it immediately generates a ping response which may take different hops back to the manager. This is the simplest command you can send to a mote, so it is used to verify that the mote is operating correctly and to measure the round-trip time.

Via the Manager's CLI

Log into the CLI interface of the Manager as described in Interacting With the Manager. Send a `ping` command to the mote 2. The Mote ID can be found by using the `sm` command as shown earlier. The mote with Mote ID=2 will respond with the temperature and voltage, and the Manager will print the round-trip time for this request-response.

```
> ping 2
Sending ping request to mote 2
> Ping response from mote 2, time=2656 msec v=2885 t=23
```

Via the Manager's API

> ⚠ The APIs refer to motes by MAC address, while the CLI often uses Mote ID. To convert between the two, we use the information returned by the `sm` command in CLI.

1. Using the *ping* command, enter the MAC address of the mote you joined earlier in the **macAddr** field and press **send**.
    - The manager returns a callback ID for the ping response, and a *pingReply* event (*eventId* 449) once the mote has responded. Besides round-trip timing information, this reply also contains the mote's temperature, supply voltage and hops the packet took.

> ℹ The manager *ping* command, while similar in function, is not the same as a unix/linux or DOS `ping` command, which results in an ICMP echo command being sent to the device.

For more information

For more details on the API and interacting with a Mote, refer to:

- SmartMesh WirelessHART User's Guide
- SmartMesh WirelessHART Mote API Guide

7.4.2 Common Problems

The application "hangs" when I send a command to the device

If you are connected to a SmartMesh IP Mote or SmartMesh WirelessHART Mote, this happens when the mote is not running in **slave** mode.

See the Troubleshooting section of this guide for a description of the mote modes and how to change them.

The application won't connect to my mote

- Is the mote switched on?
- Have you connected the application to the API port of the device?
- Is another application already connected to that port?

The table below details the serial setting for the mote:

device	serial port number	usage	baudrate	data bits	parity	stop bits
SmartMesh WirelessHART Mote	third*	CLI	9600	8	N	1
	fourth*	API	115200**	8**	N**	1**

*: refers to the serial ports created by the FTDI drivers. See the FTDI driver installation guide.

**: default values.

Trademarks

Eterna, Mote-on-Chip, and SmartMesh IP, are trademarks of Dust Networks, Inc. The Dust Networks logo, Dust, Dust Networks, and SmartMesh are registered trademarks of Dust Networks, Inc. LT, LTC, LTM and  are registered trademarks of Linear Technology Corp. All third-party brand and product names are the trademarks of their respective owners and are used solely for informational purposes.

Copyright

Disclaimer

This documentation is provided “as is” without warranty of any kind, either expressed or implied, including but not limited to, the implied warranties of merchantability or fitness for a particular purpose.

This documentation might include technical inaccuracies or other errors. Corrections and improvements might be incorporated in new versions of the documentation.

Linear Technology does not assume any liability arising out of the application or use of any products or services and specifically disclaims any and all liability, including without limitation consequential or incidental damages.

Linear Technology products are not designed for use in life support appliances, devices, or other systems where malfunction can reasonably be expected to result in significant personal injury to the user, or as a critical component in any life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness. Linear Technology customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify and hold Linear Technology and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Linear Technology was negligent regarding the design or manufacture of its products.

Linear Technology reserves the right to make corrections, modifications, enhancements, improvements, and other changes to its products or services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to Dust Network's terms and conditions of sale supplied at the time of order acknowledgment or sale.