

SmartMesh IP Tools Guide

Table of Contents

1	About This Guide	6
1.1	Related Documents	6
1.2	Conventions Used	8
1.3	Revision History	9
2	Introduction	10
3	Installation	11
3.1	Before You Begin	11
3.1.1	What You Need	11
3.2	Setup	12
3.2.1	System Overview	12
3.2.2	Step 1 - Prepare the Hardware	13
3.2.3	Step 2a - Installing FTDI Serial Drivers	14
3.2.4	Step 2b - Installing Serial Mux	21
3.2.5	Step 2c - Installing the SmartMesh SDK	23
3.2.6	Step 2d - Installing Stargazer	28
3.3	Troubleshooting	30
3.3.1	Linux FTDI Driver installation	30
3.3.2	Macintosh OS X FTDI Driver Installation	31
3.3.3	Trouble Connecting to Manager	32
3.3.4	Not Getting Notifications	33
3.3.5	Changing Network ID	34
3.3.6	Master/Slave	35
4	Serial Terminal Client	37
4.1	TeraTerm	37
4.2	PuTTY	37
4.3	minicom	38
4.4	Microsoft Windows HyperTerminal	38
5	Serial API Multiplexer (Serial Mux)	40
5.1	Overview	40
5.2	Serial Mux Configuration	41
5.2.1	Step 1: Edit the Serial Mux Configuration File	41
5.2.2	Step 2: Restart the Serial Mux Windows Service	43
5.2.3	Advanced Serial Mux Configuration	44
5.2.4	Serial Mux with Multiple Managers	45
5.3	Serial Mux Protocol	48
5.3.1	Basic Operation	48
5.3.2	Protocol	49
5.3.3	Connections	50
5.3.4	Info command	51

5.3.5	Subscriptions and Notifications	51
5.3.6	Disconnection	52
5.3.7	Serial Mux Definitions	52
6	Stargazer GUI	54
6.1	Upgrading Stargazer	54
6.1.1	Remove the Existing Application	54
6.2	Using Stargazer	56
6.2.1	Overview	56
6.2.2	Managing the Network	61
7	SmartMesh IP SDK	80
7.1	About SmartMeshSDK	80
7.2	SmartMeshSDK Features	80
7.3	Document Organization	80
7.4	Folder Contents	81
7.5	Example Applications	81
7.5.1	Running An Application	81
7.5.2	Color Coding	82
7.5.3	Overview	83
7.5.4	APIExplorer	84
7.5.5	DC2126A	88
7.5.6	HrListener	89
7.5.7	InstallTest	90
7.5.8	LBRCConnection	92
7.5.9	LEDPing	94
7.5.10	MgrListener	101
7.5.11	MuxConfig	101
7.5.12	OTAP Communicator	105
7.5.13	PkGen	108
7.5.14	SensorDataReceiver	112
7.5.15	SimpleIPMgr and SimpleIPMote	114
7.5.16	TempMonitor	116
7.5.17	Upstream	118
7.5.18	Xively	122
7.6	Architecture	126
7.6.1	Overview	126
7.6.2	An Example: Structure of the APIExplorer Application	127
7.7	dustUI Library	128
7.7.1	Overview	128
7.7.2	Module Description	128
7.8	SmartMeshSDK Library	133
7.8.1	Overview	133
7.8.2	Module Description	133
8	Interacting with a Network	135

8.1	Introduction	135
8.2	A First Network	135
8.2.1	Overview	135
8.2.2	Common Problems	140
8.3	Interacting with the Manager	142
8.3.1	Introduction	142
8.3.2	Common Problems	153
8.4	Interacting with a Mote	154
8.4.1	Overview	154
8.4.2	Common Problems	178
8.5	Advanced Topics	179
8.5.1	Exercise the API Programmatically	179
8.5.2	Log HDLC Frames	186
8.5.3	Upstream Communication	188
8.5.4	Downstream Communication	193
8.5.5	Internet Integration	198
9	Low-power Border Router	208
9.1	What is an LBR?	208
9.2	Documentation Organization	208
9.3	Overview	209
9.3.1	Goals of an LBR	209
9.3.2	Services	209
9.4	Test Drive	215
9.4.1	Running	215
9.4.2	Demonstration Resources	215
9.5	Installation	216
9.5.1	Requirements	216
9.5.2	Installation Steps	217
9.6	User Guide	222
9.6.1	Security Levels	222
9.6.2	User Account Types	222
9.6.3	Installing the LBR's Keying Material	223
9.6.4	Adding Users	225
9.6.5	Managing Users	231
9.6.6	Backup and Recovery	232
9.7	CLI Guide	233
9.7.1	add	233
9.7.2	backup	234
9.7.3	disconnect	235
9.7.4	help	236
9.7.5	loglevel	237
9.7.6	passwordremove	238
9.7.7	passwordset	239

9.7.8	publickeyremove	240
9.7.9	publickeyset	241
9.7.10	quit	242
9.7.11	remove	243
9.7.12	seclevel	244
9.7.13	status	245
9.7.14	users	246
9.7.15	version	247
10	On-chip Application Protocol	248
10.1	Protocol	248
10.1.1	Packet Format	248
10.1.2	Communication	249
10.1.3	OAP Payload	251
10.1.4	Tag-Length-Value(TLV) Encoding	253
10.1.5	Using OAP to interact with an application	254
10.2	OAP in SmartMesh Motes	256
10.2.1	Introduction	256
10.2.2	Notifications	264
10.3	OAP Examples	266
10.3.1	Turning on the INDICATOR_0 LED	266
10.3.2	Temperature Sample Notification	266
10.3.3	Getting app info	267

1 About This Guide

1.1 Related Documents

The following documents are available for the SmartMesh IP network:

Getting Started with a [Starter Kit](#)

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

User's Guide

- [SmartMesh IP 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 IP 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 IP Manager API Guide](#) - used for programmatic interaction with a manager. This document covers connecting to the API and its command set.
- [SmartMesh IP Mote CLI Guide](#) - used for human interaction with a mote (e.g. during development of a sensor application, or for troubleshooting). This document covers connecting to the CLI and its command set.
- [SmartMesh IP 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 IP 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 IP Application Notes](#) - Cover a wide range of topics specific to SmartMesh IP networks and topics that apply to SmartMesh networks in general.

Documents Useful When Starting a New Design

- The Datasheet for the [LTC5800-IPM SoC](#), or one of the [modules](#) based on it.
- The Datasheet for the [LTC5800-IPR SoC](#), or one of the [embedded managers](#) based on it.

- A [Hardware Integration Guide](#) for the mote/manager SoC or [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 load 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 Configuration Guide](#).

Other Useful Documents

- A glossary of wireless networking terms used in SmartMesh documentation can be found in the [SmartMesh IP 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	03/18/2013	Initial release
2	09/18/2013	Added LTP5901/2 to OAP pins
3	10/22/2013	New SMSDK apps added; Other minor corrections
4	04/04/2014	Document HRListener example application;
5	10/28/2014	Clarified analog input units in OAP; Other minor changes

2 Introduction

This document covers the [installation](#) and use of various tools available to interact with a SmartMesh IP 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 and Manager over a USB-to-serial link. Next, the user can interact with the CLI of the Mote and Manager via a [serial terminal client](#). A [Serial Multiplexer \(Mux\)](#) allows for multiple concurrent connections to the Manager API - several tools make use of this API:

- The [SmartMesh SDK](#), which is a Python-based set of tools used to demonstrate various aspects of the Mote and Manger APIs. The SDK can connect directly to the API of the mote, and can connect to the Manager API either via the Serial Mux or directly.
- The [Stargazer GUI](#), which allows you to visualize and interact with the network.
- The [Low-power Border Router](#) (LBR) which allows you to send to and receive data from an IPv6 host on your LAN or the internet.

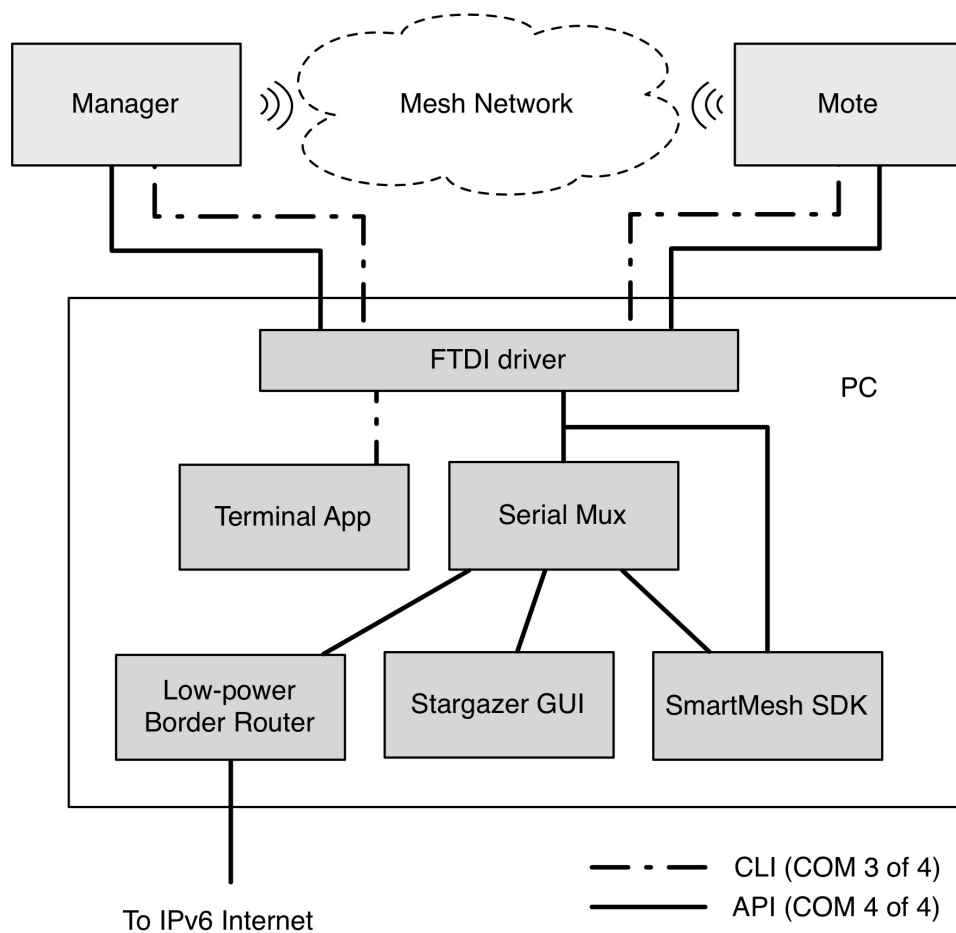



Figure 1 - software tools for interacting with a SmartMesh IP network.

3 Installation

3.1 Before You Begin

3.1.1 What You Need

The following hardware elements are required for manager and mote interaction in the [Interacting with the Manager](#) and [Interacting with a Mote](#) tutorials:

Quantity	Description
1	Computer running Microsoft Windows (XP, 7), with two available USB ports
2	DC9006 Eterna Interface Board
2	<p>DC9003 Eterna Eval/Dev Board :</p> <ul style="list-style-type: none"> • A board with a yellow sticker has been programmed as a SmartMesh IP Manager (DC9003A-A) • A board with a white sticker has been programmed as a SmartMesh IP Mote (DC9003A-B) <div style="border: 1px solid #ccc; background-color: #fff9c4; padding: 5px; margin-top: 10px;"> <p> The combination of a DC9006 Eterna Interface Board and a DC9003A-A manager is a DC9001.</p> </div>
2	USB micro cables to connect your computer to the DC9006

The following software elements are required for manager and mote interaction in the [Interacting with the Manager](#) and [Interacting with a Mote](#) tutorials:

Name	Description
FTDI Drivers	USB-to-Serial drivers for communication with DC9006 boards
Serial Terminal Client	Software used to interact with the devices Command Line Interface (CLI)
Serial Mux	Software used to connect multiple clients to the manager (e.g. APIExplorer and Stargazer)
SmartMesh SDK	Software development kit used to interact with the devices' Application Programming Interface (API)

The following hardware elements are required to run the [A First Network](#) tutorial:

Quantity	Description
1	<p>SmartMesh IP Starter Kit (DC9000)</p> <p>Includes 5 DC9003A-B (SmartMesh IP Motes)</p> <p>- a maximum of either 32 or 100 motes may be used, depending upon whether the manager has external RAM</p>


The following software elements are required to run the [A First Network](#) tutorial:

Name	Description
Stargazer	GUI application for visualizing a network and interacting with motes

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.

 NOTE: The SmartMesh IP Manager looks identical to a Mote out of the box. The Yellow sticker ([DC9003A-A](#)) identifies the Manager. The interface boards for both the Manager and Mote are identical ([DC9006](#)). [DC9001](#) refers to the combination of a Manager and the interface board ([DC9003A-A](#) and [DC9001](#))

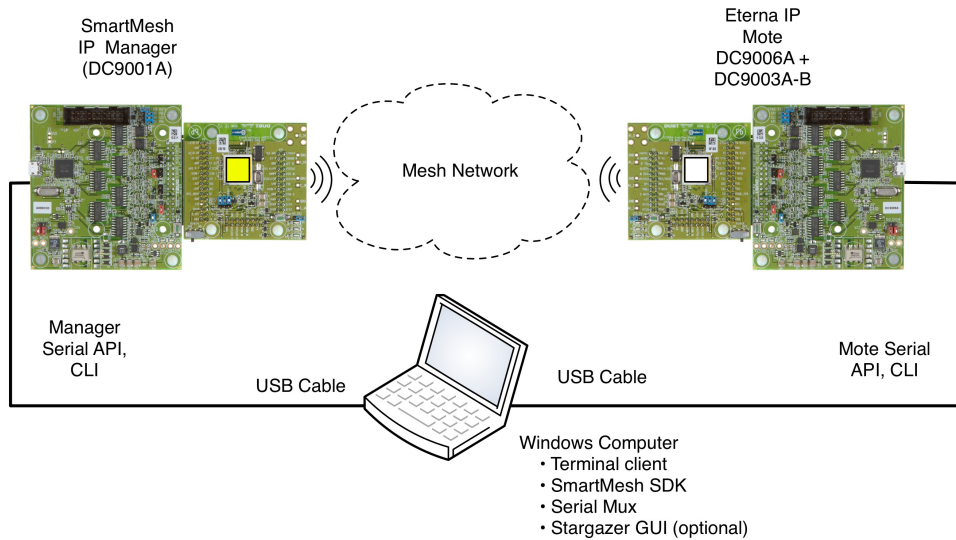


Figure 1 - Eval kit components

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 (and custom Terminal Client if desired)
 - b: Serial Mux
 - c: Python (if not already installed) and SmartMesh SDK

You will then move to the Basic Communications section and:

1. Use APIExplorer to establish a PC to Manager ([DC9001](#)) connection
2. Use APIExplorer to establish a PC to Mote ([DC9003A-B](#)) 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

In order to communicate with the [DC9003A-A](#) Manager and a [DC9003A-B](#) mote, you will need to connect a [DC9006](#) interface board to each using the board-to-board connector, as shown in the figure:

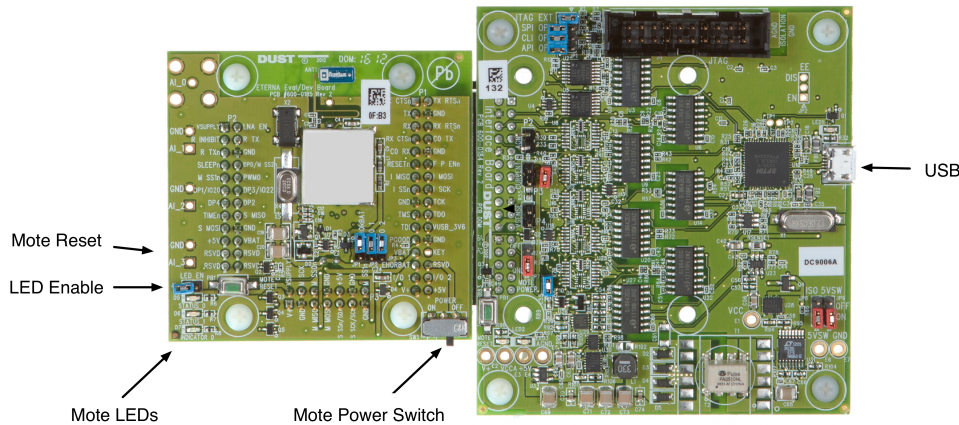


Figure 1 - DC9003 board (left) connected to DC9006 board (right)

- Turn the slide switch marked "Power" on the DC9003 boards to ON.
 - On the manager (yellow sticker), the blue LED on the board should light up. If not check the jumper connection (LED_EN) that enables the LEDs.
 - On the mote (white sticker), one of the yellow LEDs (STATUS_0) should begin blinking to indicate that the mote is searching for network. If not, check the jumper connection (LED_EN) that enables the LEDs.
- 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 DC9006 board and a computer, the Mote and Manager may appear to be operating (as seen by the LEDs) in spite of the power switch being off. The 4 COM ports for each device will appear but you will not be able to communicate with them reliably. Make sure that the power switch on all boards is set to on to ensure proper operation.

DC9003 boards ship with the LED_EN jumper shorted. This is so that you will see the Status LEDs for join behavior. These LEDs draw > 100x the power that the mote does when in a network, so for longer term evaluation or power measurements, the LED_EN jumper should be removed.

3.2.3 Step 2a - Installing FTDI Serial Drivers

Installing Serial Drivers

Driver installation has three steps:

- Download FTDI driver software
- Connect a manager ([DC9001](#)) and run through driver setup
- Connect a mote ([DC9006](#) + [DC9003A-B](#)) 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.

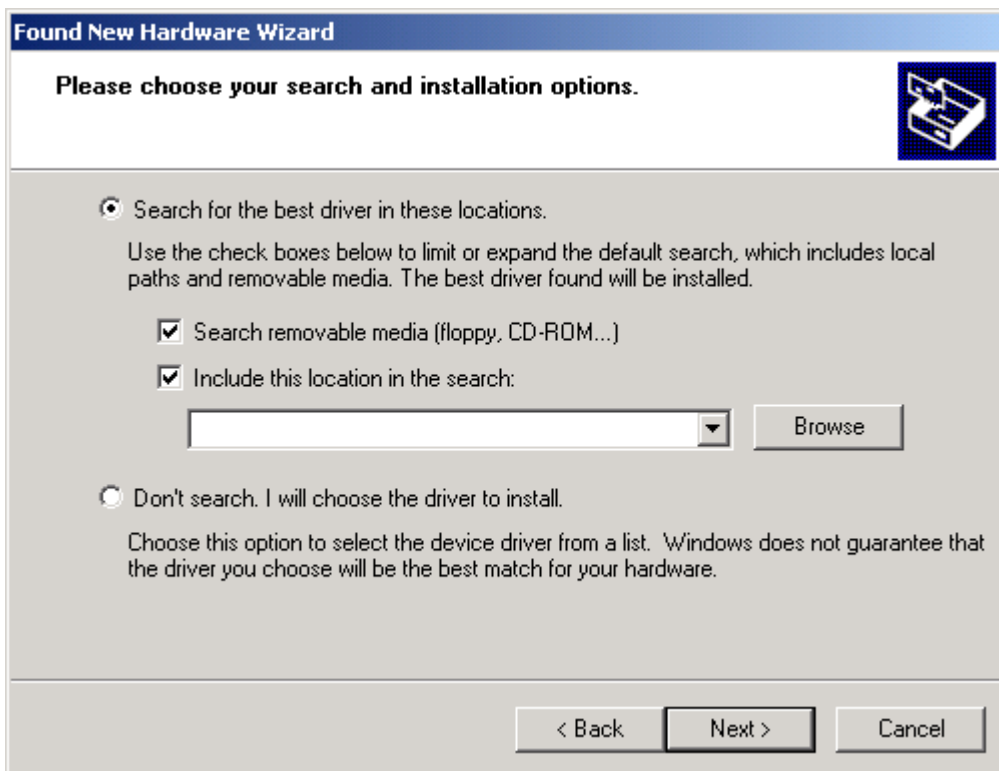
If the Found New Hardware Wizard does not appear, do the following:

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.

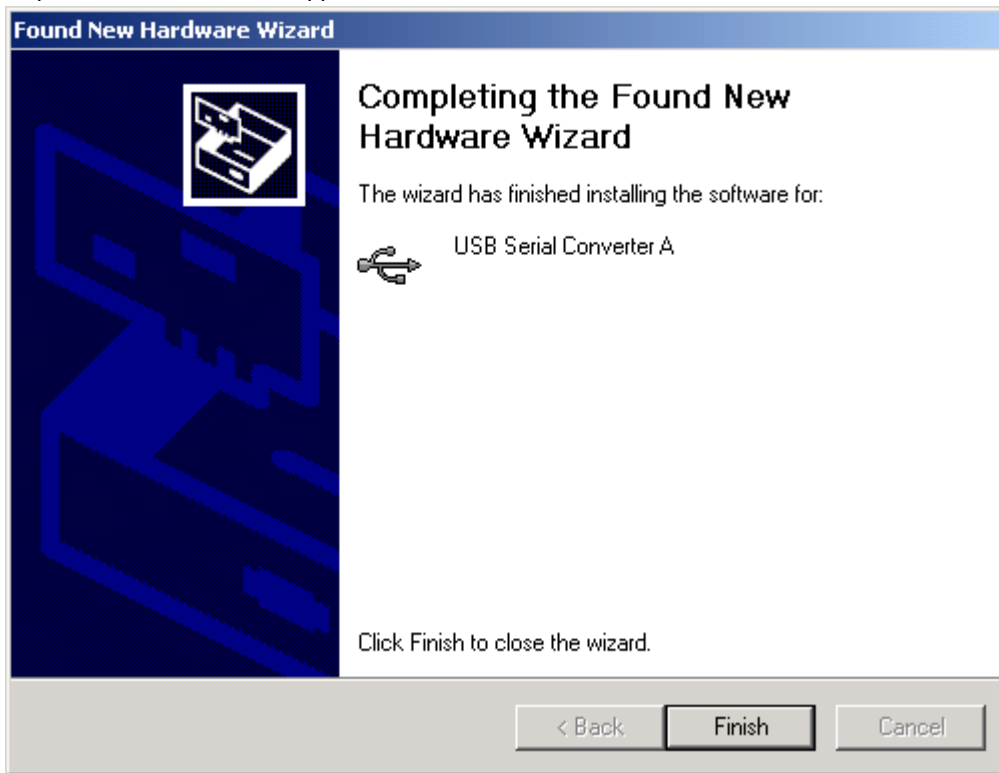
2. In the Wizard, click the option to **Install from a list or specific location** and click **Next**.




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

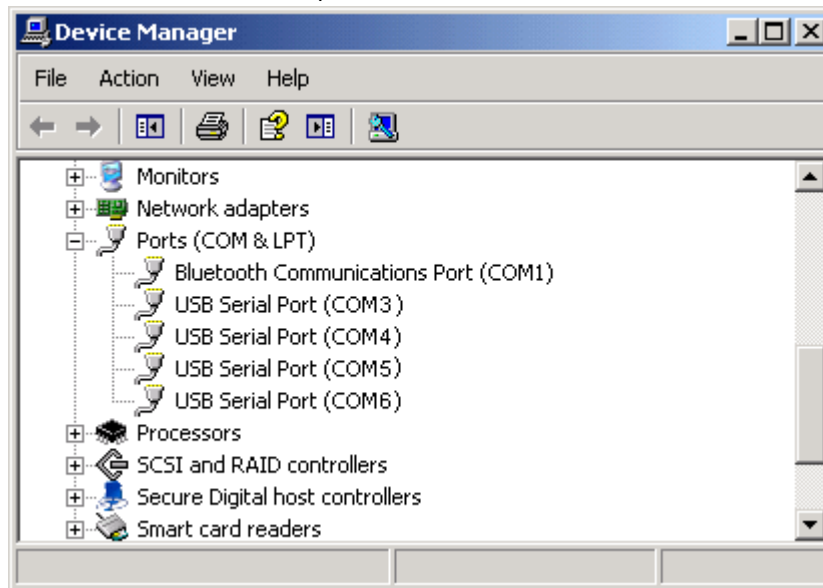


4. After the Wizard installs the software, click **Finish**.
5. When the Found New Hardware Wizard reappears, repeat steps 2 through 4 to continue the installation. Repeat these steps each time the Wizard appears.



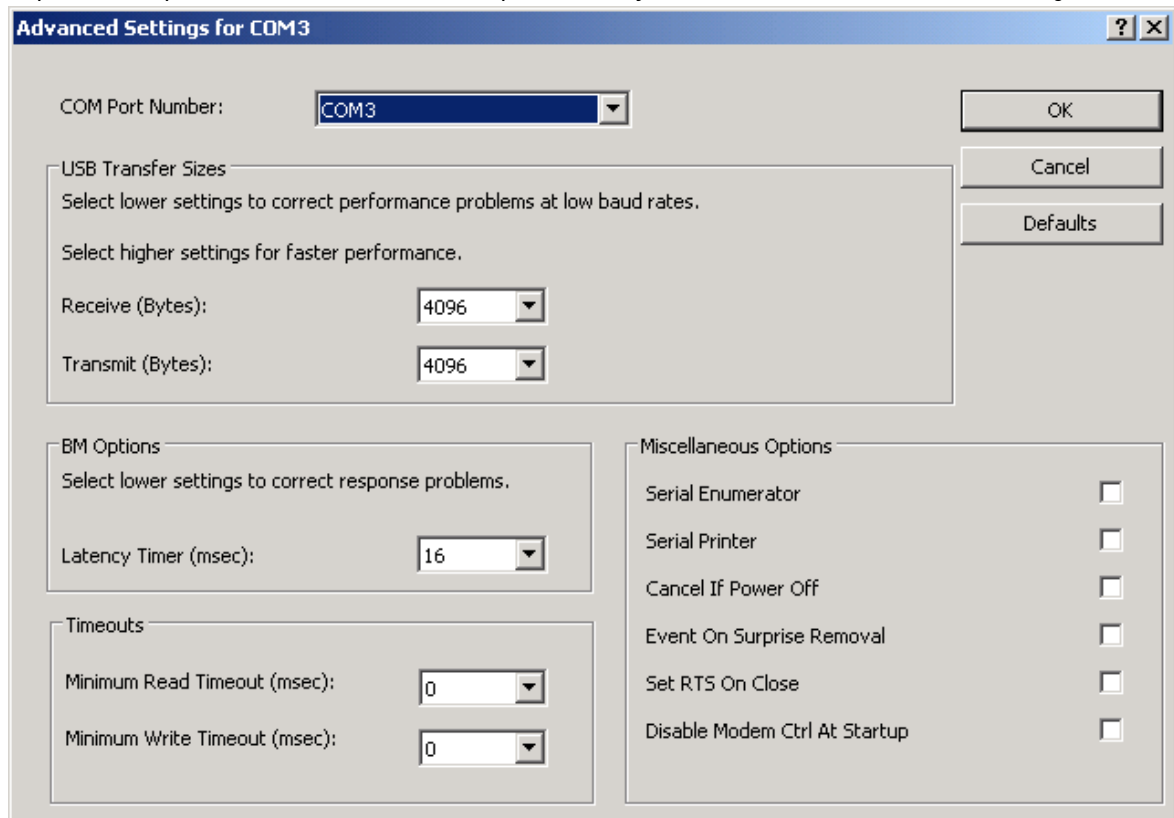
 Because of the way Windows works, you may be prompted to go through the Wizard up to eight times to complete the installation and mapping of the USB port. The manager will install a total of four virtual serial ports, along with the USB devices to control them.


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 Manager and always use this port for plugging in the Manager. Do the same for the mote. Doing this will ensure the COM port assignment will be preserved.

Function of Serial Ports

After installing the [DC9006](#), four serial ports are created on your computer. You can interact with each device over two different serial ports:

- The Command Line Interface (CLI) serial port. Use a [third-party serial terminal](#) software to connect to it.
- The Application Programming Interface (API) serial port. Use [SmartMeshSDK](#)-based applications or [Stargazer](#) to connect to it.

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 IP Manager	third*	CLI	9600	8	N	1
	fourth*	API	115200**	8**	N**	1**
SmartMesh IP 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 devices connected to your computer. We will refer to those numbers throughout the 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 Serial Mux

Overview

Since the Manager has a single serial port dedicated to the its API, normally only a single client would be able to connect at a time. The Serial API Multiplexer (Serial Mux) is a windows service that allows for multiple concurrent connections to the manager's API port, e.g. the [Stargazer GUI](#) and [APIExplorer](#) (found in the SmartMesh SDK).

Installation

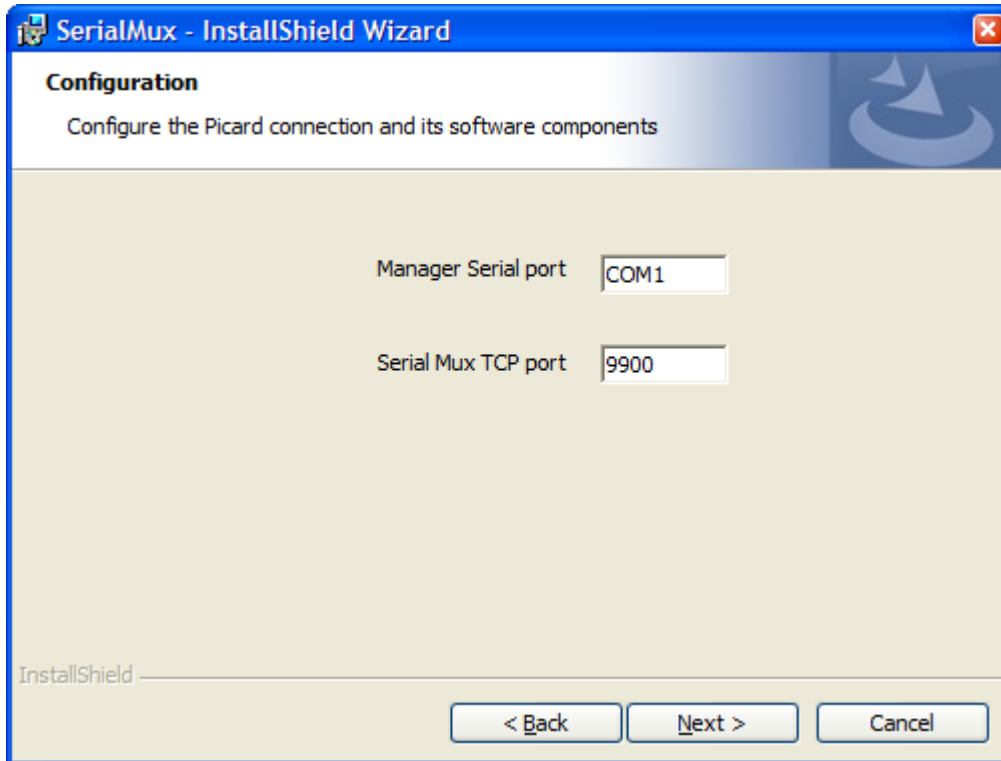
The Serial Mux software is distributed in a zip archive file named `SerialMuxInstaller` with the Serial Mux version appended. Unless you know that you need a particular version for compatibility, choose the latest version.


Install Serial Mux on the computer you connected to the manager. The Windows Installer automatically installs the runtime components required by the Serial Mux if they are not already present. The Installer also includes a click-through prompt with terms and conditions of use. Accepting these terms is required for the installation to proceed.

To install the Serial Mux (on Windows):

1. Unzip the installer zip archive.
2. Launch the Serial Mux Installer (`setup.exe`).
3. Follow the Installation Wizard to install the Serial Mux. A shortcut will be added to your desktop. The Serial Mux requires the VC++ 2010 runtime library. If the system does not have this library installed, the installer will request permission to install it. You must allow the installer to install the runtime library for the Serial Mux to function properly.

- When the Configuration screen appears, enter the serial port (COM port on Windows) that was assigned to the manager Serial API when you connected the manager.



 The API serial port will be the 4th one in the list of 4 devices installed when the FTDI driver was installed. For example, if serial ports COM3, COM4, COM5 and COM6 were installed, the API port would be the fourth one of these, COM6.

Reconfiguring the COM port

If you:

- Forget to note the correct COM port
- Move the manager from one USB port to another
- Launch Stargazer before installing the Serial Mux

you can reconfigure the serial port used by the Serial Mux with the Serial Mux Configurator (currently included in the [SmartMesh SDK](#)).

For more information see the [Serial API Multiplexer \(Serial Mux\)](#).

3.2.5 Step 2c - 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.

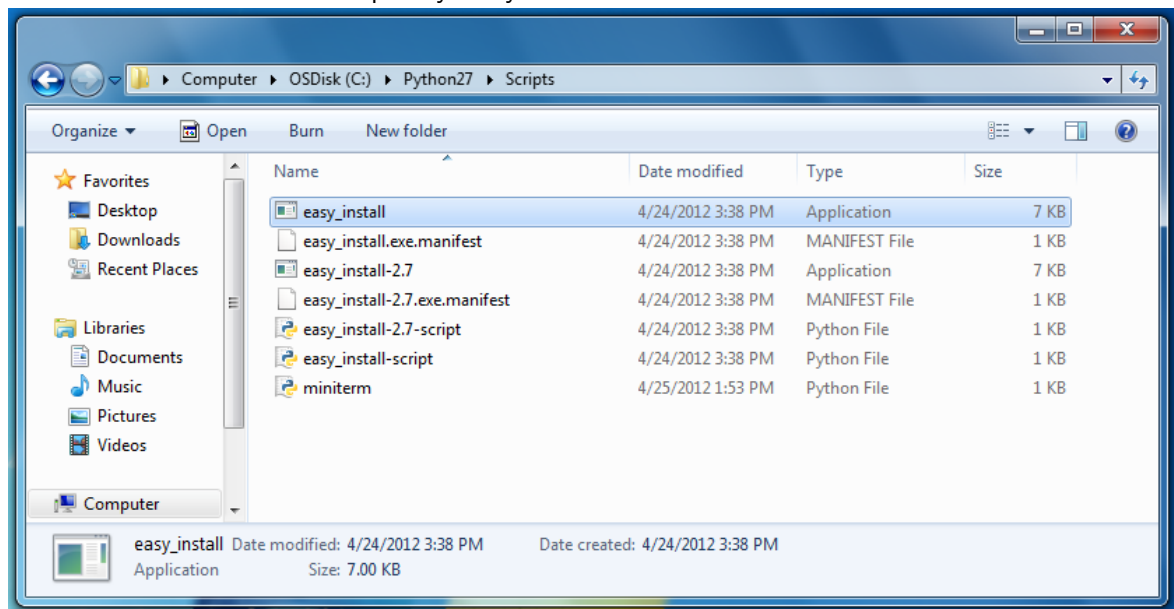
Install Python 2.7

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.

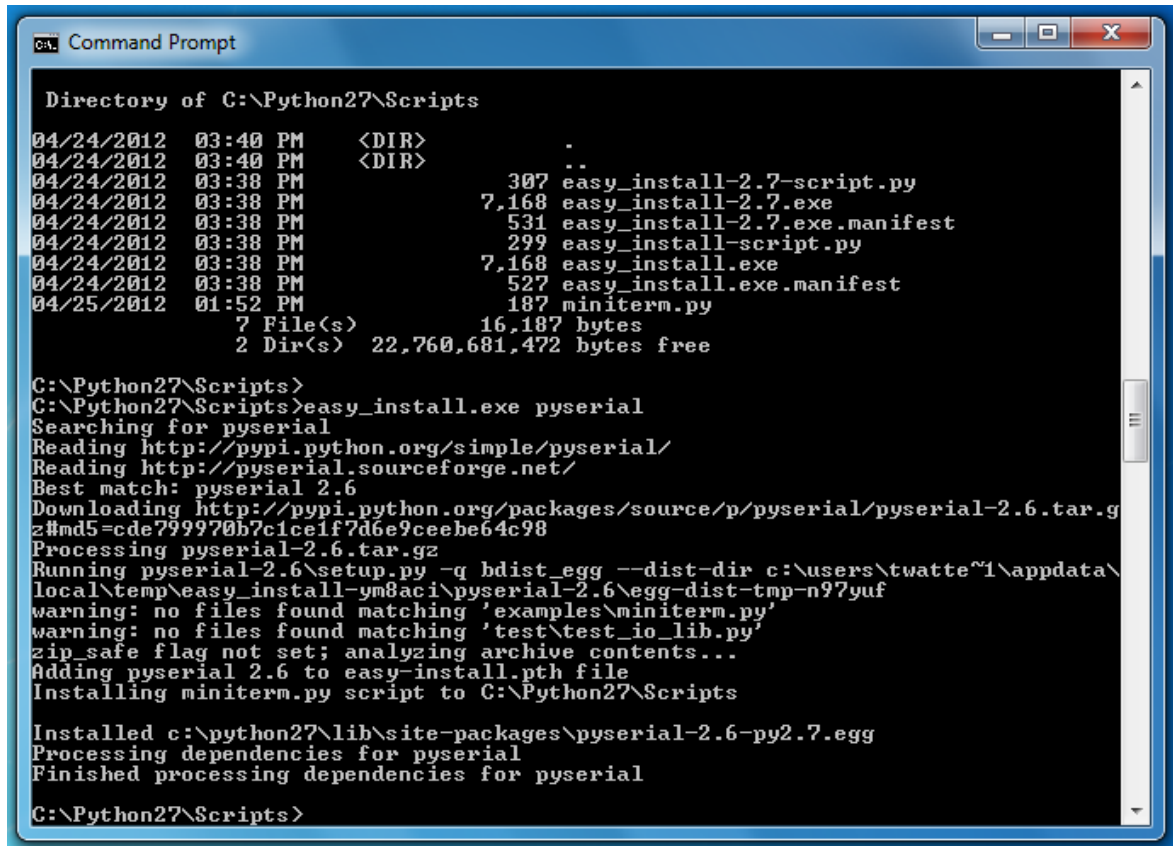
Install PySerial

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`



```

C:\> Command Prompt

Directory of C:\Python27\Scripts
04/24/2012  03:40 PM    <DIR>          .
04/24/2012  03:40 PM    <DIR>          ..
04/24/2012  03:38 PM                307 easy_install-2.7-script.py
04/24/2012  03:38 PM            7,168 easy_install-2.7.exe
04/24/2012  03:38 PM            531 easy_install-2.7.exe.manifest
04/24/2012  03:38 PM            299 easy_install-script.py
04/24/2012  03:38 PM            7,168 easy_install.exe
04/24/2012  03:38 PM            527 easy_install.exe.manifest
04/25/2012  01:52 PM            187 miniterm.py
              7 File(s)          16,187 bytes
              2 Dir(s)        22,760,681,472 bytes free

C:\Python27\Scripts>
C:\Python27\Scripts>easy_install.exe pyserial
Searching for pyserial
Reading http://pypi.python.org/simple/pyserial/
Reading http://pyserial.sourceforge.net/
Best match: pyserial 2.6
Downloading http://pypi.python.org/packages/source/p/pyserial/pyserial-2.6.tar.gz#md5=cde799970b7c1cef7d6e9ceebe64c98
Processing pyserial-2.6.tar.gz
Running pyserial-2.6\setup.py -q bdist_egg --dist-dir c:\users\twatte~1\appdata\local\temp\easy_install-ym8aci\pyserial-2.6\egg-dist-tmp-n97yuf
warning: no files found matching 'examples\miniterm.py'
warning: no files found matching 'test\test_io_lib.py'
zip_safe flag not set; analyzing archive contents...
Adding pyserial 2.6 to easy-install.pth file
Installing miniterm.py script to C:\Python27\Scripts

Installed c:\python27\lib\site-packages\pyserial-2.6-py2.7.egg
Processing dependencies for pyserial
Finished processing dependencies for pyserial

C:\Python27\Scripts>
  
```

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`.

3.2.6 Step 2d - Installing Stargazer

Stargazer GUI

The Stargazer GUI is an optional component that allows for visualization and interaction with the network. Make sure that if you are installing stargazer that you have previously installed the Serial drivers (step 2a) and Serial Mux (step 2b).

Requirements

Stargazer requires a computer running Windows XP or Windows 7 with a SmartMesh IP Manager connected and the [Serial Mux](#) installed. For detailed installation instructions for these components, refer to the [Setup section](#) of the [SmartMesh IP Tools Guide](#).

Download Stargazer

Find the download in the Dust Networks area of the [Linear Design Tools](#) site. The Stargazer download link is in the "Software Utilities" section. This will download the latest version.

- Unzip the file to extract the installer (setup.exe) and other installation files.
- Run the installer. You may be required to download a .NET framework if it is not already installed.
- A shortcut may be created on your desktop to launch Stargazer

Detailed instructions on using Stargazer can be found in the [Stargazer GUI](#) section of this guide.

Installing Stargazer

The Stargazer distribution is available as a `StargazerInstaller` zip file.

Install Stargazer on the computer that has the Serial Mux installed and is connected to the manager. The Installer automatically installs the Microsoft .NET Framework 4.0 if it is not already installed. The Installer also includes a click through prompt with terms and conditions of use. Accepting these terms is required for the installation to proceed.

1. Launch the Stargazer Installer (`setup.exe`).
2. Follow the Installation Wizard to install the Stargazer application. A shortcut for the Stargazer application will be added to your desktop. Stargazer requires the .NET 4.0 Client runtime. If the system does not have this component installed, the installer will request permission to install it. You must allow the installer to install the .NET runtime for Stargazer to function properly.




Files will be installed in different locations depending on which version of Windows are used.

- For Windows XP, the executables are installed in `C:\Program Files\Dust Networks\Stargazer`, and the configuration files are installed in `C:\Documents and Settings\All Users\Application Data\Dust Networks\Stargazer\Default`.
- For Windows 7, the executables are installed in `C:\Program Files\Dust Networks\Stargazer` (or `C:\Program Files (x86)\Dust Networks\Stargazer` on 64-bit systems), and the configuration files are installed in `C:\ProgramData\Dust Networks\Stargazer\Default`.

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.

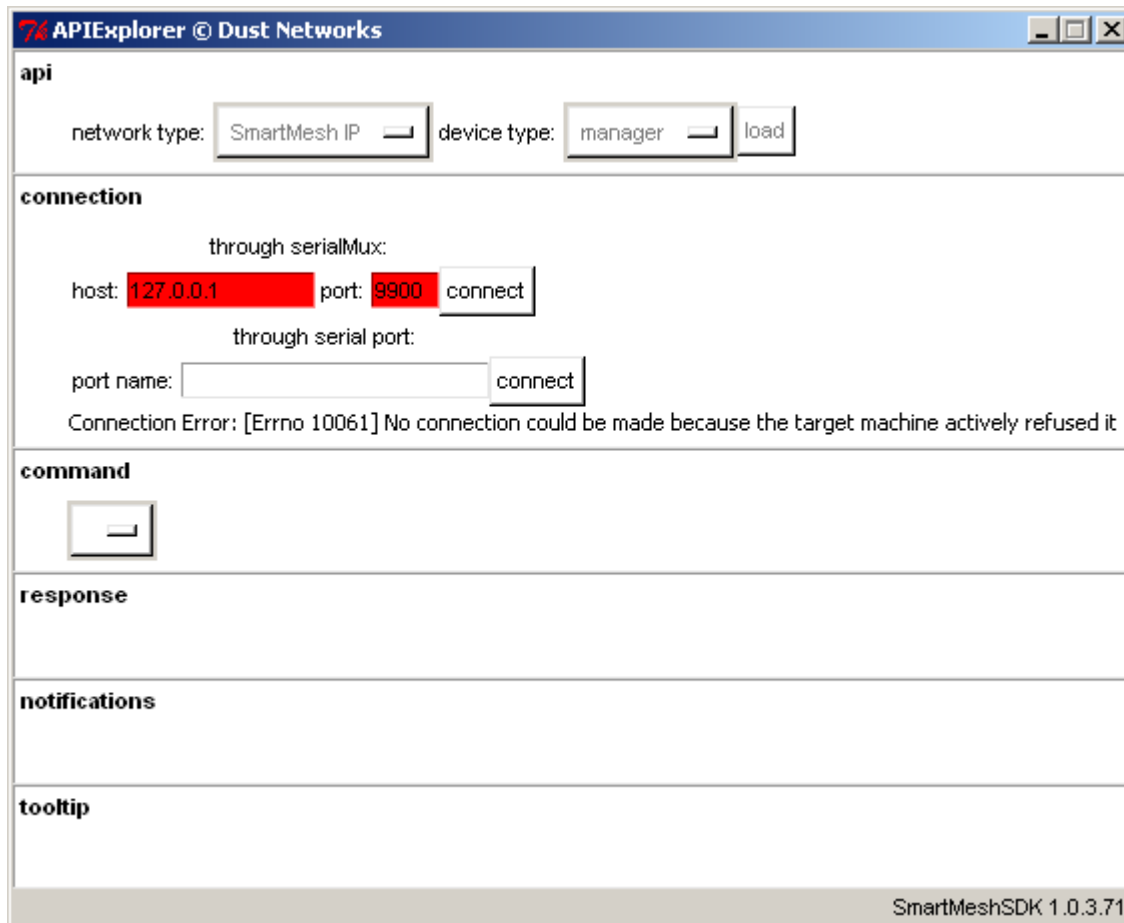
```
$ screen /dev/tty.usbserial-01234567C 9600

> help
...
```

3.3.3 Trouble Connecting to Manager

Symptom - Connection refused error

When attempting to connect to the manager through the Serial Mux, a connection refused error is displayed, as shown in the figure for APIExplorer:



Resolution

The connection refused error can have a number of causes:

- Cause #1 - The Serial Mux is not running
 - Verify that it is running. See [Serial Mux Configuration](#) in the [SmartMesh IP Tools Guide](#) for an explanation of how to manage services.
- Cause #2 - The Serial Mux is running, but is listening to the wrong TCP port (e.g. in the figure above, if the Serial Mux was listening on port 9901)
 - Use the MuxConfig tool in the [SmartMesh SDK](#) to determine the correct TCP port.

- Cause #3 - The Serial Mux is running and is on the correct port, but the manager is not connected
 - Go to the Windows Device manager and verify that the COM port you expect the manager on is listed, and that the MuxConfig tool shows that the Serial Mux is looking for the right COM port. Verify that the manager is powered on and a blue LED (INDICATOR_0) is lit.

Symptom - Connection Error

When attempting to connect to the manager through a serial, a Connection Error is shown.

Resolution


The Connection Error can have several causes:

- Cause #1 - The manager is not powered on
 - Verify that the manager is powered on and a blue LED (INDICATOR_0) is lit.
- Cause #2 - The serial port provided does not exist
 - If the port previously existed
 - Go to the Windows Device manager and unplug/replug the manager. Note which bank of COM ports shows up and select the correct one in the tool you are using (e.g. APIExplorer).
 - If no ports show up when you plug in the manager, verify that the DC9006A board shows 3 LEDs lit, and that you have installed the [FTDI VCP](#) drivers

3.3.4 Not Getting Notifications

With the SmartMesh IP Manager, notifications are suppressed by default unless they are subscribed to. The *subscribe* API is documented in the [SmartMesh IP Manager API Guide](#) and can be tested using APIExplorer in the [SmartMesh SDK](#). To subscribe to all currently defined notifications, a bitmap of 0x76 (118) is used.

3.3.5 Changing Network ID

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

The network identifier (or netid) is the 16-bit identifier of your network. It is set to 1229 by default. Follow the steps below to change it to your own netid on each device you have :

- Connect the device to your computer over USB
- Open the CLI port of that device using serial terminal, using the setting above
- Switch on your device
- If you are connected to a SmartMesh IP Manager, issue the following commands (here we'll set the netid to 100):

```
> login user
> set config netid 100
> minfo
ipmote ver 1.0.3 #12
state:      Oper
mac:        00:17:0d:00:00:38:03:89
moteid:     1
netid:      100
blSwVer:    9
UTC time:   1025665212:339500
reset st:   100
> reset system
System reset by CLI
Reset status: 100, 0
548 : **** AP connected. Network started
```

- if you are connected to a SmartMesh IP Mote, type in the following commands:

```
> mset netid 100
> mget netid
netid = 100
> reset

SmartMesh IP mote, ver 1.1.0.37
```

- Repeat for all your devices.

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

3.3.6 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 \geq 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
```

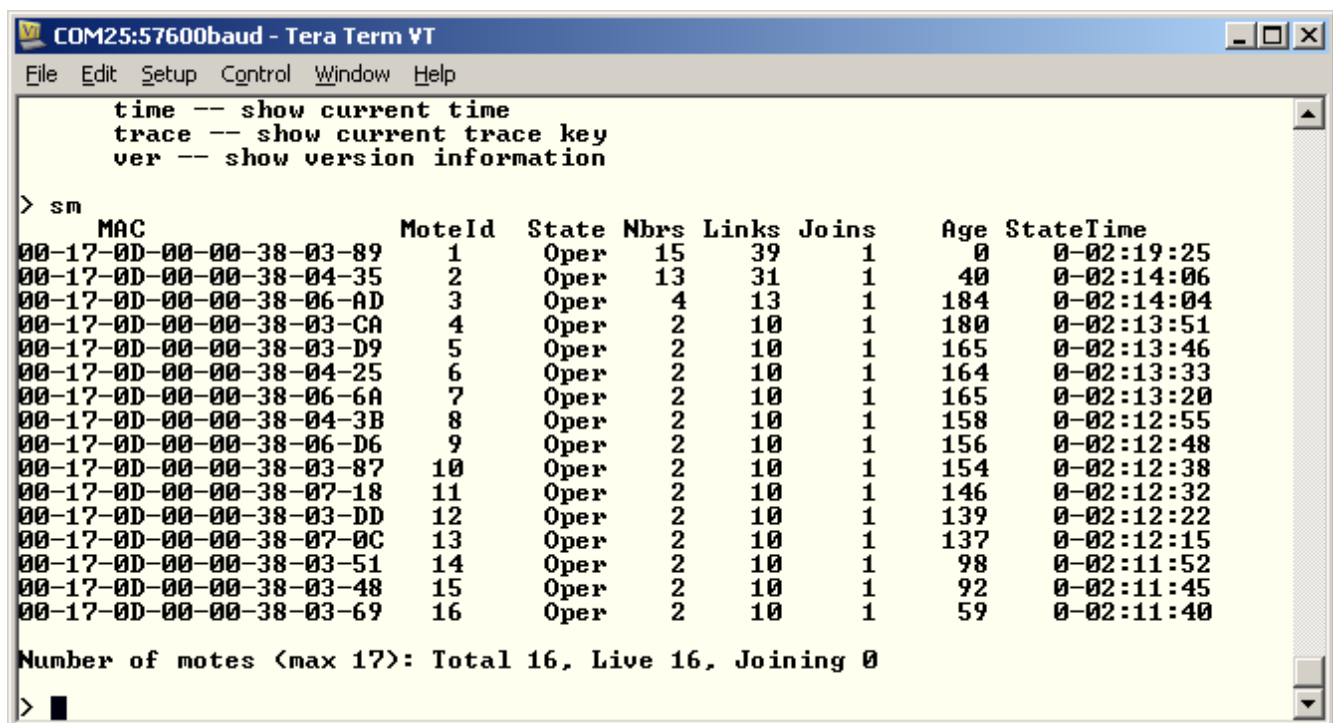
 You must reset the mote for the `set mode` command to take effect. Once set, the mode persists through reset.

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://tssh2.sourceforge.jp/index.html.en>



```

COM25:57600baud - Tera Term VT
File Edit Setup Control Window Help
time -- show current time
trace -- show current trace key
ver -- show version information
> sm
MAC MoteId State Nbrs Links Joins Age StateTime
00-17-0D-00-00-38-03-89 1 Oper 15 39 1 0 0-02:19:25
00-17-0D-00-00-38-04-35 2 Oper 13 31 1 40 0-02:14:06
00-17-0D-00-00-38-06-AD 3 Oper 4 13 1 184 0-02:14:04
00-17-0D-00-00-38-03-CA 4 Oper 2 10 1 180 0-02:13:51
00-17-0D-00-00-38-03-D9 5 Oper 2 10 1 165 0-02:13:46
00-17-0D-00-00-38-04-25 6 Oper 2 10 1 164 0-02:13:33
00-17-0D-00-00-38-06-6A 7 Oper 2 10 1 165 0-02:13:20
00-17-0D-00-00-38-04-3B 8 Oper 2 10 1 158 0-02:12:55
00-17-0D-00-00-38-06-D6 9 Oper 2 10 1 156 0-02:12:48
00-17-0D-00-00-38-03-87 10 Oper 2 10 1 154 0-02:12:38
00-17-0D-00-00-38-07-18 11 Oper 2 10 1 146 0-02:12:32
00-17-0D-00-00-38-03-DD 12 Oper 2 10 1 139 0-02:12:22
00-17-0D-00-00-38-07-0C 13 Oper 2 10 1 137 0-02:12:15
00-17-0D-00-00-38-03-51 14 Oper 2 10 1 98 0-02:11:52
00-17-0D-00-00-38-03-48 15 Oper 2 10 1 92 0-02:11:45
00-17-0D-00-00-38-03-69 16 Oper 2 10 1 59 0-02:11:40
Number of motes <max 17>: Total 16, Live 16, Joining 0
> █
  
```

4.2 PuTTY

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

```

COM25 - PuTTY
sim
MAC                MoteId  State  Nbrs  Links  Joins   Age  StateTime
00-17-0D-00-00-38-03-89    1    Oper   15    39     1     0   0-02:20:27
00-17-0D-00-00-38-04-35    2    Oper   13    31     1    102  0-02:15:08
00-17-0D-00-00-38-06-AD    3    Oper    4    13     1   246  0-02:15:06
00-17-0D-00-00-38-03-CA    4    Oper    2    10     1   242  0-02:14:53
00-17-0D-00-00-38-03-D9    5    Oper    2    10     1   227  0-02:14:48
00-17-0D-00-00-38-04-25    6    Oper    2    10     1   226  0-02:14:35
00-17-0D-00-00-38-06-6A    7    Oper    2    10     1   227  0-02:14:22
00-17-0D-00-00-38-04-3B    8    Oper    2    10     1   220  0-02:13:57
00-17-0D-00-00-38-06-D6    9    Oper    2    10     1   218  0-02:13:50
00-17-0D-00-00-38-03-87   10    Oper    2    10     1   216  0-02:13:40
00-17-0D-00-00-38-07-18   11    Oper    2    10     1   208  0-02:13:34
00-17-0D-00-00-38-03-DD   12    Oper    2    10     1   201  0-02:13:24
00-17-0D-00-00-38-07-0C   13    Oper    2    10     1   199  0-02:13:17
00-17-0D-00-00-38-03-51   14    Oper    2    10     1   160  0-02:12:54
00-17-0D-00-00-38-03-48   15    Oper    2    10     1   154  0-02:12:47
00-17-0D-00-00-38-03-69   16    Oper    2    10     1   121  0-02:12:42

Number of motes (max 17): Total 16, Live 16, Joining 0
>

```

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

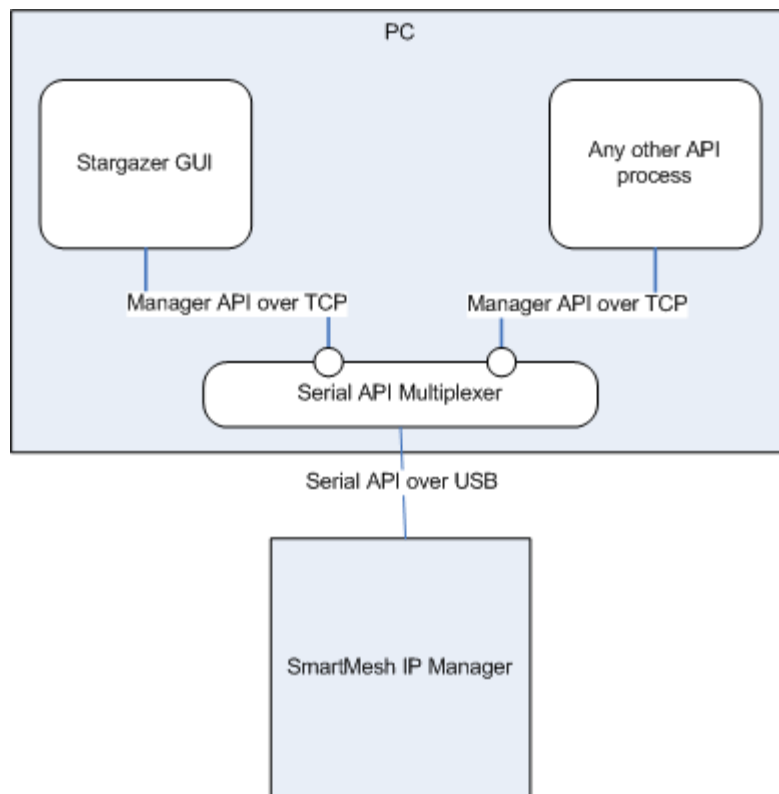
> _

Connected 0:00:06 Auto detect 57600 8-N-1 SCROLL CAPS NUM Capture Print echo

5 Serial API Multiplexer (Serial Mux)

5.1 Overview

This section describes the Serial API Multiplexer (“Serial Mux”) for the SmartMesh IP Manager. The Serial Mux is a simple multiplexer that allows multiple processes to communicate with the Manager’s Serial API over a TCP connection. On Windows, the Serial Mux can be installed as a background service using the Serial Mux Installer. The [Stargazer GUI](#) application and [SmartMesh SDK](#) both communicate with the manager through the Serial Mux.



- Installation is covered in the [Setup](#) section of this guide.
- The [Serial Mux Configuration](#) section contain instructions about manually configuring the Serial Mux.
- The [SmartMesh SDK](#) contains a [MuxConfig](#) tool for editing Serial Mux configurations.
- The [Serial Mux Protocol](#) section describes how a client can communicate with the Manager through the Serial Mux.

5.2 Serial Mux Configuration

The Serial Mux uses a serial port (COM port) to communicate with the Manager and accepts client connections on a specific TCP port. The serial port and TCP port configuration parameters are stored in a configuration file whose location varies by OS. This section describes how to manually edit the configuration parameters. The instructions in this section are for reference only - it's significantly easier to use the [MuxConfig](#) GUI tool to add and edit configurations.

The procedure is to:

1. Edit the Serial Mux configuration file
2. Restart the Serial Mux Windows service

5.2.1 Step 1: Edit the Serial Mux Configuration File

On Windows XP, the configuration file is located at:

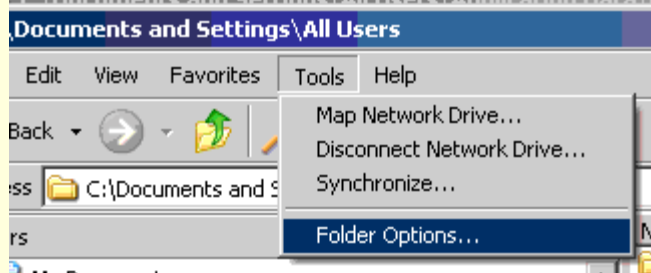
```
C:\Documents and Settings\All Users\Application Data\Dust Networks\SerialMux\Default\serial_mux.cfg
```

On Windows 7, the configuration file is located at:

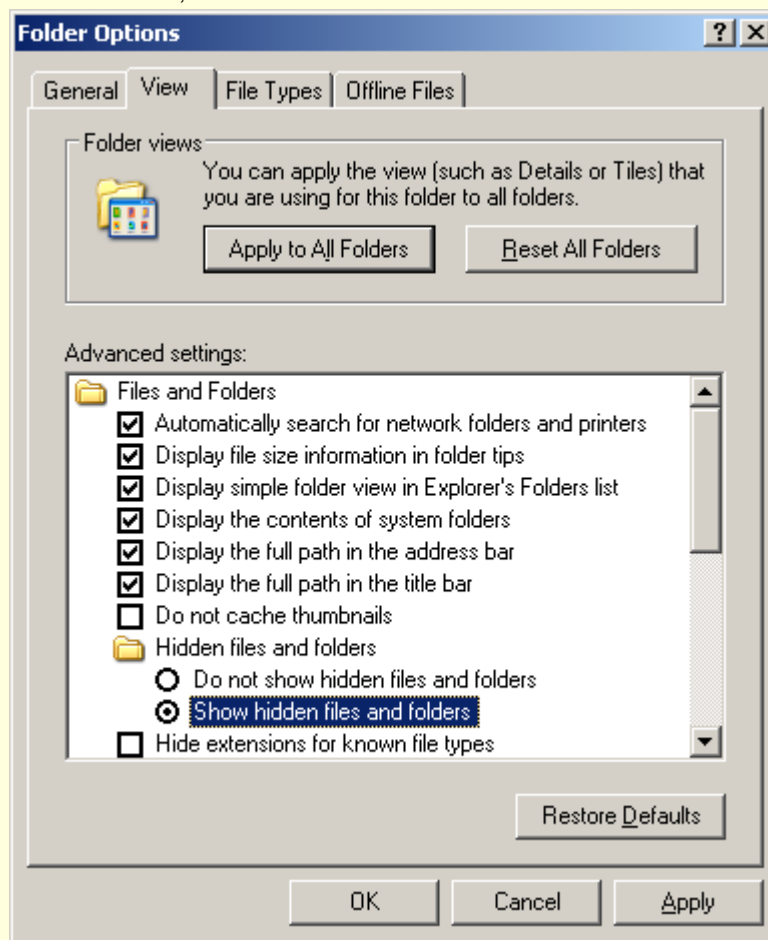
```
C:\ProgramData\Dust Networks\SerialMux\Default\serial_mux.cfg
```

⚠ By default, the *Application Data* folder is hidden by Windows. To see it:

1. in your Windows Explorer window, select **Tools > Folder Options**



2. In the "View" tab, select "Show hidden files and folders"



3. When you are done editing the Serial Mux configuration file, repeat these steps but select the "Do not show hidden files and folders" option.

Open the configuration file with a text editor. It contains the following lines:

```
# Configuration file for Serial Mux
port = COM34
listen = 9900
```

Where:

- *port* is the serial (COM) port the Serial Mux listens to (default value is COM1)
- *listen* is the TCP port the Serial Mux accepts connections on (default value is 9900)

Change the settings to the serial port matching your setup and choose a TCP port that doesn't conflict with other services.

5.2.2 Step 2: Restart the Serial Mux Windows Service

The Serial Mux Windows Service runs in the background at all times. The Serial Mux only reads its configuration file when it starts, so you need to restart the service to have changes to the configuration file take effect.

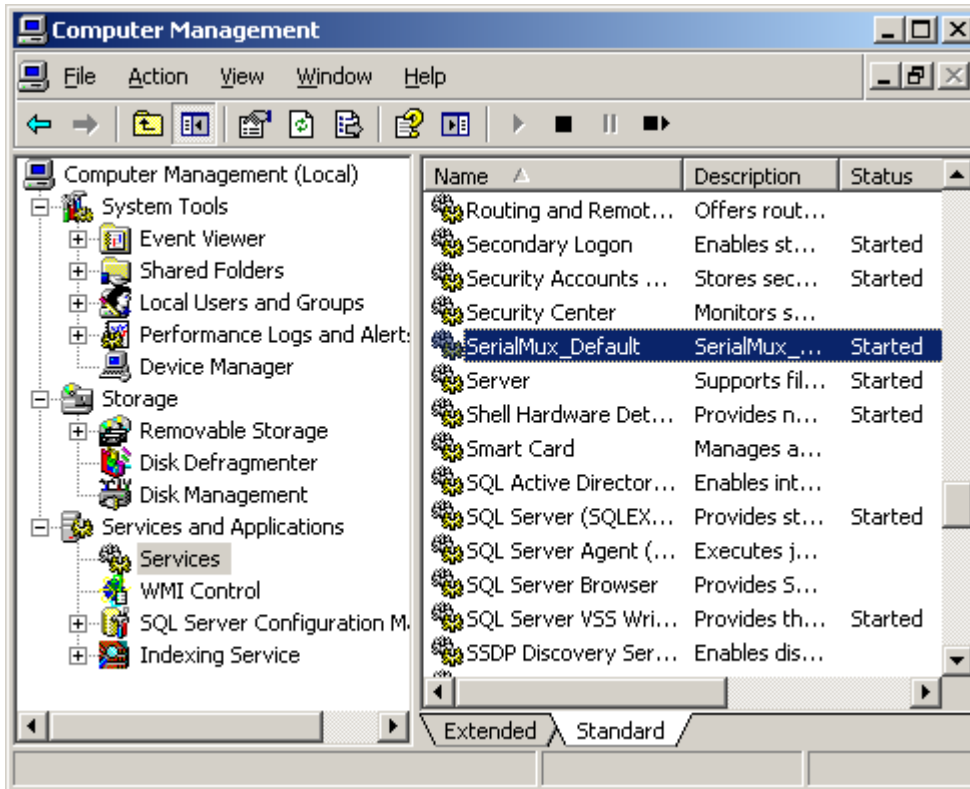
1. In your **Start** Menu, select **Settings > Control Panel**
2. Select **Administrative Tools**



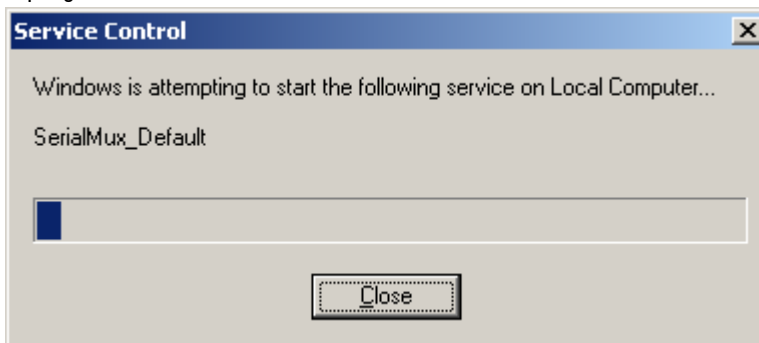
3. Select **Computer Management**



4. Navigate to **Services** and **Applications > Services**. In the list on the right, right-click on the **SerialMux_Default** service and select **restart**



5. A progress bar indicates when the service has been restarted



5.2.3 Advanced Serial Mux Configuration

The Serial Mux configuration parameters are set in a configuration file.

On Windows XP, the configuration file is located at:

```
C:\Documents and Settings\All Users\Application Data\Dust Networks\Serial
Mux\Default\serial_mux.cfg
```

On Windows 7, the configuration file is located at:

```
C:\ProgramData\Application Data\Dust Networks\Serial Mux\Default\serial_mux.cfg
```

The Installer writes a simple default configuration file based on user input:

```
# Serial Mux Configuration
port = COM6
listen = 9900
```

Configuration file parameters

Parameter name	Description
port	Serial port to which the Manager is connected. Configured in the Installer or with the Serial Mux Configurator.
listen	TCP port for client connections. The default port is 9900.
authToken	Authentication token that clients must use in the Hello message to authenticate. The default token is the byte string 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37
accept-anyhost	Allow connections to the Serial Mux from any computer. By default, only connections from the local machine are allowed.
log-file	Specify the name of the log file to write to. The log file is also used as a lock file to detect other Serial Mux processes using the same configuration.
log-level	Specify the detail level of log messages to be printed. <code>trace</code> , <code>info</code> , <code>warning</code> , <code>error</code> are valid values.
log-num-backups	Number of backup log files to keep.
log-max-size	Log file size (in bytes) at which to rotate log files.

5.2.4 Serial Mux with Multiple Managers

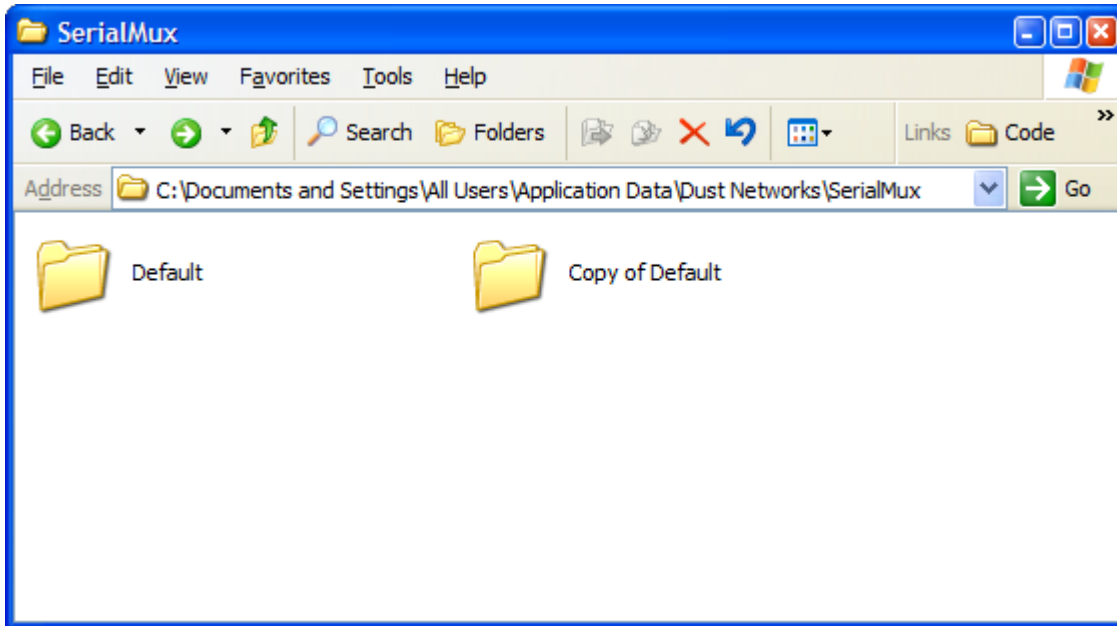
The standard installation of the Serial Mux supports a connection to one Manager. In order to support multiple Managers, multiple copies of the Serial Mux service must be created, one for each Manager.

The instructions in this section are for reference only. It's significantly easier to use the [MuxConfig](#) GUI tool to add and edit configurations.

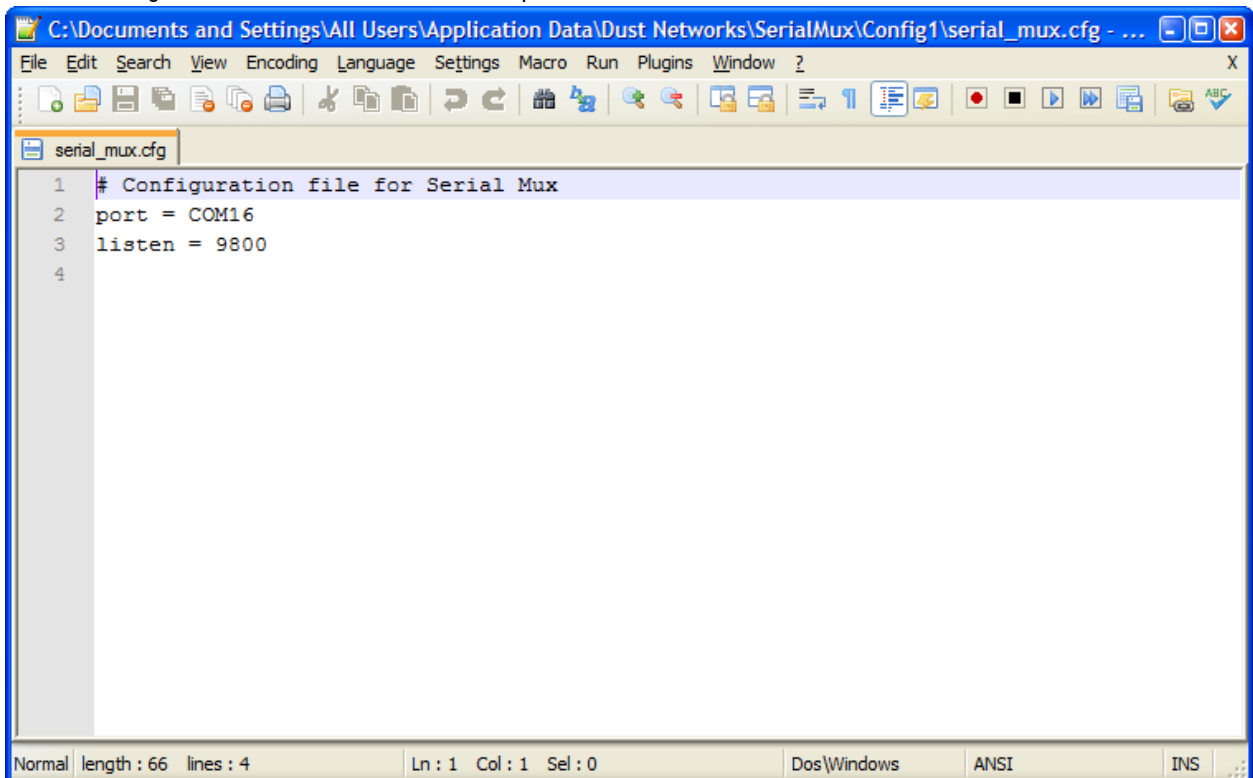
Creating a Serial Mux Configuration

The following steps will allow you to create an additional Serial Mux configuration.

1. Copy the Serial Mux Default configuration directory and rename it - here we renamed Copy of Default to Config1)



2. Update the copied configuration file. The port should match the second manager's API (4th) port, and the listener port must be changed to avoid conflicts. Here we use port 9800.




3. Create a new Serial Mux service using the `sc` command. For consistency, we recommend naming the service to match the name of the configuration directory, e.g. *SerialMux_Config1*.


From a Windows command prompt, enter all the following in a single line. We have broken it up into two lines here to fit it all on the page:

```
> sc create SerialMux_Config1 start= auto binPath= "\"C:\Program Files\Dust  
Networks\SerialMux\serial_mux.exe\""  
--daemon --directory "\"C:\Documents and Settings\All Users\Application Data\Dust  
Networks\SerialMux\Config1\""
```

The `binPath` parameter contains the command line to be run for the service, `<path>/serial_mux.exe --daemon --directory <config path>`. The space after the parameter name and proper quoting is important.

Warnings:

 The Installer assumes that the configuration directory will be Default and the service name will be `SerialMux_Default`. After making these changes, the Installer will not be able to cleanly remove all of the Serial Mux services. If you remove the Serial Mux from your computer, you will need to manually remove the Serial Mux services.

 There is no configurable option to Stargazer to adjust the Serial Mux port, so Stargazer can only communicate with a Serial Mux on the default TCP port.

Testing Your Installation

At this point if you have not rebooted your computer, the second Serial Mux service is not running. To confirm that you've successfully created it:

1. In your **Start** Menu, select **Settings > Control Panel**
2. Select **Administrative Tools**
3. Select **Computer Management**
4. Navigate to **Services and Applications > Services**. In the list on the right, right-click on the **SerialMux_Config1** service and select **start**

5. A progress bar indicates when the service has been started - if you receive an error at this point, verify that the path to the duplicated configuration directory is correct. If it isn't (for example, if you included an extra space) you will need to delete the service and re-create it with the proper parameters.
 1. To delete the service, run the following command from a Windows command prompt:

```
> sc delete SerialMux_Config1
```

Once you have verified that the two services are running, you can use the [API Explorer application](#) included in the SmartMesh SDK to confirm that you can connect to both managers.

1. Launch two instances of the API Explorer.
2. Configure both for Manager API.
3. Configure one to connect to the Serial Mux on port 9900, the other connect to the Serial Mux on port 9800.
4. Connect to each manager.
5. You should be able to issue commands to both managers - *getMoteCfgByID* with an ID of 1 will show you the MAC addresses of your managers.

5.3 Serial Mux Protocol

The Serial API Multiplexer provides an API similar to the SmartMesh IP Manager Serial API with some changes to the messaging protocol and a couple of command extensions. The Serial Mux uses a TCP connection with a client instead of a connection over the Manager serial port. With the exception of the protocol layer changes, a client can communicate with the Serial Mux in much the same way that it would communicate with the Manager.

5.3.1 Basic Operation

The Serial Mux tries to connect to the Manager immediately when it starts. If the Serial Mux can not connect, it periodically retries until a connection is established. After connecting to the Manager, the Serial Mux listens (on a well-known, configurable port) for TCP connections from clients. When a client connects, it is expected to send a Hello message to the Serial Mux. The Hello message contains a secret token that indicates the client is authorized to connect and the client's protocol version. If the secret token or protocol version doesn't match, the Serial Mux sends a HelloResp with an error and drops the connection. Otherwise, the Serial Mux begins reading the client connection for commands to forward to the Manager.

The Manager Serial API can only process one outstanding request at a time, so clients should limit the rate that requests are sent to the Serial Mux. The Serial Mux will only read a single request at a time from a client's TCP connection. While servicing requests from clients one at a time, the Serial Mux reads from the IP Manager Serial API. Responses are routed to the appropriate client. Notifications are copied to all subscribed clients and ack'ed to the Manager Serial API. Clients write requests and read both responses and notifications from the same TCP connection to the Serial Mux. If the Serial API connection with the Manager resets, the Serial Mux will close any open API clients and reconnect to the Manager. It is the responsibility of the client to reconnect with the Serial Mux.

5.3.2 Protocol

Requests and responses have a short header to identify a messages within the TCP stream. The command type precedes the message data. The command types and associated request and response data are the same structures used in the [SmartMesh IP Manager API Guide](#) with the addition of the Hello and Info messages described below.

Requests

Parameter	Type	Description
headerToken	INT8U[4]	4 byte message start sequence
length	INT16U	Length of the remainder of the message
reserved	INT16U	Reserved. Set to 0
commandType	INT8U	Command type as described in Serial Mux Command Types
data	INT8U[]	Request data as described in Serial API

Responses

Parameter	Type	Description
headerToken	INT8U[4]	4 byte message start sequence
length	INT16U	Length of the remainder of the message
reserved	INT16U	Reserved. Set to 0
commandType	INT8U	Command type as described in Serial Mux Command Types
data	INT8U[]	Response data as described in Serial API (includes response code)

The data types, byte ordering and transmission order for the headers and command structures are as described in the [SmartMesh IP Manager API Guide](#).

5.3.3 Connections

The Serial Mux listens for TCP connections from clients on a configurable TCP port.

Handshake

When the client connects to the Serial Mux, the Serial Mux expects to receive a Hello message before any other commands are sent.

Hello Request

Parameter	Type	Description
version	INT8U	Client protocol version
authentication	INT8U[8]	Authentication secret

Hello Response

Parameter	Type	Description
rc	INT8U	Response code
version	INT8U	Manager protocol version

Example connection

To send a Hello to the Serial Mux, the client establishes a TCP connection to the Mux and sends the following byte stream:

headerToken	length	id	commandType	Hello:version	Hello:authentication
0xa7 0x40 0xa0 0xf5	0x0b	0x00 0x00	0x01	0x04	0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37

If the authentication secret matches, the Serial Mux would respond with:

headerToken	length	id	commandType	response code	Hello Response:version
0xa7 0x40 0xa0 0xf5	0x04	0x00 0x00	0x01	0x00	0x04

To send the *getNetworkInfo* command, the client sends:

headerToken	length	id	commandType
0xa7 0x40 0xa0 0xf5	0x03	0x00 0x00	0x40

The Serial Mux responds with:

headerToken	length	id	commandType	response code	getNetworkInfo response
0xa7 0x40 0xa0 0xf5	0x22	0x00 0x00	0x40	0x00	0x00 0x00 0x02 0x1C 0x52 0x00 0x01 0x64 0x48 0x00 0x00 0x08 0xFC 0x00 0xFE 0x80 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x17 0x0D 0x00 0x00 0x30 0x01 0xC7

5.3.4 Info command

The Serial Mux Info command is provided to allow clients to query the Serial Mux version.

Info Request

Parameter	Type	Description
-----------	------	-------------

Info Response

Parameter	Type	Description
protocolVersion	INT8U	Manager Serial API protocol version
majorVersion	INT8U	Serial Mux major version
minorVersion	INT8U	Serial Mux minor version
releaseVersion	INT8U	Serial Mux release version
buildVersion	INT16U	Serial Mux build number

5.3.5 Subscriptions and Notifications

The Serial Mux allows clients to subscribe to notifications from the manager using the same subscribe command provided by the Serial API. In the subscribe request, the client lists the types of notifications that it wishes to receive. Several clients may be subscribed for the same types of notifications. The Serial Mux serializes and sends notifications for each client on the client's TCP connection. The client may see notification messages interleaved with command responses. The Serial Mux acknowledges notifications to the manager and handles the complexity of managing different notifications for different clients.

5.3.6 Disconnection

The Serial Mux maintains a timer while waiting for a command response. If the timer times out, the Serial Mux:

- resets the Manager connection
- closes all connected clients
- tries to reconnect to the Manager by sending a Hello message

If the Serial Mux can not write to a client connection, the connection is closed.

5.3.7 Serial Mux Definitions

This section lists constants and pre-defined values used in the API structures.

Protocol constants

The Header Token is the 4 byte sequence:

```
0xa7 0x40 0xa0 0xf5
```

Command Types

Command Name	Type	Description
Hello	1	Client Hello message
Serial Mux Info	2	Serial Mux Info

All other command types and payloads are defined by the SmartMesh IP Manager Serial API.

Response Codes

Response Code	Value	Description
OK	0	Command executed without error
Invalid Command	1	Invalid command type
Invalid Argument	2	Invalid argument
Invalid Authentication	3	The Hello message contained an invalid authentication token

Unsupported Version	4	The Hello message contained an unsupported version
Command Timeout	5	The Command response from the Manager timed out

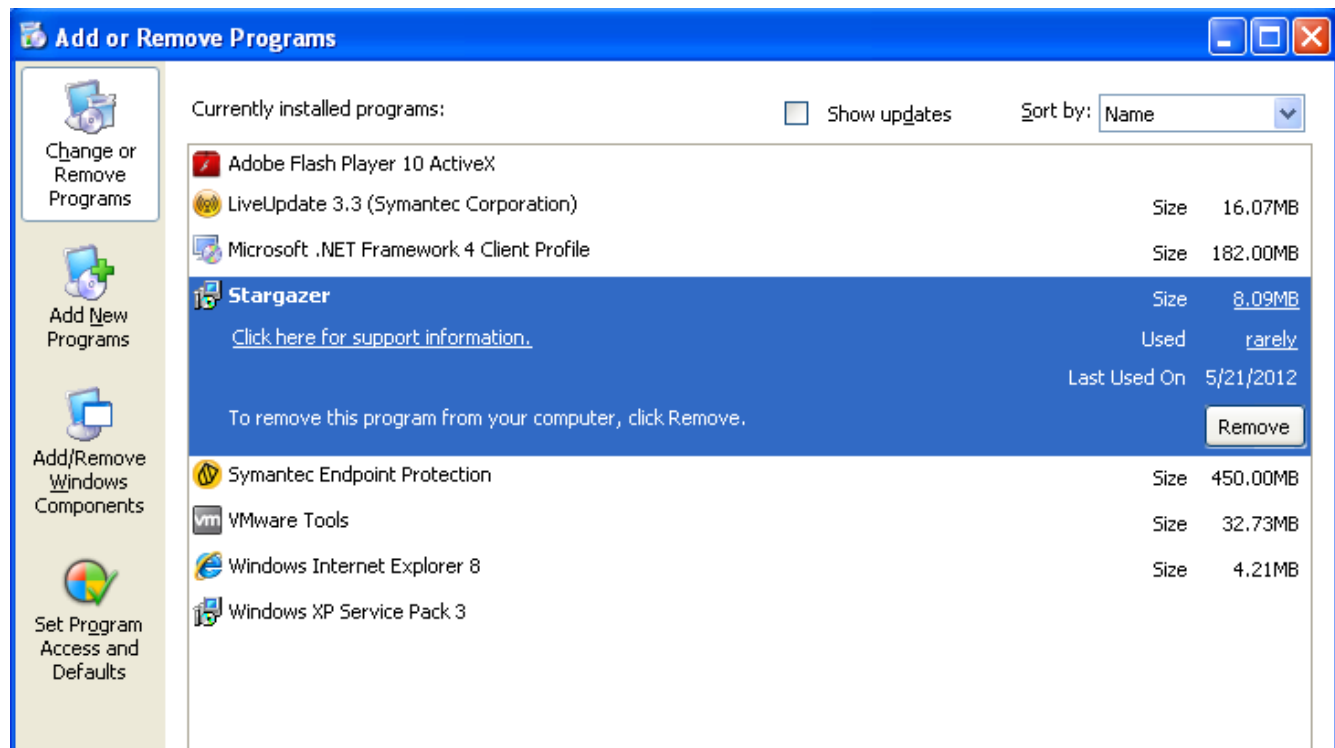
6 Stargazer GUI

6.1 Upgrading Stargazer

The Stargazer Installer and the other tool Installers do not always support upgrades in place. In order to upgrade to a new version, you will need to manually remove the currently installed version. This is especially true for upgrades from Stargazer 1.0.5.50 to 1.0.5.51 where the components of 1.0.5.50 have been separated into separate Installers.

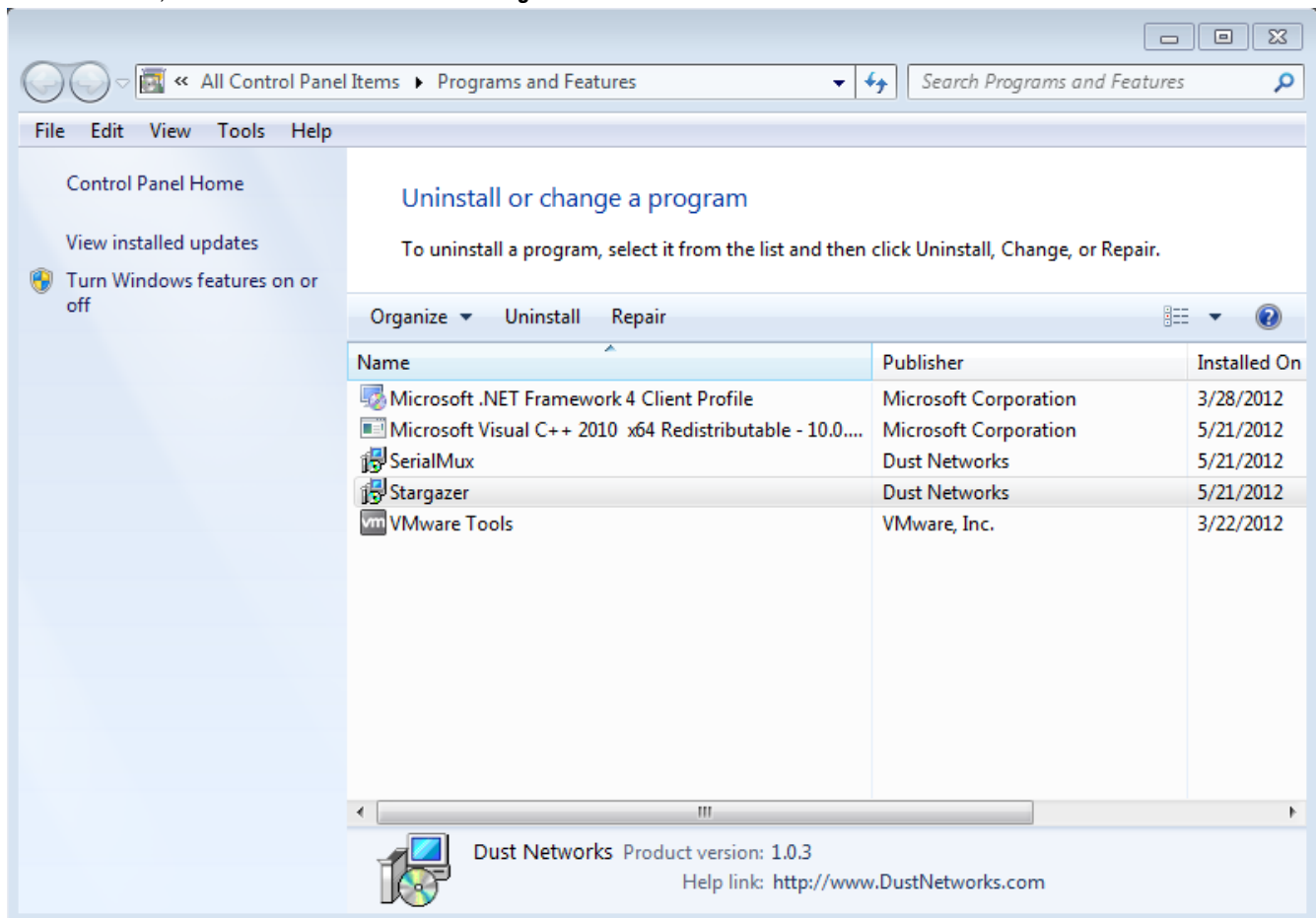
6.1.1 Remove the Existing Application

On Windows XP, from the Control Panel, select **Add / Remove Programs**.



Select **Stargazer** (or the tool that you are upgrading) from the list and click the **Remove** button.

On Windows 7, from the Control Panel select **Programs and Features**:



Select **Stargazer** (or the tool that you are upgrading) from the list, right click and select **Uninstall**. Any dependencies that were installed, such as the .NET Client Framework, do not need to be removed unless the installation instructions indicate otherwise.



Once the existing installation has been removed, you can proceed with the normal installation process through the Installer.

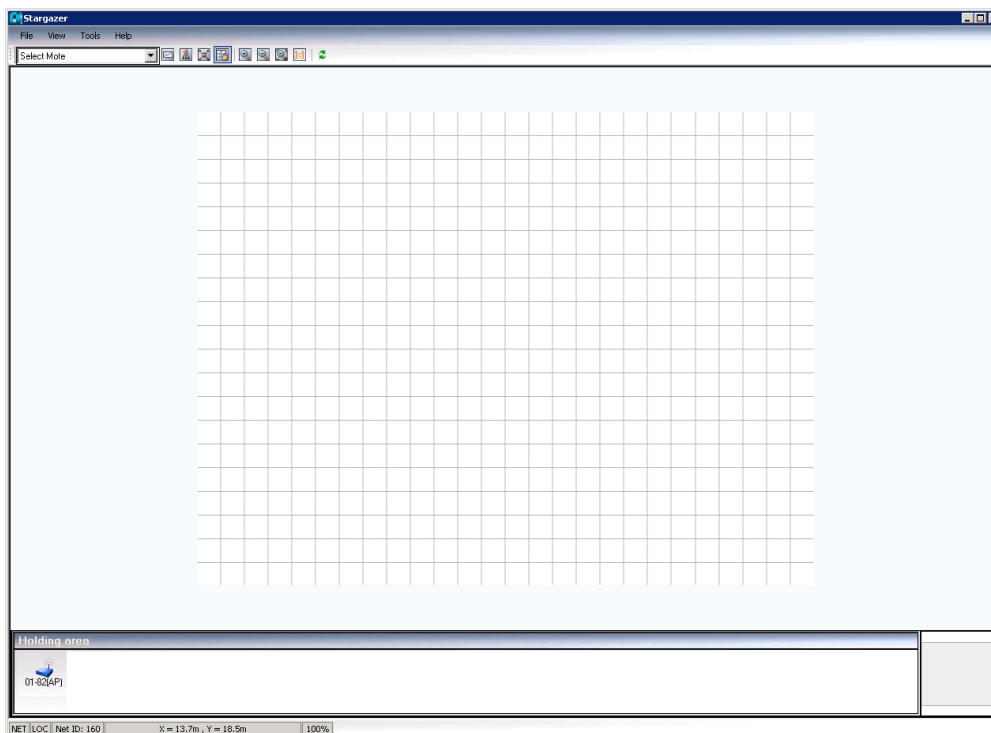
6.2 Using Stargazer

6.2.1 Overview

Stargazer allows you to configure and manage your networks from a Windows-based computer connected to the SmartMesh IP Manager. Stargazer graphically displays the network topology—simply point and click to view current data, display the status of motes and paths, and monitor network statistics and alarms. Stargazer can optionally overlay the network on an imported JPG or PNG image of your site map.

Launching Stargazer


Double-click Stargazer icon  on your desktop. When you start Stargazer, the application immediately connects to the manager and graphically displays the network in a Stargazer window, initially a blank grid. Note that the manager (indicated as “AP”) appears in the **Holding Area** at the bottom of the window. 



Deploying the Motes

The next step is to deploy the evaluation modules (motes). To work with Stargazer, the SmartMesh IP Mote must be in **master** mode. If you previously switched a mote to **slave** mode to use with [APIExplorer](#), [TempMonitor](#), or other SmartMesh SDK example application, return it to **master** mode:

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


 Motes in the starter kit ([DC9000](#)) 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 IP User's Guide](#).

The moment you turn on a mote, it begins searching for nearby motes and starts to join the network. Within minutes, the motes form a reliable multichannel mesh network.

Follow these guidelines when installing the motes:

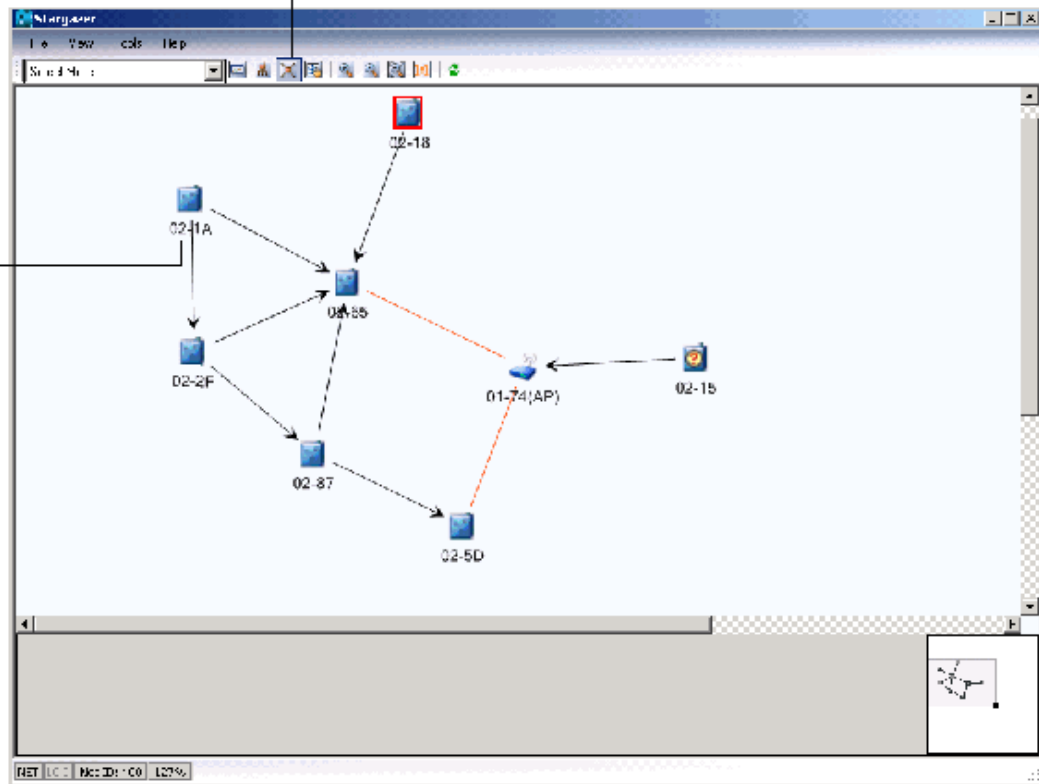
- Install the motes closest to the manager first.
- The optimum distance between motes depends on your RF environment. The recommended distance is within 10 to 30 meters for indoor installations. The recommended distance for outdoor installations is within 50 meters. Please be aware that the module enclosure is not weatherproof.
- Install the motes at least two meters above the ground.
- Make sure that the manager has at least three motes within range.
- Make sure that each mote has at least three motes within range. If the manager is within range, it may count as one of these motes.

To deploy the motes:

1. If not already installed, install a CR2032 battery, positive side up. Slide the **Power** switch to the **On** position to activate the mote.
2. If the LED_EN jumper is installed, the **Status 0** LED will be blinking, indicating that the mote is activated and searching for the network. If the mote fails to power on, try resetting it by pressing the MOTE RESET button. If the LEDs still do not come on, the battery needs to be replaced.
3. Place the mote according to the guidelines described above.
4. Repeat steps 1 through 3 for the remaining motes. As each mote connects to the network, its icon appears in the SmartMesh window.
5. Click the **Radio Space** button  in the window toolbar to display the network topology with the manager in the center. By default, motes are identified by the last four digits of their MAC address, the unique address assigned to a mote at the factory. You can change how motes are identified by changing Preferences (File menu).
6. After all the motes are deployed, go to the next section, Reviewing Network Connectivity.


Radio Space button

Mote identifier



The mote **Status LEDs** provide the following information about network connectivity, and are made available by installing the LED_EN jumper.



Status 0	Status 1	Mote State
Blinking	Off	The mote is searching for the network
On	Off	The mote has found the network and is attempting to join
On	On	The mote has joined the network and is operational

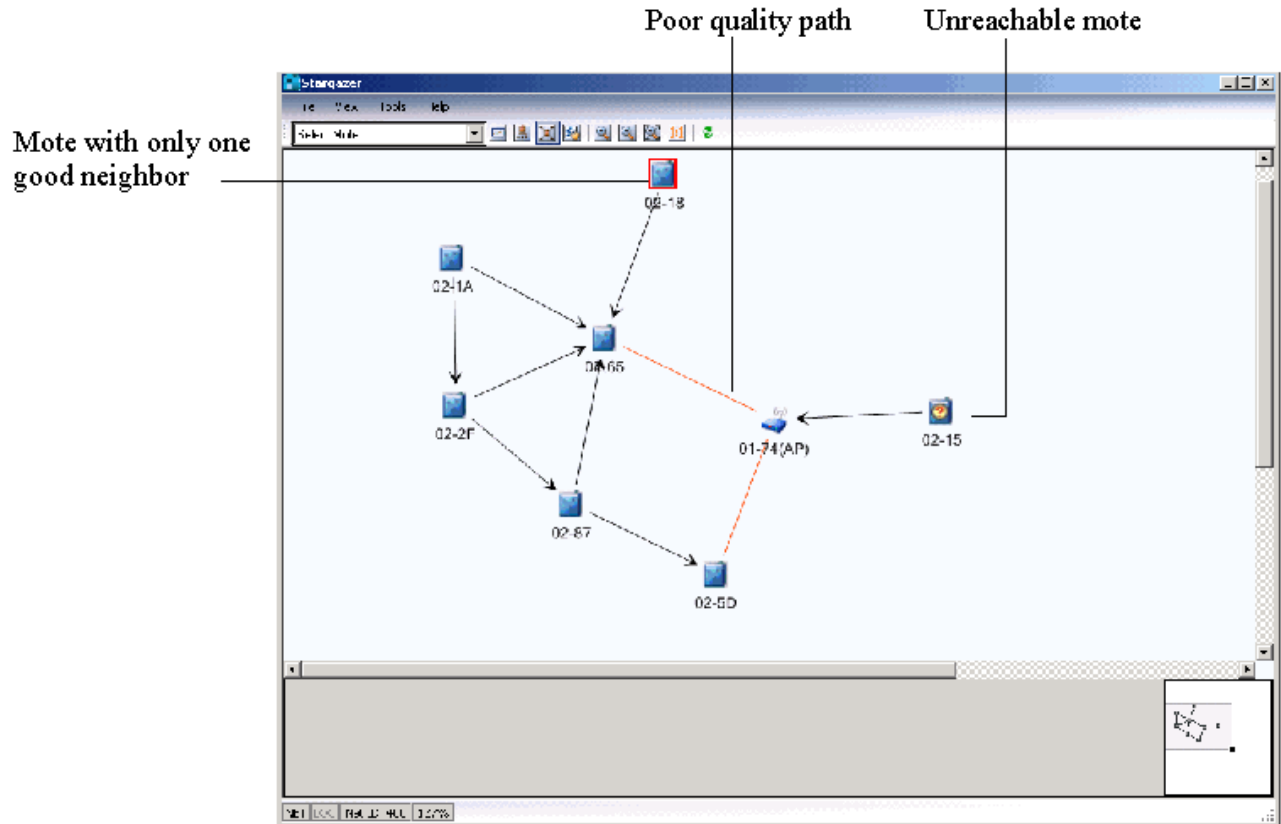
 LEDs are only activated when the mote is configured in Master mode.

Reviewing Network Connectivity

At this point, you have deployed all your motes and they have joined the network. Each mote has at least one parent, and data is flowing to the manager. The SmartMesh IP Manager automatically performs network optimization by making continuous, proactive adjustments in network links to improve overall latency and power consumption while maintaining high reliability. You may notice the network changing over the course of several hours as it evolves towards a more optimal state. The next step is to review network connectivity.

To review network connectivity:


1. Click the *Refresh* button  in the toolbar to manually update the display.
2. Identify connectivity problems:
 1. Look for unreachable motes that have not joined the network. Unreachable motes are marked with a  icon.
 2. Look for more than one mote that does not have two good neighbors - motes with fewer than a specified number of good neighbor (2 by default) are highlighted in red. A good neighbor is one whose path quality is > 50% (default) - note that quality is based on measurements for used paths, and on estimates based on RSSI for unused paths. Each mote needs at least two parents to enable redundant routing and allow the network to self-heal. There should only be one device with a single parent - this is to prevent loops.
 3. Look for paths that have weak radio signal strength (RSSI) - Paths with weak signal strength appear orange. To see the RSSI value of a path hover the cursor over the path or double-click the path to see detailed path information.

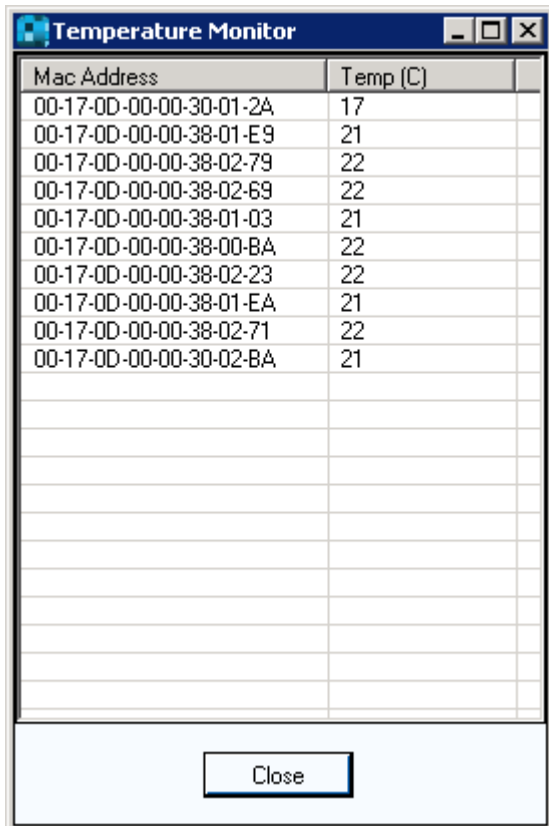


If connectivity problems persist for any mote after the network has been running for an hour, you can manually intervene by trying one of the following:

- Move the mote closer to a neighboring mote (see instructions below).
- Move the mote closer to the manager and reset the mote. To reset a mote, press the MOTE RESET button.
- Move the mote within the line of sight of other motes and reset the mote.
- Move the mote more than two feet above the floor and reset the mote.
- Add one or more motes between the problem mote and the nearest connected mote.
- Move, remove, or minimize the number of objects (especially heavy metal objects) between the mote and its neighbors or the manager. Reset the mote.


Once you see that the network has formed a fully redundant mesh, choose **Temperature Monitor** from the **Tools** menu to display real-time data from the temperature sensor on board each mote. By default each mote is publishing a temperature reading every 30 seconds.

 Publish rates persist through reset, so if you previously set a mote to publish faster or slower than 30 s, it will continue to publish at the new rate each time it joins a network.



Mac Address	Temp (C)
00-17-0D-00-00-30-01-2A	17
00-17-0D-00-00-38-01-E9	21
00-17-0D-00-00-38-02-79	22
00-17-0D-00-00-38-02-69	22
00-17-0D-00-00-38-01-03	21
00-17-0D-00-00-38-00-BA	22
00-17-0D-00-00-38-02-23	22
00-17-0D-00-00-38-01-EA	21
00-17-0D-00-00-38-02-71	22
00-17-0D-00-00-30-02-BA	21

Congratulations! You have successfully set up your SmartMesh IP network and have verified that live data is flowing to the manager.

 The **Temperature Monitor** window will only display information on motes that have joined the network. If a mote has not joined the network since the manager was reset or powered on, it will not show up in **Temperature Monitor**. This is remedied by closing **Temperature Monitor** and re-opening it once all motes have joined.

6.2.2 Managing the Network

Now that your network is up and running, you can use the Stargazer application to view real-time data and evaluate network connectivity and performance. Stargazer provides a graphical view of your network, and gives you complete control over network operations and individual motes.

Main Menus

The Stargazer screen contains the following main menus:

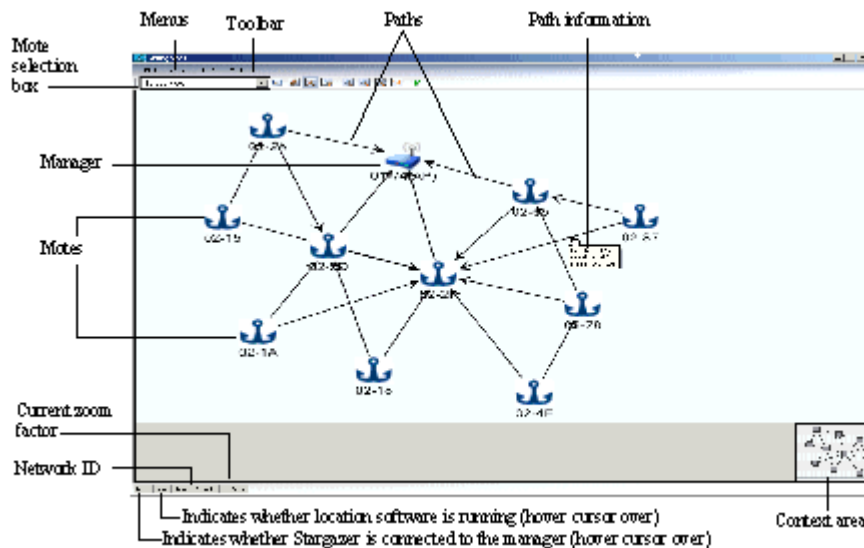
- *File* —Sets user preferences, prints information, and saves a snapshot of mote states and statistics.
- *View* —Displays the network in different layouts, zooms in or out.

- *Tools* —Configures network and mote settings, shows mote and path information, monitors network traffic, configures sensors, and performs other network management operations, such as resetting motes or changing the Network ID.
- *Help* —Displays version information for Stargazer and customer service contact information.

In addition, a popup shortcut menu displays when you right-click a selection, such as a mote or path. For a complete description of menu commands, see the Command Reference section below.

Viewing the Network


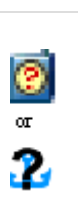


The Stargazer window can display the network topology in three views—a Manual layout that overlays the network on a map or a grid, a Hierarchical “tree-like” layout, and a Radio Space layout that displays the network topology with the manager in the center.



Stargazer Window—Radio Space View

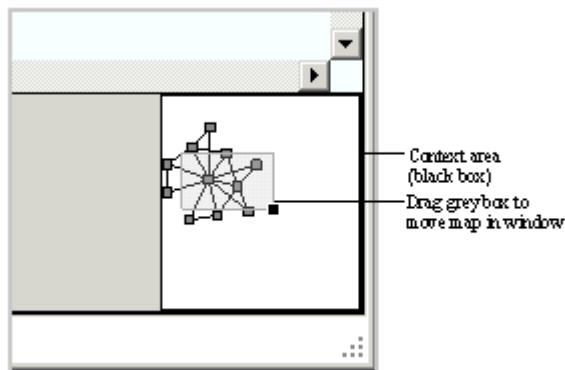
The following icons appear in all topology views:

	<p>SmartMesh IP Manager — The manager controls the network and publishes data received from the wireless mesh network.</p>
<p> or </p>	<p>Operational mote — This is a mote that is connected to the network. When an operational mote is in the Holding Area in Manual view, it appears as in all views. When the mote has been dragged to the main window in Manual view, its icon changes to in all views. The mote identifier is set in Preferences.</p>

	<p>Highlighted mote —Motes are highlighted in red when their connectivity (number of good neighbors) is below the threshold set in Preferences. By default, two neighbors are required for good connectivity.</p>
 <p>or</p> 	<p>Lost mote —This is a mote that is not currently reporting to the network, but was in the network at some time in the past.</p>
	<p>Paths —The arrows indicate the communication paths between motes, and always points toward the mote's parent. Paths are highlighted in red if their quality is lower than the threshold set in Preferences - quality is determined by path stability for used paths, and by RSSI (signal level) for discovered but unused paths.</p>



Use these actions to move the network within the Stargazer window:




- **Click and drag** —Click anywhere in the window and drag to move the network image.
- **Use the context area** —Drag the grey box within the context area (see Context Area) to move the network image. The context area is located in the lower right corner of the window and contains an image of the entire network. The grey box indicates the part of the network that is currently displayed.
- **Use the scroll wheel** —Scroll to enlarge or reduce the size of the network image.
- **Drag an icon** —In Manual view, drag to position a mote or the manager on the map or grid.



Context Area





You can also use the following Toolbar buttons to zoom in or out and refresh the window.

	<p>Zoom In —Enlarges the map size and zooms in on the center. You can also use the mouse scroll wheel to zoom.</p>
	<p>Zoom Out —Reduces the network size. You can also use the mouse scroll wheel to zoom.</p>

	Fit in Window —Sizes the network to fit in the window.
	Actual Size —Enlarges the network to 100%, i.e the icons are drawn at their native resolution.
	Refresh —Updates mote information in the window.

Changing the Network View

Use these Toolbar buttons to switch between the network views:

	Table —Displays a table listing the motes and information about their status. For more information, see Viewing Mote Information later in this chapter.
	Hierarchical —Displays the network topology with manager at the top. This layout makes it easier to see how many hops a mote is from the manager.
	Radio Space —Displays the network topology with the manager in the center.
	Manual —Displays the network topology over a site map or grid. This allows you to place motes where they are located geographically.

Displaying Mote and Path Information

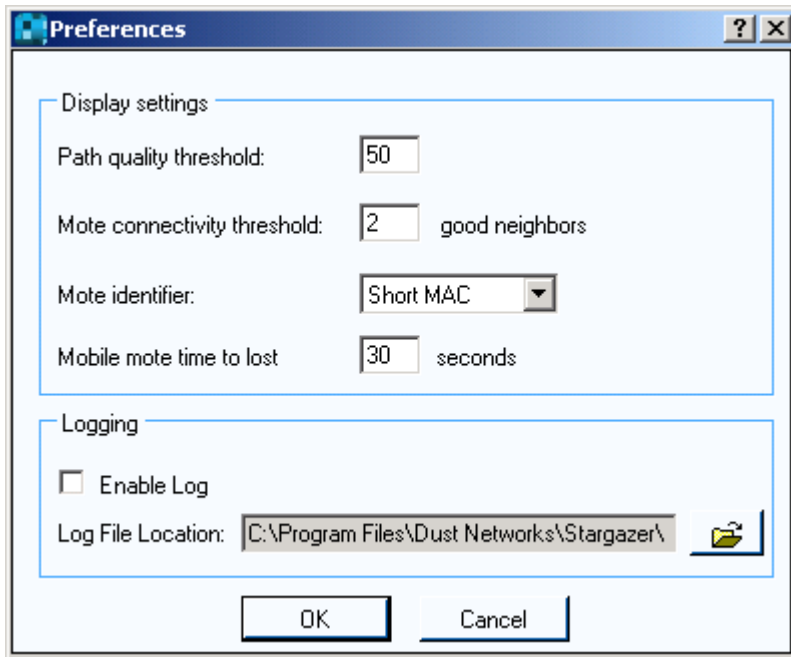
With a few clicks, you can display more detailed network information about a selected mote or path. For example, you can:

- Hover the cursor over a mote to see connectivity information and location coordinates.
- Double-click a mote to view configuration details.
- Hover over a path to see the path quality and RSSI.
- Double-click a path to view additional details.

You can view detailed information about all motes or paths by clicking Motes or Paths on the Tools menu.

Changing Stargazer Preferences

Use the Preferences command on the File menu to change display preferences and turn on logging. You may be requested to turn on logging if you are having problems with your network. The log can be used for troubleshooting purposes.



Preferences Window

You can set the following preferences:

Field	Description
Path quality threshold	Sets the threshold (between 0 and 100%) for highlighting paths in Stargazer. Paths are highlighted in orange if they do not meet the path quality threshold. Paths $\geq 50\%$ are considered to be good
Mote connectivity threshold	Sets the threshold for highlighting motes in Stargazer. Motes are highlighted in red if they do not meet the mote connectivity threshold. Enter the minimum number of <i>good neighbors</i> a mote must have. A mote is considered a good neighbor if its path has good signal strength and path quality.
Mote identifier	Specifies how motes are identified in Stargazer. You can choose from the following options: <ul style="list-style-type: none"> <i>Long MAC</i>—Displays the mote's 8-byte EUI address. The MAC address is the unique address assigned to the mote at the factory. For example: <i>00-17-0D-00-00-38-01-74</i> <i>Short MAC</i>—Displays the last 2 bytes of the mote's MAC address. For example: <i>01-74</i> <i>Mote ID</i>—Displays the mote's network assigned ID. The mote ID is based upon the order in which the mote joined the network. Mote IDs start at 2, and the manager is always numbered 1.
Enable log	Enables a log that can be used to troubleshoot problems with the network.

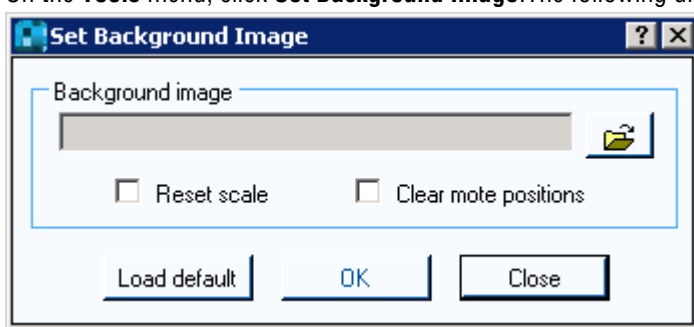
Log file location	Specifies the location of the log.
-------------------	------------------------------------

Importing a Background Image

Stargazer can display an imported site map (PNG or JPG) in the background, allowing you to geographically position the motes on the map. By default, the background displays a grid.


To import a map background:

1. Click the **Manual** button  on the Toolbar to switch to Manual view.
2. On the **Tools** menu, click **Set Background Image**. The following dialog box appears:



3. Click the **Browse** button, navigate to where the site map image is stored on your computer, and click **Open**.
4. Click **OK**. The site map appears in the Manual view in Stargazer.

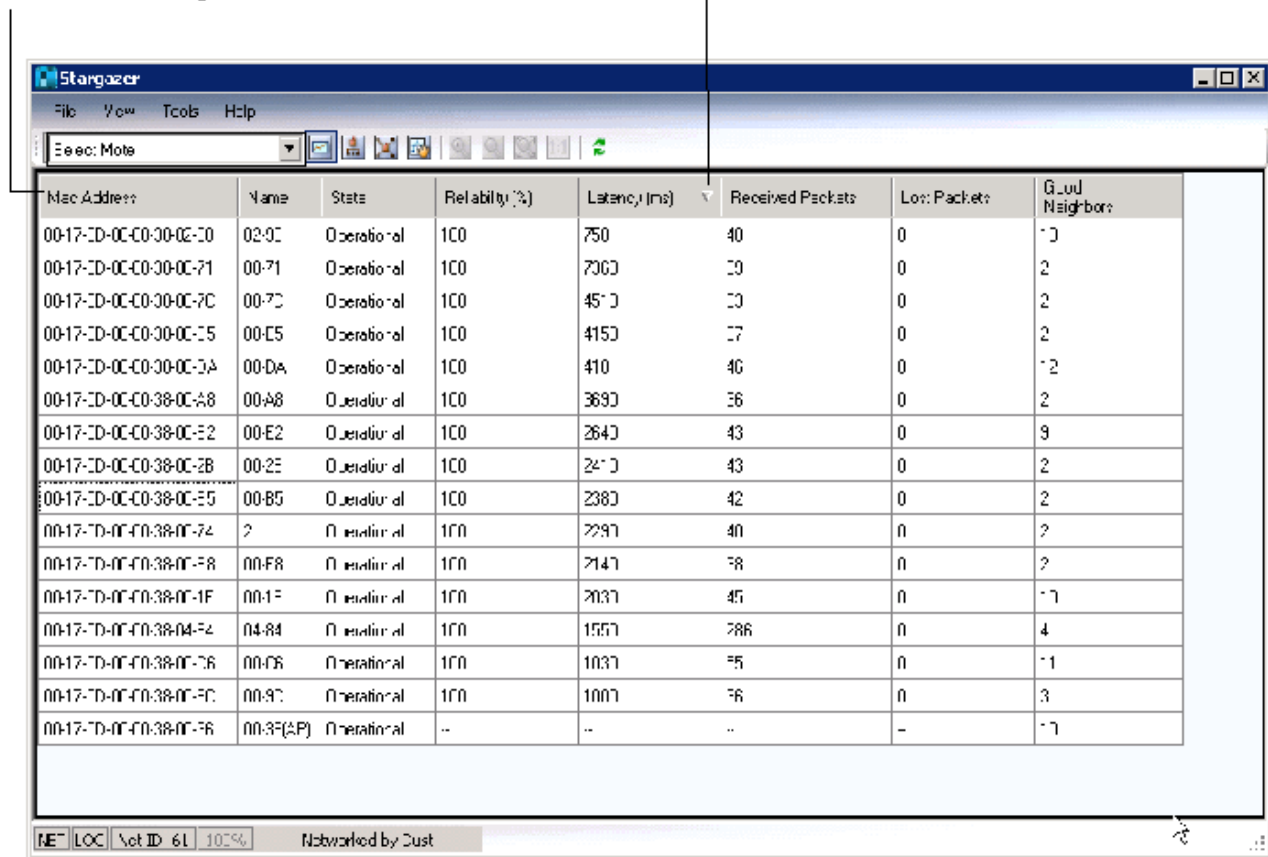
Viewing Mote Information

Click the Table icon  in the Toolbar or click Motes on the Tools menu to view the motes in a list format that can be sorted by column. The mote list shows the current mote status and statistics. Mote statistics include packet statistics gathered by the mote and provided in the health report it sends to the manager every 15 minutes, and data latency statistics, which are calculated by the manager. These statistics show when motes are functioning poorly: these motes will have higher latency than expected or lost packets displayed.

Use the Refresh button to update information in the window. To sort the list by column category, click in the column heading. Single-click for an ascending sort; double-click for a descending sort. The sort icon in the column heading indicates the direction of the sort.

Column heading

Sort icon



Mac Address	Name	State	Reliability (%)	Latency (ms)	Received Packets	Lost Packets	Good Neighbors
00-17-D0-0C-30-02-70	02-0C	Operational	100	750	40	0	1
00-17-D0-0C-30-0C-71	00-71	Operational	100	7000	20	0	2
00-17-D0-0C-30-0C-7C	00-7C	Operational	100	4500	20	0	2
00-17-D0-0C-30-0C-75	00-75	Operational	100	4150	27	0	2
00-17-D0-0C-30-0C-3A	00-DA	Operational	100	410	46	0	2
00-17-D0-0C-38-0C-A8	00-A8	Operational	100	3690	26	0	2
00-17-D0-0C-38-0C-E2	00-E2	Operational	100	2640	43	0	9
00-17-D0-0C-38-0C-2B	00-2B	Operational	100	2400	43	0	2
00-17-D0-0C-38-0C-25	00-85	Operational	100	2380	42	0	2
00-17-D0-0C-38-0C-74	?	Operational	100	2290	40	0	2
00-17-D0-0C-38-0C-F8	00-F8	Operational	100	2140	28	0	2
00-17-D0-0C-38-0C-1F	00-1F	Operational	100	2030	45	0	1
00-17-D0-0C-38-04-F4	04-84	Operational	100	1550	286	0	4
00-17-D0-0C-38-0C-76	00-76	Operational	100	1030	25	0	1
00-17-D0-0C-38-0C-9C	00-9C	Operational	100	1000	26	0	3
00-17-D0-0C-38-0C-56	00-3F(AP)	Operational	--	--	--	--	1

Mote Table View

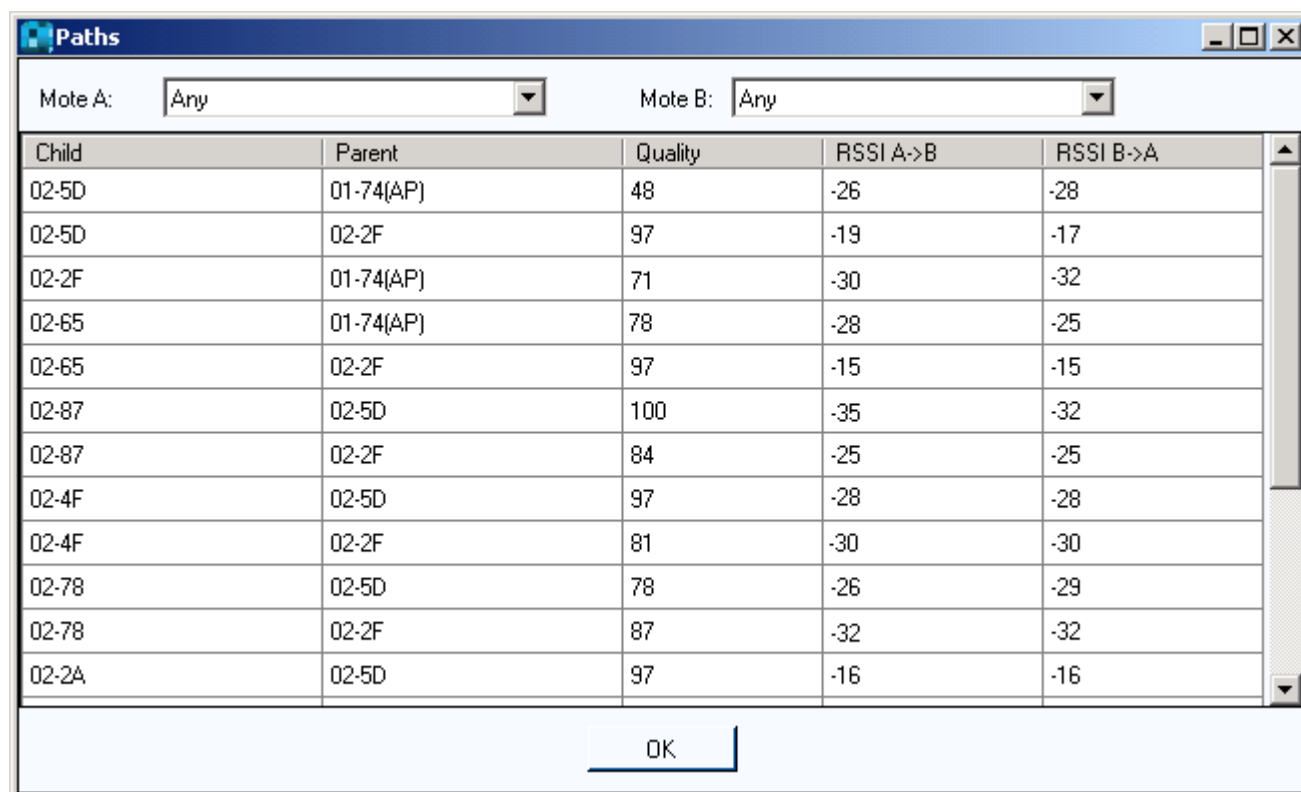
The mote table displays the following information:

Field	Description
MAC Address	The unique address assigned to the mote at the factory.
Name	The mote identifier, as set in Preferences.
State	The current state of the mote in the network: <ul style="list-style-type: none"> • <i>Negotiating</i> indicates the mote is attempting to join the network. • <i>Operational</i> indicates the mote is connected to the network and reporting data. • <i>Lost</i> indicates the mote is not currently in the network.
Reliability	Measures the percentage of packets that were delivered to the manager without getting lost.
Latency	The average time (in milliseconds) required for a data packet to travel from the originating source to its destination.

Received Packets	The number of packets the manager has received from the mote.
Lost Packets	The number of packets the manager expected but did not receive.
Good Neighbors	The number of neighboring motes with a path quality higher value that is than the threshold set in Preferences

Viewing Path Information

Click Paths on the Tools menu to display the path quality and signal strength for all paths in the network. You can display this same information for a selected path by double-clicking the path.



Child	Parent	Quality	RSSI A->B	RSSI B->A
02-5D	01-74(AP)	48	-26	-28
02-5D	02-2F	97	-19	-17
02-2F	01-74(AP)	71	-30	-32
02-65	01-74(AP)	78	-28	-25
02-65	02-2F	97	-15	-15
02-87	02-5D	100	-35	-32
02-87	02-2F	84	-25	-25
02-4F	02-5D	97	-28	-28
02-4F	02-2F	81	-30	-30
02-78	02-5D	78	-26	-29
02-78	02-2F	87	-32	-32
02-2A	02-5D	97	-16	-16

Paths Window

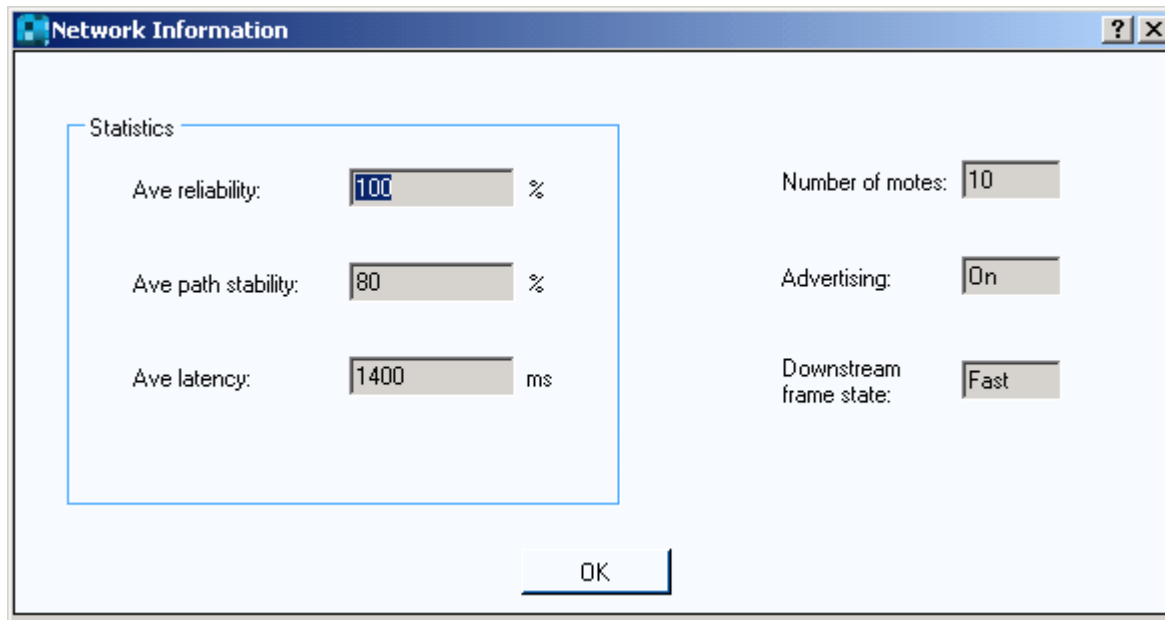
The Paths window displays the following information.

Field	Description
Child	The mote that is sending upstream data on this path.
Parent	The mote that is receiving upstream data on this path.

Quality	The path quality indicated by a value between 0 and 100 in %. A path quality of $\geq 50\%$ is considered good. Paths $< 30\%$ are optimized out if possible.
RSSI A->B	Radio signal strength in dBm for transmissions from the child to the parent.
RSSI B->A	Radio signal strength in dBm for transmissions from the parent mote to the child mote.

Viewing Network Statistics

The SmartMesh IP Manager generates data reliability, path stability, and data latency statistics based on the health reports it receives from each mote every 15 minutes. The Network Information window provides a lifetime summary of network statistics and current network information.



Network Information Window

The Network Information window displays the following information.

Field	Description
Average reliability	The percentage of expected data packets that were actually received. One hundred percent reliability means that every expected data packet was received. The reported values are network averages.

Average stability	Path stability measures the success rate for mote-to-mote transmissions. It is the percentage of data packets for which the sending mote has received an acknowledgement. If a transmitting mote does not receive an acknowledgement, it may resend on an alternate path. Due to the unique benefits of mesh routing, data reliability can be 100% even with very low path stability.
Average latency	The average time (in milliseconds) required for a data packet to travel from the originating mote to the manager. Data latency varies across the network. The value displayed represents the average data latency.
Number of motes	Number of motes in the network.
Advertising	When advertising is on, advertisement packets announcing the presence of the network are frequently sent out by the motes and manager. Motes trying to join the network listen for these packets in order to join. When advertising is turned off, advertisements are not sent out.
Downstream frame state	Indicates whether the network is currently operating at Fast or Normal speed. When the network is forming, it runs faster to facilitate mote joining. Once the network is formed, it runs at a slower (normal) speed to save power.

To view network statistics:

- On the **Tools** menu, click **Network** and then **Information**.

Clearing Network Statistics

When you want to collect a new set of network statistics, you can delete the current average data reliability, path stability, and data latency statistics from the manager. The manager continues collecting statistics going forward, providing a new set for your reference.

To clear all statistics:

- On the **Tools** menu, click **Network** and click **Clear Statistics**.

Monitoring Network Traffic

Click Traffic Monitor on the Tools menu to monitor network communications between Stargazer and the manager. The Traffic Monitor window shows request/response packets exchanged between Stargazer and the manager and event and data notifications received from motes.

Traffic Monitor				
<input checked="" type="checkbox"/> Request/Response		<input checked="" type="checkbox"/> Notifications		<input type="checkbox"/> Scroll to new packets
				Clear
Time	Direction	Mote	Length	Payload
5/26/2011 6:50:17 PM	RX	02-78	100	1404003D240251000203A000170D0000380278F0B9F0B903550100FF020402...
5/26/2011 6:50:17 PM	TX	02-78	44	2C00170D000038027800F0B9F0B900015601FF020401
5/26/2011 6:50:18 PM	TX	02-78	44	2C00170D000038027800F0B9F0B900015701FF020400
5/26/2011 6:50:18 PM	RX	02-78	100	1404003D240252000A316000170D0000380278F0B9F0B903560100FF020401...
5/26/2011 6:50:22 PM	RX	02-78	100	1404003D24025600042E5000170D0000380278F0B9F0B903570100FF020400...
5/26/2011 6:50:22 PM	RX	02-2A	102	1404003D24025600058DE000170D000038022AF0B9F0B900030501FF010500...
5/26/2011 6:50:22 PM	RX	02-15	94	1404003D24025600084D0000170D0000380215F0B9F0B900030500FF010500...
5/26/2011 6:50:23 PM	RX	02-2F	94	1404003D2402570002328000170D000038022FF0B9F0B900030500FF010500...
5/26/2011 6:50:24 PM	RX	02-4F	94	1404003D2402580001388000170D000038024FF0B9F0B900020500FF010500...
5/26/2011 6:50:25 PM	RX	02-18	94	1404003D2402590002887000170D0000380218F0B9F0B900010500FF010500...
5/26/2011 6:50:28 PM	RX	02-5D	94	1404003D24025C0005014000170D000038025DF0B9F0B900040500FF010500...
5/26/2011 6:50:29 PM	RX	02-1A	94	1404003D24025D000927C000170D000038021AF0B9F0B900010500FF010500...

Traffic Monitor Window

The Traffic Monitor window displays the following information.

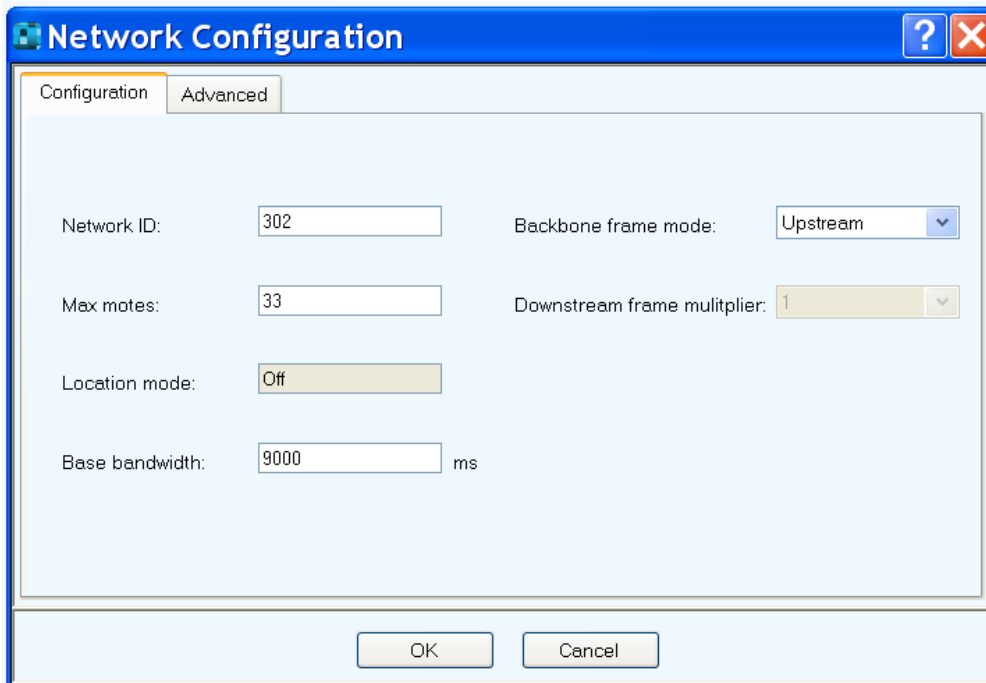
Field	Description
Request/Response check box	When selected, the Traffic Monitor displays request/response packets exchanged between Stargazer and the manager.
Notifications check box	When selected, the Traffic Monitor displays response packets and data packets received by Stargazer from motes.
Scroll to new packets	When selected, continuously scrolls to the latest packets.
Clear	Clears the Traffic Monitor display.
Time	Time the packet was sent by Stargazer or received by Stargazer.
Direction	Direction in which the packet traveled. TX indicates a packet sent by Stargazer. RX indicates a packet received by Stargazer from a mote.
Mote	The mote that sent the packet to Stargazer or received the packet from Stargazer. The mote is identified by its MAC address or mote ID (depending on how Preferences are set).
Length	Packet length, including header information.
Payload	The packet content, in hexadecimal.

Low Latency Mode

SmartMesh IP provides a low latency mode that enhances the speed of upstream communications (mote to manager) and bidirectional communications when a backbone of line powered motes is present. Although the latency improvement is more obvious in control systems (such as lighting control and building automation) than in GUI-based interfaces like Stargazer, you may use Stargazer to configure low latency mode.

To configure the network for low latency:

1. On the **Tools** menu, click **Network** and then click **Configuration**. The following window appears.



2. In the **Backbone frame mode** field , select one of the following options:
 1. **Upstream** —Increases the speed of event notifications sent from the mote to the manager.
 2. **Bidirectional** —Increases the speed of request/response communications between the mote and manager.
3. Click **OK** .
4. On the **Tools** menu, click **System** and then click **Reset System** . This restarts the manager's software processes and wireless connection. The new configuration settings take effect after the network reforms.


Communicating with Mote Applications


The *Communicate with Application* window allows you to use Stargazer to communicate with the mote's built in applications. The default mote shipped with the evaluation kits contains four applications that run on the internal Cortex M3 micro processor, and make use of Eterna's built in features. These are;

- Temperature, using Eterna's built in temperature sensor
- Analog Inputs (4 channels),

- Digital Inputs (4 channels),
- and Digital Outputs (3 channels).

Each application and channel can be individually configured to sample and publish data.

 The analog inputs are 0-1.8V. Applying a signal outside of this range can damage the device.

 For lowest power operation, termination of unused digital inputs (D0-D3) when operating in **master** mode is recommended.

To access the *Communicate with Application* window, select a mote, right click, and select **Communicate with Application**. A window appears with a list of sensor inputs and outputs (channels) and several configuration parameters for each. Note that Stargazer does not display a mote's current status until it is actually queried. To update this window with the mote's current status, click on the **Refresh** button. Note that this action will need to be repeated for each tab to get a complete status of all of the applications running on the mote.

Communicate with 00-17-0D-00-00-30-02-C1
_ □ ×

Analog Inputs
Digital Inputs
Digital Outputs

A0:	Enabled <input type="checkbox"/>	Rate <input type="text"/>	Sample Count <input type="text"/>	Format <input type="text"/>	Value <input type="text"/>
A1:	Enabled <input type="checkbox"/>	Rate <input type="text"/>	Sample Count <input type="text"/>	Format <input type="text"/>	Value <input type="text"/>
A2:	Enabled <input type="checkbox"/>	Rate <input type="text"/>	Sample Count <input type="text"/>	Format <input type="text"/>	Value <input type="text"/>
A3:	Enabled <input type="checkbox"/>	Rate <input type="text"/>	Sample Count <input type="text"/>	Format <input type="text"/>	Value <input type="text"/>
Temp:	Enabled <input type="checkbox"/>	Rate <input type="text"/>	Sample Count <input type="text"/>	Format <input type="text"/>	Value <input type="text"/>

Save
Refresh

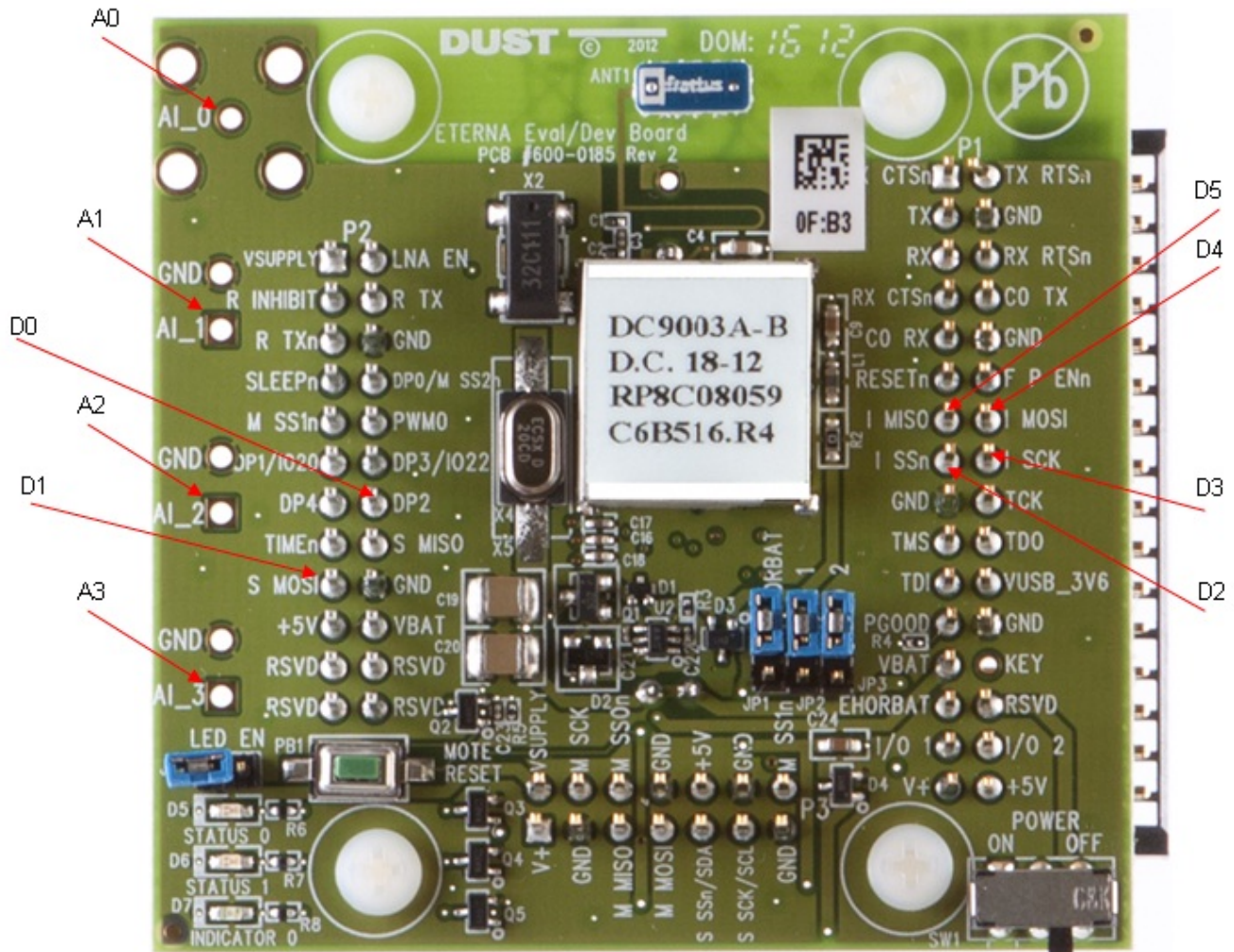
Traffic monitor

TX
 RX
 Scroll to new packets
 Clear

Time	Direction	Mote	Length	Payload

The mote must be in Master mode for the applications listed here to work.

Please refer to the following diagram for I/O locations on the Eternal Evaluation board.



Analog Input Applications

The Analog Inputs tab provides access to two applications, namely Temperature monitoring and analog I/O monitoring. Each field in the display window is described as;

Field	Description
Refresh button	Polls the sensor application on the mote for the current sensor configuration information and data values. The tab area of the window appears grey while the information is being obtained.
Temp:	Embedded temperature sensor on the mote. By default, the temperature sensor is enabled and configured to report the temperature every 30 seconds. Measurements will appear in the Temperature Monitor window available from the Tools menu.
A0 through A3	Analog Input channels

Enabled	When checked, enables data reporting from the sensor on that channel
Rate	Data sample rate, in milliseconds. The rate can be set from 1,000 to 300,000 milliseconds. <ul style="list-style-type: none"> For example, setting the rate at 30000 ms configures the sensor to sample data every 30 seconds.
Sample Count	A sample count of 1 means one data sample is sent per packet. A sample count of 3 means three samples are provided in each packet. If the specified number of samples does not fit in the packet (there is room for 27 samples), additional samples that don't fit are discarded, and the mote sends the packet when the sample count has been reached.
Format	Defines how data is reported in the packet: <ul style="list-style-type: none"> All includes all data samples (samples that do not fit in the packet are dropped) Stats provides the maximum, minimum, and average of all samples
Value	For the Temperature application, the Value will be the temperature measured in degrees Celcius. For the Analog inputs, the Value shows the current voltage reading on the input pin when you click the Refresh button. <ul style="list-style-type: none"> The voltage can be from 0 to 1.8 V, and is represented by a value from 0 to 1024. For example, a value of 636 equates to 1.11 V: $(636/1024 \times 1.8 = 1.11 \text{ V})$ <p>Note: If there is nothing connected to the pin, it will float and the value can be anything within the valid range.</p>
Save button	Saves changes made to the configuration. The tab area of the window appears grey while the mote is being configured.
Traffic Monitor	Displays traffic sent and received by Stargazer. For details, see "Monitoring Network Traffic" above .
Payload	Packet payload. Note that this payload does not include the packet header, as it does in the main Traffic Monitor window. For details, see the sample payloads provided in the "On-chip Application Protocol" section of this Guide.

To configure an analog sensor (A2 or Temperature):

In Stargazer, right-click the mote and select **Communicate with Application**.

The **Communicate with Application** dialog box appears. The **Analog Inputs** tab is selected by default.

1. Click **Refresh** to obtain the current configuration information from the sensor applications on the mote.

Stargazer sends a series of commands to the sensor application on the mote requesting the current configuration settings for all analog sensors.

1. Select the **Enabled** check box for the analog input.
2. Specify the **Rate**, **Sample Count** and **Format**, as described in the table above.
3. Click **Save**.

Stargazer sends a configuration command to the sensor application on the mote. The **Analog Inputs** tab appears grey while the command is sent and a response is received. When the configuration is complete, the data notifications from sensor start to appear in the **Traffic Monitor** at the bottom of the window.

- Note: When you configure a mote to publish at a rate faster than 1 packet every 3 seconds, there may be a delay of several minutes after pressing **Save** before the manager assigns the additional links needed to support the data rate. During this period, the mote may be unable to receive additional configuration commands since they must be processed in order. If a configuration command does not appear to take effect (the **Traffic Monitor** does not reflect the change), try pressing **Save** again.

You can click **Refresh** at any time to update the **Value** field with the current sensor reading. The **Value** field does not update automatically, the **Refresh** button must be pressed.

Digital Input Applications

The Digital Inputs tab provides access to an application that monitors up to four digital inputs. Each field in the display window is described as;

Field	Description
Refresh button	Polls the sensor application on the mote for the current sensor configuration information and data values. The tab area of the window appears grey while the information is being obtained.
D0 through D3	Digital Input channels
Enabled	When checked, enables data reporting from the sensor on that channel
Rate	Data sample rate, in milliseconds. The rate can be set from 1,000 to 300,000 milliseconds. <ul style="list-style-type: none"> • For example, setting the rate at 30000 ms configures the sensor to sample data every 30 seconds.
Sample Count	A sample count of 1 means one data sample is sent per packet. A sample count of 3 means three samples are provided in each packet. If the specified number of samples does not fit in the packet (there is room for 54 samples), additional samples that don't fit are discarded, and the mote sends the packet when the sample count has been reached.
Format	Defines how data is reported in the packet: <ul style="list-style-type: none"> • <i>All</i> includes all data samples (samples that do not fit in the packet are dropped) • <i>On change</i> provides a sample when the input changes from 0 to 1 or vice versa • <i>On rising</i> provides a sample whenever a rising edge (0 to 1 transition) is seen • <i>On falling</i> provides a sample whenever a falling edge (1 to 0 transition) is seen

Value	<p>Either a 1 or 0.</p> <p>See the LTC5800-IPM Datasheet for minimum I/O levels.</p> <p>Note: If there is nothing connected to the pin, it will float and the value can be anything within the valid range.</p>
Save button	Saves changes made to the configuration. The tab area of the window appears grey while the mote is being configured.
Traffic Monitor	Displays traffic sent and received by Stargazer. For details, see "Monitoring Network Traffic" above.
Payload	Packet payload. Note that this payload does not include the packet header, as it does in the main Traffic Monitor window. For details, see the sample payloads provided in the "On-chip Application Protocol" section of this Guide.

To configure an digital input (D0-D3):

In Stargazer, right-click the mote and select **Communicate with Application**.

The **Communicate with Application** dialog box appears. Select the **Digital Inputs** tab.

1. Click **Refresh** to obtain the current configuration information from the sensor applications on the mote.

Stargazer sends a series of commands to the sensor application on the mote requesting the current configuration settings for all analog sensors.

1. Select the **Enabled** check box for the digital input.
2. Specify the **Rate**, **Sample Count** and **Format**, as described in the table above.
3. Click **Save**.

Stargazer sends a configuration command to the sensor application on the mote. The **Digital Inputs** tab appears grey while the command is sent and a response is received. When the configuration is complete, the data notifications from sensor start to appear in the **Traffic Monitor** at the bottom of the window.

- Note: When you configure a mote to publish at a rate faster than 1 packet every 3 seconds, there may be a delay of several minutes after pressing **Save** before the manager assigns the additional links needed to support the data rate. During this period, the mote may be unable to receive additional configuration commands since they must be processed in order. If a configuration command does not appear to take effect (the **Traffic Monitor** does not reflect the change), try pressing **Save** again.

You can click **Refresh** at any time to update the **Value** field with the current sensor reading. The **Value** field does not update automatically, the **Refresh** button must be pressed.

Digital Output Applications

The Digital Outputs tab provides access to two output pins D4 and D5, and the INDICATOR_0 (blue) LED on the [DC9003](#) board. Each field in the display window is described as;

Field	Description
Refresh button	Polls the sensor application on the mote for the current sensor configuration information and data values. The tab area of the window appears grey while the information is being obtained.
D4, D5, Indicator	Digital Output channels. Indicator is connected to the INDICATOR_0 LED.
Value	The value field can take one of three states: <ul style="list-style-type: none"> • <i>No change</i> - the value is unchanged. Can be used when configuring other outputs when an existing output is already enabled. • 0 - set the output to low (gnd) • 1 - set the output to high (Vsupply)
Save button	Saves changes made to the configuration. The tab area of the window appears grey while the mote is being configured.
Traffic Monitor	Displays traffic sent and received by Stargazer. For details, see "Monitoring Network Traffic" above.
Payload	Packet payload. Note that this payload does not include the packet header, as it does in the main Traffic Monitor window. For details, see the sample payloads provided in the "On-chip Application Protocol" section of this Guide.

To configure an Digital Output (D4, D5, or Indicator):

In Stargazer, right-click the mote and select **Communicate with Application**.

The **Communicate with Application** dialog box appears. Select the **Digital Outputs** tab.

1. Click **Refresh** to obtain the current configuration information from the sensor applications on the mote.

Stargazer sends commands to the mote sensor application requesting the current configuration settings for all analog sensors.

1. Specify the **Value** for the digital output as described in the table above.
2. Click **Save**.

Stargazer sends a configuration command to the sensor application on the mote. The **Digital Outputs** tab appears grey until a response is received. You can click **Refresh** at any time to review the current settings.

7 SmartMesh IP 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.

7.1 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.

7.2 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.

7.3 Document Organization

The remainder of this guide is organized as follows, and is 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.

7.4 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

7.5 Example Applications

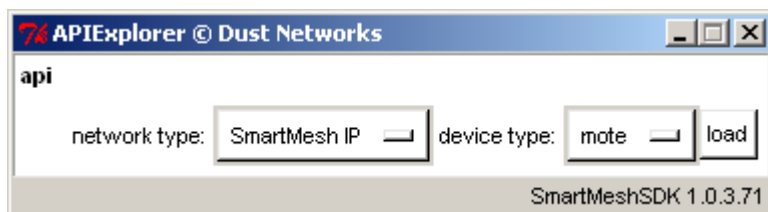
7.5.1 Running An Application

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

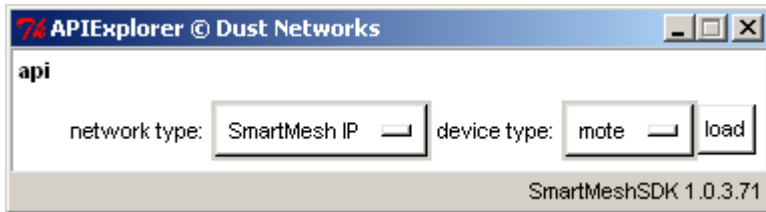


- ✔ This is the recommended way of running the application if you do **not** wish to modify the behavior of the application.

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



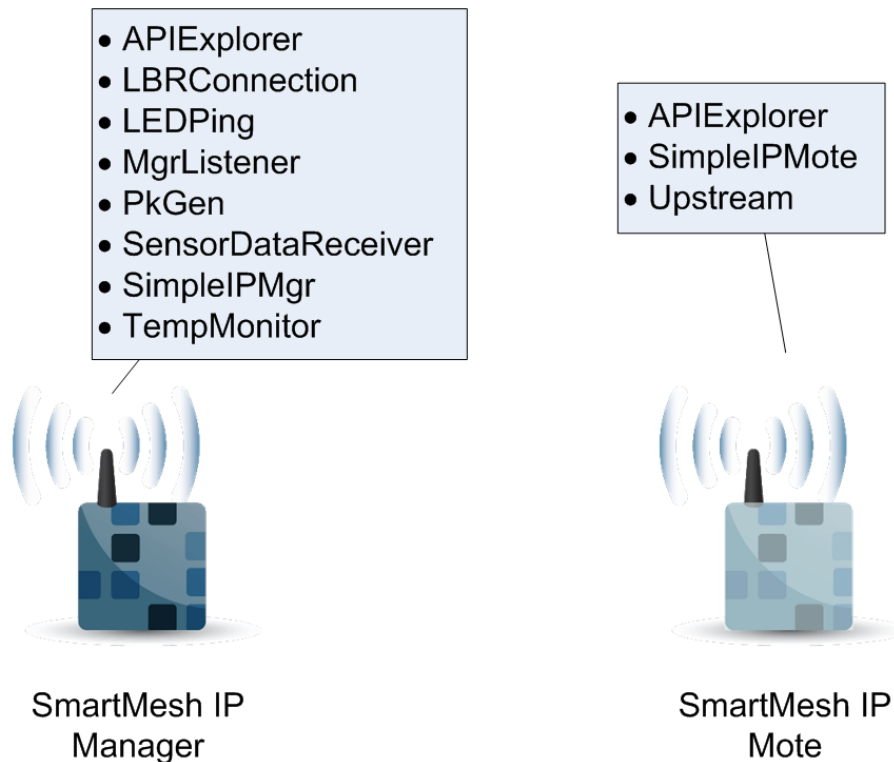
This is the recommended way of running the application if you are modifying the behavior of the application by editing its source code.

7.5.2 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 a non-existing serial port.

7.5.3 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.
- **LBRConnection:** The LBRConnection application allows you to connect to a running LBR to view information about LBR and counts of the packets flowing through it.
- **LEDPing:** The LEDPing application continuously toggles the LED on a SmartMesh IP Mote running in **master** mode with the default firmware.
- **MgrListener:** The MgrListener application connects to the SmartMesh IP Manager and subscribes to all types of notifications, displaying a counter for each type received.
- **MuxConfig:** The Serial Mux Configurator provides a GUI editor for managing Serial Mux configurations.
- **OTAPCommunicator:** OTAPCommunicator is a tool for loading firmware onto motes via Over-the-Air Programming (OTAP).
- **PkGen:** PkGen connects to either a SmartMesh IP Manager or SmartMesh WirelessHART Manager, allowing you to send commands to the packet generation application on the corresponding motes running in **master** mode.
- **SensorDataReceiver:** The SensorDataReceiver application connects to the SmartMesh IP Manager and displays data sent by the `Upstream` application connected to the SmartMesh IP Mote.
- **Simple:** SimpleIPMgr and SimpleIPMote are console (non-graphical) applications which show how to interact programmatically with the API of the respective devices. They are meant to serve as sample code.

- **TempMonitor**: TempMonitor connects to either a SmartMesh IP Manager or SmartMesh WirelessHART Manager, allowing you to send commands to the temperature sampling application on the corresponding motes running in **master** mode.
- **Upstream**: The Upstream application takes the SmartMesh IP Mote through the joining, service request, and socket binding state machines, and allows the user to enter dummy "sensor" data and send it to the SmartMesh IP Manager or any Internet host.

New in 1.0.4:

- **DC2126A**: Sample application which receives and parses data from the DC2126A board.
- **Xively**: Publishes sensor data to Xively and subscribes to changes.

7.5.4 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: device type:

connection

through serialMux:

host: port:

Connection successful.

command

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 of 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.
- Responses 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 receive 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](#)

7.5.5 DC2126A

Introduction

Sample application which receives and parses data from the DC2126A board.

It can connect to:

- The SmartMesh IP Manager



This application is only useful if you have one or more DC2126A boards running in your SmartMesh IP network.

Running

You can run the DC2126A application:

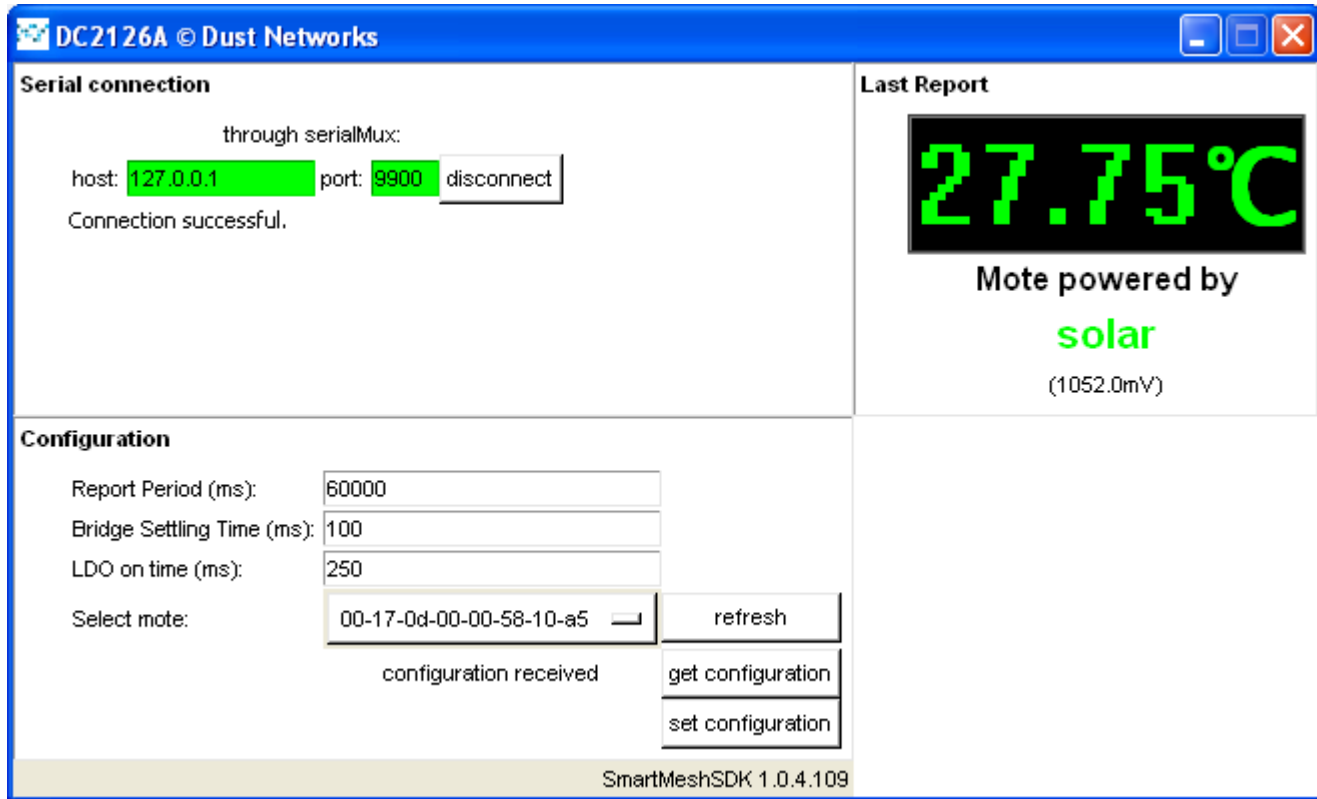
- by double-clicking on the Windows executable at `/win/ DC2126A.exe`
- by double-clicking on its source files at `/src/ bin/DC2126A/DC2126A.py` (may require additional steps on non-Windows OSes)

Description

DC2126A is a GUI-based sample application. It connects to your SmartMesh IP Manager over a serialMux or serial connection. If you have one or more DC2126A boards running in your SmartMesh IP network, it will automatically parse and display the data that each board gathers.

- Make sure your DC2126A board is running and is part of your SmartMesh IP network.
- Start the DC2126A sample application and connect it to your SmartMesh IP Manager. You can connect either over the serialMux or serial port.
- When the DC2126A sample application connects to the SmartMesh IP Manager, it retrieves the motes in the network, and populates the **Select mote** drop-down menu.
- Each time the DC2126A board publishes data, that data is displayed in the upper-right hand corner of the DC2126A sample application.
- To see the data generated by a different DC2126A board in your SmartMesh IP network, select its MAC address using the **Select mote** drop-down menu.
- Clicking on the **get configuration** button sends a packet to the mote selected in the **Select mote** drop-down menu, requesting its current configuration. The mote answers with its current configuration, which is displayed in the configuration form.
- To modify the configuration of a particular mote: enter the report period, bridge settling time, and LDO on time; select mote's MAC address in the **Select mote** drop-down menu; and click **set configuration**.

- To update the MAC address in the **Select mote:** drop-down menu, click on the **refresh** button.



7.5.6 HrListener

Introduction

The HrListener application connects to the SmartMesh IP Manager, subscribes to health report notifications, and displays a table containing a count of each type of HR received.

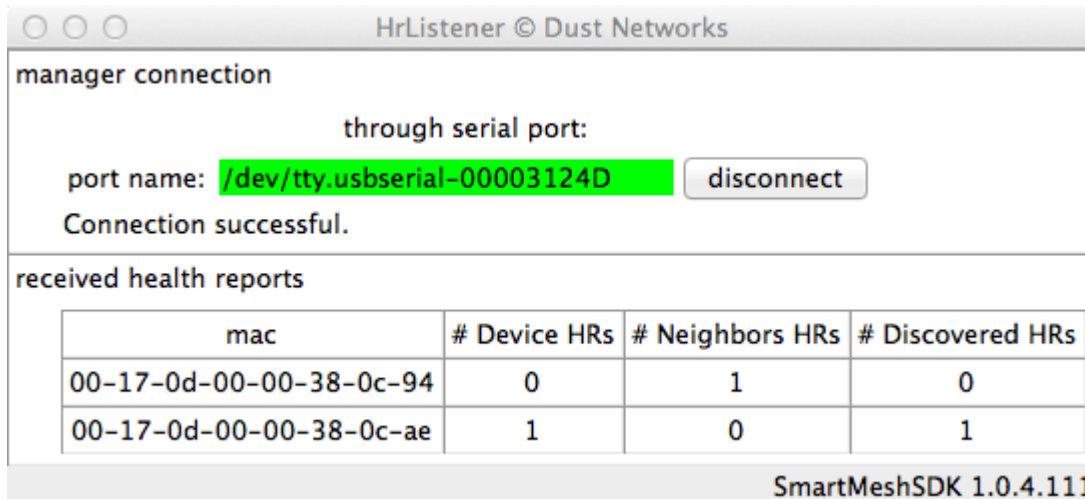
Running

You can run the HrListener application:

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

Description

HrListener is a GUI based application which connects to the SmartMesh IP Manager, subscribes to health report notifications, and displays a table containing a count of each type of HR received. It does not parse the contents of each HR other than to determine type.



HrListener logs parsed health reports to `receivedHRs.log`, found in the same directory as the application (either `.exe` or `.py`). A sample is shown below:

```

2014-06-05 10:49:04,629 [INFO] ===== START LOGGING HEALTH REPORTS =====
2014-06-05 10:56:00,294 [INFO] from 00-17-0d-00-00-38-0c-86:
- Device:
- badLinkFrameId : 0
- badLinkOffset : 0
- badLinkSlot : 0
- batteryVoltage : 3093
- charge : 168
- numMacDropped : 0
- numRxLost : 0
- numRxOk : 7
- numTxBad : 0
- numTxFail : 0
- numTxOk : 25
- queueOcc : 65
- temperature : 25
2014-06-05 10:59:26,829 [INFO] ===== STOP LOGGING HEALTH REPORTS =====

```

7.5.7 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://sourceforge.net/projects/pywin32/files/pywin32/
Press Enter to exit.
```

7.5.8 LBRConnection

Introduction

The Low-power Border Router (LBR) sits between a SmartMesh IP Manager and the internet. It converts between the IPv6 packet format of the internet to the 6LoWPAN packet format of the SmartMesh IP network to enable communication between computers on the Internet ("internet hosts") and SmartMesh IP Motes.

A sample LBR implementation that runs on a Linux computer is available from Linear.com. The LBRConnection application allows you to connect to a running LBR to view information about LBR and counts of the packets flowing through it.

Running

You can run this application:

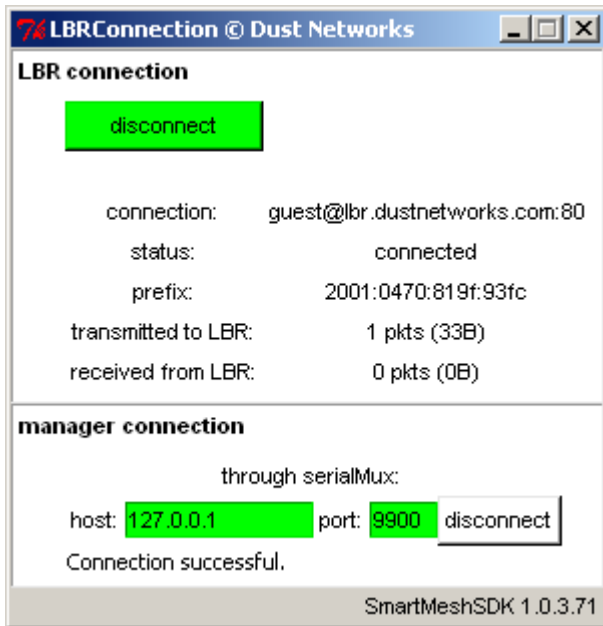
- By double-clicking on the Windows executable at `/win/LBRConnection.exe`.
- By double-clicking on its source files at `/src/bin/LBRConnection/LBRConnection.py`.

Description

GUI organization

The LBRConnection application consists of two frames:

- The "LBR connection" frame allows you to connect to an [LBR](#).
- The "manager connection" frame allows you to connect to the SmartMesh IP Manager.



Internally, the application subscribes to ip notifications from the SmartMesh IP Manager. When it receives such a notification, it forwards the packet to the LBR it is connected to. Similarly, when receiving data from the LBR, it forwards it to the SmartMesh IP Manager.

Using .lbrauth files

To connect to the LBR, the LBRConnection application needs to know the following information:

- IP address of the LBR
- TCP port number the LBR is listening on
- Username of the network
- Credentials

The credentials needed depend on the security level, as described in the [LBR User Guide](#). Rather than having to type this information in fields, the LBRConnection application parses "LBR authentication files" (*.lbrauth). One such file is provided with your SmartMesh SDK installation.

Information displayed

The "LBR connection" frame displays the following information:

field	description
connection	When connected to the LBR, indicates the username and address of the LBR as <code>username@address</code> .
status	Displays the current status, either <code>connected</code> or <code>disconnected</code> .
prefix	When connected to the LBR, indicates the IPv6 prefix assigned by the LBR to your network.

transmitted to LBR	Number of packets and bytes transmitted to the LBR.
received from LBR	Number of packets and bytes received from the LBR.

Walk-through

This application is used in the following tutorials:

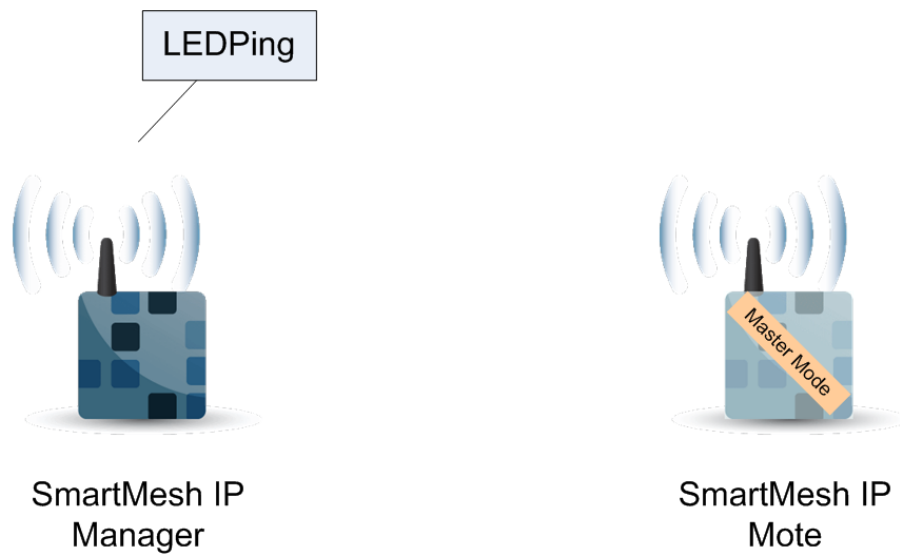
- For SmartMesh IP:
 - [Internet Integration](#)

7.5.9 LEDPing

Introduction

The LEDPing application continuously toggles the LED on a SmartMesh IP Mote running in **master** mode with the default firmware.

Setup



Running

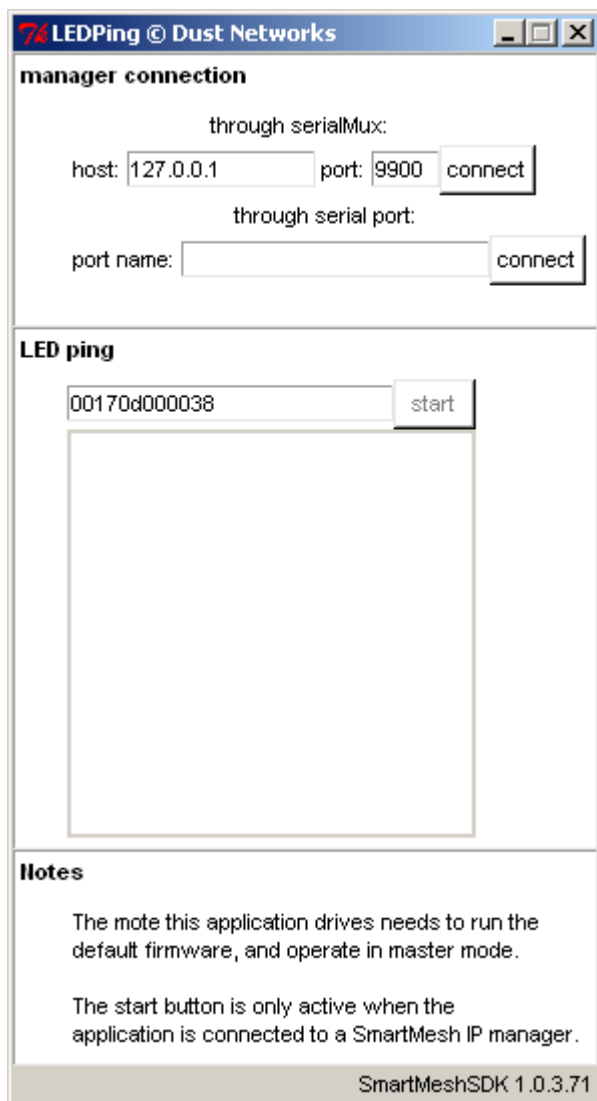
You can run this application:

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

GUI organization

The LEDPing application consists of two frames:

- The **manager connection** frame allows you to connect to the SmartMesh IP Manager.
- The **LED ping** frame allows you to select the SmartMesh IP Mote or interest, and monitor its LED state.
- The **Notes** frame contains some help information.



Description

The default firmware of the SmartMesh IP Mote implements the [On-chip Application Protocol \(OAP\)](#). The LEDPing application connects to the SmartMesh IP Manager and sends OAP packets to any SmartMesh IP Mote in the network, to continuously toggle its `INDICATOR` LED. That is, after pressing the **start** button:

- The LEDPing application sends an OAP packet to the selected SmartMesh IP Mote to set its LED on.
- Upon receiving the packet, the SmartMesh IP Mote set the LED, and sends an application-level acknowledgment.
- Upon receiving this application-level acknowledgment, the LEDPing application sends a new OAP packet to set the LED off.
- These steps repeat until the **stop** button is pressed.

You can use this application "see" the round-trip time in the network. With the default configuration, the LED will blink every 2-3 seconds. Changing the networks configuration (e.g. by switching on the backbone mode) can result in a much faster blinking rate.

Using the application

- Start the LEDPing application:

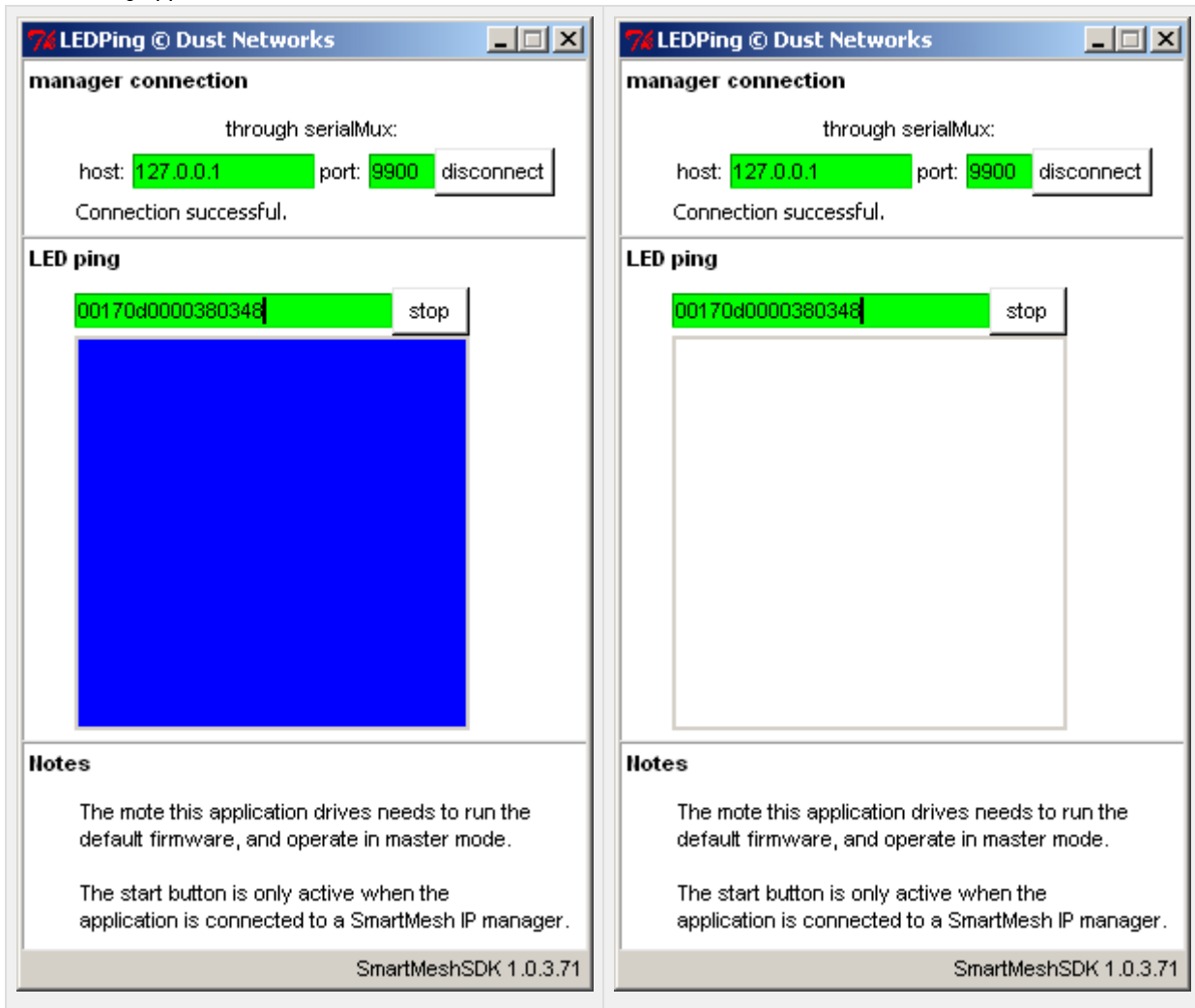


- Connect the application to your SmartMesh IP Manager, either through the SerialMux, or directly over the serial port. The fields in the **manager connection** frame turn green, indicating a successful connection.



- Enter the full MAC address of any mote currently in operational mode in the network. Press the **start** button.

- The LEDPing application will blink blue at the same time as the LED on the SmartMesh IP Mote.



- At any time, click the **stop** button to interrupt the blinking.



7.5.10 MgrListener

Introduction

The MgrListener application connects to the SmartMesh IP Manager and subscribes to all types of notifications, displaying a counter for each type received.

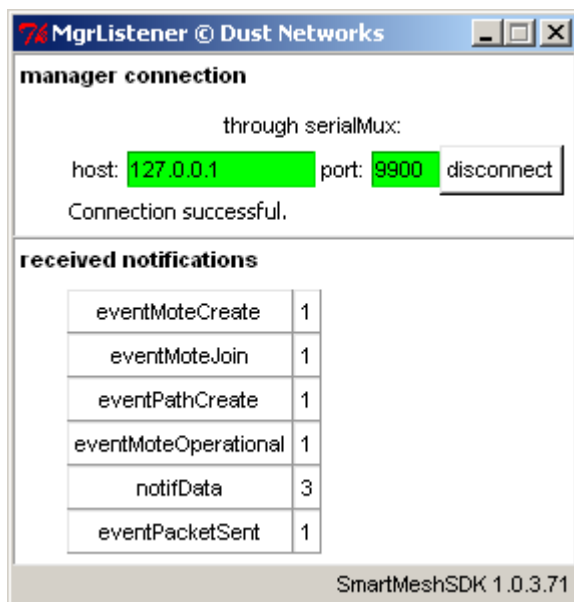
Running

You can run the MgrListener application:

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

Description

MgrListener is a GUI based application which connects to the SmartMesh IP Manager, subscribes to all notifications, and displays the number of each notification it receives.



7.5.11 MuxConfig

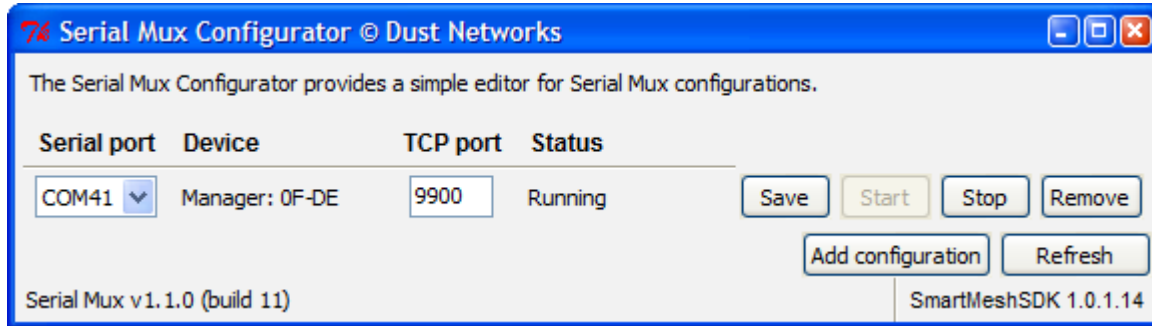
Overview

The Serial Mux Configurator provides a GUI editor for managing Serial Mux configurations.

Each configuration represents a Serial Mux process (which runs as a background service on Windows) that is connected to a manager. If you have multiple managers connected to your computer, you will need to set up multiple Serial Mux configurations.

The Configurator allows you to:

- add, update and remove Serial Mux configurations, e.g. if you connect a manager to a different serial port,
- start and stop the Serial Mux processes, and
- view basic device information for connected devices.

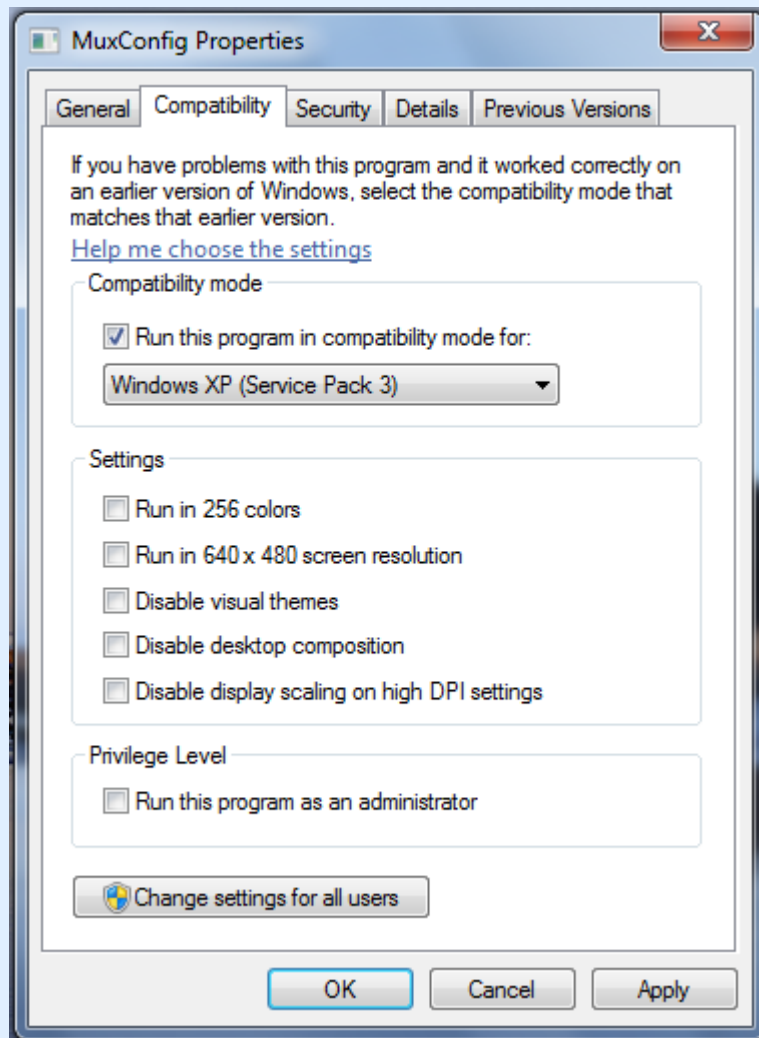


Running the Configurator

Currently, the Configurator is distributed with the SmartMesh SDK. The Configurator requires the [Serial Mux](#) to be installed. Run the Configurator by double-clicking the pre-compiled Windows executable, `MuxConfig.exe`.

i Under Windows 7, the Configurator must be run under XP compatibility mode. Some versions of Windows 7 do not support XP compatibility mode, for these versions, use SmartMeshSDK v1.0.2.53 or later and allow the Configurator to run as an administrator.

1. Locate `MuxConfig.exe`, right click and select **Properties**.
2. Under the Compatibility tab, select **Run this program in compatibility mode for:** and select **Windows XP (Service Pack 3)**. Or if compatibility mode is not available, select **Run this program as an administrator**. Click **OK**.



Using the Configurator

On startup, the Configurator populates its window with a list of the current Serial Mux configurations. The list of serial ports (in the Serial Port combobox) is populated with the list of USB serial ports. The list of serial ports may include ports associated with non-Dust devices.

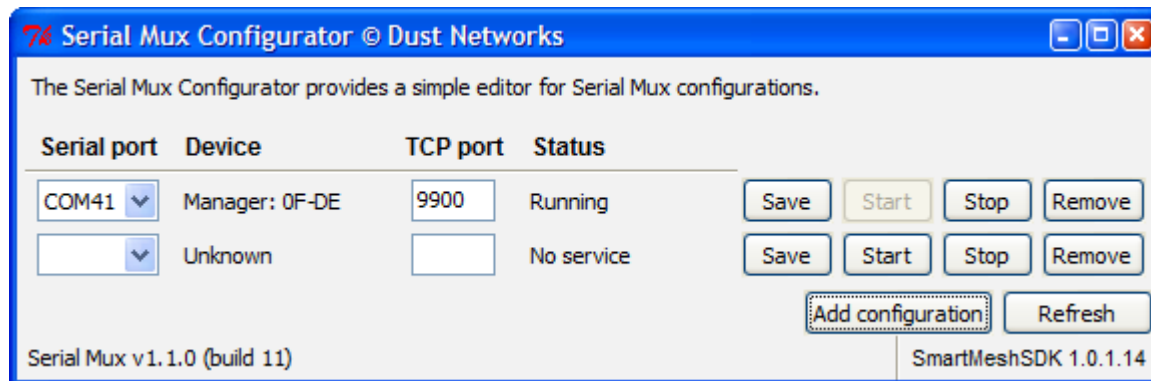
Editing a configuration

To edit a configuration, change the Serial port and/or TCP port fields. Click *Save* to update the configuration. If the Serial Mux service is already running, the Configurator will stop and restart the service.

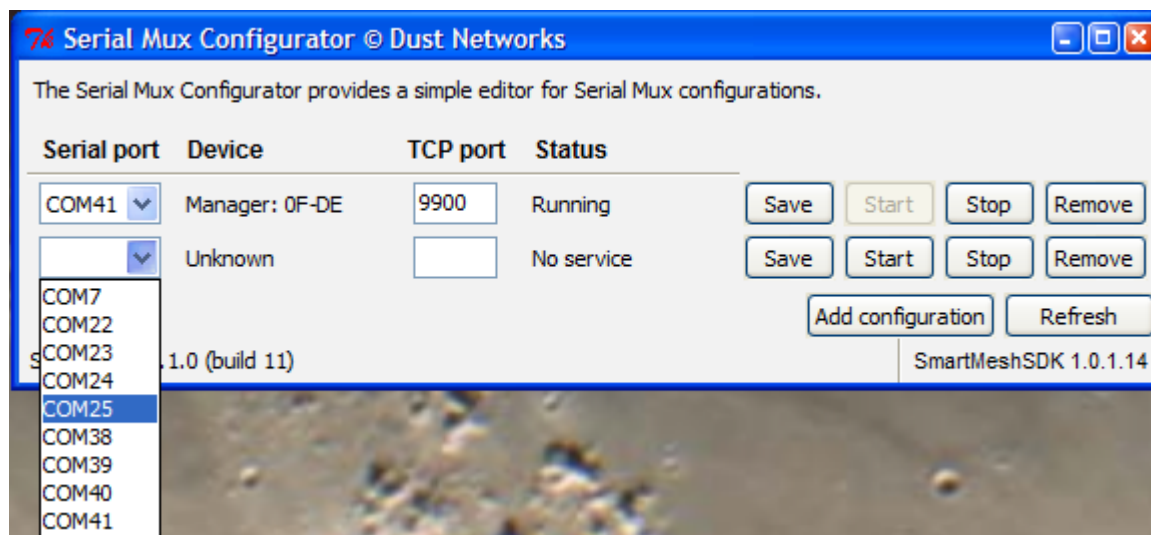
The TCP port must be different from other TCP ports.

Creating a new configuration

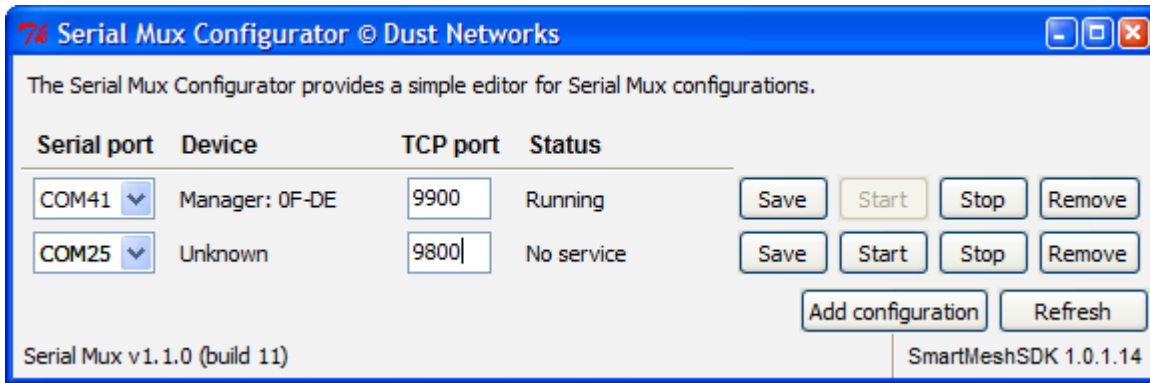
Clicking the **Add configuration** button will create an empty configuration.



Edit the Serial port and TCP port fields to fill in the desired configuration. The TCP port must be chosen so there are no conflicts with other Serial Mux services. (Conflicts with the TCP ports used by other services on the computer are also possible.)

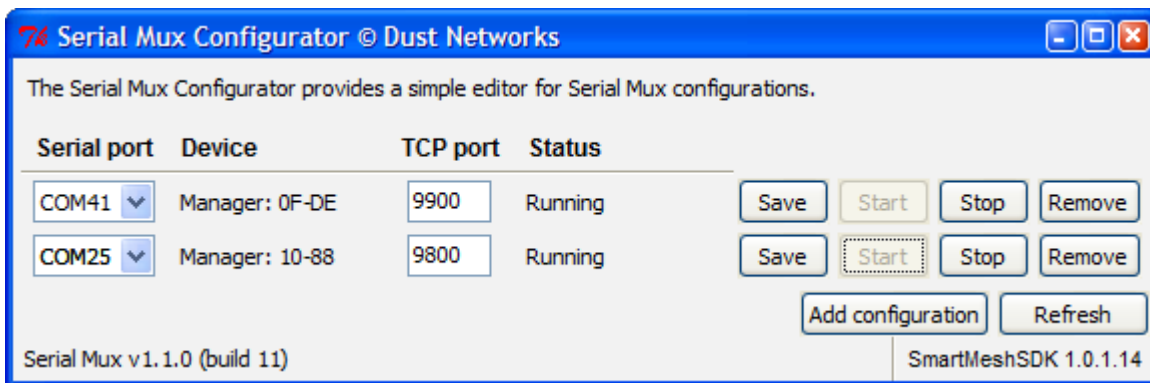


Click **Save** to update the configuration.



Click **Start** to start the Serial Mux process.

If the Device field changed from **Unknown** to **Manager**, then the Serial Mux has connected to the Manager. Use the APIExplorer or another interactive tool to connect to the Manager on the Serial Mux's TCP port.



In this example, connecting the APIExplorer to port 9900 (the Serial Mux default) will connect to the Manager with a MAC address ending in 0F-DE and connecting to port 9800 will connect to the Manager with a MAC address ending in 10-88.

Removing a configuration

Click the **Remove** button to remove a specific Serial Mux configuration. If the associated Serial Mux process is running, it will be stopped and the Serial Mux configuration will be removed.

7.5.12 OTAP Communicator

Introduction

OTAPCommunicator is a tool for loading firmware onto motes via Over-the-Air Programming (OTAP).

The OTAP process sends software updates or files through the wireless network so that software upgrades can be distributed in a network without requiring physical access to each mote. The OTAP process reliably sends the file data in small chunks, querying the mote for completeness and resending chunks that a mote did not receive. The OTAP process generally takes a long time (the duration can be measured in hours) to load a complete mote image over the network.

The OTAP Communicator is provided as part of the SmartMesh SDK as a command line executable for Windows and as Python source code.

The OTAP Communicator requires the [Serial Mux](#) to be connected with the SmartMesh IP Manager.

Command line arguments

The OTAP Communicator is run from the Windows command line prompt. The program takes as input the path to the file(s) to load and optionally a list of destination motes. Files are treated as normal files to be loaded into the mote's file system unless the file has an `.otap` or `.otap2` extension (SmartMesh IP Motes require the `.otap2` file format).

```
OTAPCommunicator.exe [-v] [-p port] [-m MAC]... <file>...
```

Option	
-v	Enable verbose output for debugging.
-p	Specify the TCP port of the Serial Mux.
-m	Specify the MAC address of the destination mote(s). This option can be provided multiple times. If the <code>-m</code> option is not specified, the file (or image) will be distributed to all motes.

Example

When the OTAP Communicator runs, it prints some helpful messages.

```
$ OTAPCommunicator.exe -m 00-17-0d-00-00-38-01-8d mote_ip_1_1_0_36_oski.otap2
Welcome to the OTAP communicator console
Loading mote_ip_1_1_0_36_oski.otap2
Starting OTAP for mote_ip_1_1_0_36_oski.otap2
Starting handshake with 1 motes
Handshake completed, 1 motes accepted
Starting data transmission of 2798 blocks, 1 motes left
...<long wait>...
Starting status query to 1 motes
Data complete. No motes left on incomplete list
Starting commit for 'mote_ip_1_1_0_36_oski.otap2' to 1 motes
00-17-0D-00-00-38-01-8D committed mote_ip_1_1_0_36_oski.otap2 [FCS=0x6ca0]
Successful OTAP to:
00-17-0D-00-00-38-01-8D
```

This output shows:

- The OTAP Communicator loaded the `mote_ip_1_1_0_36_oski.otap2` file.
- The OTAP Communicator initiated an OTAP session with 1 mote (because one mote was listed on the command line).
- 1 mote accepted the OTAP Handshake, which means the OTAP Communicator will start sending the file to the mote.
- The file data was transmitted successfully.

- Each successful commit shows the MAC address and FCS.
- A list of all successes and failures.

Detailed output is logged to the `otap_communicator.log` log file in the directory from which the OTAP Communicator was run.

7.5.13 PkGen

Introduction

PkGen connects to either a SmartMesh IP Manager or SmartMesh WirelessHART Manager, allowing you to send commands to 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:

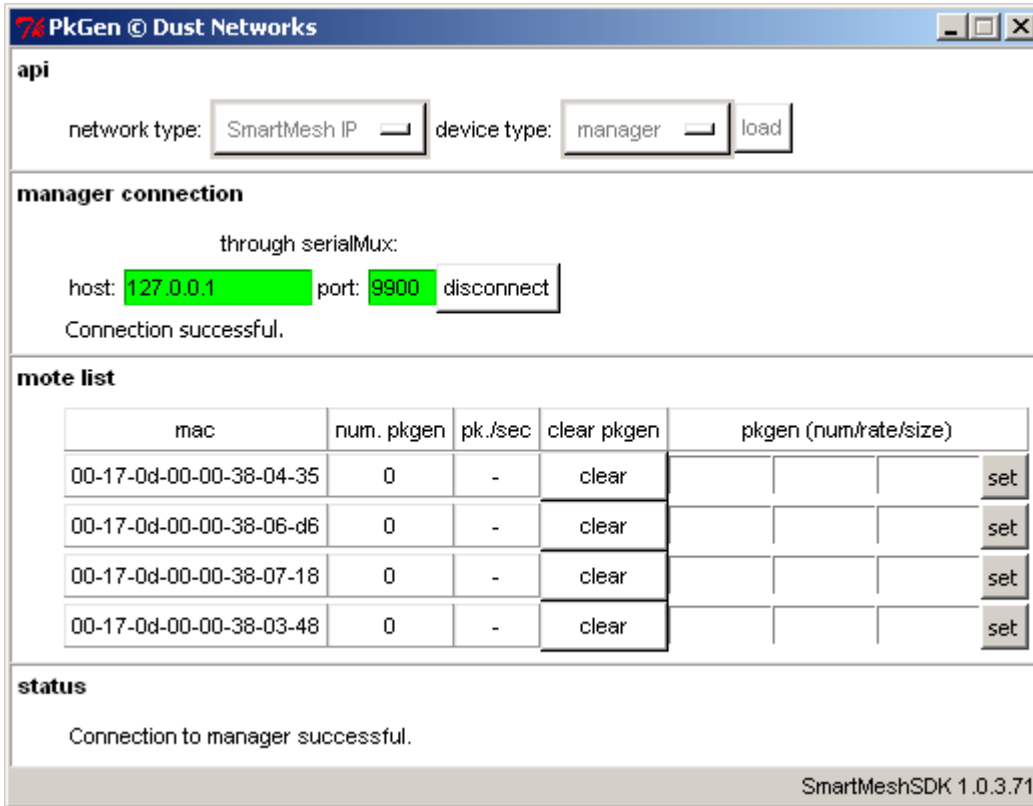
- by double-clicking on the Windows executable at `/win/ PkGen.exe`
- by double-clicking on its source files at `/src/ bin/PkGen/PkGen.py` (may require additional steps on non-Windows OSes)

Description

The PkGen application consists of four frames:

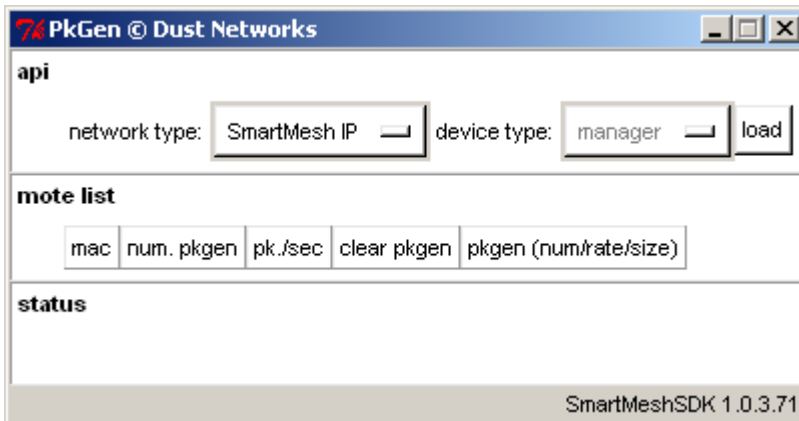
- The **api** frame allows you to specify whether you are connecting to a SmartMesh IP or SmartMesh WirelessHART network. Since you are connecting to a manager, the "device type" selector is disabled.
- The **manager connection** frame allows you to specify how to connect to the manager.
- Once connected, 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.

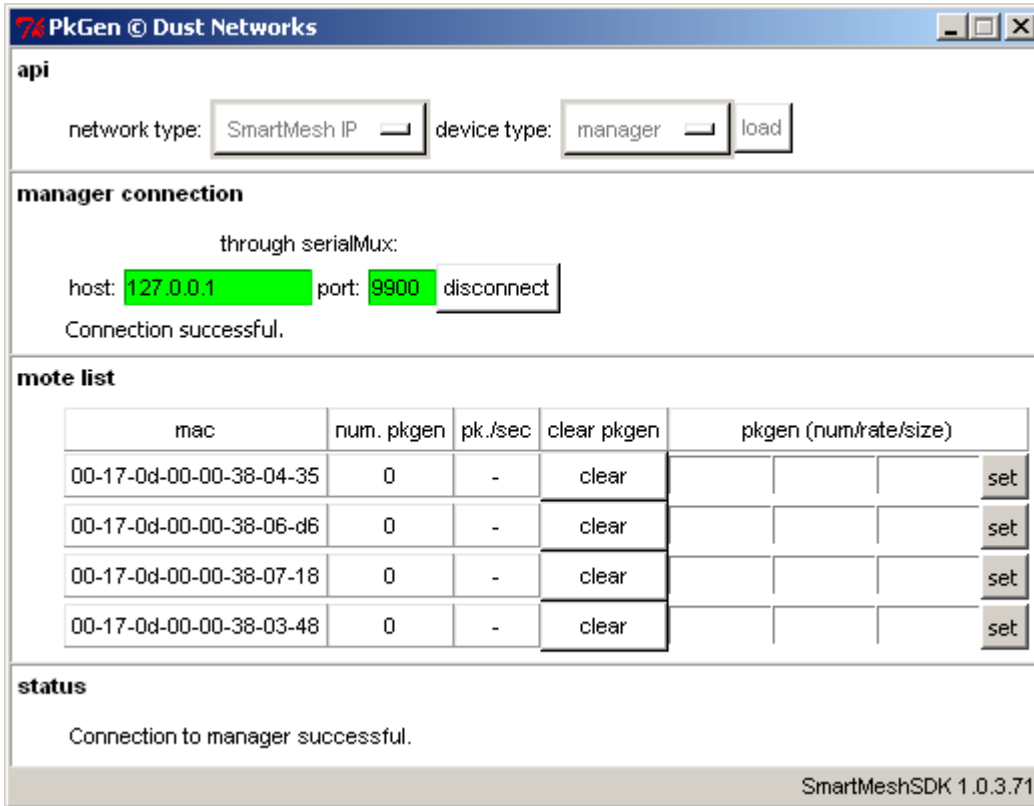


Steps

- 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.



api

network type: device type:

manager connection

through serialMux:

host: port:

Connection successful.


mote list


mac	num. pkgen	pk./sec	clear pkgen	pkgen (num/rate/size)		
00-17-0d-00-00-38-04-35	0	-	<input type="button" value="clear"/>	<input type="text"/>	<input type="text"/>	<input type="text"/> <input type="button" value="set"/>
00-17-0d-00-00-38-06-d6	0	-	<input type="button" value="clear"/>	<input type="text"/>	<input type="text"/>	<input type="text"/> <input type="button" value="set"/>
00-17-0d-00-00-38-07-18	0	-	<input type="button" value="clear"/>	<input type="text"/>	<input type="text"/>	<input type="text"/> <input type="button" value="set"/>
00-17-0d-00-00-38-03-48	0	-	<input type="button" value="clear"/>	<input type="text"/>	<input type="text"/>	<input type="text"/> <input type="button" value="set"/>

status

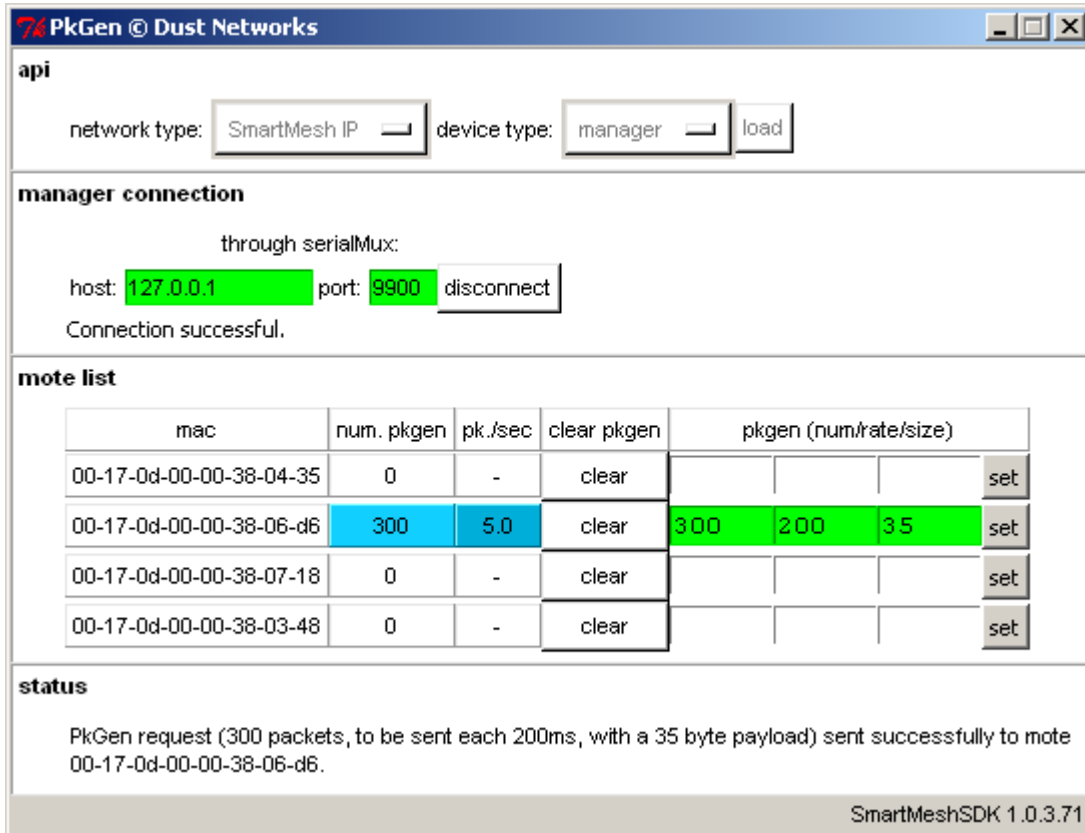
Connection to manager successful.

SmartMeshSDK 1.0.3.71

 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** or **slave** mode. You can only interact with motes that you know are running in master mode.

- For the mote of your 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.



PkGen @ Dust Networks

api
network type: SmartMesh IP device type: manager load

manager connection
through serialMux:
host: 127.0.0.1 port: 9900 disconnect
Connection successful.

note list

mac	num. pkgen	pk./sec	clear pkgen	pkgen (num/rate/size)			
00-17-0d-00-00-38-04-35	0	-	clear				set
00-17-0d-00-00-38-06-d6	300	5.0	clear	300	200	35	set
00-17-0d-00-00-38-07-18	0	-	clear				set
00-17-0d-00-00-38-03-48	0	-	clear				set

status
PkGen request (300 packets, to be sent each 200ms, with a 35 byte payload) sent successfully to mote 00-17-0d-00-00-38-06-d6.

SmartMeshSDK 1.0.3.71

- 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.

7.5.14 SensorDataReceiver

Introduction

The SensorDataReceiver application connects to the SmartMesh IP Manager and displays data sent by the `Upstream` application connected to the SmartMesh IP Mote.

Running

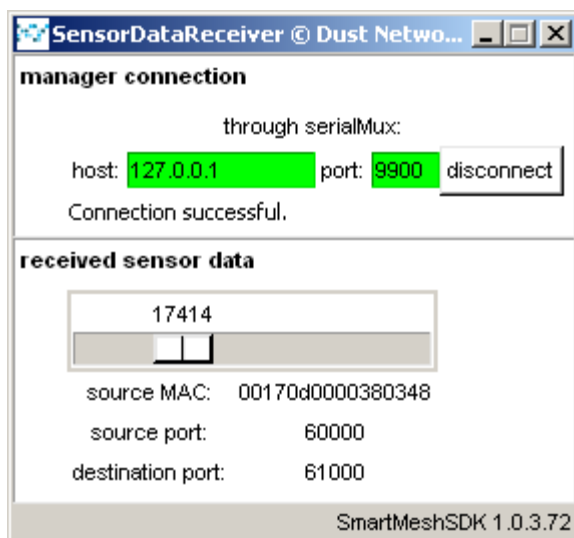
You can run this application:

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

Description

The SensorDataReceiver application consists of two frames:

- the **manager connection** frame allows you to connect to the SmartMesh IP Manager.
- the **received sensor data** displays the data received from an Upstream application connected to a SmartMesh IP Mote.



Internally, the application subscribes to data notifications from the SmartMesh IP Manager. When it receives such a notification, it sets the two first bytes on the slider and displays information about the packet.

field	description
source MAC	The MAC address of the mote sending the data.
source port	Source UDP port number, i.e. port number associated with the socket on the SmartMesh IP Mote through which the data was sent.

destination port	UDP port number on the SmartMesh IP Manager on which the data was received.
timestamp generated	Network timestamp, in seconds, at which the data was generated by the SmartMesh IP Mote.
timestamp received	Computer timestamp at which the data was received at the SmartMesh IP Manager.



You will only see data in this application if you are also using the `Upstream` application connected to a mote.

7.5.15 SimpleIPMgr and SimpleIPMote

Introduction

SimpleIPMgr and SimpleIPMote are console (non-graphical) applications which show how to interact programmatically with the API of the respective devices. They are meant to serve as sample code.

Running

This sample application is a collection of different scripts:

- SimpleIPMgr which exercises the API of the SmartMesh IP Manager
- SimpleIPMote which exercises the API of the SmartMesh IP Mote

You can run those scripts 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

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

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

===== Step 1. API exploration =====
=====
Load the API definition of the IP mote
done.
=====
List all the defined command IDs:
[1, 2, 6, 7, 8, 9, 12, 16, 17, 18, 21, 22, 23, 24, 36, 40]
=====
List all the defined command names:
['setParameter', 'getParameter', 'join', 'disconnect', 'reset', 'lowPowerSleep',
 'testRadioRx', 'clearNV', 'requestService', 'getServiceInfo', 'openSocket', 'closeSocket', 'bindSocket', 'sendTo', 'search', 'testRadioTxExt']
=====
Get the command name of command ID 2:
getParameter
=====
Get the command ID of command name 'getParameter':
2
=====
List the subcommand of command 'getParameter':
['macAddress', 'networkId', 'txPower', 'joinDutyCycle', 'eventMask', 'moteInfo',
 'netInfo', 'moteStatus', 'time', 'charge', 'testRadioRxStats', 'OTAPLockout', 'moteId', 'ipv6Address', 'routingMode', 'appInfo', 'powerSrcInfo', 'powerCostInfo']
```

```
' , 'mobilityType', 'advKey', 'sizeInfo', 'autoJoin']
=====
Get a description of the getParameter.moteStatus command:
The getParameter<moteStatus> command is used to retrieve current mote state and
ther dynamic information.
=====
List the name of the fields in the getParameter.moteStatus request:
[]
=====
List the name of the fields in the getParameter.moteStatus response:
['state', 'reserved_0', 'reserved_1', 'numParents', 'alarms', 'reserved_2']
=====
Print the format of the getParameter.moteStatus 'state' response field:
int
=====
Print the length of the getParameter.moteStatus 'state' response field:
1
=====
Print the valid options of the getParameter.moteStatus 'state' response field:
[0, 1, 2, 3, 4, 5, 6, 7, 8]
=====
Print the description of each valid options of the getParameter.moteStatus 'stat
e' response field:
['init', 'idle', 'searching', 'negotiating', 'connected', 'operational', 'discon
nected', 'radiotest', 'promiscuous listen']

===== Step 2. Connecting to a device =====
Do you want to connect to a device? [y/n] y
Enter the serial port of the IP mote's API (e.g. COM30) COM6
=====
Creating connector
done.
=====
Connecting to IP mote
done.

===== Step 3. Getting information from the device =====
=====
Retrieve the moteStatus, through 'raw' API access:
{'numParents': 0, 'reserved_1': 0, 'reserved_0': 0, 'reserved_2': 0, 'state': 1,
  'RC': 0, 'alarms': 0}
=====
Retrieve the moteStatus, through function-based API access:
Tuple_dn_getParameter_moteStatus(RC=0, state=1, reserved_0=0, reserved_1=0, numP
arents=0, alarms=0, reserved_2=0)

===== Step 4. Disconnecting from the device =====
=====
Disconnecting from IP mote
done.
Script ended. Press Enter to exit.
```

7.5.16 TempMonitor

Introduction

TempMonitor connects to either a SmartMesh IP Manager or SmartMesh WirelessHART Manager, allowing you to send commands to 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

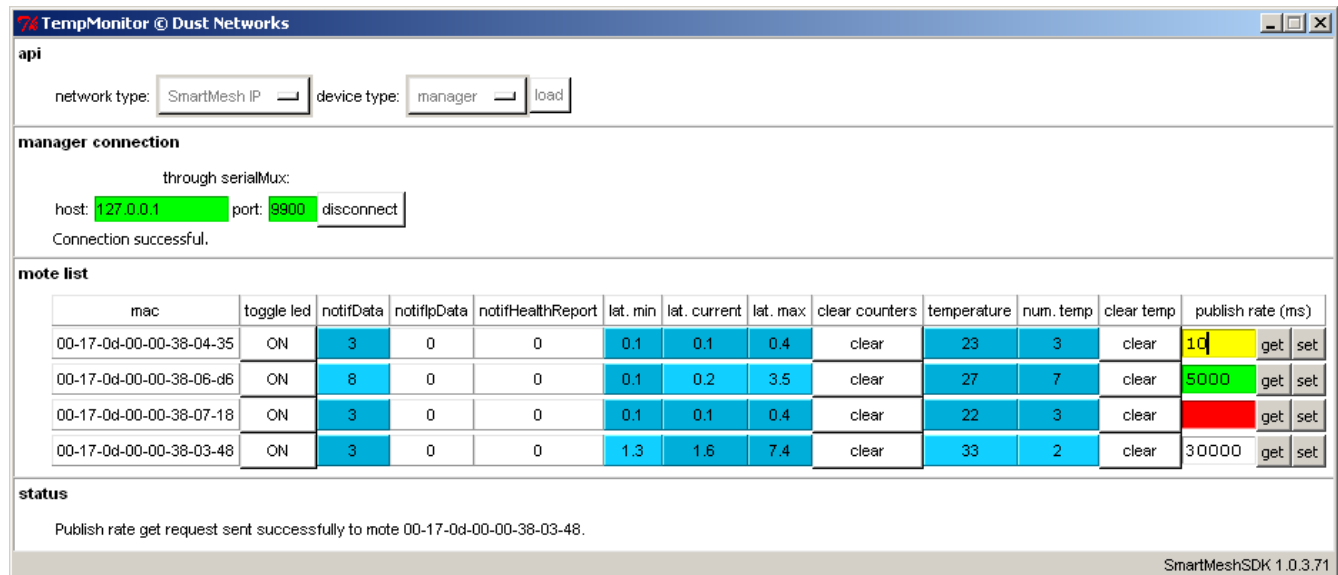
TempMonitor Connects to the Manager and interacts with motes running in **master** mode.

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.



The screenshot shows the TempMonitor application window with the following sections:

- api**: network type: SmartMesh IP, device type: manager, load
- manager connection**: through serialMux: host: 127.0.0.1, port: 9900, disconnect. Connection successful.
- mote list**: A table with columns: mac, toggle led, notifData, notifIpData, notifHealthReport, lat. min, lat. current, lat. max, clear counters, temperature, num. temp, clear temp, publish rate (ms), get, set.
- status**: Publish rate get request sent successfully to mote 00-17-0d-00-00-38-03-48.

mac	toggle led	notifData	notifIpData	notifHealthReport	lat. min	lat. current	lat. max	clear counters	temperature	num. temp	clear temp	publish rate (ms)	get	set
00-17-0d-00-00-38-04-35	ON	3	0	0	0.1	0.1	0.4	clear	23	3	clear	10	get	set
00-17-0d-00-00-38-06-d6	ON	8	0	0	0.1	0.2	3.5	clear	27	7	clear	5000	get	set
00-17-0d-00-00-38-07-18	ON	3	0	0	0.1	0.1	0.4	clear	22	3	clear		get	set
00-17-0d-00-00-38-03-48	ON	3	0	0	1.3	1.6	7.4	clear	33	2	clear	30000	get	set

SmartMeshSDK 1.0.3.71

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.

7.5.17 Upstream

Introduction

The Upstream application takes the SmartMesh IP Mote through the joining, service request, and socket binding state machines, and allows the user to enter dummy "sensor" data and send it to the SmartMesh IP Manager or any Internet host.

Running

You can run the Upstream application:

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

Description

Upstream © Dust Networks
_ □ ×

sensor data to send

17414

destination IPv6 address	dest. UDP port	
ff020000000000000000000000000002	<input style="width: 50px;" type="text" value="61000"/>	send to manager
20010470006600170000000000000002	<input style="width: 50px;" type="text" value="61000"/>	send to host

Sent successfully

join state machine

READYTOSEND	active	ready to send	289.791s
BINDSOCKET	done	bind the socket	0.016s
OPENSOCKET	done	open a socket	0.015s
SERVICEGRANTED	done	service granted	0.000s
REQUESTINGSERVICE	done	requesting service	61.561s
JOINED	done	joined	0.000s
OPERATIONAL	done	mote is operational	63.561s
JOINREQUESTSENT	done	join request sent	7.795s
SEARCHING	done	searching for a network	14.605s
CONFIGURED	done	configured	0.000s
CONFIGURING_DC	done	configuring the join duty cycle	0.047s
CONFIGURE	done	start configuring the mote	0.000s
ASSESSMOTESTATE	done	evaluate what the current mote state is	0.016s
WAITFORINITIALNOTIF	done	wait for initial notifications	1.921s

mote connection

through serial port:

port name:

COM25
disconnect

Connection successful.

tip

Remember to reset your mote before starting this application.


Note: The "send to manager" and "send to host" button become active only once the mote has reached the READYTOSEND state.

SmartMeshSDK 1.0.3.71

Upstream consists of three frames:

- the **mote connection** frame allows you to connect the application to a SmartMesh IP Mote.
- the **join state machine** frame shows the current state of the mote as it joins (states described below). The right-most column shows the time spent in each state.
- the **sensor data to send** frame allows the user to choose a 16-bit value (mimicking a sensor), and to send that data to either the manager, or a host on the Internet.

Joining states

 For the SmartMesh IP Mote to go to all following states, you need to reset it before connecting this Upstream application. You can do so by

- sending a `reset` command using the APIExplorer application, or
- using the `reset` CLI command, or
- powering the SmartMesh IP Mote off, then on.

name of the state	description
WAITFORINITIALNOTIF	Wait for possible notifications, such as a <code>BOOT</code> notification is the SmartMesh IP Mote was just reset. This state is left after a given timeout, regardless of whether a notification was received.
ASSESSMOTESTATE	Evaluate the current mote's state: <ul style="list-style-type: none"> • retrieve the mote's status; if it's operational, jump to the state <code>READYTOSEND</code> • retrieve the services already in place in this mote; jump to the state <code>READYTOSEND</code> of the services are already in place. • otherwise, proceed to the <code>CONFIGURE</code> step.
CONFIGURE	A transition state.
CONFIGURING_DC	Configure the join duty cycle to 100%.
CONFIGURED	A transition state. When reaching this state, the SmartMesh IP Mote is configured and ready to join the network.
SEARCHING	Issue a <code>join</code> command. Wait to receive an advertisement from the network (a <code>JOINSTART</code> event notification).
JOINREQUESTSENT	Wait for the SmartMesh IP Mote to become operational (an <code>OPERATIONAL</code> event notification).

OPERATIONAL	Wait for the base bandwidth to be in place (a SVC_CHG event notification).
JOINED	A transition state. When reaching this state, the SmartMesh IP Mote is part of the network.
REQUESTINGSERVICE	Request service to the manager. Wait to receive an indication that the service was installed (a SVC_CHG event notification).
SERVICEGRANTED	A transition state. When reaching this state, the SmartMesh IP Mote has obtained its service and can start sending data.
OPENSOCKET	Open a UDP socket.
BINDSOCKET	Bind that socket to UDP port 60000.
READYTOSEND	The final state of the joining process.

Sending sensor data

The "sensor data to send" frame is only active when the mote has reached the `READYTOSEND` state. Once that state is reached, you can set a sensor value, and decide to send:

- to the manager by pressing the **sendto manager** button.
- to a host on the Internet by specifying its IPv6 address, and pressing the **send to host** button.

✔ Sending to the manager simply means sending data to the manager's well-known IPv6 address (`ff02::2`, or `ff020000000000000000000000000002` when written out).

⚠ This application is meant to be used in conjunction with the `SensorDataReceiver` application connected to the Manager.

⚠ For the data to be able to be sent into the Internet, the manager needs to be connected the [Low-Power Border Router](#) using the [LBRConnection](#) application.

7.5.18 Xively

Introduction

Publishes sensor data to Xively and subscribes to changes.

Xively (<https://xively.com/>) is a cloud-based service on which you can publish sensor data. Xively offers a number of libraries so you can easily develop applications to visualize and interact with the data stored on Xively.

The Xively sample application inside the SmartMesh SDK:

- Sends the temperature data generated by each SmartMesh IP Motes running the default firmware in **master** mode. This allows you to see an timeline of your data at <https://xively.com/>, or on a Xively-enabled application you develop.
- Subscribes to the datastreams corresponding to the LED of each SmartMesh IP Motes running the default firmware in **master** mode. This allows you to actuate the LED on an evaluation mote from <https://xively.com/>, or from a Xively-enabled application you develop.

Running

You can run the Xively application:

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

Description


The Xively sample application connects to your SmartMesh IP Manager, and subscribes for temperature data. To be able to send that information to the Xively service, it requires you to enter a Xively Master API key.

You must first create a developer account with Xively at <http://www.xively.com>.

One you have created that account, log into <http://www.xively.com> and create a Master API key:

- From the Web Tools dropdown, select "Settings".
- Under the Settings section, select Master Keys.
- Click on Add Master Key.
- Enter a title for the key, and select all boxes, i.e. READ, CREATE, UPDATE, DELETE, and Access Private Fields.

Copy that Master key.

 Make sure that you have a SmartMesh IP network running with one or more SmartMesh IP Motes running the default firmware in **master** mote.

To publish data to Xively:

- Start the Xively sample application
- Connect the Xively sample application to your SmartMesh IP Manager, either through the serialMux, or through a serial port.
- The list of motes in your network is automatically populated
- Enter the Xively Master API key you created in the steps above in the **Xively API key box**, and click **set**.
- Each time a mote publishes temperature:
 - the **data received** counter increments
 - the **published** counter increments, indicating the Xively sample application is attempting to publish the data to Xively.
 - the **published OK** counter increments, indicating that publication was successful.

Xively Publisher © Dust Networks

manager connection

through serialMux:
 host: 127.0.0.1 port: 9900 disconnect
 Connection successful.

Xively API key

set

mote list

mac	data received	published	published OK	clear	see data online
00-17-0d-00-00-38-03-ca	7	7	7	clear	open browser
00-17-0d-00-00-38-03-87	7	7	7	clear	open browser
00-17-0d-00-00-38-03-d9	7	7	6	clear	open browser
00-17-0d-00-00-38-06-d6	7	7	6	clear	open browser
00-17-0d-00-00-38-04-25	7	7	7	clear	open browser
00-17-0d-00-00-38-06-ad	7	7	7	clear	open browser
00-17-0d-00-00-38-03-dd	7	7	6	clear	open browser
00-17-0d-00-00-38-04-35	7	7	7	clear	open browser
00-17-0d-00-00-38-07-18	6	6	6	clear	open browser
00-17-0d-00-00-38-03-69	7	7	6	clear	open browser
00-17-0d-00-00-38-07-0c	7	7	7	clear	open browser
00-17-0d-00-00-38-06-6a	7	7	6	clear	open browser

tooltip

SmartMeshSDK 1.0.4.109

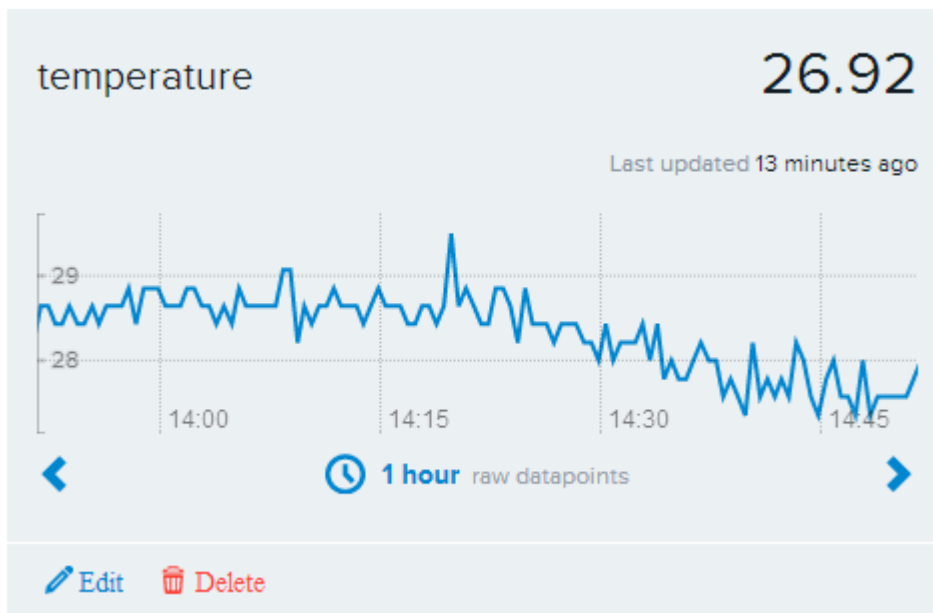
To see your data:

- Log into <http://www.xively.com>, and navigate to **Web Tools, Manage**.

- A new product was created entitled "SmartMesh IP Starter Kit".




- Clicking on that product lists the devices in your SmartMesh IP network.
- Each device has two datastreams associated:
- the "temperature" datastream holds the timeline of the temperature published.



- the "led" datastream is used for actuating the mote's LED.

To actuate the mote's LED:

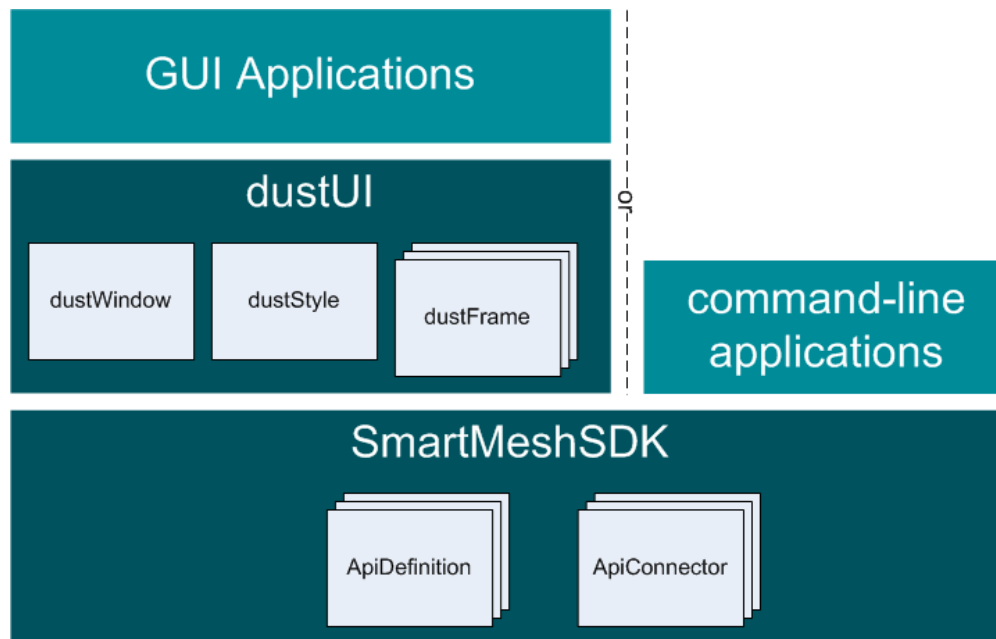
 On the mote, make sure the **LED_EN** jumper is in place.

- On a mote's device page on <http://www.xively.com>, click on the LED datastream.
- This causes the LED on that board to switch on.

- Repeat the steps above and enter **0** to switch the LED off.

7.6 Architecture

7.6.1 Overview

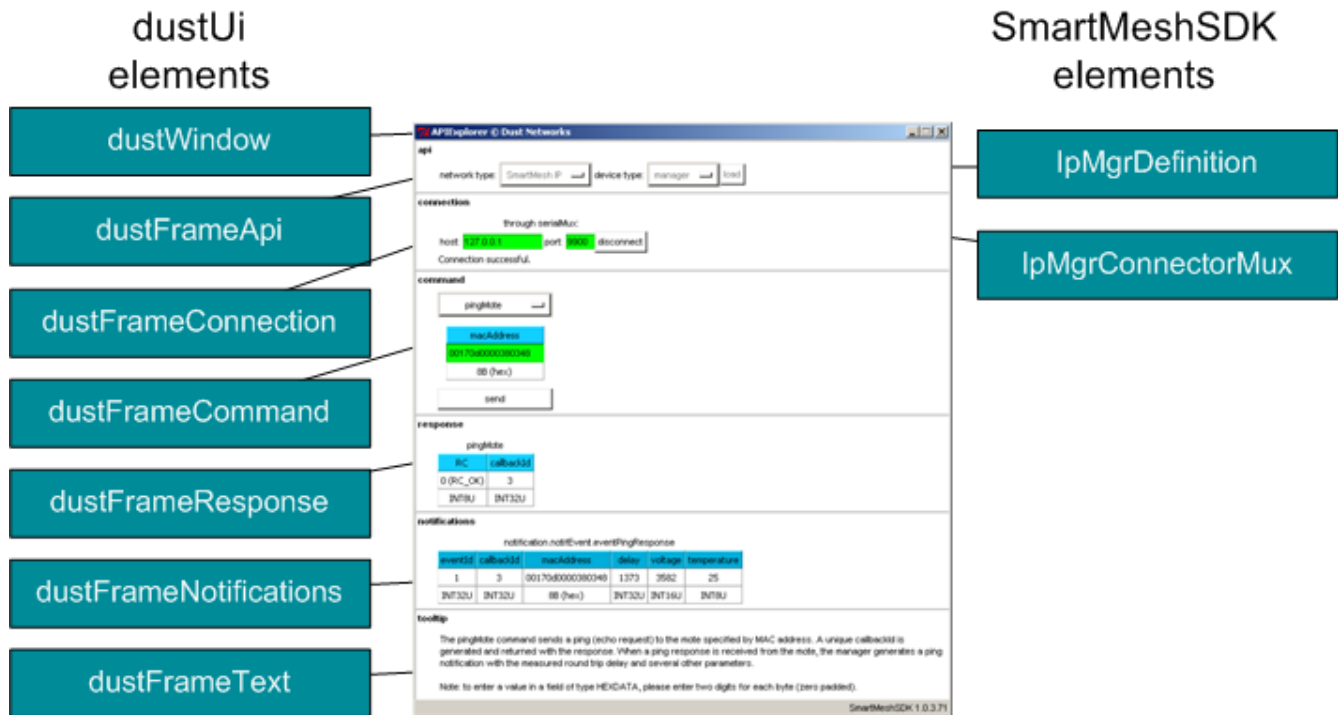


The SmartMesh SDK consists of three layers:

- 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

7.6.2 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.

7.7 dustUI Library

7.7.1 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.

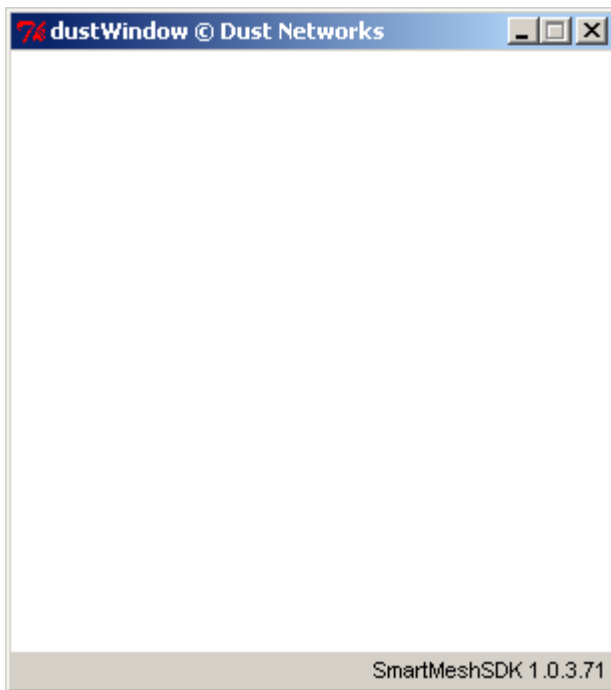
7.7.2 Module Description

- ✔ 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



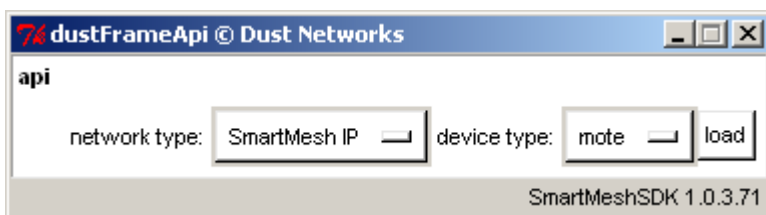
The Window of each application, which includes:

- 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

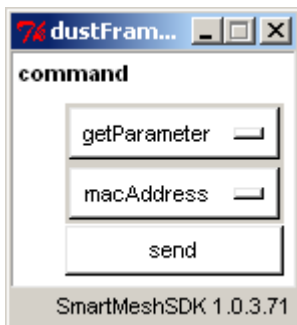
This is the parent class for all dustFrames, and is meant to be inherited.

dustFrameApi.py



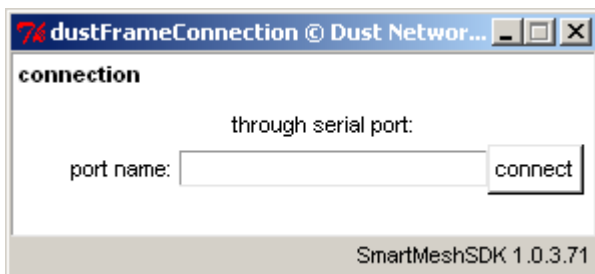
Specify which API definition to load.

dustFrameCommand.py



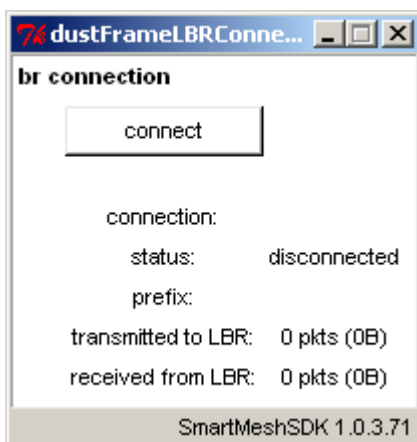
Browse the available commands and send them.

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



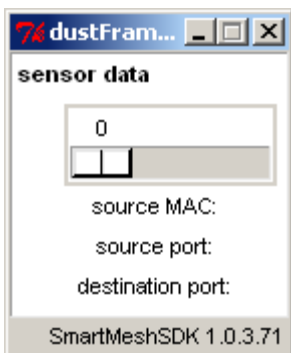
Display the fields contained in a response.

dustFrameNotifications.py



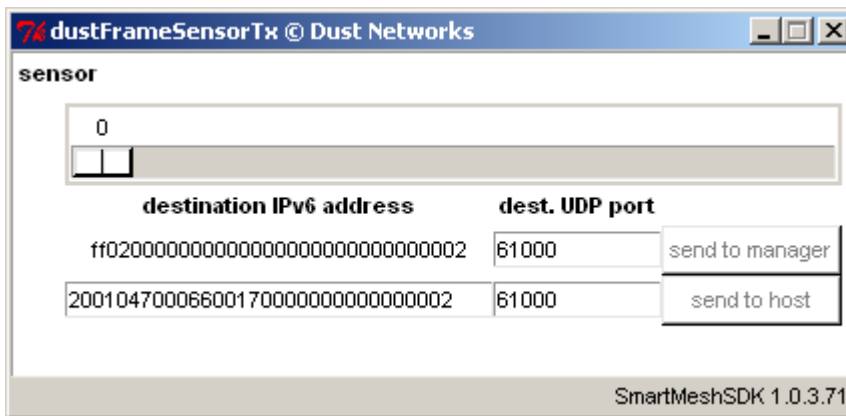
Display the fields contained in a notification.

dustFrameSensorData.py



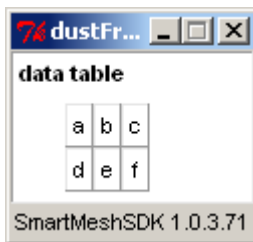
Display sensor data.

dustFrameSensorTx.py



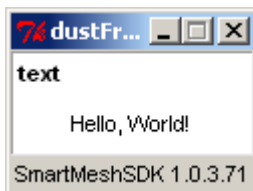
Send sensor data.

dustFrameTable.py



Display a list of lists as a 2D table.

dustFrameTooltip.py



Display arbitrary text.

7.8 SmartMeshSDK Library

7.8.1 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.

7.8.2 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
<code>IpMgrDefinition.py</code>	SmartMesh IP Manager
<code>IpMoteDefinition.py</code>	SmartMesh IP Mote
<code>HartMgrDefinition.py</code>	SmartMesh WirelessHART Manager
<code>HartMoteDefinition.py</code>	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
<code>IpMgrConnectorMux</code>	SmartMesh IP Manager	SerialMux
<code>IpMgrConnectorSerial</code>	SmartMesh IP Manager	serial
<code>IpMoteConnector</code>	SmartMesh IP Mote	serial
<code>HartMgrConnector</code>	SmartMesh WirelessHART Manager	XML-RPC
<code>HartMoteConnector</code>	SmartMesh WirelessHART Mote	serial

8 Interacting with a Network

8.1 Introduction

This section contains a collection of tutorials, each focusing on a particular aspect of a SmartMesh IP network. No prior knowledge about wireless sensor networking or Dust Networks products is required.

Basic Tutorials:

- In [A First Network](#), you will switch on your network with default settings and use the [Stargazer](#) GUI application to watch the network form.
- In [Interacting with the Manager](#), you will log into the SmartMesh IP Manager and use the APIExplorer application to extract information about the SmartMesh IP Manager and the SmartMesh IP Motes connected to it.
- In [Interacting with a Mote](#), you will use the APIExplorer application to control a SmartMesh IP Mote, mimicking the behavior of an external micro-controller. You will have that mote join a network and send data to the SmartMesh IP Manager.

Advanced Topics:

- [Exercise the API programmatically](#) shows how the SmartMeshSDK can be used from a script rather than a Graphical User Interface.
- [Log HDLC Frames](#) uses the logging capabilities of the SmartMeshSDK modules to follow the stream of bytes sent over a serial connection with a SmartMesh IP Mote.
- [Upstream Communication](#) introduces the Upstream application, which drives a SmartMesh IP Mote through the joining, service request, and socket binding state machines, and allows the user to send data to the SmartMesh IP Manager.
- [Downstream Communication](#) uses the APIExplorer application to send data from the SmartMesh IP Manager to a SmartMesh IP Mote.
- [Internet Integration](#) shows you how to connect the SmartMesh IP Manager to a [Low-power Border Router](#) so a SmartMesh IP Mote can exchange data directly with computers on the Internet.

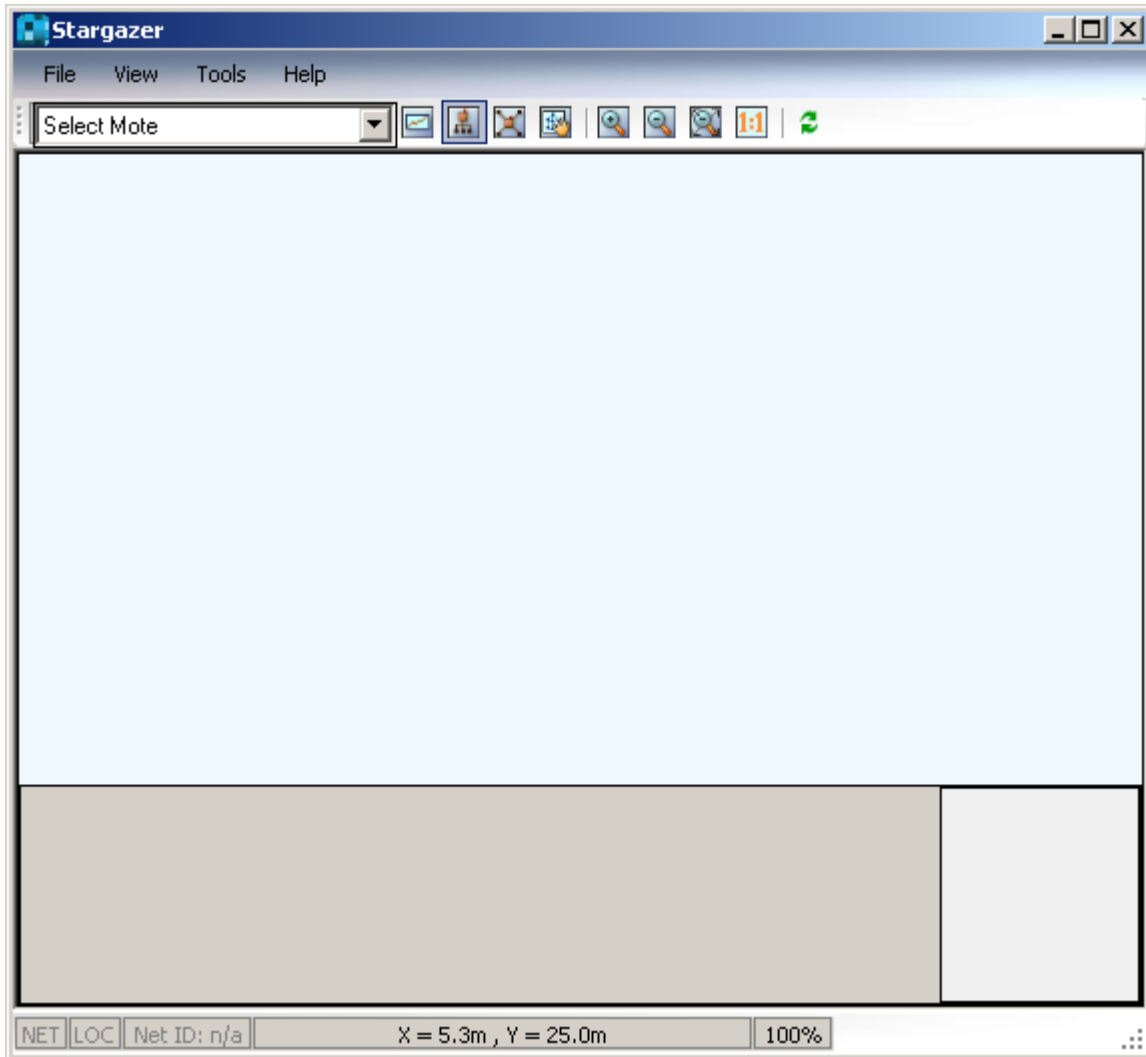
8.2 A First Network

8.2.1 Overview

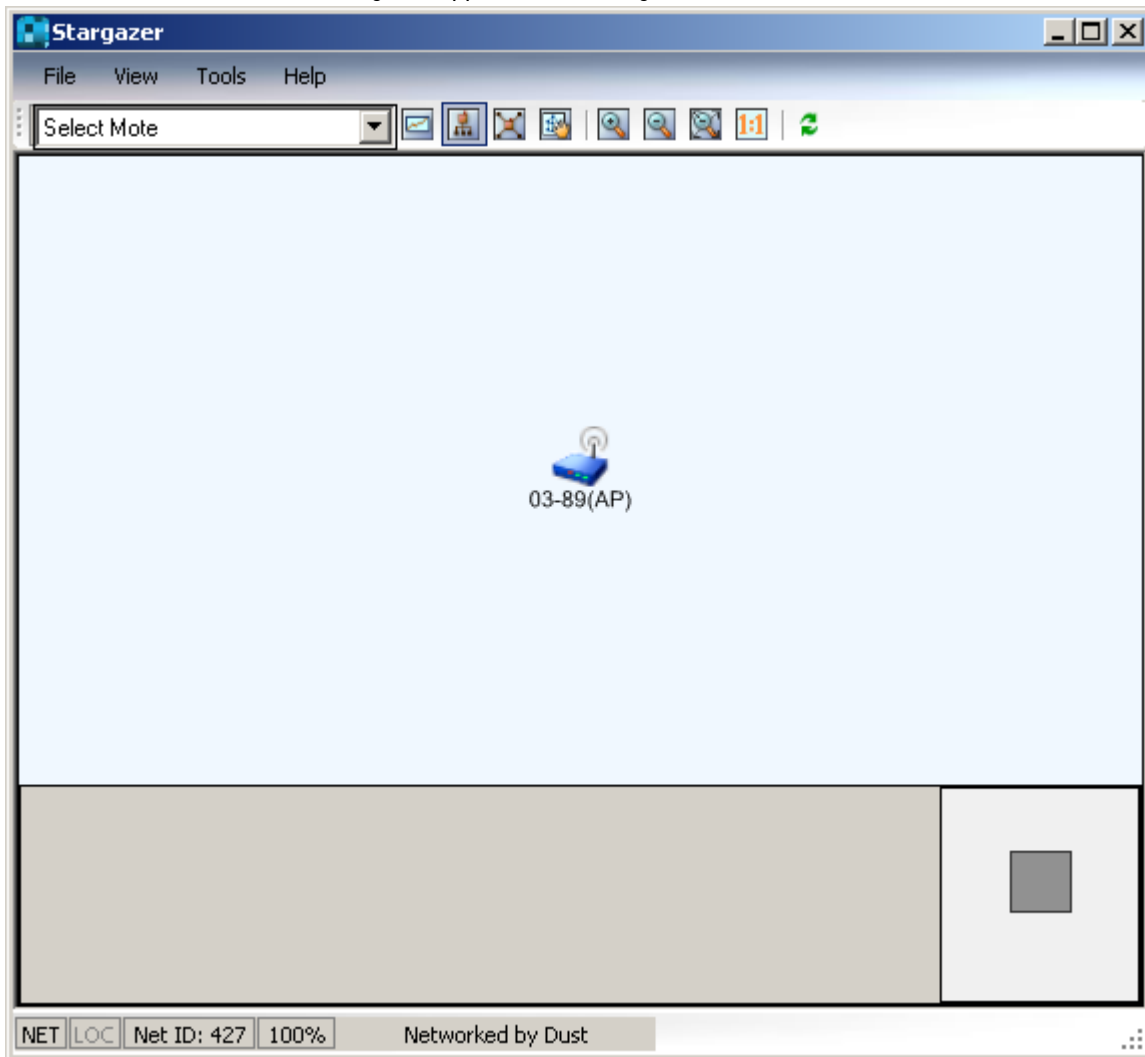
In this tutorial, you will use the [Stargazer GUI](#) and a SmartMesh IP Manager to watch a mesh network form. It assumes that you have previously followed the [installation](#) steps outlined earlier in this document.

Building the network

1. Switch off all the motes and manager.
2. Connect the SmartMesh IP Manager to your computer.
3. Start the Stargazer application. The following window opens:

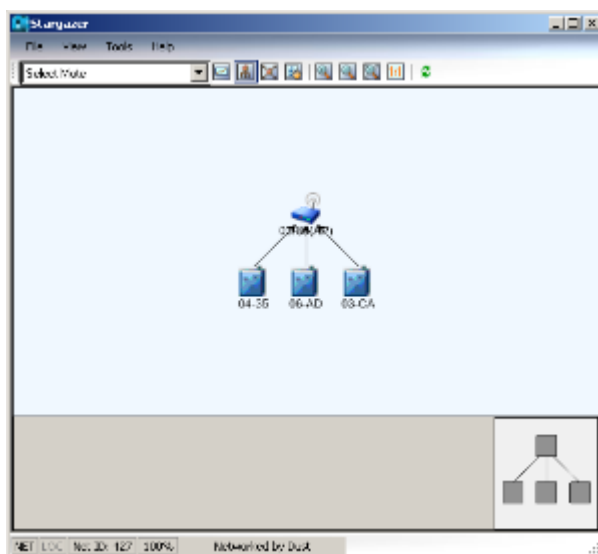


4. Switch on the SmartMesh IP Manager. It appears in the Stargazer window:

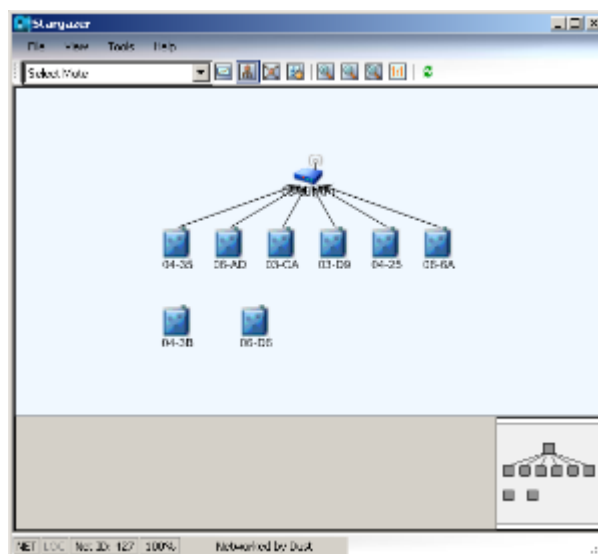


5. Switch on the SmartMesh IP Motes. The order in which you switch them on does not matter since the network self-organizes as the SmartMesh IP Motes join. You can watch the network build:

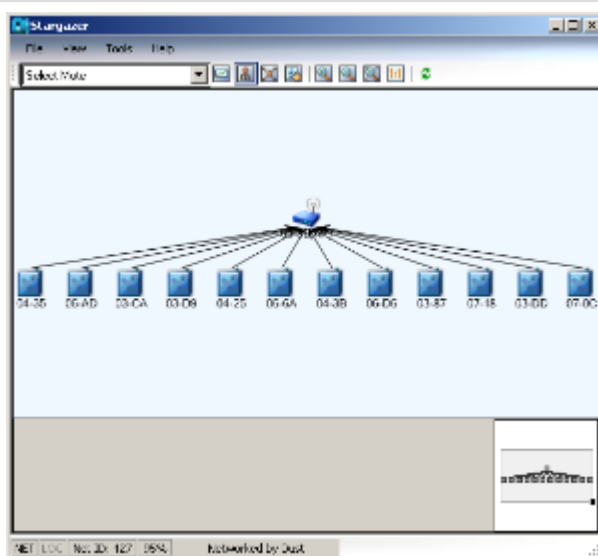
3 notes



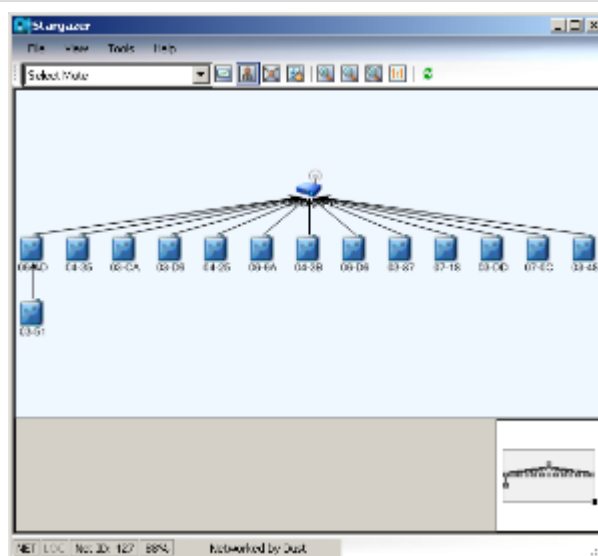
8 notes



12 notes



14 notes

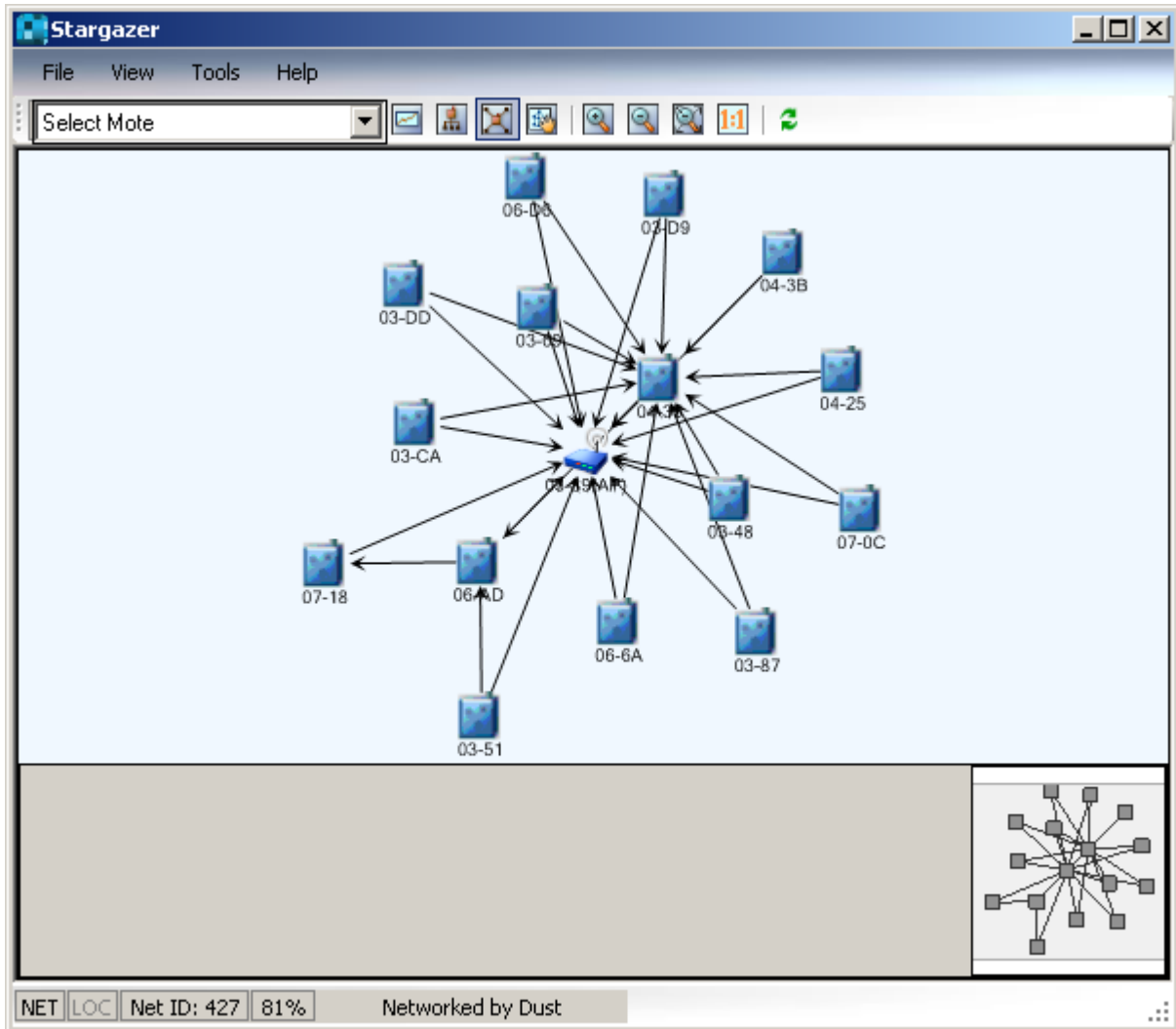


If you previously switched a mote to slave mode to use with [APIExplorer](#), [TempMonitor](#), or other SDK examples, return it to **master** mode:

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

⚠ Motes in the starter kit (DC9000) ship in **master** mode. Mote modes and how to switch between them are discussed in the [SmartMesh IP User's Guide](#). The starter kit contains 5 motes, but 32 motes can be added to the manager. If your manager has additional external RAM, it supports 100 motes.

- Once all nodes have joined, you can see how the topology transforms into a redundant mesh as the SmartMesh IP Manager applies its optimization rules:



✔ Use the icons above the display region to switch between different views:

Stargazer Radio Space View

Stargazer Tabular View

Mac Address	Name	State	Reliability (%)	Latency (ms)	Received Packets	Lost Packets	Good Neighbors
00-17-0D-00-00-38-03-89	03-89(AP)	Operational	--	--	--	--	14
00-17-0D-00-00-38-04-35	04-35	Operational	100	1660	6	0	1
00-17-0D-00-00-38-06-AD	06-AD	Operational	100	1210	8	0	2
00-17-0D-00-00-38-03-CA	03-CA	Operational	100	2080	7	0	1
00-17-0D-00-00-38-03-D9	03-D9	Operational	100	1470	6	0	1
00-17-0D-00-00-38-04-25	04-25	Operational	100	2200	7	0	1
00-17-0D-00-00-38-06-6A	06-6A	Operational	100	1700	7	0	1
00-17-0D-00-00-38-04-3B	04-3B	Operational	100	2100	7	0	1
00-17-0D-00-00-38-06-D6	06-D6	Operational	100	1560	7	0	1
00-17-0D-00-00-38-03-87	03-87	Operational	100	1310	7	0	1
00-17-0D-00-00-38-07-18	07-18	Operational	100	1580	7	0	1
00-17-0D-00-00-38-03-DD	03-DD	Operational	100	830	7	0	1
00-17-0D-00-00-38-07-0C	07-0C	Operational	100	1180	7	0	1
00-17-0D-00-00-38-03-51	03-51	Operational	100	1730	7	0	1
00-17-0D-00-00-38-03-48	03-48	Operational	100	970	7	0	1
00-17-0D-00-00-38-03-69	03-69	Operational	100	2160	7	0	1

See [Using Stargazer](#) for more details on the Stargazer GUI.

✔ You are now ready to evaluate the performance of a SmartMesh IP network - see the application note [How to Evaluate Network and Device Performance](#) for recommendations.

8.2.2 Common Problems

No notes appear in Stargazer

- Is the SmartMesh IP Manager switched on?
Stargazer "learns" about your network by communicating with the SmartMesh IP Manager; if it is off, Stargazer has no information to display.

- Is the device you are connected to a SmartMesh IP Manager?

To find out, connect to the devices CLI port and (settings 9600 baud, 8 data bits, No parity, 1 stop bit, no flow control), and type the following:

```
> login user
> help
help <command>
Commands:
  mtrace
  mset
  mget
  minfo
  mstats
  mfs
  mstacks
  mlinks
  mnbrs
  mlog
  delete
  log
  login
  logout
  exec
  ping
  reset
  set
  show
  showi
  sm
  su
  trace
>
```

If you get an output not similar to the above, the device is not an SmartMesh IP Manager; Stargazer will not display any information.

- Is the port the SerialMux is listening to the API port of SmartMesh IP Manager?

If not, the SerialMux can not communicate with the manager.

Follow the steps in the [Serial Mux Configuration](#) guide to verify and change the configuration.

A mote is missing from Stargazer

- Is the SmartMesh IP Mote switched on?
- Is the SmartMesh IP 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 SmartMesh IP Mote configured to the same Network ID as the SmartMesh IP Manager? See the [Troubleshooting](#) section of this guide for instructions on verifying/changing the network ID.

8.3 Interacting with the Manager

8.3.1 Introduction

In this step, you will interact with the SmartMesh IP Manager (DC9003A-A+ DC9006) over CLI using a [Terminal Client](#), and via API, using the [APIExplorer](#) application.

Interacting with CLI

The SmartMesh IP Manager has two serial ports:

- the Command Line Interface (CLI) port to interact directly via a serial terminal software
 - the Application Programming Interface (API) port to interact using the SmartMeshSDK.
1. Connect your serial terminal client to the CLI port (3rd of 4) of the SmartMesh IP Manager (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:
  mlog
  login
  logout
```

3. Login into the manager to get access to more commands.

```
> login user
> help
help <command>
Commands:
  mtrace
  mset
  mget
  minfo
  mlog
  mfs
  mseti
  mgeti
  mshow
  mxtal
  delete
  log
  login
  logout
  exec
  ping
  radiotest
  onechan
  reset
  set
  show
  showi
  sm
  su
  trace
>
```

4. Use the `sm` commands (short for "show motes") to obtain the list of connected motes. At this point only the internal Access Point (MoteId 1) will be connected.

```
> sm
      MAC                MoteId  State Nbrs Links Joins   Age StateTime
00-17-0D-00-00-38-06-6A    1    Oper    0   12    1     0  0-00:00:37
Number of motes (max 33): Total 1, Live 1, Joining 0
>
```

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

```

> show mote 1
Mote #1, mac: 00-17-0D-00-00-38-06-6A
  State: Oper, Hops: 0.0, Uptime: 0-00:03:22, Age: 0
  Power. Route/TplgRoute.
  Power Cost: Max 65535, FullTx 0, FullRx 0
  Number of neighbors (parents, descendants): 0 (0, 0)
  Number of links      : 12
    Compressed         : 11
    Upstream (tx/rx)   : 0 (0/0)
    Downstream(tx/rx) : 1 (1/0)
  Neighbors:
>

```

6. Use the `show stat` command to get statistics about your network.

```

> show stat
Manager Statistics -----
  established connections: 1
  dropped connections      : 0
  transmit OK              : 0
  transmit error           : 0
  transmit repeat         : 0
  receive OK               : 0
  receive error            : 0
  acknowledge delay avrg  : 0 msec
  acknowledge delay max   : 0 msec
Network Statistics -----
  reliability: 0% (Arrived/Lost: 0/0)
  stability: 0% (Transmit/Fails: 0/0)
  latency: 0 msec
Motes Statistics -----
  Mote   Received  Lost  Reliability Latency Hops
>

```

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

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

```

> logout

```


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

- [SmartMesh IP User's Guide](#)
- [SmartMesh IP Manager CLI Guide](#)

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.

The [SerialMux](#) that you installed in [setup](#) is a Windows service. It runs in the background and listens to the API port of your SmartMesh IP Manager. It allows multiple applications to connect to it over a TCP socket, thereby sharing the SmartMesh IP Mote's API port.

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

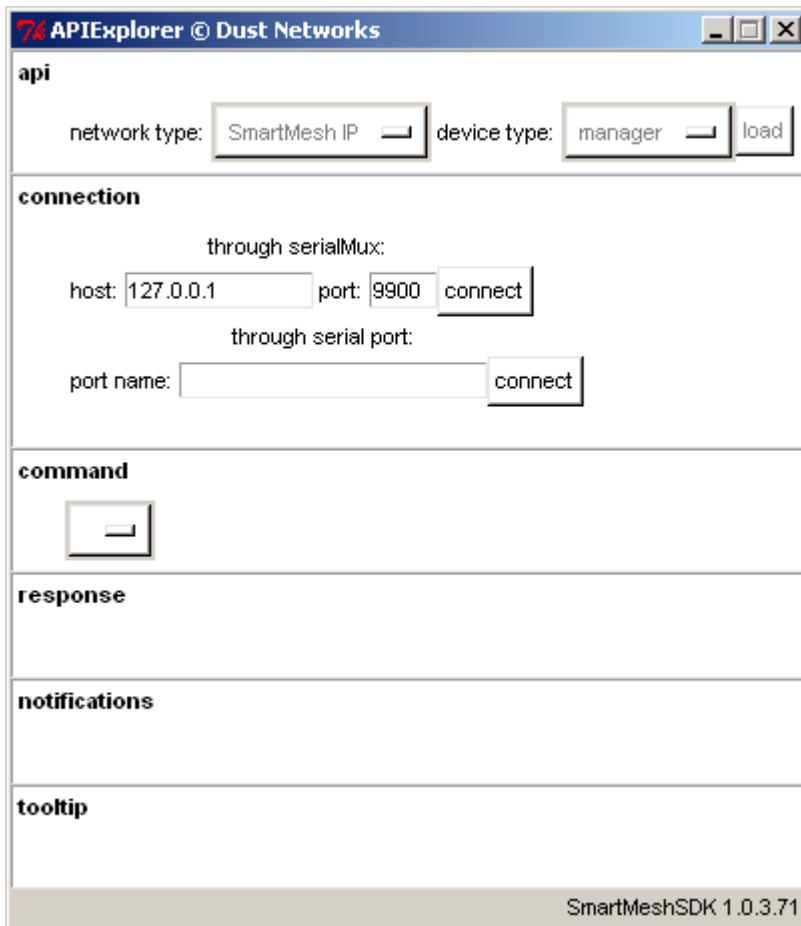
Connect to the Manager

1. Make sure your SmartMesh IP Manager is connected to your computer.
2. In the `SmartMeshSDK` directory, double click on the `win/APIExplorer.exe` program. This opens the APIExplorer's window.



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

4. Click the **load** button.



The screenshot shows a window titled "API Explorer © Dust Networks". The interface is divided into several sections:

- api**: Contains "network type:" with a dropdown menu set to "SmartMesh IP", "device type:" with a dropdown menu set to "manager", and a "load" button.
- connection**: Contains two options:
 - "through serialMux:" with "host:" (127.0.0.1) and "port:" (9900) fields, and a "connect" button.
 - "through serial port:" with a "port name:" field and a "connect" button.
- command**: Contains a single button.
- response**: An empty text area.
- notifications**: An empty text area.
- tooltip**: An empty text area.

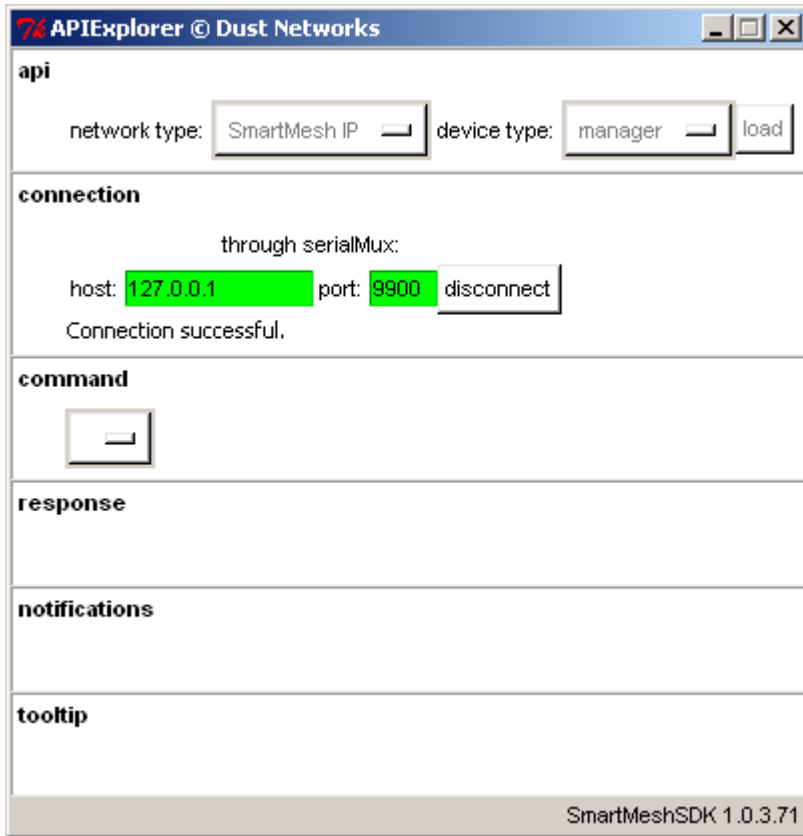
At the bottom right of the window, the version "SmartMeshSDK 1.0.3.71" is displayed.

5. In the **connection** frame, you are presented with two options to connect to the SmartMesh IP Manager; we will connect through the SerialMux.

In the SerialMux section of the **connection** frame, enter the following:

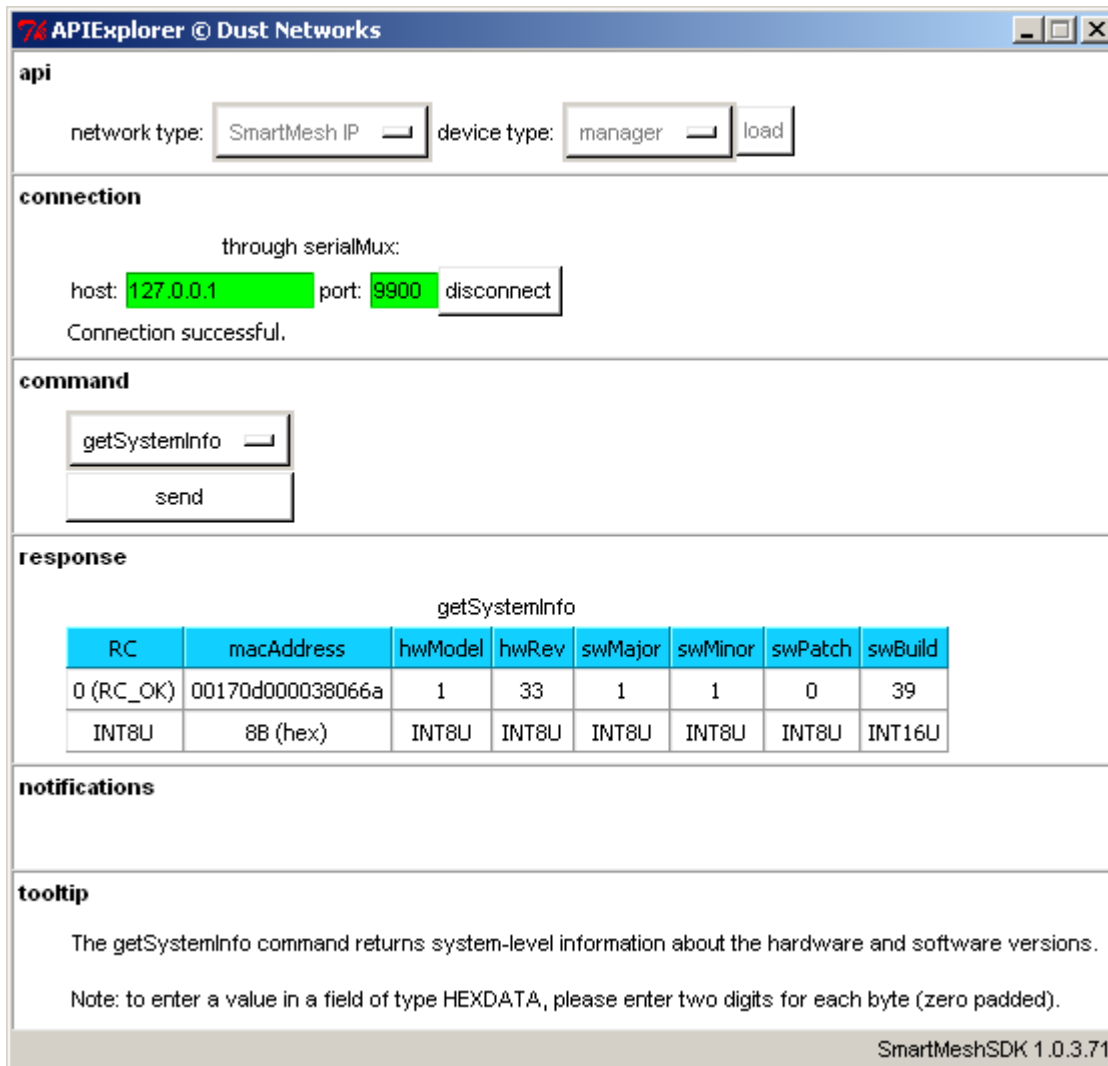
- host: 127.0.0.1
- port: 9900

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



Obtain Information about the Manager and the Network

- The drop-down menu in the **command** frame lists all the commands defined in the [SmartMesh IP Manager API Guide](#). Select *getSystemInfo*, and press **send**. The response prints in the **response** frame, and contains the name, value and format for each field.



The screenshot shows the API Explorer interface with the following configuration and results:

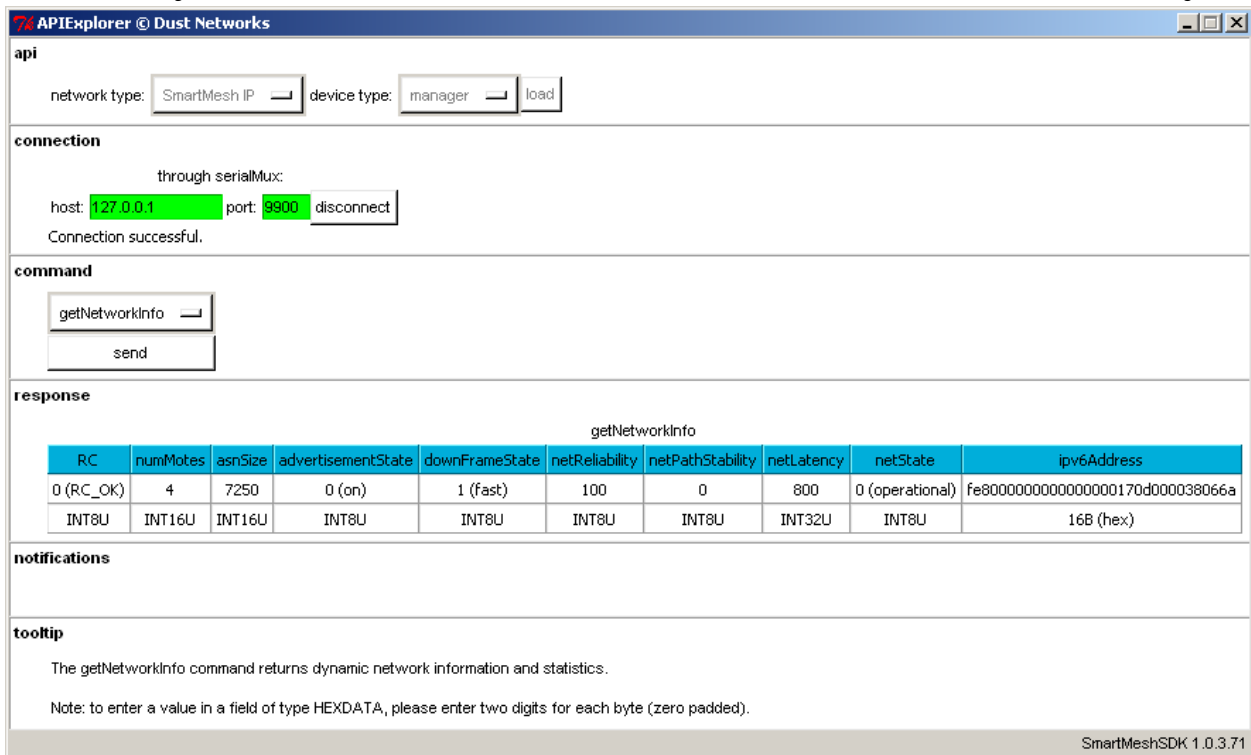
- api**: network type: SmartMesh IP, device type: manager, load
- connection**: through serialMux: host: 127.0.0.1, port: 9900, disconnect. Connection successful.
- command**: getSystemInfo, send
- response**:

getSystemInfo							
RC	macAddress	hwModel	hwRev	swMajor	swMinor	swPatch	swBuild
0 (RC_OK)	00170d000038066a	1	33	1	1	0	39
INT8U	8B (hex)	INT8U	INT8U	INT8U	INT8U	INT8U	INT16U
- notifications**: (empty)
- tooltip**: The getSystemInfo command returns system-level information about the hardware and software versions. 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 image above indicates:
 - The MAC address of the SmartMesh IP Manager is 00:17:0d:00:00:38:06:6a
 - Information about the hardware and software of the SmartMesh IP Manager

1. Similar, issue a *getNetworkInfo* command to obtain information about the mesh network connected to the manager.



The screenshot shows the API Explorer interface with the following sections:

- api**: network type: SmartMesh IP, device type: manager, load
- connection**: through serialMux: host: 127.0.0.1, port: 9900, disconnect. Connection successful.
- command**: getNetworkInfo, send
- response**:

getNetworkInfo									
RC	numMotes	asnSize	advertisementState	downFrameState	netReliability	netPathStability	netLatency	netState	ipv6Address
0 (RC_OK)	4	7250	0 (on)	1 (fast)	100	0	800	0 (operational)	fe800000000000000170d000038066a
INT8U	INT16U	INT16U	INT8U	INT8U	INT8U	INT8U	INT32U	INT8U	16B (hex)
- notifications**: (empty)
- tooltip**: The getNetworkInfo command returns dynamic network information and statistics. 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

2. The image above indicates that:
- 4 motes are connected to the manager (**numMotes** field)
 - a slot is 7.25ms long (**asnSize** field)

Consult the [SmartMesh IP Manager API Guide](#) for details about all commands and fields.

Obtain Information about a Mote

1. Select the *getMoteConfig* command and enter:

- 0000000000000000 in the **MAC** field
- True in the **next** field

1. After pressing **send**, the manager returns information about the first device in the network. This is the mote of the SmartMesh IP Manager itself, called the *Access Point*, or AP (note the field **isAP** is 1).

The screenshot shows the API Explorer interface for Dust Networks. It displays the configuration for the `getMoteConfig` command and the resulting response table.

api
network type: SmartMesh IP device type: manager load

connection
through serialMux:
host: 127.0.0.1 port: 9900 disconnect
Connection successful.

command
getMoteConfig
macAddress: 0000000000000000 next: True
8B (hex) BOOL
send

response
getMoteConfig

RC	macAddress	moteId	isAP	state	reserved	isRouting
0 (RC_OK)	00170d000038066a	1	True	4 (operational)	1	True
INT8U	8B (hex)	INT16U	BOOL	INT8U	INT8U	BOOL

notifications

tooltip

The `getMoteConfig` command returns a single mote description as the response. The command takes two arguments, a MAC Address and a flag indicating whether the MAC Address refers to the requested mote or to the next mote in managers memory. This command may be used to iterate through all motes known by the manager by starting with the `macAddress` parameter set to 0 and `next` set to true, and then using the MAC Address of that response as the input to the next call.

The mote MAC address is used in all query commands, but space constraints require the neighbor health reports to use the Mote ID for identification. Therefore, both identifiers are present in the mote structure.

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

2. Make note of the value in the **macAddress** field, which contains the MAC address of the AP, in our case 00170d000038066a.

✔ **Right-click** on any field in a table to **copy**.

Right-click in an editable text field to **paste**.

Subscribe to Notifications

In the sections above, you have sent commands to the SmartMesh IP Manager, which has answered immediately with responses. When an event happens, the SmartMesh IP Manager 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 IP Manager API Guide](#).

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

1. Select the *subscribe* command and enter:

- `FFFFFFFF` in the **filter** field
- `00000000` in the **unackFilter** field

1. After pressing **send**, you have subscribed to all available notifications.

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

API Explorer © Dust Networks

api

network type: device type:

connection

through serialMux:

host: port:

Connection successful.

command

filter	unackFilter
fffffff	00000000
4B (hex)	4B (hex)

response

subscribe

RC
0 (RC_OK)
INT8U

notifications

notification.notifData

utcSecs	utcUsecs	macAddress	srcPort	dstPort	data
1025665699	575750	00170d0000380718	61625	61625	00000500ff0105000000003d226aa30008cebe0000753001100016
8B (int)	INT32U	8B (hex)	INT16U	INT16U	hex

tooltip

The subscribe command indicates that the manager should send the external application the specified notifications. It contains two filter fields:


- filter is a bitmask of flags indicating the types of notifications that the client wants to receive
- unackFilter allows the client to select which of the notifications selected in filter should be sent acknowledged. If a notification is sent as 'acknowledged', the subsequent notification packets will be queued while waiting for response.

Each subscription request overwrites the previous one. If an application is subscribed to data and then decides he also wants events he should send a subscribe command with both the data and event flags set. To clear all subscriptions, the client should send a subscribe command with the filter set to zero. When a session is initiated between the manager and a client, the subscription filter is initialized to zero.

The subscribe bitmap uses the values of the notification type enumeration. Some values are unused to provide backwards compatibility with earlier APIs.

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

 **For more information**

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

- [SmartMesh IP User's Guide](#)
- [SmartMesh IP Manager API Guide](#)

8.3.2 Common Problems

I get no output over the CLI

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

The table below details the serial setting for all devices:

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

I get no output when connected to the manager over the SerialMux

It is possible that APIExplorer can connect to the SerialMux (the connection fields turns green), but the SerialMux cannot connect to the SmartMesh IP Manager. This may be because the SerialMux is configured to listen to a serial port which is not the API port of your SmartMesh IP Manager.

To change the configuration of your SerialMux, follow the [Serial Mux Configuration](#) guide.

8.4 Interacting with a Mote

8.4.1 Overview

In this step, you will interact with the SmartMesh IP Mote ([DC9003A-B+ DC9006](#)) over CLI using a [Terminal Client](#), and via API, using the [APIExplorer](#) application.

Interacting with the Mote CLI

The SmartMesh IP Mote has two serial ports:

- the CLI port to interact directly over a serial terminal
 - the API port to interact using the SmartMeshSDK
1. Open your serial terminal client on the CLI port of the SmartMesh IP Mote (settings 9600 baud, 8 data bits, No parity, 1 stop bits, no flow control)
 2. Type `help` to get the list of available commands

```
> help
help <command>
Commands:
  mtrace
  mset
  mget
  minfo
  mlog
  mfs
  mseti
  mgeti
  mshow
  mxtal
  set
  get
  radiotest
  trace
  reset
  loc
  info
  restore
```

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

```
Net stack v1.1.0.0
state: Search
mac: 00:17:0d:00:00:38:03:48
moteid: 0
netid: 423
blSwVer: 9
ldrSwVer: 1.0.3.12
UTC time: 1025665219:790010
reset st: 600, b96bf7dd
```

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

```
> reset
> SmartMesh IP mote, ver 1.1.0.41 (0x0)
> minfo
Net stack v1.1.0.0
state: Search
mac: 00:17:0d:00:00:38:03:48
moteid: 0
netid: 423
blSwVer: 9
ldrSwVer: 1.0.3.12
UTC time: 1025665201:513787
reset st: 100, 0
> 7084 : Joining
    8817 : Connected
    14538 : Active
> minfo
Net stack v1.1.0.0
state: Oper
mac: 00:17:0d:00:00:38:03:48
moteid: 2
netid: 423
blSwVer: 9
ldrSwVer: 1.0.3.12
UTC time: 1025666079:45928
reset st: 100, 0
```

Interacting with API using APIExplorer

SmartMesh SDK

The SmartMesh SDK is a Python package which simplifies the integration of a SmartMesh IP 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 IP Mote over a Windows serial COM port and interact with its API using the SmartMesh SDK.

Connect to the Device

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

Make sure your SmartMesh IP Mote is operating in **slavemode** by issuing the following command on the mote CLI:

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



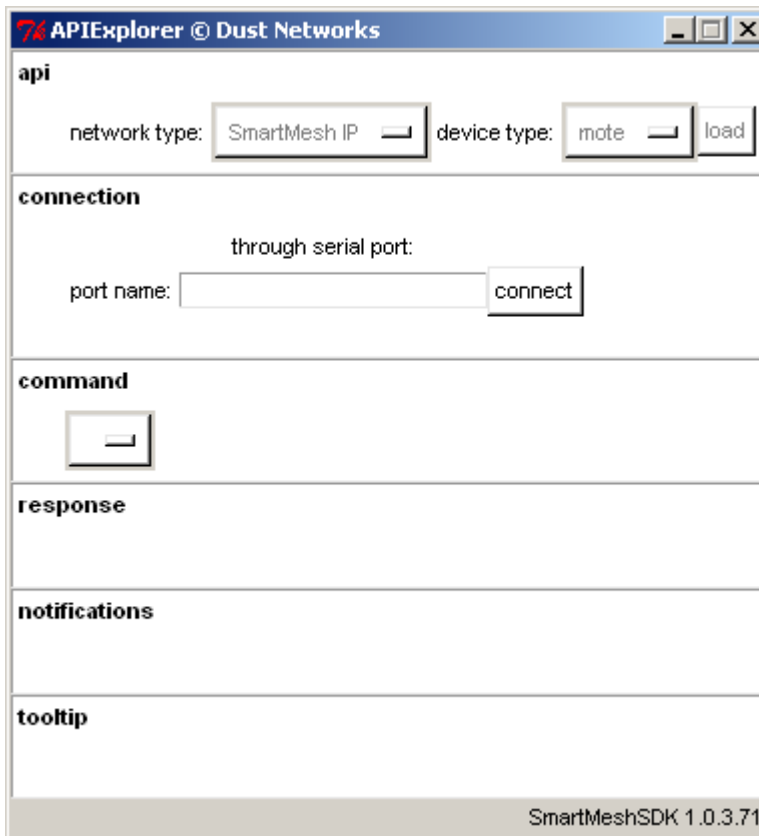
Motes in the starter kit ([DC9000](#)) ship in **master** mode. Mote modes and how to switch between them are discussed in the [Troubleshooting](#) section of this guide and also in the [SmartMesh IP User's Guide](#)

2. In the SmartMeshSDK directory, double click on the win/APIExplorer.exe application. This opens the APIExplorer's window.



3. Tell the application you want to connect to a SmartMesh IP Mote by selected the following:
 - network type: SmartMeshIP
 - device type: mote

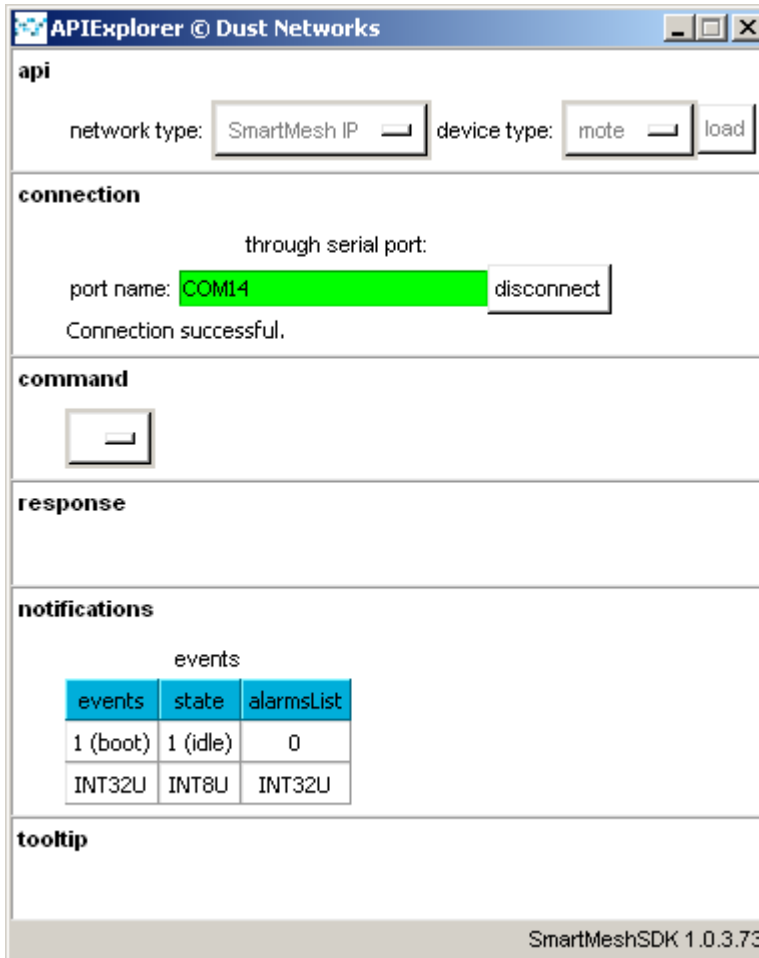
4. Click the **load** button.



5. In the **connection** frame, enter the following:

- port name: your SmartMesh IP Mote's API port number

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



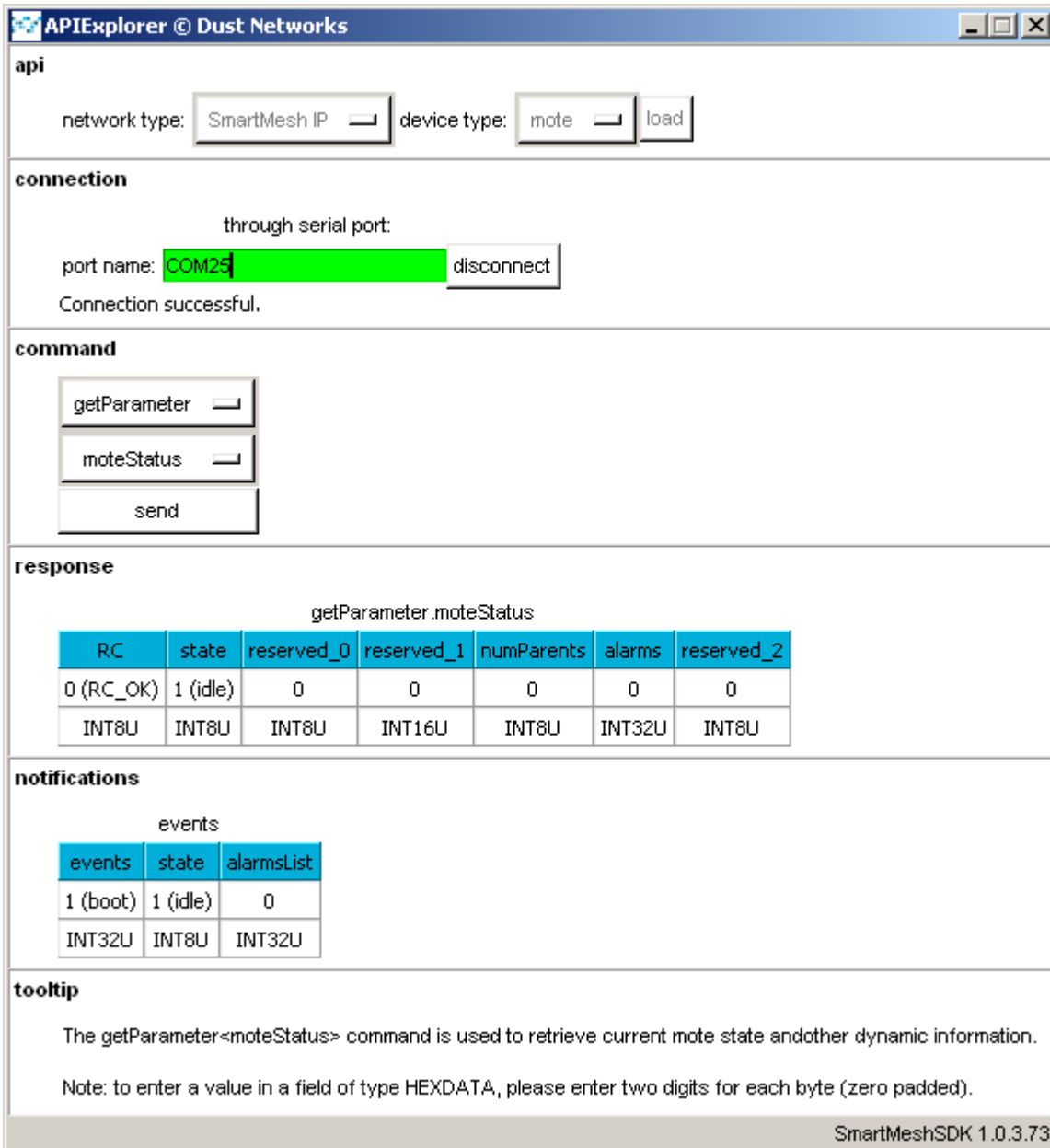
Getting Notifications from the Mote

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

When a SmartMesh IP 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 API Explorer application. This explains the boot mote event displayed in the screen shot above.

Obtain Information About the Mote

- The drop-down menu in the **command** frame lists all the commands defined in the [SmartMesh IP Mote 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.



The screenshot shows the API Explorer interface with the following sections:

- api**: network type: SmartMesh IP, device type: mote, load
- connection**: through serial port: port name: COM25, disconnect. Connection successful.
- command**: getParameter, moteStatus, send
- response**:

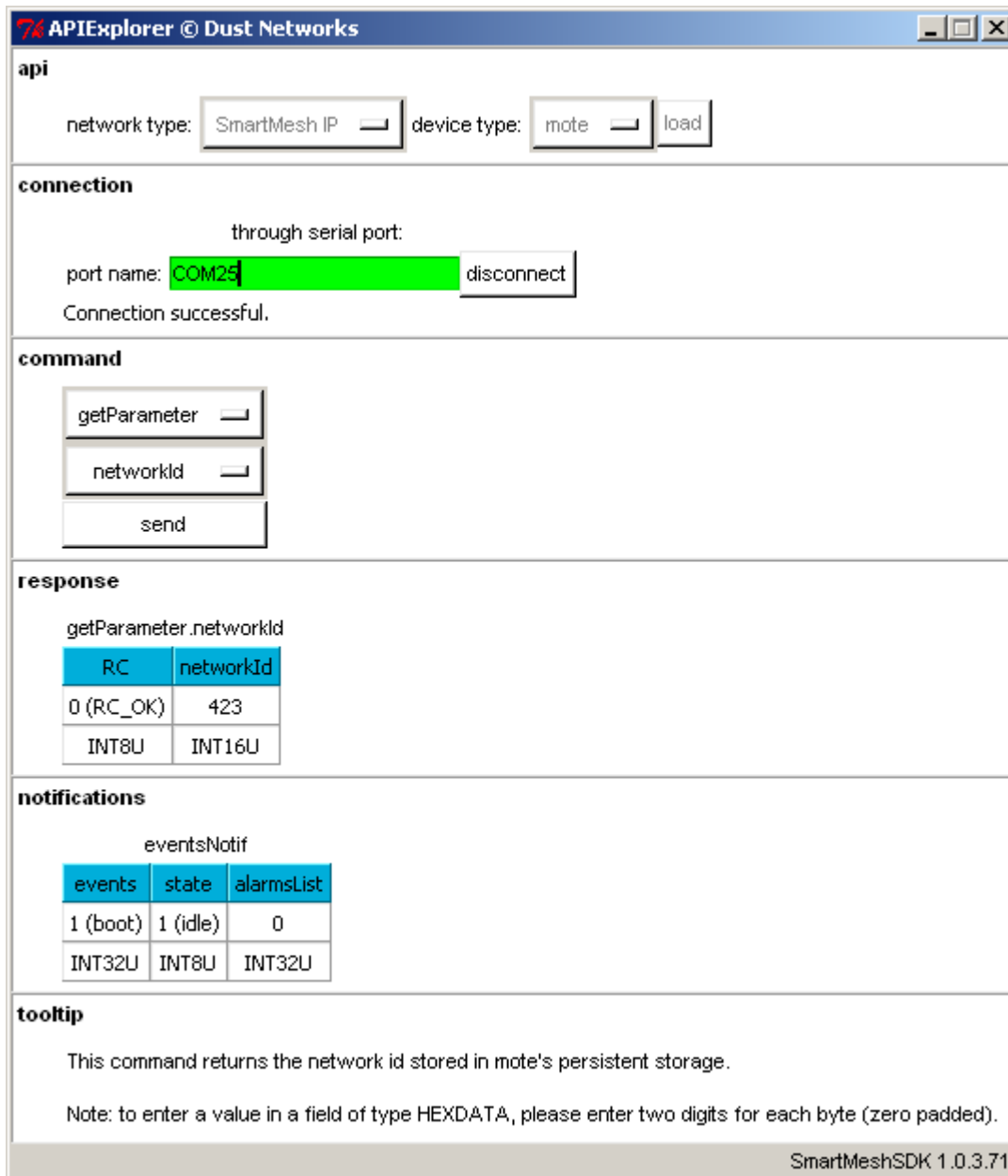
getParameter.moteStatus						
RC	state	reserved_0	reserved_1	numParents	alarms	reserved_2
0 (RC_OK)	1 (idle)	0	0	0	0	0
INT8U	INT8U	INT8U	INT16U	INT8U	INT32U	INT8U
- notifications**:

events		
events	state	alarmsList
1 (boot)	1 (idle)	0
INT32U	INT8U	INT32U
- tooltip**: The getParameter<moteStatus> command is used to retrieve current mote state and other dynamic information. Note: to enter a value in a field of type HEXDATA, please enter two digits for each byte (zero padded).

SmartMeshSDK 1.0.3.73

- The image above indicates that the mote is in `idle` state, i.e. it is not trying to join a network.

3. Similarly, issue a `getParameter.networkId` command to verify that your SmartMesh IP Mote is configured with the correct network ID.



The screenshot shows the API Explorer interface with the following sections:

- api**: network type: SmartMesh IP, device type: mote, load
- connection**: through serial port: port name: COM25, disconnect. Connection successful.
- command**: getParameter, networkId, send
- response**:

getParameter.networkId	
RC	networkId
0 (RC_OK)	423
INT8U	INT16U
- notifications**:

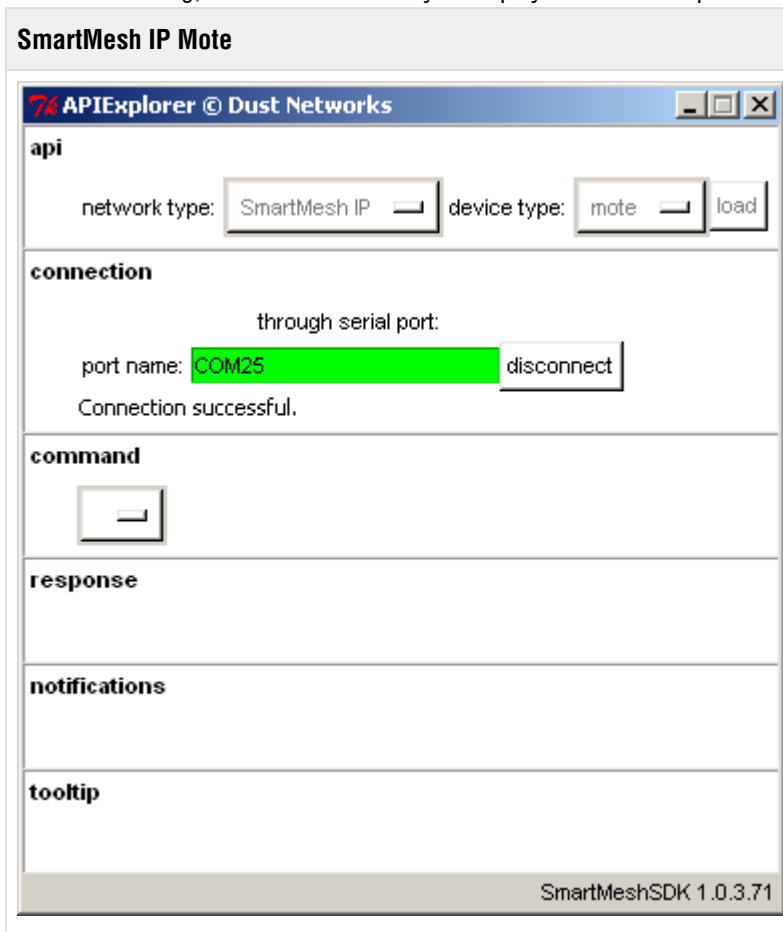
eventsNotif		
events	state	alarmsList
1 (boot)	1 (idle)	0
INT32U	INT8U	INT32U
- tooltip**: This command returns the network id stored in mote's persistent storage. Note: to enter a value in a field of type HEXDATA, please enter two digits for each byte (zero padded).

SmartMeshSDK 1.0.3.71

Have the Mote Join the Network

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

3. For easier reading, we recommend that you display the two APIExplorer windows side-by-side.



4. Reset the SmartMesh IP Mote by calling its `reset` command. After a few seconds, you should receive a `boot` event notification at the SmartMesh IP Mote.

SmartMesh IP Mote

APIExplorer © Dust Networks

api

network type: SmartMesh IP device type: mote load

connection

through serial port:

port name: COM25 disconnect

Connection successful.

command

reset

send

response

reset

RC
0 (RC_OK)
INT8U

notifications

eventsNotif

events	state	alarmsList
1 (boot)	1 (idle)	0
INT32U	INT8U	INT32U

tooltip

The reset command initiates a soft-reset of the device. The device will initiate the reset sequence shortly after sending out the response to this command. Resetting a mote directly can adversely impact its descendants; to disconnect gracefully from the network, use the disconnect command

Note: to enter a value in a field of type HEXDATA, please enter two digits for each byte (zero padded).

SmartMeshSDK 1.0.3.0

- At the SmartMesh IP Manager, *subscribe* to all notifications. If you have a network running, you may receive notifications from time to time.

SmartMesh IP Manager

APIExplorer © Dust Networks

api

network type: device type:

connection

through serialMux:

host: port:

Connection successful.

command

filter	unackFilter
ffffffff	00000000
4B (hex)	4B (hex)

response

subscribe

RC
0 (RC_OK)
INT8U

notifications

tooltip

The subscribe command indicates that the manager should send the external application the specified notifications. It contains two filter fields:

- filter is a bitmask of flags indicating the types of notifications that the client wants to receive
- unackFilter allows the client to select which of the notifications selected in filter should be sent acknowledged. If a notification is sent as 'acknowledged', the subsequent notification packets will be queued while waiting for response.

Each subscription request overwrites the previous one. If an application is subscribed to data and then decides he also wants events he should send a subscribe command with both the data and event flags set. To clear all subscriptions, the client should send a subscribe command with the filter set to zero. When a session is initiated between the manager and a client, the subscription filter is initialized to zero.

The subscribe bitmap uses the values of the notification type enumeration. Some values are unused to provide backwards compatibility with earlier APIs.

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

- At the SmartMesh IP Mote, call the *join* command. This causes it to join the network, which will trigger the following notifications on both the SmartMesh IP Mote and the SmartMesh IP Manager:

at the SmartMesh IP Manager		at the SmartMesh IP Mote	
notification	explanation	notification	explanation
		joinStart	heard an advertisement and sent a join request to the SmartMesh IP Manager
eventsMoteJoin	received a join request		
eventPathCreate	created a path for the new SmartMesh IP Mote		
		operational	the SmartMesh IP Mote is part of the network.
eventMoteOper	the path was installed correctly, the SmartMesh IP Mote is considered operational		
		scvChange	the SmartMesh IP Mote has received its base bandwidth

7. After the SmartMesh IP Mote has joined:

SmartMesh IP Manager

API Explorer @ Dust Networks

api

network type: device type:

connection

through serialMux:

host: port:

Connection successful.

command

filter	unackFilter
fffffff	00000000
4B (hex)	4B (hex)

response

subscribe

RC

0 (RC_OK)

INT8U

notifications

notification.notifEvent.eventMoteOperational

eventId	macAddress
5	00170d0000380348
INT32U	8B (hex)

tooltip

The subscribe command indicates that the manager should send the external application the specified notifications. It contains two filter fields:

- filter is a bitmask of flags indicating the types of notifications that the client wants to receive
- unackFilter allows the client to select which of the notifications selected in filter should be sent acknowledged. If a notification is sent as 'acknowledged', the subsequent notification packets will be queued while waiting for response.

Each subscription request overwrites the previous one. If an application is subscribed to data and then decides he also wants events he should send a subscribe command with both the data and event flags set. To clear all subscriptions, the client should send a subscribe command with the filter set to zero. When a session is initiated between the manager and client, the subscription filter is initialized to zero.

The subscribe bitmap uses the values of the notification type enumeration. Some values are unused to provide backwards compatibility with earlier APIs.

Note: to enter a value in a field of type HEXDATA, please enter two digits for each byte (zero padded).

Taskbar: Chrome, Firefox, Edge, VS Code, apiexplorer_ipjoin_don..., C:\Documents and Set..., apiexplorer_ipjoin_sub..., C:\

SmartMesh IP Mote

API Explorer © Dust Networks

api

network type: device type:

connection

through serial port:

port name:

Connection successful.

command

response

join

RC
0 (RC_OK)
INT8U

notifications

eventsNotif

events	state	alarmsList
128 (svcChange)	5 (operational)	0
INT32U	INT8U	INT32U

tooltip

The join command requests that mote start searching for the network and attempt to join. The mote must be in the IDLE state for this command to be valid. Note that the join time will be affected by the maximum current setting.

Note: to enter a value in a field of type HEXDATA, please enter two digits for each byte (zero padded).

SmartMeshSDK 1.0.3.

Have the Mote Request a Service



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

1. At the SmartMesh IP Mote, issue a *requestService* command to request bandwidth to send one packet every 10s to the SmartMesh IP Manager:
 - **destAddr** to 65534 (the well-known address of the SmartMesh IP Manager)
 - **serviceType** is *bandwidth*
 - **value** is 10000 (the period between transmissions, in milliseconds)

- After pressing **send**, the SmartMesh IP Manager receives the request, installs the request bandwidth and signals the requesting SmartMesh IP Mote that the bandwidth has been installed. This results in a `svcChange` event notification at the SmartMesh IP Mote.

API Explorer @ Dust Networks

api

network type: device type:

connection

through serial port:

port name:

Connection successful.

command

destAddr	serviceType	value
65534	bandwidth	10000
INT16U	INT8U	INT32U

response

requestService

RC
0 (RC_OK)
INT8U

notifications

eventsNotif

events	state	alarmsList
128 (svcChange)	5 (operational)	0
INT32U	INT8U	INT32U

tooltip

The `requestService` command may be used to request a new or changed service level to a destination device in the mesh. This command may only be used to update the service to a device with an existing connection (session).

Whenever a change in bandwidth assignment occurs, the application receives a `serviceChanged` event that it can use as a trigger to read the new service allocation.

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

You can verify the allocated bandwidth by using the `getMoteInfo` command at the SmartMesh IP Manager and the `getServiceInfo` command at the SmartMesh IP Mote.

SmartMesh IP Mote

API Explorer © Dust Networks

api

network type: device type:

connection

through serial port:

port name:

Connection successful.

command

destAddr	type
65534	bandwidth <input type="text"/>
INT16U	INT8U

response

getServiceInfo

RC	destAddr	type	state	value
0 (RC_OK)	fffe	0 (bandwidth)	0 (completed)	5850
	INT8U	2B (hex)	INT8U	INT32U

notifications

eventsNotif

events	state	alarmsList
128 (svcChange)	5 (operational)	0
INT32U	INT8U	INT32U

tooltip

The getServiceInfo command returns information about the service currently allocated to the mote.

Note: to enter a value in a field of type HEXDATA, please enter two digits for each byte (zero padded).

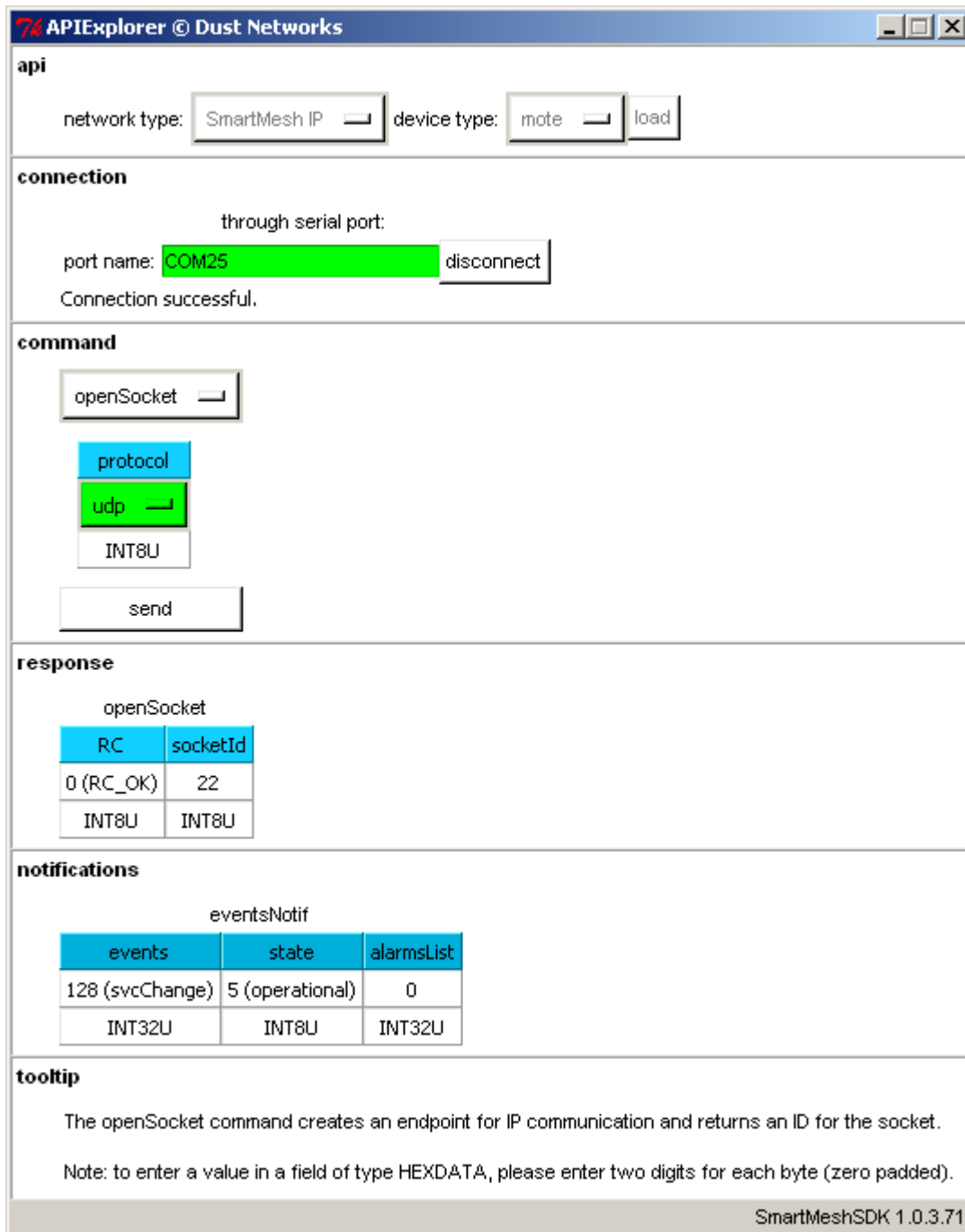
SmartMeshSDK 1.0.3.71

The bandwidth allocated is expressed in time between transmits, *i.e.* a small number indicates a larger bandwidth. The SmartMesh IP Manager uses a safety margin when allocating bandwidth, so the period really allocated is smaller than the one requested.

Have the Mote Prepare a UDP Socket

i In this section, the SmartMesh IP Mote opens a new UDP socket and binds it to a UDP port number.

1. At the SmartMesh IP Mote, use the *openSocket* command to create a new socket. Make note of the value returned in *socketId*.



The screenshot shows the API Explorer interface for Dust Networks. The window title is "API Explorer © Dust Networks". The interface is divided into several sections:

- api**: network type: SmartMesh IP, device type: mote, load
- connection**: through serial port: port name: COM25, disconnect. Connection successful.
- command**: openSocket, protocol, udp, INT8U, send
- response**:

openSocket	
RC	socketId
0 (RC_OK)	22
INT8U	INT8U
- notifications**:

eventsNotif		
events	state	alarmsList
128 (svcChange)	5 (operational)	0
INT32U	INT8U	INT32U
- tooltip**: The openSocket command creates an endpoint for IP communication and returns an ID for the socket. 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

- Use that **socketId** value as a handler to bind that newly created socket to a UDP port number of your choice (in our case 60000).

API Explorer © Dust Networks

api
network type: SmartMesh IP device type: mote load

connection
through serial port:
port name: COM25 disconnect
Connection successful.

command
bindSocket

socketId	port
22	60000
INT8U	INT16U

send

response
bindSocket

RC
0 (RC_OK)
INT8U

notifications
eventsNotif

events	state	alarmsList
128 (svcChange)	5 (operational)	0
INT32U	INT8U	INT32U

tooltip
Bind a previously opened socket to a port. When a socket is created, it is only given a protocol family, but not assigned a port. This association must be performed before the socket can accept connections from other hosts.
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

You cannot send data without opening and binding a socket. This is because the socket is used to identify the UDP source port of the data you are sending.

Have the Mote Send Data to the Manager



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

1. At the SmartMesh IP Mote, use the *sendTo* command to create an send a packet to the manager. Note that, in a SmartMesh IP network, the manager's IPv6 address is `ff02::2`, which translates to the hexadecimal number `ff020000000000000000000000000002`.

- At the SmartMesh IP Mote, a `txDone` notification is generated when the packet is accepted for transmission. The `packetId` specified in that notification corresponds to the `packetId` specified in the `sendTo` command.

The screenshot shows the API Explorer interface with the following sections:

- api**: network type: SmartMesh IP, device type: mote, load
- connection**: through serial port: port name: COM25, disconnect. Connection successful.
- command**: sendTo

socketId	destIP	destPort	serviceType	priority	packetId	payload
22	ff020000000000000000000000000002	61000	bandwidth	medium	1234	abcd
INT8U	16B (hex)	INT16U	INT8U	INT8U	INT16U	hex
- response**: sendTo

RC
0 (RC_OK)
INT8U
- notifications**: txDoneNotif

packetId	status
1234	0 (ok)
INT16U	INT8U
- tooltip**: Send a packet into the network. If the command returns RC_OK, the mote has accepted the packet and has queued it up for transmission. A `txDone` notification will be issued when the packet has been sent, if and only if the packet ID passed in this command is different from `0xffff`. You can set the packet ID to any value. The notification will contain the packet ID of the packet just sent, allowing association of the notification with a particular packet. The destination port should be in the range `0xF0B8-F0BF` (61624-61631) to maximize payload.
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

- At the SmartMesh IP Manager, the reception of that data packet will trigger a `notifData` notification.

API Explorer © Dust Networks

api

network type: device type:

connection

through serialMux:

host: port:

Connection successful.

command

filter	unackFilter
fffffff	00000000
4B (hex)	4B (hex)

response

subscribe

RC
0 (RC_OK)
INT8U

notifications

notification.notifData

utcSecs	utcUsecs	macAddress	srcPort	dstPort	data
1025665929	190500	00170d0000380348	60000	61000	abcd
8B (int)	INT32U	8B (hex)	INT16U	INT16U	hex

tooltip

The subscribe command indicates that the manager should send the external application the specified notifications. It contains two filter fields:

- filter is a bitmask of flags indicating the types of notifications that the client wants to receive
- unackFilter allows the client to select which of the notifications selected in filter should be sent acknowledged. If a notification is sent as 'acknowledged', the subsequent notification packets will be queued while waiting for response.

Each subscription request overwrites the previous one. If an application is subscribed to data and then decides he also wants events he should send a subscribe command with both the data and event flags set. To clear all subscriptions, the client should send a subscribe command with the filter set to zero. When a session is initiated between the manager and a client, the subscription filter is initialized to zero.

The subscribe bitmap uses the values of the notification type enumeration. Some values are unused to provide backwards compatibility with earlier APIs.

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

4. You can verify that the packet at the SmartMesh IP Manager corresponds to the packet sent at the SmartMesh IP Mote, i.e. sent from UDP port 60000 to UDP port 61000, with payload `abcd`. The sender and receiver UDP port numbers do not need to match.

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 CLI

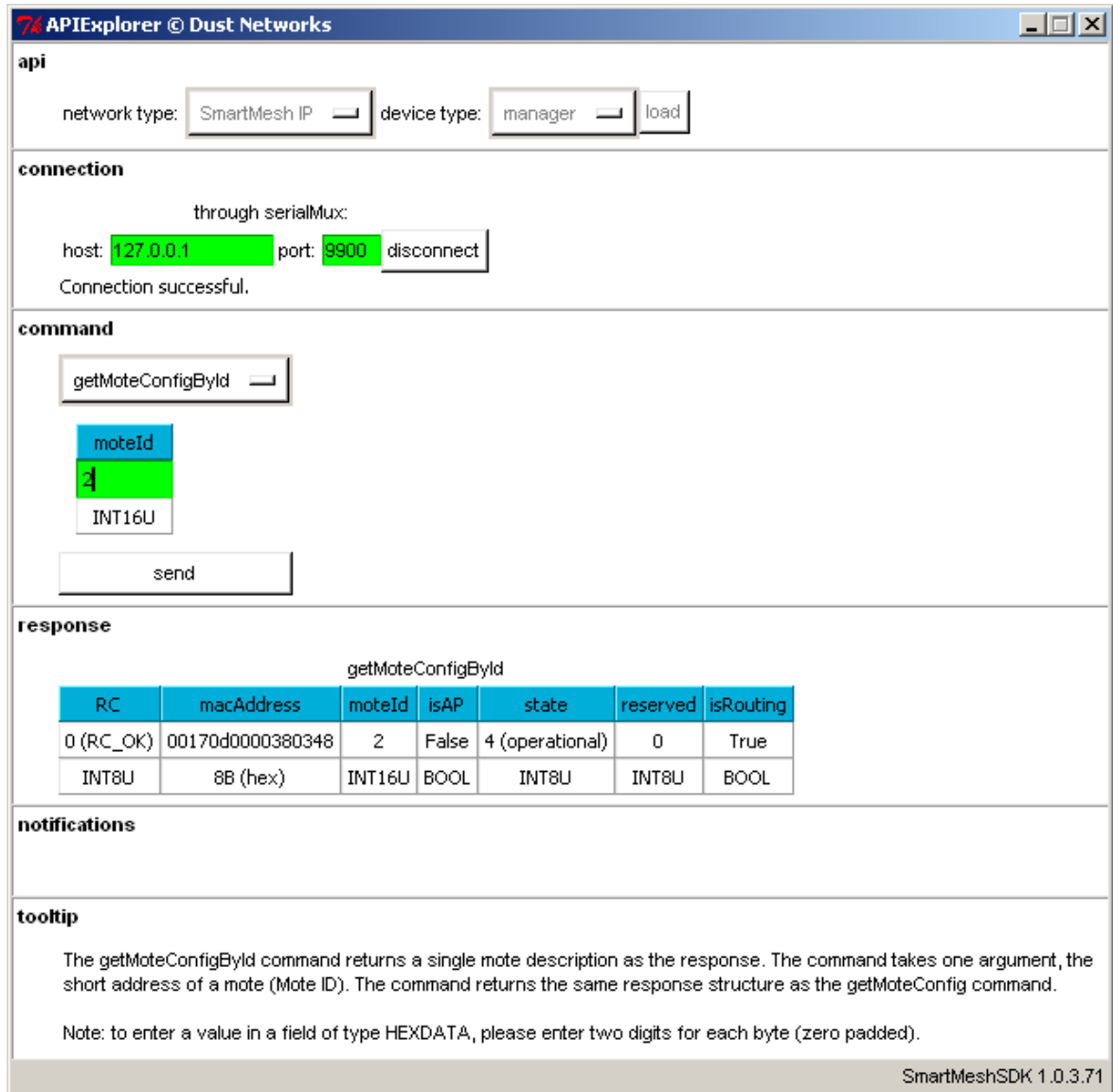
Log into the CLI interface of the Manager as described in [Interacting With the Manager](#). Send a ping command to the mote 2. The mote's ID can be found by using the `sm` command as shown earlier. The Mote with ID=2 will respond with the round trip delay time, temperature and supply voltage.

```
> ping 2
Sending ping request to mote 2
> Ping response from mote 2, time=339 msec v=3582 t=31
```

Via API

 The APIs refer to motes by MAC address, while the CLI often uses `moteID`. To convert between the two, we use the `getMoteConfigByID` command.

1. Using the `getMoteConfigById` command, enter 2 in the **moteId** field and press **send**.
 - The manager returns information about mote 2 in the network.



The screenshot shows the API Explorer interface for Dust Networks. The 'api' section has 'network type' set to 'SmartMesh IP' and 'device type' set to 'manager'. The 'connection' section shows a host of '127.0.0.1' and port '9900', with a 'disconnect' button and the message 'Connection successful.'. The 'command' section has 'getMoteConfigById' selected, with 'moteId' set to '2' and a 'send' button. The 'response' section displays a table for the 'getMoteConfigById' command.

RC	macAddress	moteId	isAP	state	reserved	isRouting
0 (RC_OK)	00170d0000380348	2	False	4 (operational)	0	True
INT8U	8B (hex)	INT16U	BOOL	INT8U	INT8U	BOOL

The 'notifications' section is empty. The 'tooltip' section contains the following text:

The `getMoteConfigById` command returns a single mote description as the response. The command takes one argument, the short address of a mote (Mote ID). The command returns the same response structure as the `getMoteConfig` command.

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

2. Select the `pingMote` command and enter the MAC address of the mote you discovered above in the **macAddress** field.
3. You will receive a response indicating that the command was taken into account, and, a few seconds later, a notification that the mote you just pinged has replied. Besides round-trip timing information, this reply also contains the supply voltage and temperature at that mote.

APIExplorer © Dust Networks

api

network type: device type:

connection

through serialMux:

host: port:

Connection successful.

command

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



To receive the eventPingResponse notification at the manager, you need to have subscribed to this type of notifications.

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

- [SmartMesh IP User's Guide](#)
- [SmartMesh IP Mote API Guide](#)

8.4.2 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 all devices:

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

8.5 Advanced Topics

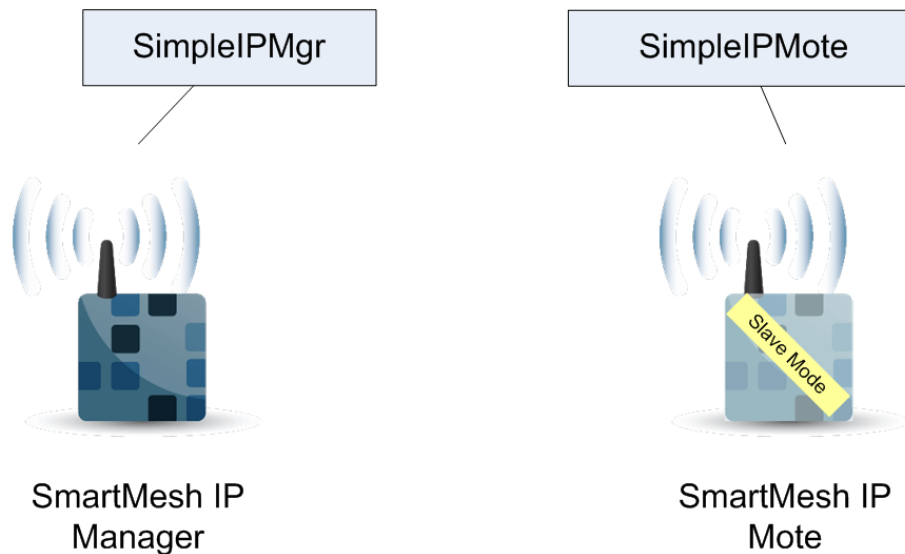
8.5.1 Exercise the API Programmatically

Overview

In this step, you will exercise the API of the SmartMesh IP Mote through an interactive script, rather than a graphical application.

It highlights the fact that you can use the SmartMesh SDK as a basis for your own applications. The sample applications used can be used as a starting point, and will therefore be run directly from the Python source code, rather than from a pre-compiled executable.

Setup Overview



Steps

The following sections use the following scripts part of the SmartMesh SDK:

- SmartMeshSDK/bin/Simple/SimpleIpMgr.py
- SmartMeshSDK/bin/Simple/SimpleIpMote.py



For more information

This section gives you a glimpse for how to use the SmartMeshSDK programmatically.

For a complete description, refer to the [SmartMeshSDK](#) documentation.

Interacting with the SmartMesh IP Manager



Make sure your SmartMesh IP Manager is connected to your computer, and switched on.

Run SimpleIpMgr.py

1. Open Windows Browser to your SmartMeshSDK directory.
2. Navigate to /bin/Simple.
3. Double-click on SimpleIpMgr.py. A Command Window appears.
4. At the line:

```
Do you want to connect to a manager over SerialMux? [y/n]
```

type `y` and press enter.

5. At the line:

```
Enter the SerialMux's host (leave blank for 127.0.0.1)
```

press Enter if your SerialMux runs on the same machine as this script.

6. At the line:

```
Enter the SerialMux's port (leave blank for 9900)
```

press Enter if your SerialMux runs with default TCP port settings.

7. The script ends at the line:

```
Script ended. Press Enter to exit.
```

Press Enter to close the Window.

The following trace contains the complete output of the script:

```
Simple Application which interacts with the IP manager - (c) Dust Networks

===== Step 1. Connecting to the manager =====
Do you want to connect to a manager over SerialMux? [y/n] y
Enter the SerialMux's host (leave blank for 127.0.0.1)
Enter the SerialMux's port (leave blank for 9900)
=====
Creating connector
done.
=====
Connecting to IP manager
done.

===== Step 2. Getting information from the network =====
=====
Retrieve the network info
Tuple_dn_getNetworkInfo(RC=0, numNotes=1, asnSize=7250, advertisementState=0, do
wnFrameState=1, netReliability=100, netPathStability=100, netLatency=400, netSta
te=0, ipv6Address=(254, 128, 0, 0, 0, 0, 0, 0, 0, 23, 13, 0, 0, 56, 6, 106))

===== Step 3. Disconnecting from the device =====
=====
Disconnecting from IP manager
done.
Script ended. Press Enter to exit.
```

The script goes through 3 steps:

1. It connects to a SmartMesh IP Manager.
2. It retrieves the status of the network attached to the SmartMesh IP Manager.
3. It disconnects from the SmartMesh IP Manager.

Understand SimpleIpMgr.py

Open the `SimpleIpMgr.py` script with a text editor to see its contents (do not double-click on it which, by default, will run the script rather than opening it).

The file contains comments throughout to allow you to match it against the printout at the execution.

A few keys for understanding:

- the `IpMgrConnector` object (and its instance `connector`) is the entity which physically connects to the SmartMesh IP Manager, over the `SerialMux`.
- `raw_input()` is a Python function which halts a script and waits for a user to type in a string and press Enter.



For more information

For a complete description, refer to the [SmartMeshSDK](#) documentation.

Interacting with the SmartMesh IP Mote



Make sure your SmartMesh IP Mote is connected to your computer, is switched in, and is operating in **slave** mode.

Run SimpleIpMote.py

1. Open Windows Browser to your SmartMeshSDK directory.
2. Navigate to `/bin/Simple`.
3. Double-click on `SimpleIpMote.py`. A Command Window appears.
4. At the line:

```
Do you want to connect to a device? [y/n]
```

type `y` and press Enter.

5. At the line:

```
Enter the serial port of the IP mote's API (e.g. COM30)
```

enter the serial port of your SmartMesh IP Mote's API port and press Enter.

6. The script ends at the line:

```
Script ended. Press Enter to exit.
```

Press Enter to close the Window.

The following code block contains the complete output of the script:

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

===== Step 1. API exploration =====
=====
Load the API definition of the IP mote
done.
=====
List all the defined command IDs:
[1, 2, 6, 7, 8, 9, 12, 16, 17, 18, 21, 22, 23, 24, 36, 40]
=====
List all the defined command names:
['setParameter', 'getParameter', 'join', 'disconnect', 'reset', 'lowPowerSleep',
 'testRadioRx', 'clearNV', 'requestService', 'getServiceInfo', 'openSocket', 'closeSocket', 'bindSocket', 'sendTo', 'search', 'testRadioTxExt']
=====
Get the command name of command ID 2:
getParameter
=====
Get the command ID of command name 'getParameter':
2
=====
List the subcommand of command 'getParameter':
['macAddress', 'networkId', 'txPower', 'joinDutyCycle', 'eventMask', 'moteInfo',
 'netInfo', 'moteStatus', 'time', 'charge', 'testRadioRxStats', 'OTAPLockout', 'moteId', 'ipv6Address', 'routingMode', 'appInfo', 'powerSrcInfo', 'powerCostInfo', 'mobilityType', 'advKey', 'sizeInfo', 'autoJoin']
=====
Get a description of the getParameter.moteStatus command:
The getParameter<moteStatus> command is used to retrieve current mote state and other dynamic information.
=====
List the name of the fields in the getParameter.moteStatus request:
[]
=====
List the name of the fields in the getParameter.moteStatus response:
['state', 'reserved_0', 'reserved_1', 'numParents', 'alarms', 'reserved_2']
=====
Print the format of the getParameter.moteStatus 'state' response field:
int
=====
Print the length of the getParameter.moteStatus 'state' response field:
1
=====
```

```
Print the valid options of the getParameter.moteStatus 'state' response field:
[0, 1, 2, 3, 4, 5, 6, 7, 8]
=====
Print the description of each valid options of the getParameter.moteStatus 'state'
response field:
['init', 'idle', 'searching', 'negotiating', 'connected', 'operational', 'discon
nected', 'radiotest', 'promiscuous listen']

===== Step 2. Connecting to a device =====
Do you want to connect to a device? [y/n] y
Enter the serial port of the IP mote's API (e.g. COM30) COM6
=====
Creating connector
done.
=====
Connecting to IP mote
done.

===== Step 3. Getting information from the device =====
=====
Retrieve the moteStatus, through 'raw' API access:
{'numParents': 0, 'reserved_1': 0, 'reserved_0': 0, 'reserved_2': 0, 'state': 1,
 'RC': 0, 'alarms': 0}
=====
Retrieve the moteStatus, through function-based API access:
Tuple_dn_getParameter_moteStatus(RC=0, state=1, reserved_0=0, reserved_1=0, numP
arents=0, alarms=0, reserved_2=0)

===== Step 4. Disconnecting from the device =====
=====
Disconnecting from IP mote
done.
Script ended. Press Enter to exit.
```

The script goes through 4 steps:

1. It loads the list of commands of the SmartMesh IP Mote API and uses SmartMesh SDK functions to obtain details about those commands.
2. It connects to a SmartMesh IP Mote.
3. It retrieves the status of the SmartMesh IP Mote.
4. It disconnects from the SmartMesh IP Mote.

Understand SimpleIpMote.py

Open the `SimpleIpMote.py` script with a text editor to see its contents (do not double-click on it which, by default, will run the script rather than opening it).

The file contains comment throughout to allow you to match it against the printout at the execution.

A few keys for understanding:

- the `IpMoteDefinition` object (and its instance `apidef`) represents the SmartMesh IP Mote API definition, i.e. the list of commands you can send to a SmartMesh IP Mote.
- the `IpMoteConnector` object (and its instance `connector`) is the entity which physically connects to the API port of a SmartMesh IP Mote.



For more information

For a complete description, refer to the [SmartMeshSDK](#) documentation.

Common Problems

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 all devices:

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

I get no output when connected to the manager over the SerialMux

It is possible that APIExplorer can connect to the SerialMux (the connection fields turns green), but the SerialMux cannot connect to the SmartMesh IP Manager. This may be because the SerialMux is configured to listen to a serial port which is not the API port of your SmartMesh IP Manager.

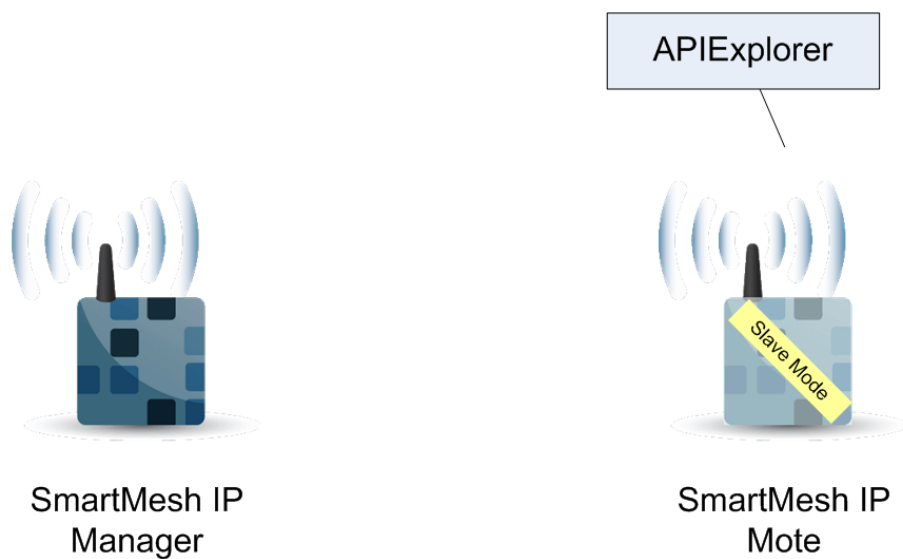
To change the configuration of your SerialMux, follow the [Serial Mux Configuration](#) guide.

8.5.2 Log HDLC Frames

Overview

In this step, you will use the APIExplorer's logging capability to see the raw bytes exchanged between your computer and the SmartMesh IP Mote.

Setup Overview



Steps

For more information

This section gives you a glimpse of how to use the logging capabilities of the SmartMeshSDK.

For a complete description, refer to the [SmartMeshSDK](#) documentation.

1. Open a Windows Browser window to navigate to your SmartMeshSDK directory.
2. Navigate to `win/`.
3. If present, erase the file `APIExplorer.log`.
4. Double-click on `APIExplorer.exe` to launch the APIExplorer application.
5. Connect to your SmartMesh IP Mote.

6. Send a *reset* command and wait to receive the `boot` event notifications.
7. Disconnect from the SmartMesh IP Mote and close the APIExplorer application.
8. Open the newly created `APIExplorer.log` file with a text editor. If you are running the APIExplorer from the windows command line from a different directory, the `APIExplorer.log` file will instead be in that directory.

`APIExplorer.log` contains the activity of all the modules in the APIExplorer application. An example output is provided below:

```

2012-12-12 17:32:16,131 [Crc:DEBUG] calculating for data=[15, 9, 8, 0, 0, 0, 1, 1, 0, 0, 0, 0]
2012-12-12 17:32:16,131 [Crc:DEBUG] fcs=0xd767
2012-12-12 17:32:16,131 [Hdlc:DEBUG]
receivedFrame:
- payload: 0f 09 08 00 00 00 01 01 00 00 00 00
- fcs:      d7 67
- valid:    True
2012-12-12 17:32:16,131 [SerialConnector:DEBUG] cmdId=15 length=9 isResponse=False packetId=0
payload=[0, 0, 0, 1, 1, 0, 0, 0, 0]
2012-12-12 17:32:16,131 [SerialConnector:DEBUG] <----- moteToPc DATA (0) -----
2012-12-12 17:32:16,131 [SerialConnector:DEBUG] _sendInternal cmdId=15 retry=0 isResponse=True
serializedFields=[]
2012-12-12 17:32:16,131 [Crc:DEBUG] calculating for data=[15, 0, 1, 0]
2012-12-12 17:32:16,145 [Crc:DEBUG] fcs=0xff57
2012-12-12 17:32:16,145 [Hdlc:DEBUG]
packetToSend:
- payload: 0f 00 01 00
- fcs:      ff 57
- valid:    True
2012-12-12 17:32:16,145 [SerialConnector:DEBUG] ----- moteToPc ACK (0) after 0.015 ----->
2012-12-12 17:32:16,145 [SerialConnector:DEBUG] ack sent
2012-12-12 17:32:16,145 [ByteArraySerializer:DEBUG] deserialize ...
- type=notification
- id=15
- byteArray=[0, 0, 0, 1, 1, 0, 0, 0, 0]
2012-12-12 17:32:16,145 [ByteArraySerializer:DEBUG] ... deserialized into
- nameArray=['events']
- returnFields={'alarmsList': 0, 'state': 1, 'events': 1}
2012-12-12 17:32:25,584 [Hdlc:INFO] disconnect
2012-12-12 17:32:25,584 [SerialConnector:INFO] hdlc notification: connection state=False
2012-12-12 17:32:25,584 [Hdlc:INFO] thread ended

```

Some keys for understanding the file:

- Every entry follows the same format:

```
<date> <timestamp> [<module>:<loglevel>] <logmessage>
```

- The entries from the `Hdlc` module indicate the exact bytes sent over the serial port.
- The entries from the `SerialConnector` module indicate how those bytes are serialized/deserialized.

Common Problems

APIExplorer.log is very long

The logging output from the APIExplorer is appended to the end of the `APIExplorer.log` file each time you start the application. To restart a clean logging session, remove the `APIExplorer.log` file before starting the APIExplorer application.

8.5.3 Upstream Communication

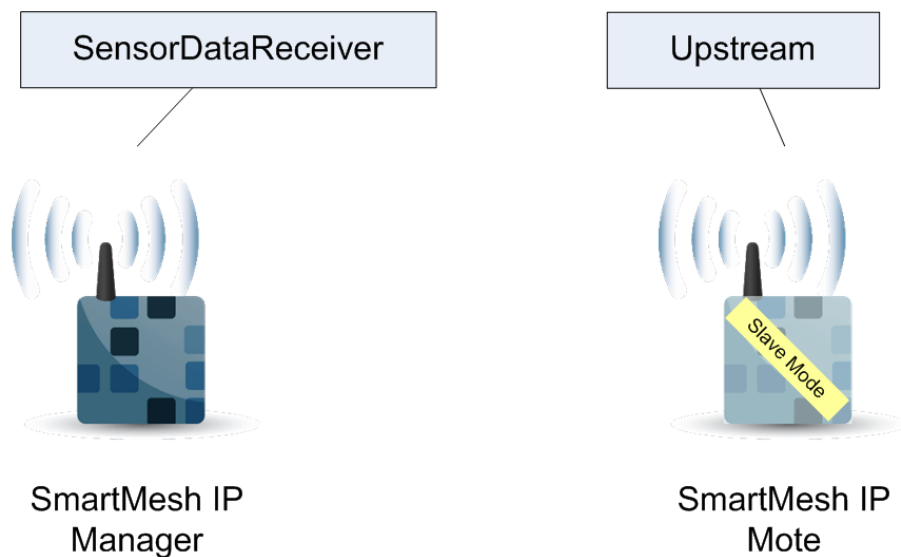
Definition

Upstream communication goes from a SmartMesh IP Mote to the SmartMesh IP Manager.

Overview

In this step, you will use the UpStream application to drive the SmartMesh IP Mote through a state machine which will take it from power-up to sending data.

Setup Overview



Steps

- ✔ The UpStream application can be seen as a programmatic version of the steps followed in the [Basic Walk-Through](#) using the APIExplorer.

1. Setup your SmartMesh IP Mote:
 1. Verify that your SmartMesh IP Mote is configured with the correct network id, and is operating in **slave** mode.
 2. Reset your SmartMesh IP Mote, either by power cycling it, or by issuing a *reset* command with the APIExplorer.
 3. Navigate to your SmartMeshSDK directory, then to `/win/`.
 4. Double-click on the `Upstream.exe` program. The application opens.
 5. Enter the name of the API port of your SmartMesh IP Mote, and press **connect**.
 6. The Upstream application drives your SmartMesh IP Mote through the following state machine:
 - Wait for a possible initial `boot` event notification.
 - Configure the join duty cycle.
 - Issue the *join* command. The SmartMesh IP Mote starts searching for a network.
 - When the SmartMesh IP Mote has joined the network, request a service.
 - Once that service has been granted, the mote reaches the `READYTOSEND` state, which enables the "sensor data to send" frame.
2. Setup your SmartMesh IP Manager:
 1. Navigate to your SmartMeshSDK directory, then to `/win/`.
 2. Double-click on the `SensorDataReceiver.exe` program. The application opens.
 3. Use one of the "manager connection" options to connect to your SmartMesh IP Manager.
3. Send data from the SmartMesh IP Mote to the SmartMesh IP Manager:
 1. On the UpStream application, connected to your SmartMesh IP Mote, change the value of the slider to any value. This emulates the value collected by a sensor.
 2. Press the "send to manager" button. This sends the following UDP packet from your SmartMesh IP Mote to your SmartMesh IP Manager:
 - UDP source port `61000`.
 - UDP destination port `61000`, or as set in the UpStream application.
 - The packet contains two bytes of data representing the value of the slider on the UpStream application.
 3. On the SensorDataReceiver application, connected to your SmartMesh IP Manager, the data is displayed in the "received sensor data" frame when the packet is received.

at the SmartMesh IP Manager

The screenshot shows a window titled "SensorDataReceiver © Dust Netwo...". It is divided into two main sections:

- manager connection**: This section shows the connection method as "through serialMux:". Below this, the host is "127.0.0.1" and the port is "9900". There is a "disconnect" button. A status message below reads "Connection successful."
- received sensor data**: This section displays a numerical value "17414" in a box. Below the box is a small bar chart with two bars of different heights. Further down, the following details are listed:
 - source MAC: 00170d0000380348
 - source port: 60000
 - destination port: 61000

At the bottom right of the window, the version "SmartMeshSDK 1.0.3.72" is displayed.

at theSmartMesh IP Mote

Upstream © Dust Networks

sensor data to send

17414

destination IPv6 address

dest. UDP port

ff020000000000000000000000000002	61000	send to manager
20010470006600170000000000000002	61000	send to host

Sent successfully

join state machine

READYTOSEND	active	ready to send	289.791s
BINDSOCKET	done	bind the socket	0.016s
OPENSOCKET	done	open a socket	0.015s
SERVICEGRANTED	done	service granted	0.000s
REQUESTINGSERVICE	done	requesting service	61.561s
JOINED	done	joined	0.000s
OPERATIONAL	done	mote is operational	63.561s
JOINREQUESTSENT	done	join request sent	7.795s
SEARCHING	done	searching for a network	14.605s
CONFIGURED	done	configured	0.000s
CONFIGURING_DC	done	configuring the join duty cycle	0.047s
CONFIGURE	done	start configuring the mote	0.000s
ASSESSMOTESTATE	done	evaluate what the current mote state is	0.016s
WAITFORINITIALNOTIF	done	wait for initial notifications	1.921s

mote connection

through serial port:

port name: COM25 disconnect

Connection successful.


tip

Remember to reset your mote before starting this application.

Note: The "send to manager" and "send to host" button become active only once the mote has reached the READYTOSEND state.

SmartMeshSDK 1.0.3.71

The "send to host" button allows you to send data to any IP address. This functionality requires routing through an LBR and is covered in a separate section.

 For the UpStream application to stay generic, it does not configure the Network ID of the mote. For your convenience, the corresponding code is present, but commented out, in the `UpStream.py` source file.

Common Problems

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 all devices:

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

My mote does not join

- Depending on its join duty cycle and the state of your network, it can take a SmartMesh IP Mote several minutes to join.
- Verify that your SmartMesh IP Mote is configured on the same network ID as your SmartMesh IP Manager

8.5.4 Downstream Communication

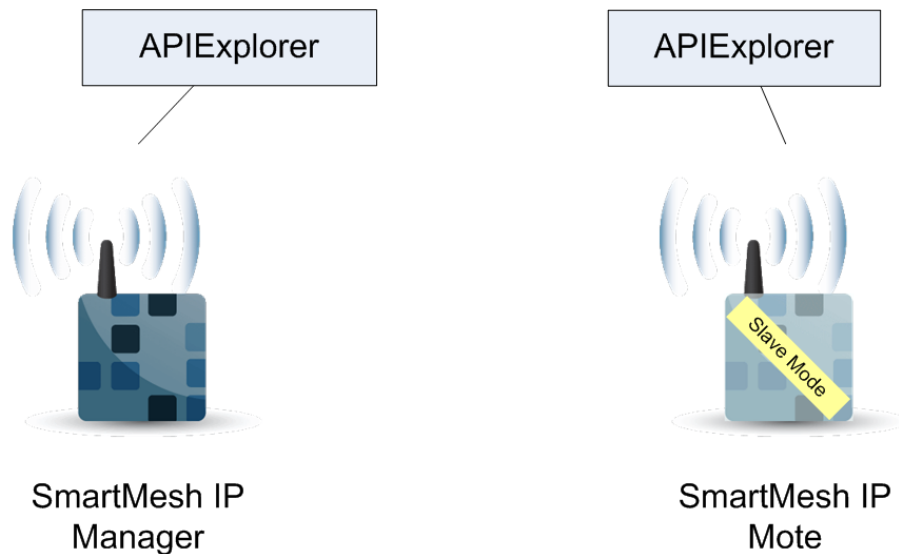
Definition

Downstream communication goes from a SmartMesh IP Manager to the SmartMesh IP Mote.

Overview

In this step, you will use the APIExplorer to send data from the SmartMesh IP Manager to an already-connected SmartMesh IP Mote.

Setup Overview



Steps

1. Follow the steps described in the [Basic Walk-Through](#) to connect a SmartMesh IP Mote to a SmartMesh IP Manager, and open a socket on UDP port 60000.
2. Using the APIExplorer connected to the SmartMesh IP Mote, issue a `getParameter.macAddress` to read its MAC address. In our case, 00170d0000380348.
3. If it's not already done, start a second APIExplorer and connect it to the SmartMesh IP Manager.

4. In that second APIExplorer, issue a *sendData* command and fill in the following data:
 - The **macAddress** field should contain the MAC address as returned by the SmartMesh IP Mote.
 - **priority:Medium**. Without any other traffic in the network, this priority does not change packet propagation.
 - **srcPort**: 61000, or a 16-bit port number of your choice.
 - **dstPort**: 60000, i.e. the UDP port number of the socket opened on the SmartMesh IP Mote.
 - **options**: 0 (none).
 - **data**: 1234, or another hexadecimal payload of your choice.
5. Press **send**. The packet is sent from the SmartMesh IP Manager to the SmartMesh IP Mote.

The reception of the packet triggers a *receive* notification at the SmartMesh IP Mote, indicating that the packet comes from IPv6 address `ff02::2` (the well-known IPv6 address of the SmartMesh IP Manager), and from UDP port 61000.

at the SmartMesh IP Manager

APIExplorer © Dust Networks

api

network type: device type:

connection

through serialMux:

host: port:

Connection successful.

command

macAddress	priority	srcPort	dstPort	options	data
00170d0000380348	Medium	61000	60000	0	1234
8B (hex)	INT8U	INT16U	INT16U	INT8U	hex

response

sendData

RC	callbackId
0 (RC_OK)	1
INT8U	INT32U

notifications

notification.notifEvent.eventPacketSent

eventId	callbackId	rc
1	1	0
INT32U	INT32U	INT8U

tooltip

The sendData command sends a packet to a mote in the network. The response contains a callbackId. When the manager injects the packet into the network, it will generate a packetSent notification. It is the application layers responsibility send a response from the mote, and to timeout if no response is received.

The sendData command should be used by applications that communicate directly with the manager. If end-to-end (application to mote) IP connectivity is required, the application should use the sendIP command. For a more comprehensive discussion of the distinction, see the SmartMesh IPNetwork User Guide.

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

at the SmartMesh IP Mote

APIExplorer © Dust Networks

api

network type: device type:

connection

through serial port:

port name:

Connection successful.

command

socketId	port
22	60000
<input type="text" value="INT8U"/>	<input type="text" value="INT16U"/>

response

bindSocket

RC
0 (RC_OK)
<input type="text" value="INT8U"/>

notifications

receiveNotif

socketId	srcAddr	srcPort	payload
22	ff020000000000000000000000000002	61000	1234
<input type="text" value="INT8U"/>	<input type="text" value="16B (hex)"/>	<input type="text" value="INT16U"/>	<input type="text" value="hex"/>

tooltip

Bind a previously opened socket to a port. When a socket is created, it is only given a protocol family, but not assigned a port. This association must be performed before the socket can accept connections from other hosts.

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

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 all devices:

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

I get no output when connected to the manager over the SerialMux

It is possible that APIExplorer can connect to the SerialMux (the connection fields turns green), but the SerialMux cannot connect to the SmartMesh IP Manager. This may be because the SerialMux is configured to listen to a serial port which is not the API port of your SmartMesh IP Manager.

To change the configuration of your SerialMux, follow the [Serial Mux Configuration](#) guide.


My mote does not join

- Depending on its join duty cycle and the state of your network, it can take a SmartMesh IP Mote several minutes to join.
- Verify that your SmartMesh IP Mote is configured on the same network ID as your SmartMesh IP Manager

8.5.5 Internet Integration

Overview

In this step, you will use the UpStream application to send data from your SmartMesh IP Mote to the <http://motedata.dustnetworks.com/> webpage.

 The computer connected to the SmartMesh IP Manager needs to be connected to the Internet.

Setup Overview

- LBRConnection
- SensorDataReceiver




SmartMesh IP
Manager

Upstream




SmartMesh IP
Mote

Steps

 You will be connecting two different applications to your SmartMesh IP Manager at the same time. You therefore need to connect through the SerialMux.

Send data to the manager

 The first steps described below are the same as the steps from the [Upstream Communication](#) tutorial.

1. Setup your SmartMesh IP Mote:
 1. Verify that your SmartMesh IP Mote is configured with the correct network id, and is operating in **slave** mode.
 2. Reset your SmartMesh IP Mote, either by power cycling it, or by issuing a *reset* command with the APIExplorer.
 3. Navigate to your SmartMeshSDK directory, then to `/win/`.
 4. Double-click on the `Upstream.exe` program. The application opens.
 5. Enter the name of the API port of your SmartMesh IP Mote, and press **connect**.
 6. The Upstream application drives your SmartMesh IP Mote through the following state machine:
 - Wait for a possible initial `boot` event notification.
 - Configure the join duty cycle.
 - Issue the *join* command. The SmartMesh IP Mote starts searching for a network.
 - When the SmartMesh IP Mote has joined the network, request a service.
 - Once that service has been granted, the mote reaches the `READYTOSEND` state, which enables the "sensor data to send" frame.
2. Setup your SmartMesh IP Manager:
 1. Navigate to your SmartMeshSDK directory, then to `/win/`.
 2. Double-click on the `SensorDataReceiver.exe` program. The application opens.
 3. Use one of the "manager connection" options to connect to your SmartMesh IP Manager.
3. Send data from the SmartMesh IP Mote to the SmartMesh IP Manager:
 1. On the UpStream application, connected to your SmartMesh IP Mote, change the value of the slider to any value. This emulates the value collected by a sensor.
 2. Press the "send to manager" button. This sends the following UDP packet from your SmartMesh IP Mote to your SmartMesh IP Manager:
 - UDP source port `61000`.
 - UDP destination port `61000`, or as set in the UpStream application.
 - The packet contains two bytes of data representing the value of the slider on the UpStream application.
 3. On the SensorDataReceiver application, connected to your SmartMesh IP Manager, the data is displayed in the "received sensor data" frame when the packet is received.

at the SmartMesh IP Manager

The screenshot shows a window titled "SensorDataReceiver © Dust Netwo...". It is divided into two main sections: "manager connection" and "received sensor data".

manager connection

through serialMux:
host: 127.0.0.1 port: 9900 disconnect
Connection successful.

received sensor data

17414

source MAC: 00170d0000380348
source port: 60000
destination port: 61000

SmartMeshSDK 1.0.3.72

at theSmartMesh IP Mote

Upstream © Dust Networks

sensor data to send

17414

destination IPv6 address

dest. UDP port

ff020000000000000000000000000002

61000

send to manager

20010470006600170000000000000002

61000

send to host

Sent successfully

join state machine

READYTOSEND	active	ready to send	289.791s
BINDSOCKET	done	bind the socket	0.016s
OPENSOCKET	done	open a socket	0.015s
SERVICEGRANTED	done	service granted	0.000s
REQUESTINGSERVICE	done	requesting service	61.561s
JOINED	done	joined	0.000s
OPERATIONAL	done	mote is operational	63.561s
JOINREQUESTSENT	done	join request sent	7.795s
SEARCHING	done	searching for a network	14.605s
CONFIGURED	done	configured	0.000s
CONFIGURING_DC	done	configuring the join duty cycle	0.047s
CONFIGURE	done	start configuring the mote	0.000s
ASSESSMOTESTATE	done	evaluate what the current mote state is	0.016s
WAITFORINITIALNOTIF	done	wait for initial notifications	1.921s

mote connection

through serial port:

port name: COM25 disconnect

Connection successful.

tip

Remember to reset your mote before starting this application.

Note: The "send to manager" and "send to host" button become active only once the mote has reached the READYTOSEND state.

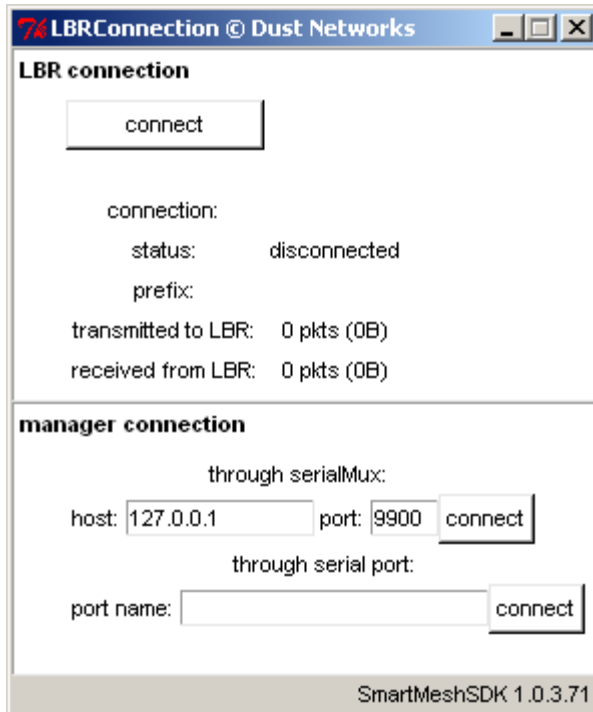
SmartMeshSDK 1.0.3.71

The "send to host" button allows you to send data to any IP address. This functionality requires routing through an LBR and is covered in a separate section.

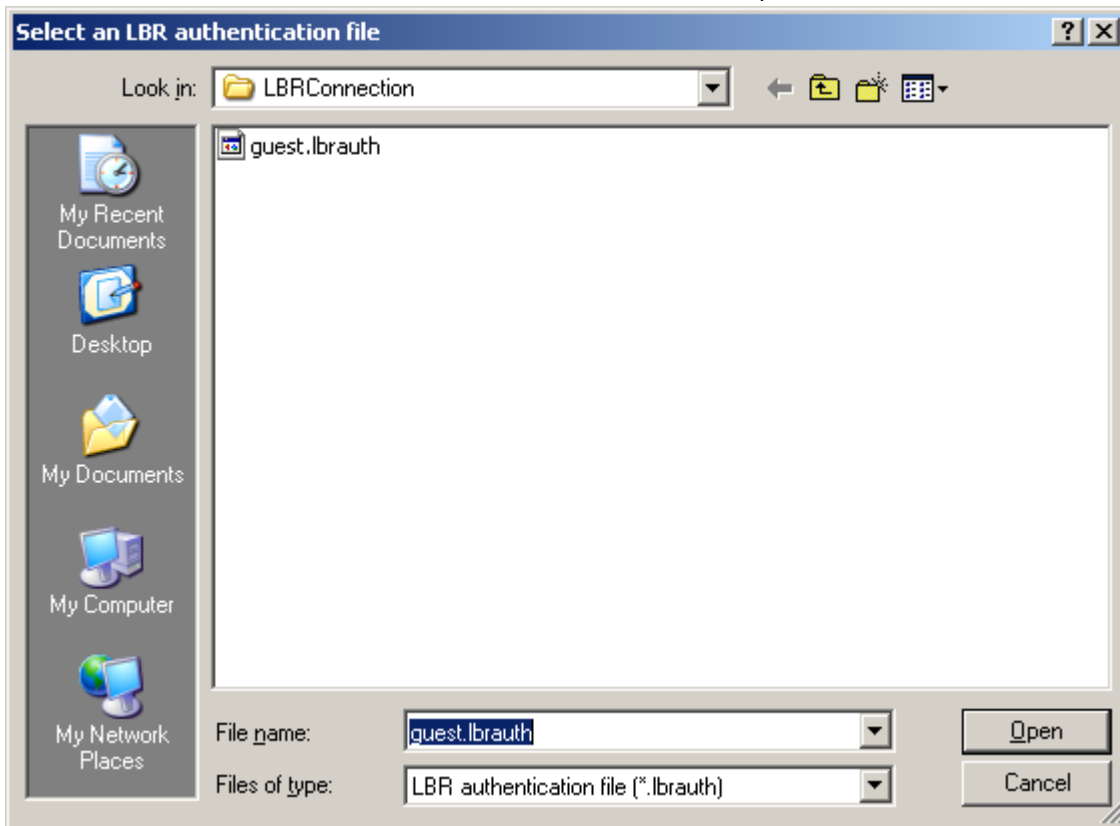
Connect the manager to the LBR

i The Low-power Border Router (LBR) is a server on the Internet which compresses/decompresses the 6LoWPAN headers into IPv6 headers. Refer to the [Low-power Border Router](#) guide for details.

1. Navigate to your SmartMesh SDK directory, then to /win/.
2. Double click on LBRConnection.exe script. The application opens.

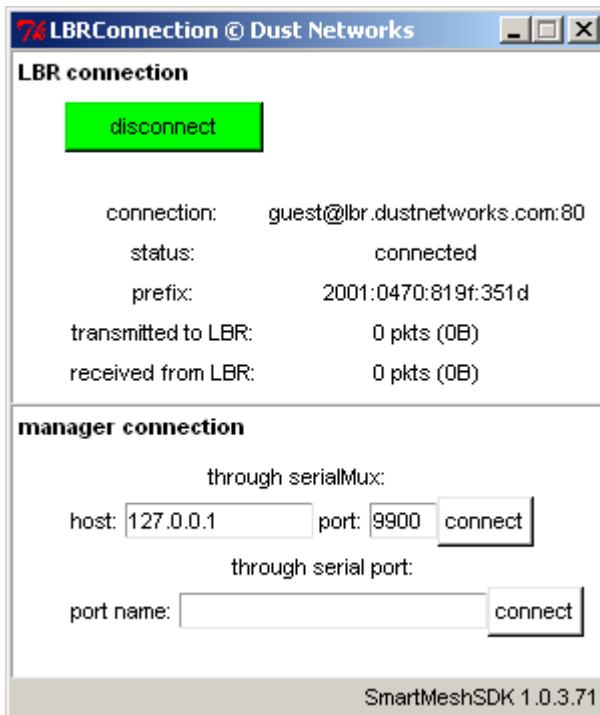


- In the **LBR connection** frame, click on **connect**, a file browser window opens.

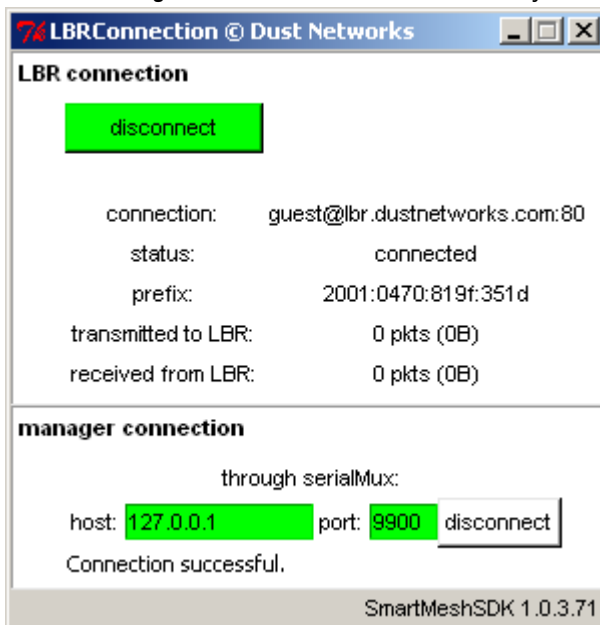


- Select the `guest.lbrauth` file, and click **open**. This file contains the address of the LBR, and the credentials you need to authenticate to it.

- The button becomes green and the field informs you of the IPv6 prefix the LBR has assigned you (in our case 2001:0470:819f:351d). Make note of that prefix.




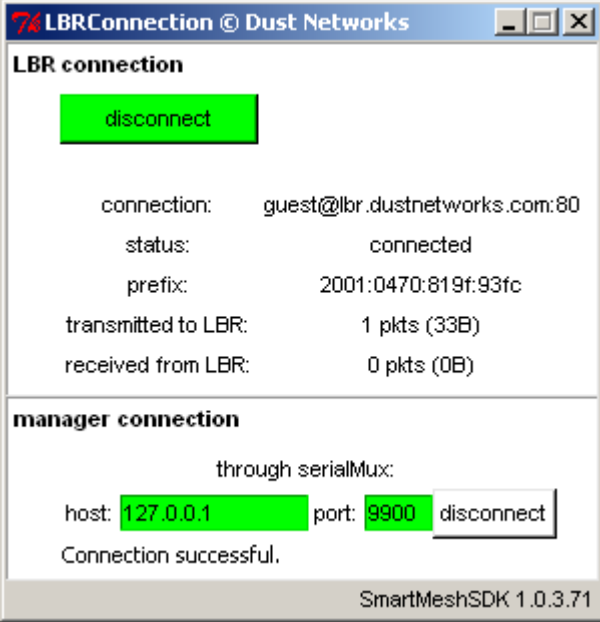
- Use the **manager connection** frame to connect to your SmartMesh IP Manager.



Send data to motedata.dustnetworks.com

- Open the <http://motedata.dustnetworks.com/> page in your browser. The 12 gauges represent the data sent by the last 12 SmartMesh IP Motes which have connected to the system.

2. In your UpStream application, make sure that
 1. The second IPv6 destination address is set to 20010470006600170000000000000002, the IPv6 address of the motedata.dustnetworks.com server.
 2. The second destination UDP port is set to 61000, the UDP port the motedata.dustnetworks.com server is listening on.
3. In your Upstream application, set the slider in **sensor data to send** to some value (in our case 25091), and press the **send to host** button.
4. This triggers the following sequence of events:
 1. Your SmartMesh IP Mote sends a UDP packet to IPv6 address 20010470006600170000000000000002, UDP port 61000 containing the value you selected on the slider as payload.
 2. The packet reaches your SmartMesh IP Manager, which sees that it is not addressed to it, and hands it over to the LBRConnection application.
 3. The LBRConnection application forwards your packet to the LBR. You can see the **transmitted to LBR** counter incrementing.
 4. The LBR decompresses your packet from 6LoWPAN to IPv6, and injects the packet into the IPv6 Internet.
 5. The IPv6 Internet routes the packet to the motedata.dustnetworks.com server.
 6. The motedata.dustnetworks.com server stores the data you sent.
 7. Your browser periodically updates the <http://motedata.dustnetworks.com/> page, causing the gauge representing your data to update.

at http://motedata.dustnetworks.com/	at the SmartMesh IP Manager
	

at the SmartMesh IP Mote

Upstream © Dust Networks

sensor data to send

25091

destination IPv6 address	dest. UDP port	
ff020000000000000000000000000002	61000	send to manager
20010470006600170000000000000002	61000	send to host

Sent successfully

join state machine

READYTOSEND	active	ready to send	11.145s
BINDSOCKET	done	bind the socket	0.000s
OPENSOCKET	done	open a socket	0.015s
SERVICEGRANTED	done	service granted	0.000s
REQUESTINGSERVICE	done	requesting service	60.113s
JOINED	done	joined	0.016s
OPERATIONAL	done	mote is operational	52.595s
JOINREQUESTSENT	done	join request sent	7.377s
SEARCHING	done	searching for a network	29.103s
CONFIGURED	done	configured	0.000s
CONFIGURING_DC	done	configuring the join duty cycle	0.047s
CONFIGURE	done	start configuring the mote	0.000s
ASSESSMOTESTATE	done	evaluate what the current mote state is	0.000s
WAITFORINITIALNOTIF	done	wait for initial notifications	1.954s

mote connection

through serial port:

port name: COM6 disconnect

Connection successful.

tip

Remember to reset your mote before starting this application.

Note: The "send to manager" and "send to host" button become active only once the mote has reached the READYTOSEND state.

SmartMeshSDK 1.0.3.71

✔ On the Upstream application, use the **send to manager** and **send to host** buttons to send either to the manager (the data appears in the SensorDataReceiver application) or into the Internet (the data appears at <http://motedata.dustnetworks.com/>).

Note that these data streams are independent, *i.e.* data sent into the Internet does not appear in the SensorDataReceiver application or vice-versa.

Common Problems

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 all devices:

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

My mote does not join

- Depending on its join duty cycle and the state of your network, it can take a SmartMesh IP Mote several minutes to join.
- Verify that your SmartMesh IP Mote is configured on the same network ID as your SmartMesh IP Manager

9 Low-power Border Router

9.1 What is an LBR?

A Low-power Border Router (LBR) is the networking device which allows you to connect your SmartMesh IP network to the Internet. The LBR sits between a SmartMesh IP Manager and the internet. It converts between the IPv6 packet format of the internet to the 6LoWPAN packet format of the SmartMesh IP network to enable communication between computers on the Internet ("internet hosts") and SmartMesh IP Motes.

The LBR performs three functions:

- **Connectivity.** It allows your SmartMesh IP Motes to send data to servers on the Internet, and for hosts on the Internet to send data to your SmartMesh IP Motes.
- **Compression.** Its compression/decompression engine translates IPv6 into 6LoWPAN as data flows between the Internet and the SmartMesh IP network.
- **Addressing.** It manages a pool of IPv6 addresses, thereby configuring each SmartMesh IP Mote with a unique, globally reachable IPv6 address.

 The term "Low-power Border Router" is defined by the [IETF work group ROLL](#) in [RFC6550](#).

The LBR is a computer program which can run in two modes:

- in **standalone** mode, it runs on the computer connected to the SmartMesh IP Manager, and handles a single SmartMesh IP network
- in **server** mode, it runs on a server which can be located anywhere on the Internet, and accepts connections from multiple SmartMesh IP networks

9.2 Documentation Organization

- [Overview](#).
- [Test Drive](#). This page shows you how to connect to a demonstration LBR and send data from your motes into the Internet. This allows you to try the LBR out without installing anything.
- [Installation](#). This page shows you how to install an LBR. If you just want to try the LBR functionality out, we recommend you take the [Test Drive](#).
- [User Guide](#). This page details how to administer the LBR by managing users.
- [CLI guide](#). This section is a detailed description of the Command Line Interface (CLI) of the LBR, and serves as a reference.


9.3 Overview

9.3.1 Goals of an LBR

An SmartMesh IP network is IPv6-ready: each SmartMesh IP Mote can be assigned an IPv6 address, and packets exchanged comply to the 6LoWPAN standard, a compressed version of IPv6.

The LBR links the SmartMesh IP Manager to the Internet. With this link established, SmartMesh IP Motes can generate data and send it directly to a host on the Internet. Different SmartMesh IP Motes can send data to different hosts. Similarly, hosts on the Internet can send data to individual SmartMesh IP Motes in the network.

This enables a SmartMesh IP Mote to behave exactly like any host on the Internet.

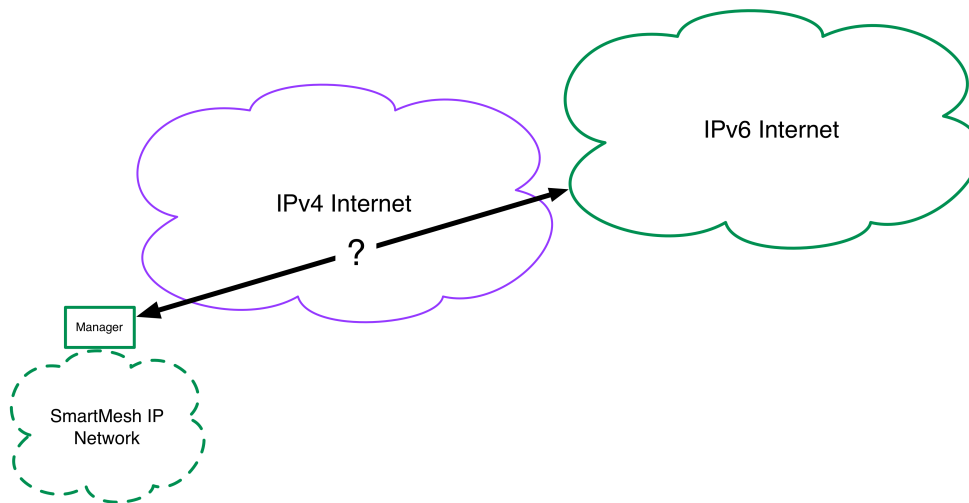
 The 6LoWPAN standard and the concept of the LBR have been developed with the Internet Engineering Task Force, the standardization body behind all Internet-related standards.

9.3.2 Services

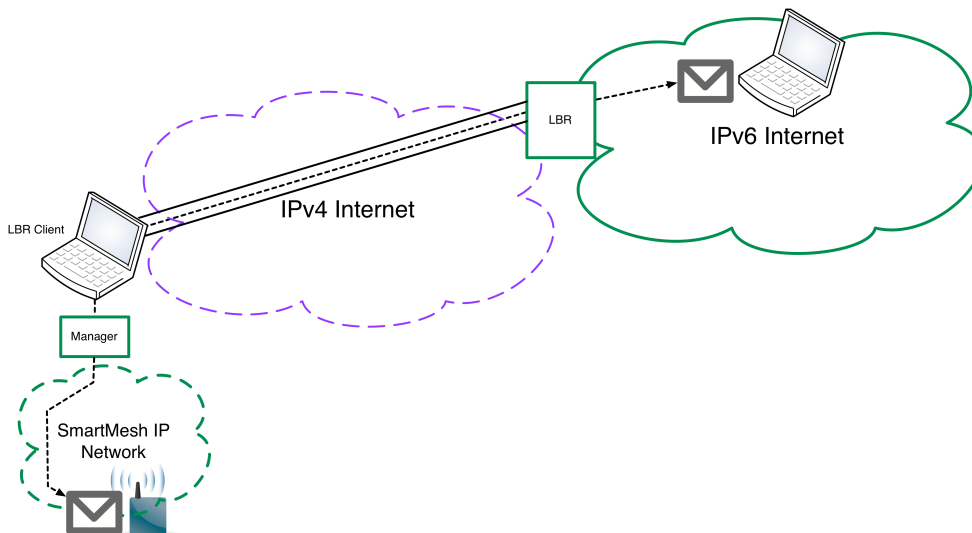
The LBR performs three functions:

- **Connectivity.** It allows your SmartMesh IP Motes to send data to servers on the Internet, and for hosts on the Internet to send data to your SmartMesh IP Motes.
- **Compression.** Its compression/decompression engine translates IPv6 into 6LoWPAN as data flows between the Internet and the SmartMesh IP network.
- **Addressing.** It manages a pool of IPv6 addresses, thereby configuring each SmartMesh IP Mote with a unique, globally reachable IPv6 address.

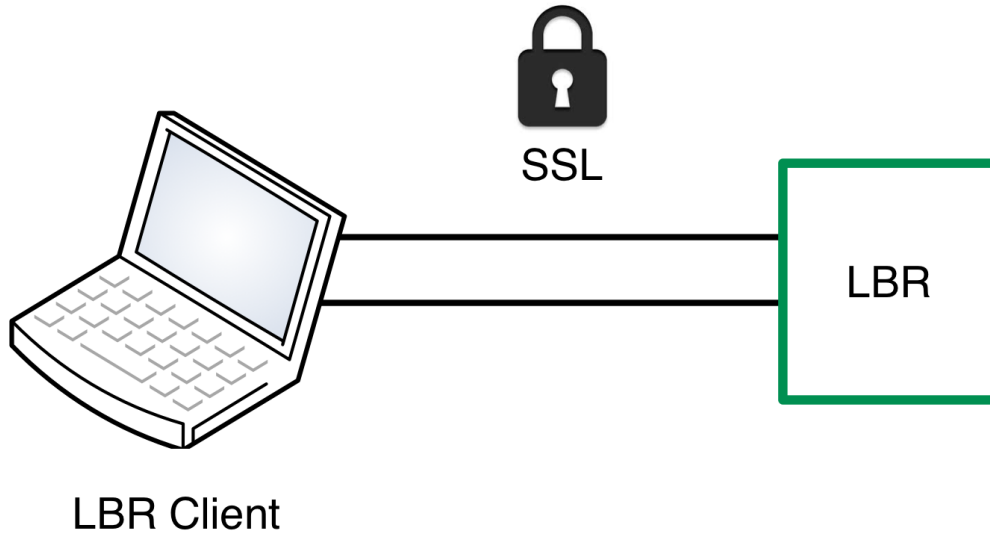
Connectivity



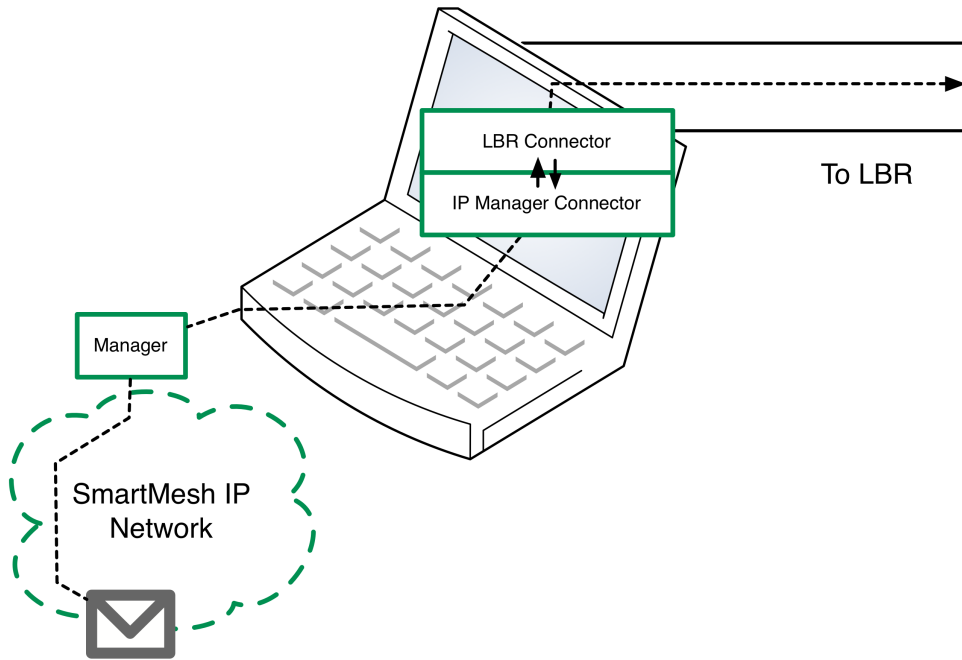
Today, there are two versions of the Internet: the older IPv4 version and the new IPv6 version. While a SmartMesh IP network is IPv6-ready, most of the current Internet still run on the older IPv4 version. The LBR understands both version, allowing for an SmartMesh IP Manager to connect to it even from a location which only supports the IPv4 Internet.



Once your SmartMesh IP Manager is connected to the LBR, the IPv6 packets exchanged between the SmartMesh IP network at the IPv6 internet are "tunneled" through the connection between the SmartMesh IP Manager and the LBR.



To enable confidentiality, data integrity and authentication, the session between the LBR client (connected to the SmartMesh IP Manager) and the LBR can be secured through a Secure Sockets Layer.

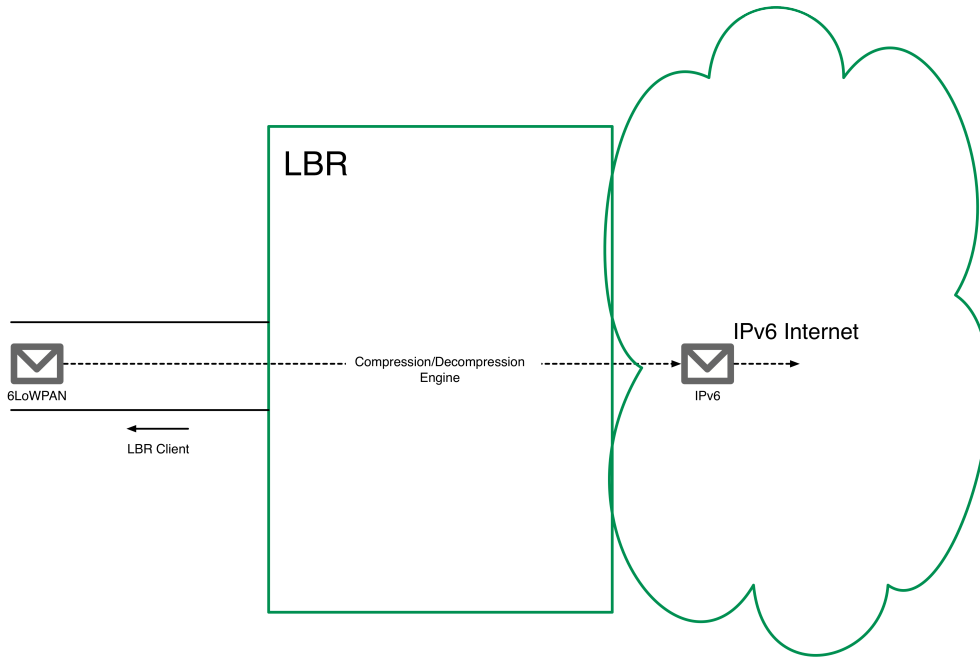


The LBR client is a computer program running on the computer physically connected to the SmartMesh IP Manager. It consists of these components:

- the `IpMgrConnector` which connects to the SmartMesh IP Manager
- the `lbrConnector` which connects to the LBR.

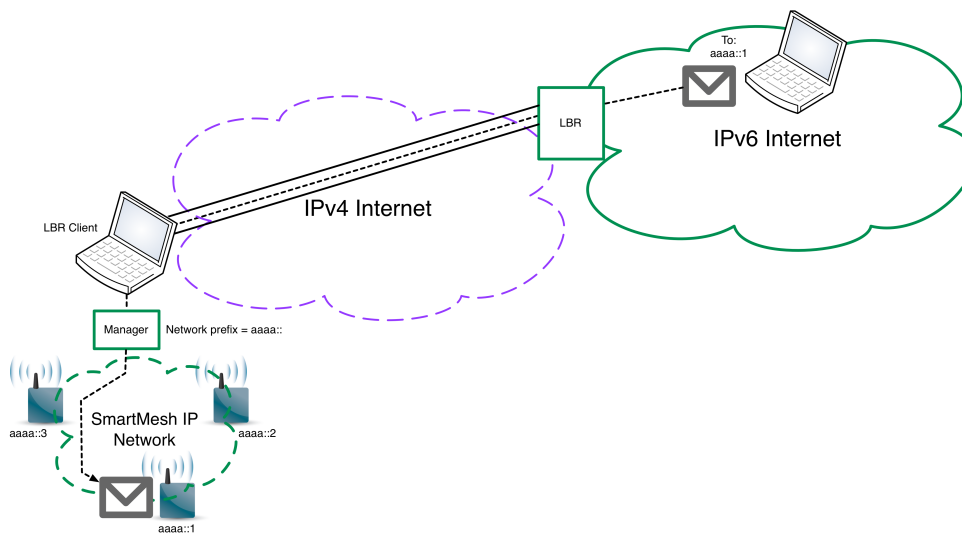
The LBR client program is part of the SmartMesh SDK.

Compression



To increase efficiency, the 6LoWPAN protocol is used inside the SmartMesh IP network, a compressed version of IPv6 protocol. The LBR contains a compression/decompression engine which turns 6LoWPAN into IPv6 on-the-fly as data is exchanged between the SmartMesh IP network and the Internet.


Addressing



When running in server mode, the LBR manages a pool of IPv6 addresses, which it assigns to the different SmartMesh IP networks connecting to it. When a client connects to it, after authenticating the client, the LBR assigns it a network "prefix", i.e. the network part of the IPv6 address of each node in the SmartMesh IP network.

This prefix is used to SmartMesh IP Mote self-configure the IPv6 address of each SmartMesh IP Mote in the network. Each SmartMesh IP Mote obtains a globally addressable, unique, IPv6 address.

9.4 Test Drive

 This page shows you how to connect to a demonstration LBR and send data from your motes into the Internet. This allows you to try the LBR out without installing anything.

9.4.1 Running

Follow the [Internet Integration](#) tutorial to connect to the demonstration resources listed below without having to install an LBR.

9.4.2 Demonstration Resources

lbr.dustnetworks.com

The following LBR is set up in server mode for demonstration purposes:

DNS name	lbr.dustnetworks.com
TCP port	80
IPv4 address	67.203.88.56
IPv6 address	2001:470:1f04:1ca6::2 (or 200104701f041ca60000000000000002 when written in full)
mote prefix*	2001:470:819f

* Indicates the /48 IPv6 prefix managed by the LBR, i.e. each user SmartMesh IP network connecting to the LBR receives a prefix of the form 2001:470:819f:xxxx/64 where xxxx depends on the user account.

 While this LBR is listening on TCP port 80, it is **not** a webserver, i.e. you can not point your browser at it.

This LBR only accepts guest accounts.

motedata.dustnetworks.com

The following data server is set up for demonstration purposes:

DNS name	motedata.dustnetworks.com
URL	http://motedata.dustnetworks.com/
IPv4 address	67.203.88.55
IPv6 address	2001:470:66:17::2 (or 20010470006600170000000000000002 when written in full)
UDP port*	61000

* Indicates the UDP port the server listens on for data sent by SmartMesh IP Motes.

It hosts:

- A daemon process which receives and stores data sent by the Upstream application
- A static web page which displays the latest data received from the last 12 SmartMesh IP Motes

9.5 Installation



This page shows you how to install an LBR. If you just want to try the LBR functionality out, we recommend you take the [Test Drive](#).

9.5.1 Requirements

Operating System

The LBR will run on any modern flavor of Linux. It requires following services to be available:

- The LBR process must be able to create/destroy **tun/tap** virtual kernel network devices. The LBR will create a new `tun` interface for each client connected, and will destroy this interface when the client disconnects.
- The LBR process must be able to bind a socket to **TCP port 80**, the default port the LBR listens on for connections.
- **IPv6** must be supported. Specifically:
 - IPv6 forwarding must be enabled
 - the LBR process must be able to assign arbitrary an IPv6 address to each `tun` interface it manages.

- The LBR process must be able to call the system commands listed in the following table:

command	description
<code>ping6</code>	a utility which send/receives ICMPv6 echo requests and responses
<code>ifconfig</code>	a utility to configure the IPv6 addresses of the <code>tun</code> interfaces the LBR manages
<code>route</code>	a utility to administer the routing table of the Linux kernel

- The LBR process must have write privileges to the `bin/dustlbr/temp/` of you LBR installation.



Most modern versions of Linux comply with the operating system requirements listed above "out-of-the-box".

Python

The LBR program requires Python 2.6 or Python 2.7 to be installed.

IPv6 connectivity

The LBR manages a /48 IPv6 prefix, which it subdivides into one /64 prefix for each SmartMesh IP which connects to it.

You can obtain a /48 prefix:

- from your network administrator if you have IPv6 support in your local network
- from most IPv6 tunnel brokers, such as [Hurricane Electric](#)

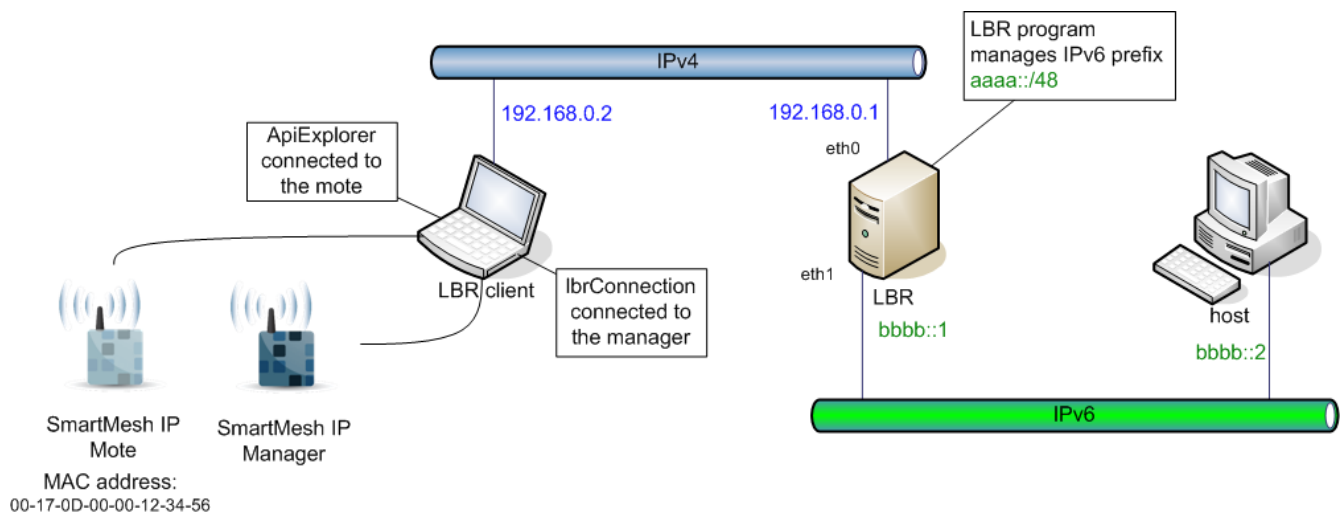
This /48 must be routed to your LBR, i.e. all traffic going to an address pertaining to that prefix should end up at your LBR.

9.5.2 Installation Steps

Follow the steps below to install the LBR and verify its operation. At the end of these steps, you will be able to:

- connect the SmartMesh IP Manager to the LBR and obtain your SmartMesh IP network's IPv6 prefix
- send data from a SmartMesh IP Mote to an Internet host
- send data from an Internet host to a SmartMesh IP Mote

Topology



The diagram above represents the connections and the addressing used for these installation steps. Replace the addresses with the ones applicable to your situation.

To be able to test your setup, you need two machines:

- the **LBR client**, which is connected to your SmartMesh IP Manager and which runs the **LBRConnection** application
- the **host**, a computer connected to IPv6 which exchanges data with the SmartMesh IP Mote

The image above shows a test setup where the LBR and LBR client, and the LBR and host sit on the same Ethernet link. This is **not** a requirement, as in a production setup, all three machine will sit at different locations.

Install the LBR

Enable IPv6 forwarding

On Debian:

```
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

On Ubuntu:

Edit the file `/etc/sysctl.conf` and uncomment the following line:

```
net.ipv6.conf.all.forwarding=1
```



You may need to restart the network service or reboot the machine for those changes to take effect.

Verify that the change was taken into account:

```
> cat /proc/sys/net/ipv6/conf/all/forwarding  
1
```

Configure the Interfaces

```
ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up  
ifconfig eth1 inet6 add bbbb::1/64
```

Verify Connectivity

Verify that the different machines can connect to each other by using the `ping` command.

- from the LBR, you should be able to ping:
 - the LBR client
 - the host
- from the LBR client, you should be able to ping:
 - the LBR
- from the host, you should be able to ping:
 - the LBR

Install the LBR program

Unzip the LBR-1.0.0.2.zip file. We call the resulting folder the LBR root folder.

Start the LBR program

```
cd <your_lbr_root_folder>
cd bin/dustlbr
python dustlbr.py aaaa:0000:0000
```

You should see the LBR command prompt:

```
Low Power Border Router (c) Dust Networks
version 1.0.0.1
>
```

Connect the LBR Client to the LBR

On the LBR client:

1. navigate to your SmartMesh SDK installation directory
2. open the file `bin/LBRConnection/guest.lbrauth` file
3. in the `lbrAddr` field, enter the IPv4 address of your LBR (in our case `192.168.0.1`) and save the file
4. double-click on `LBRConnection.py` to start the LBRConnection application
5. In the "LBR connection" frame, click `connect` and select the `guest.lbrauth` file. The application is now connected to the LBR
6. In the "manager connection" frame, connect to your SmartMesh IP Manager

The LBR has assigned your SmartMesh IP network the prefix `aaaa:0000:0000:abcd` where `abcd` is a random subprefix assigned by the LBR.

Send data from the mote to the host

On the host:

1. start the netcat utility to listen for IPv6 packets sent to UDP port 61000:

```
nc -6lu 61000
```

On the LBR client:

1. navigate to your SmartMesh SDK installation directory
2. open the APIExplorer application

3. Connect the application to your SmartMesh IP Mote.
4. Follow the instructions in the [Internet Integration](#) tutorial to send following:
 - destination address (**destIP** field): `bbbb00000000000000000000000000000002`
 - destination port (**destPort** field): `61000`
 - packet payload: `706f69` (corresponding to the ASCII string "poi")

Clicking on the send button on the SmartMesh IP Mote side results in the netcat utility on the host printing "poi".

Send data from the host to the mote

On the LBR client:

1. Using the APIExplorer application, issue a `getParameter.macAddress` command to read the MAC address of the SmartMesh IP Mote. In our case `00170d0000123456`.
2. Determine the IPv6 address of your SmartMesh IP Mote by appending its MAC address to the SmartMesh IP network's prefix. In our case:
 1. the SmartMesh IP network's prefix is `aaaa00000000abcd`
 2. the MAC address of the SmartMesh IP Mote is `00170d0000123456`
 3. the IPv6 address of the SmartMesh IP Mote is hence `aaaa00000000abcd00170d0000123456`
3. Ensure that your SmartMesh IP Mote has a socket open and bound to UDP port `60000`.
4. Keep the APIExplorer application open.

On host:

1. use the netcat utility to send the string "poi" to your SmartMesh IP Mote, port `60000`.


```
nc -6u aaaa:0000:0000:abcd:0017:0d00:0012:3456 60000
poi<type Enter to send>
<type Ctrl+C to quit the utility>
```


Pressing enter in the `nc` utility on the host results in the SmartMesh IP Mote receiving the following `receive` notification:

- `socketId`: 3, the socket bound to UDP port `60000`
- `srcIP`: `bbbb000000000000000000000000000002`, the IPv6 address of the host computer
- `srcPort`: the (random) UDP port used by the `nc` application
- `payload`: `706f69`, the hexadecimal representation of the ASCII string "poi"


This concludes the installation and test of the Low-power Border Router.

9.6 User Guide

 This page details how to administer the LBR by managing users.

 This page assumes that you have installed an LBR. If this is not the case, please follow the [installation instructions](#).

9.6.1 Security Levels

 These security levels apply only to the session between the LBR and the LBR client, not to the end-to-end session between the SmartMesh IP Mote and the Internet host.

The following security levels are supported:

level	name	description
0	none	The session between the LBR and the LBR client is a TCP session. No authentication is required from the user. Guest accounts use this security level.
1	password	The session between the LBR and the LBR client is a TCP session. The user is authenticated by a password sent in the clear after the TCP session has been established.
2	ssl	The session between the LBR and the LBR client is a SSL (Secure Sockets Layer) session. The user and the LBR are authenticated by a secure handshake based on public and private keying material. The data transmitted between the LBR and LBR client is encrypted.

Use cases:

- Since it's the most secure, we recommend to use of security level 2 (`ssl`).
- Security level 1 (`password`) should only be used in cases where SSL is not an option.
- Security level 0 (`none`) should only be used for testing and guest accounts.

9.6.2 User Account Types

The LBR supports two types of user accounts.

Guest Accounts

When connecting to the LBR using a guest account, you get the same benefits as using a regular account, i.e.:

- your network is assigned an IPv6 prefix
- the SmartMesh IP Motes in your network can exchange data with hosts on the Internet

The restrictions of a guest account are:

- the connection between the LBR client and the LBR is not secure
- each time a user connects as a guest, the LBR assigns that network a different, random prefix

You connect to the LBR using a guest account by specifying the following parameters:

username	security level
guest	0 (none)



Multiple guests can connect to the LBR at the same time.

Regular Accounts

Using a regular user account requires the LBR administrator to add the corresponding user to the user database on the LBR.

A regular user account is identified by the following:

- a unique username
- a unique 2-byte subprefix
- a security level
- credentials for authenticating the user

The type of credentials depends on the user's security level:

- level 0: none
- level 1: a password
- level 2: a pair of public/private keys for the LBR client, and the public key of the LBR

9.6.3 Installing the LBR's Keying Material



This step is required before adding any user with security level 2 (ss1).

This step consists in generating an installing a public/private keys pair on the LBR.

The keying material needs to be formatted as "PEM" (see [RFC 1422](#)), which is a base-64 encoded form wrapped with a header line and a footer line. The step below use the OpenSSL module (commonly installed in all Linux distributions), but any equivalent method can be used.

1. On your LBR, navigate to the `lbr/bin/dustlbr/keys/` directory.
2. Type the following command:

```
openssl req -new -x509 -days 365 -nodes -out servercert.pem -keyout serverkey.pem
```

3. You will be asked a number of questions about your LBR. This information will be stored in the keying material, and visible by any user connecting to your LBR.
4. This command generates the two following files:
 - the LBR's private key in `serverkey.pem`
 - the LBR's public key in `servercert.pem` (a self-signed certificate)



These keys have a limited lifetime, set in the `-days` option above. Set that lifetime to a value appropriate for your application.

You can open these files to verify that they are indeed PEM formatted.

```
-----BEGIN CERTIFICATE-----
MIICsDCCAhmGAWIBAgIJAPMne+4vew7eMA0GCSqGSIb3DQEEBQUAMEUxCzAJBgNV
BAYTAKFVMRMwEQYDVQQIEWpTb21lLVN0YXRlMSEwHwYDVQQKEzhJbnRlcm5ldCBX
aWRnaXRzIFB0eSBMdGQwHhcNMTIwNDE2MTg1MDA2WhcNMTIwNDE2MTg1MDA2WjBF
MQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEhMB8GA1UEChMYSW50
ZXJhZGQwV21kZ210cyBQdHkgTHRkMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQDd/jVNUdfo9WeocR1kEcWxNvux3aZoPtFKfPqnfEBi5502GcPulrS6ZyT1FU4
oGLrGHE/nZJS5Zvt3I8E+mDmTVQLtouf7I9MAoFZPIwVGAnQ7u+x9q3U017f7nQ1
dYn+KRtLcZkevOBWBiD1B8ps5Pev36Swsx+FTiJ+GrcpkwIDAQABo4GnMIGkMB0G
A1UdDgQWBSO2j8DzCjYEy3R3+D8k75Ptbf0jB1BgNVHSMEbjBsgBSO2j8DzCjY
Ey3R3+D8k75Ptbf0qFJpEcwRTElMAkGALUEBhMCQVUxEzARBGNVBAgTClNvbWUu
U3RhdGUxITAfBgNVBAoTGE1udGVybmV0IFdpZGdpdHMgUHR5IEEx0ZIIJAPMne+4v
ew7eMAwGA1UdEwQFMAMBAf8wDQYJKoZIhvcNAQEFBQADgYEAvHT0PRtvCbESrLka
9omcDyfnS2su3ciaPgQFZHcp+QA1HjYmsvkcYYJhovZ9kqGMHsXuLFZKgiy+J2w5
Y70vJ210BxYM6sTFvdTP9/JkGRYIOFwiTfHXCFthv54YH3T8R892R2hGBaujl3cx
x9NS9GARXCJJ4Gy4KAca8TfK2Ho=
-----END CERTIFICATE-----
```



```

-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQDd/jvNuPdfo9WeocR1kEcWXnVux3aZoPtFKfPqnfEBi5502GcP
ulrS6ZyTlFU4oGLrGHE/nZJS5ZvT3I8E+mDmTVQLtouf7I9MAoFZPIwVGAnQ7u+x
9q3UO17f7nQldYn+KRtLcZkevOBWBiD1B8ps5Pev36SWsx+FTiJ+GrcpkwIDAQAB
AoGAdD3hkYIyXm0+taMFaX4UCz2JBmoBy25FRLE0HP15LpL6dTq/tLgpVdmn+Kyt
t0ocofgzjPMopKnAkA6lATlONTA9SDsUdwhMSMi5+7xM6SEGPV/OCVzn1eTxwGue
Q41ZD8okADXjqLY/vzEiYeDrmcx9FoULbQxWUPhT9yTO6VECQQDz3e2F5K3Sy/vt
ZIenXTKeWGqHyy30i2e5W0J8oOYJqPv5PjPbN1r7rNrKZBtdvyPcYQQHyv8rhZzp
WWYdZSCbAkEA6QmtiyCo8Y1LnSrPV7lYCxFl4WB4I9+6CxsxYsQeLPrz8dBoYtAX
bNdyIDENatjHETSeahftAavvnCsCIFn+aQJAB/HH5h/ABej9SQuIW8xudLgeqFPX
KGtOMrylWtgHBnOJ2eHL4K1Z+m70JbnDjNeunGRQtExJqcpNhVCTQgkvVwJAA990
S+6KACGJ7R2l/14tEVoDqGAS/v2buM2F349EtRiibxU4dtPgX8Wgtto5z9LKtAV8
0GR/YrS5sY2hZmo4aQJBAlIhoPi2zShLLM8N9/px/ljqs5M7ElAMJCGCy8Y5c1nv
SnhSy08IFoXquKJ1BAktBDv6Li4nX8tCCeaGsRWyRSM=
-----END RSA PRIVATE KEY-----


```

9.6.4 Adding Users

.Ibrauth LBR Authentication Files

When connecting to the LBR using the SmartMesh SDK's [LBRConnection](#) application, the user has to select a LBR Authentication File. The steps below instruct you how to build such a file for each type of user.

The LBR authentication file consists of a number of "key = value" pairs.

 You need to leave at least one space on each side of the equal sign (=) in your LBR authentication file. That is:

- "key=value" is wrong
- "key = value" is right

Guest User

Configure the LBR

No special configuration is required for the LBR to accept guest users.

Create the LBR Authentication File

The LBR authentication file to distribute to the client to use in the [LBRConnection](#) application is the following.

```
# the IPv4 address of the LBR
lbrAddr = <your LBR's IP address or DNS entry>

# the TCP port of the LBR
lbrPort = 80

# username
username = guest

# security level
secllevel = 0
```

Regular User with security level 0

This section indicates how to create the following user:

username	subprefix	security level	credentials
user_secllevel_0	ab00	0 (none)	<i>none</i>

Configure the LBR

On the LBR, enter the following commands to add the new user:

```
> add user_secllevel_0 ab00
```

By default, a new user is configured with security level 0, so no further configuration is needed.

Create the LBR Authentication File

The LBR authentication file to distribute to the client to use in the [LBRConnection](#) application is the following:

```
# the IPv4 address of the LBR
lbrAddr = <your LBR's IP address or DNS entry>

# the TCP port of the LBR
lbrPort = 80

# username
username = user_seclevel_0

# security level
seclevel = 0
```

Regular User with security level 1

This section indicates how to create the following user:

username	subprefix	security level	credentials	
user_seclevel_1	ab01	1 (password)	password	user_password

Configure the LBR

On the LBR, enter the following commands to add the new user:

```
> add user_seclevel_1 ab01
> passwordset user_seclevel_1 user_password
OK.
> seclevel user_seclevel_1 password
OK.
```

Create the LBR Authentication File

The LBR authentication file to distribute to the client to use in the [LBRConnection](#) application is the following:

```
# the IPv4 address of the LBR
lbrAddr = <your LBR's IP address or DNS entry>

# the TCP port of the LBR
lbrPort = 80

# username
username = user_seclevel_1

# security level
seclevel = 1

# password
password = user_password
```

Regular User with security level 2

This section indicates how to create the following user:

username	subprefix	security level	credentials	
user_seclevel_2	ab02	2 (ssl)	public key LBR	built from the <code>servercert.pem</code> file
			public key LBR client	generated through OpenSSL
			private key LBR client	generated through OpenSSL


Generate the user's public/private keys

The first step is to generate the strings representing the public/private key pair of the LBR client:

1. Type the following command on any machine with OpenSSL installed:

```
openssl req -new -x509 -days 365 -nodes -out clientcert.pem -keyout clientkey.pem
```

2. You will be asked a number of questions about your client. This information will be stored in the keying material, and visible by the LBR when the LBR client connects.
3. This command generates the two following files:
 - the LBR client's private key in `clientkey.pem`
 - the LBR client's public key in `clientcert.pem` (a self-signed certificate)

 These keys have a limited lifetime, set in the `-days` option above. Set that lifetime to a value appropriate for your application.

You can open these files to verify that they are indeed PEM formatted.

```
-----BEGIN CERTIFICATE-----
MIICsDCCAhmGAwIBAgIJJAOBABlW4RB1lMA0GCSqGSIb3DQEBBQUAMEUxCzAJBgNV
BAYTAKFVMRMwEQYDVQQIEwB1b21lLVN0YXRlMSEwHwYDVQQKEWhJbnRlcm5ldCBX
aWRnaXRzIFB0eSBMDGQwHhcNMTIwNDE2MTkzMzAyWhcNMTIwNDE2MTkzMzAyWjBF
MQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEhMB8GA1UEChMYSW50
ZXJ1ZXQgV2lkZ210cyBqdHkgTHRkMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQCtR2w+MFGgBpp5iBwzNxyUyhb8uoLsXMfrq9aofSgsYlHER0oP67m/qkD/0DMsO
xL0nrF9Q6m2oanLMGD6sLsi4LmGwCVHmEw24sz406VgV+Pbp6Xc4xJ8jzLOUMSgc
NUu2kj0j9Vx7TD11XRvgd1OY6FuKZXVxtA0LxOjYF3U89wIDAQABO4GnMIGkMBOG
AlUdDgQWBBQoU/QBawYk8hja8V641dZtpGmvKjB1BgNVHSMebjBsgBQoU/QBawYk
8hja8V641dZtpGmvKqFJpEcwRTElMAkGALUEBhMCQVUxEzARBGNVBAgTC1NvbWUt
U3RhdGUxITAFBgNVBAoTGE1udGVybmV0IFdpZGdpdHMgUHR5IEExOZlIJAOBABlW4
RB1lMAwGA1UdEwQFMAMBAf8wDQYJKoZIhvcNAQEFBQADgYEAXjpyhcAC/Z78qyJr
KRfiprn+/376gXi3xp92KJO2SexcFmq7tnucpmrpbrcdGhENy6YmxCcEk4fOFYMLA
udFlrfXlIEXqvlWggvxd+N5+UNvX1lxCrfil08Z7PaZKxb1tpNMTmI3i0NSz162
l+3tow9UXSKf37j3ldLgXmh7pCk=
-----END CERTIFICATE-----
```

```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQCtR2w+MFGgBpp5iBwzNxyUyhb8uoLsXMfrq9aofSgsYlHER0oP6
7m/qkD/0DMsOxL0nrF9Q6m2oanLMGD6sLsi4LmGwCVHmEw24sz406VgV+Pbp6Xc4
xJ8jzLOUMSgcNUu2kj0j9Vx7TD11XRvgd1OY6FuKZXVxtA0LxOjYF3U89wIDAQAB
AOGAK5QE0vcP8DD49IuYnADEW3AO74kYxFKbii/SYyAZ/kqGzTamXptMpi81BkmZ
X9N2xt2A8zah8XK7YPzP9jml3NIzShiZLNdrsdFCi jAueXmWH/ffCzihN1Uwsm8/
8DxCAOP763y/SebuCVWXXOm7JvbGNnh0teexsWN7RNabdYEQQDcA5nPC6vI7LBx
3DILzWG69BfWQuux62C6k4IdQGv3ye3dl3lCI5y3IKw+kECQFynigtVyLarjmUD
7YEJOa4hAkeAyZ7v/ayQk55bZkAg9JTGmKzK5UsxXhGT4npgj0/Sa4wHg8S3Q22
EkVh5hgWzZtqzJw/LLRN/diQvflKyYOYFwJAfgrODgvdSSFxwBL+1MYXjAwUr9ns
vyPyavDiRLHIAm9VJzcvL5XJTRw5sSng4utyQmKlBYysIcJU2pgwyUEzIQJBAMfh
LFTVXeMqq7vbuZafablvtZhPzDncOgAixf+CzpoX+HvKp7QsIqNMa3ibywd8m01L
XXEDqwoMR7pCQIE0V3MCQQDBrvQ8izzTmWRPXTh0s2iKNUmuuKsg0Gv1QWOjyqz1
hl5s+QQ/tp3VS8ITzWIkkiF8SJDj5KymYUftM41veh0
-----END RSA PRIVATE KEY-----
```

To be able to enter those keys in the LBR, or write in the LBR authentication file, you need to convert those files into a single string by removing:

- the line returns
- the heading and trailing lines

This results in the following strings which we have truncated for display purposes:

```
MIICsDCCAhmGAWIBAgIJAObABlW4RB1lMA0GCSqGSIB3DQEbbQUAMEUxCzAJBgNVBAYTAkFVMRMwEQYDVQ... <truncated>
```

```
MIICXQIBAAKBgQCtR2w+MFGgBpp5iBwzNxyYhB8uoLsXMfrq9aofSgsYlHER0oP67m/qkD/0DmsOxL0nrF... <truncated>
```

```
MIICsDCCAhmGAWIBAgIJAPMne+4vew7eMA0GCSqGSIB3DQEbbQUAMEUxCzAJBgNVBAYTAkFVMRMwEQYDVQ... <truncated>
```

Configure the LBR

To be able to authenticate the user, the LBR needs to know **only the public key** of the user. Input the same single line string which we have again truncated here for display purposes:

```
> add user_seclevel_2 ab02
> publickeyset user_seclevel_2 MIICsDCCAhmGAWIBAgIJAObABlW4RB1lMA0GCSqGSIB3DQEbbQU... <truncated>
> seclevel user_seclevel_2 ssl
OK.
```

Create the LBR Authentication File

The LBR authentication file needs to contain both the **public and private keys of the user**, and the **public key of the LBR**. Note that we have again truncated the long keys to display here:

```
# the IPv4 address of the LBR
lbrAddr = <your LBR's IP address or DNS entry>

# the TCP port of the LBR
lbrPort = 80

# username
username = user_seclevel_2

# security level
seclevel = 2

# Client's private key
clientprivatekey = MIICXQIBAAKBgQCtR2w+MFGgBpp5iBwzNxyYhB8uoLsXMfrq9aofSgsYl... <truncated>

# Client's public key
clientpublickey = MIICsDCCAhmGAWIBAgIJAObABlW4RB1lMA0GCSqGSIB3DQEbbQUAMEUxCz... <truncated>
# LBR's public key
lbrpublickey = MIICsDCCAhmGAWIBAgIJAPMne+4vew7eMA0GCSqGSIB3DQEbbQUAMEUxCzAJB... <truncated>
```

9.6.5 Managing Users

The LBR CLI allows you to manage the users while the LBR program is running.

✔ This section gives you an overview of how to use the CLI commands to manager users. Refer to the [LBR CLI guide](#) for details about each command.

Status

Use the `users` command to get an overview of the connected users:

```
> users
  user_seclevel_1 (ab01)      connected
  user_seclevel_0 (ab00)      disconnected
  user_seclevel_2 (ab02)      disconnected
Enter 'users <someName>' to see all details about one network
```

Specify a username to obtain detailed information about that user, including statistics:

```
> users user_seclevel_1
admin:
  name:          user_seclevel_1
  subprefix:     ab01
  loglevel:      debug
  virtualIfName: tun0
security:
  seclevel:      password
  password:      user_password
connection stats:
  status:        connected
  since:         Mon Apr 16 13:11:08 2012
  connectionTime: 5 min.
  lastIpAddr:    10.10.48.124
  lastPort:      2130
packet stats:
  from the Internet:
    packets:      0 pkts
    successful:    0 pkts
    failed:        0 pkts
      compression: 0 pkts
      too long:    0 pkts
  from the mesh:
    packets:      0 pkts
    successful:    0 pkts
    failed:        0 pkts
      decompression: 0 pkts
```

Logging

The LBR logs the activity of its core, as well as for each user in the `bin/dustlbr/logs/` folder:

- `system.log` contains logging information of the LBR core.
- `user_*.log` files contain logging information about a specific user.

```
2012-04-16 13:11:08,783 [ClientConnector:DEBUG] Listen for security capabilities of the client
2012-04-16 13:11:08,786 [ClientConnector:DEBUG] Client requested requestedSeclevel=1
2012-04-16 13:11:08,789 [ClientConnector:DEBUG] Send LBR's security capabilities
2012-04-16 13:11:08,792 [ClientConnector:DEBUG] Listen for username
2012-04-16 13:11:08,795 [ClientConnector:DEBUG] Send back username
2012-04-16 13:11:08,797 [ClientConnector:DEBUG] TCP session securing: password
2012-04-16 13:11:08,800 [ClientConnector:DEBUG] Listen for password
2012-04-16 13:11:08,803 [ClientConnector:INFO] user's prefix: aaaa:0000:0000:ab01
2012-04-16 13:11:08,806 [Lbrd:INFO] user user_seclevel_1 connected
2012-04-16 13:11:10,365 [LbrCli:DEBUG] Following command entered:users
2012-04-16 13:11:17,036 [LbrCli:DEBUG] Following command entered:users user_seclevel_1
2012-04-16 13:11:20,926 [BackupEngine:DEBUG] Backing up user DB
```


You can set the log level for each user using the `loglevel` command (see [LBR CLI guide](#)).

Disconnecting

You can force a user to disconnect from the LBR by using the `disconnect` CLI command.

Removing

You can remove a user from the user database by using the CLI `remove` command.


 This also removes all statistics associated with that user.

9.6.6 Backup and Recovery

All the user information is kept in a user database, which is periodically backed up to the `bin/dustlbr/userDB.pkl` file.

When starting, the LBR program will read that file and recover the user database last backed-up into that file. To force a backup (e.g. right after adding a user), use the CLI command `backup` (see the [LBR CLI guide](#)).

9.7 CLI Guide

 This section is a detailed description of the Command Line Interface (CLI) of the LBR, and serves as a reference.

The list of CLI commands:

9.7.1 add

Description

Add a user.

Syntax

add <username> <subprefix>

Parameters

Parameter	Description
username	The name of the new user
subprefix	The 2-byte subprefix of that client, expressed as exactly 4 hexadecimal characters, e.g. 0b12

Example

```
add myuser 0bc2
```

9.7.2 backup

Description

Backup the user database to the `userDB.pk1` file.

Syntax

```
backup
```

Parameters

Parameter	Description
-----------	-------------

Example

```
backup
```

9.7.3 disconnect

Description

Disconnect a user currently connected.

Syntax

```
disconnect <username>
```

Parameters

Parameter	Description
username	The username of the user to disconnect

Example

```
disconnect myuser
```

9.7.4 help

Description

Display the list of available commands.

Syntax

help

Parameters

Parameter	Description
-----------	-------------

Example

```
> help
```

Available commands:

add (a) - add a user

backup (b) - backs up the current user database in a file

disconnect (d) - disconnect a user

help (h) - print this menu

loglevel (ll) - sets the log level for a particular user

passwordremove (pr) - removes the password of a user

passwordset (ps) - sets the password of a user

publickeyremove (pkr) - removes the public key of a user

publickeyset (pks) - sets the public key of a user

quit (q) - quit this application

remove (r) - remove a user

secllevel (sl) - sets the security level of a user

status (s) - print the general status of the LBR

users (u) - status of all users, or details about one

version (v) - print the version of the LBR

Notes:

- type '<command> ?' to get the usage

9.7.5 loglevel

Description

Change the loglevel for a given user.

Syntax

```
loglevel <username> <loglevel>
```

Parameters

Parameter	Description
username	The username of interest, or <code>all</code> to apply to all users
loglevel	The loglevel to set; the options are <code>debug</code> , <code>info</code> , <code>warning</code> , <code>error</code> or <code>critical</code>

Example

9.7.6 passwordremove

Description

Remove the password associated with a user.

Syntax

```
passwordremove <username>
```

Parameters

Parameter	Description
username	The username of interest

Example

9.7.7 passwordset

Description

Sets the password of a user.

Syntax

```
passwordset <username> <password>
```

Parameters

Parameter	Description
username	The username of interest
password	The password to set

Example

9.7.8 publickeyremove

Description

Remove the public key associated with a user.

Syntax

```
publickeyremove <username>
```

Parameters

Parameter	Description
username	The username of interest

Example

9.7.9 publickeyset

Description

Set the public key associated with a user.

Syntax

```
publickeyset <username> <public_key>
```

Parameters

Parameter	Description
username	The username of interest
public_key	The public key to set

Example

9.7.10 quit

Description

Exit the LBR application.

Syntax

quit

Parameters

Parameter	Description
-----------	-------------

Example

9.7.11 remove

Description

Remove a user from the user database.

Syntax

```
remove <username>
```

Parameters

Parameter	Description
username	The username of interest

Example

9.7.12 seclevel

Description

Set the security level associated with a user.

Syntax

```
seclevel <username> <level>
```

Parameters

Parameter	Description
username	The username of interest
level	The security level to set. Acceptable values are none, password or ssl

Example

9.7.13 status

Description

Print the status of the LBR.

Syntax

status

Parameters

Parameter	Description
-----------	-------------

Example

9.7.14 users

Description

Print an overview of which users are connected. The command `users <name>` prints details about a specific user.

Syntax

`users [username]`

Parameters

Parameter	Description
username	The name of a user

Example

9.7.15 version

Description

Prints the version of the LBR.

Syntax

version

Parameters

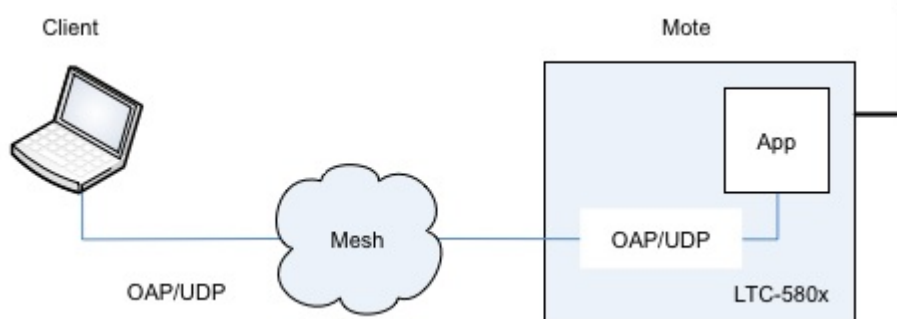
Parameter	Description
-----------	-------------

Example

10 On-chip Application Protocol


This section describes the On-chip Application Protocol (OAP). This RESTful-style protocol was developed to demonstrate how an external client can connect to, and exchange packets with applications that run on the mote. OAP runs over UDP and is conceptually similar to CoAP. It is a session-oriented protocol that has options for acknowledged and unacknowledged packets. Only one client at a time (see figure) may use the session to send requests to mote. Acknowledged traffic may only flow from client to mote. Unacknowledged traffic may only flow from mote to one or more clients.

OAP runs over UDP, port 0xF0B9 - this is in the most compressible 6LoWPAN port range to maximize the available payload. Customers are free to use OAP in their own Applications, provided they fully implement the protocol -- OAP is not part of the networking stack.



The documentation is broken-up into the following sections:

- [Protocol](#) - information about connections and general data framing
- [OAP in SmartMesh Motes](#) - information about various OAP modules in SmartMesh mote applications
- [Examples](#) - examples of commonly-used OAP exchanges.

 The mote must be in **master** mode in order to terminate OAP commands. This is the default mode for motes shipped in the starter kits ([DC9000](#) and [DC9007](#)). For lowest power operation, termination of unused digital inputs (D0-D3) when operating in **master** mode is recommended.

10.1 Protocol

10.1.1 Packet Format

OAP packets all have a common format that includes 2-byte header followed by variable length payload:

Control	Id	Payload
---------	----	---------

1 byte	1 byte	0-n bytes
--------	--------	-----------

Control field

The Control field includes the following subfields:

Field	Bit Position	Description
Transport	0	0=unacknowledged, 1=acknowledged
Response	1	0=request, 1=response
Sync	2	0=normal, 1=resync connection
Reserved	3-7	Reserved, set to 0

Id field

The ID field includes the following subfields:

Field	Bit Position	Description
Sequence number	0-3	Packet sequence number
Session ID	4-7	Session ID

10.1.2 Communication

OAP supports two traffic patterns:

- reliable request/response communication from client to the mote and
- best-effort notifications from mote to client

To start communicating with request/response traffic, the client must establish connection to the mote. OAP utilizes sequence numbers to guarantee ordering and reliable delivery. The initial sequence number is established during the first handshake and is incremented with every new request/response pair. Mote's responses always go to the source IP address and port of the request packet. To save on communication time and bandwidth, the initial handshake may include application payload.

Client: Initiating new connection with Sync request

To establish communication, the client must send a Sync request, designated by Control byte set to (*Transport=1, Response=0, Sync=1, Sequence=random #, and Session ID=0*). A valid response will have the Control byte set to (*Transport=1, Response=1, Sync=1*) and will contain the echoed sequence number. If the Sync request contains request payload, the response will also contain corresponding response payload. If the client receives a valid response, it should consider the connection established. The Session ID from the response should be included in all follow-on request packets sent to the mote and should be checked for in all response packets. Sequence number used in the exchange should be recorded as *last_sequence_number* – it will be incremented in follow-on requests.

Note: processing new Sync requests

To understand the protocol it may be helpful to understand how the mote handles incoming Sync packets. When the mote application starts, it chooses a random *current_session_id* and starts listening on UDP port associated with OAP. When it receives a packet with the Control byte set to (*Reliable=1, Sync=1, Response=0*) it goes through the following steps:

1. Increments *current_session_id*
2. Stores sequence number from the received packet in *last_sequence_number*
3. If any payload is included in the packet, processes it and prepares response packet with payload. Otherwise, the response packet has no payload.
4. Sends the response packet that has Control byte set to (*Reliable=1, Sync=1, Response=1, Sequence=last_sequence_number*) and Session ID set to *current_session_id*

Using the connection

Every request/response is associated with a new sequence number, and both client and mote must enforce a strict order of packets within a connection. A packet with an assigned sequence number must be acknowledged before a new sequence number is used. To send a new request packet, the client should do the following:

1. Increment *last_sequence_number*
2. Set Control to (*Transport=Acknowledged, Sync=0, Response=0*), *Sequence=last_sequence_number* and Session ID to *current_session_id* and send the packet
3. Wait for a reply. A valid reply must have Control byte set to (*Transport=Acknowledged, Sync=0, Response=1*), *Sequence=last_sequence_number*, and correct session ID.
4. If no reply is received, a copy of the request must be resent or connection should be considered failed. The timeout that the client should use depends on network and topology and is outside the scope of this document. We recommend at least 3 retries.

Mote receive logic for non-Sync packets is outlined below. It may be useful for understanding the protocol:

- If received packet contains an unknown session ID, the mote drops it.
- If the packet contains sequence number that is *last_sequence_number+1* (i.e. next expected packet), the mote sends a response and updates its copy of *last_sequence_number*. A copy of the response payload is cached.
- If the packet contains a sequence number equal to *last_sequence_number* (i.e. a duplicate), the mote sends back a cached copy of the response without processing the payload.
- If the packet contains any other sequence number, the mote drops it.

i For simplicity reasons, OAP protocol is designed to operate with one client at a time. If more than one client attempts to establish connection to a mote, the last one will take precedence. On the other hand, if strict ordering of request/response traffic is not required, or if multiple clients may be sending packets to a mote in parallel, establishing and maintaining a connection may not be necessary or appropriate. In such cases, the client(s) may set the Sync bit with every request packet and mote will treat it as a new connection with a single request/response exchange.

Terminating connection

There's no explicit termination of a connection. Whenever the client does not wish to continue communicating, it may stop sending packets. Note that the mote has no awareness of this, and the application will continue operating normally.

Unacknowledged Notifications from mote

Unacknowledged traffic (i.e. *Transport=Unacknowledged*, *Response=0*) may only go in one direction: Mote to Client. These packet types are used to send Notifications. The destination IP address and port must be configured on the application level before packets may flow. The Mote drops any unacknowledged traffic directed to it.

10.1.3 OAP Payload

In general, data sent inside the OAP Payload field depends on the capabilities supported by mote's application. However, all payloads in OAP have a common structure.

The request (Control includes *Transport=1*, *Response=0*) packet payload contains a Command id, followed by an address and a list of variables being accessed at that address and encoded in Tag-Length-Value format. The meaning of address and variables is explained later in this section.

Command	<Address>[<Var><Var><Var>...]
1 byte	0-n bytes

The response (Control includes *Transport=1*, *Response=1*) packet payload contains the same Command value as the request, and also carries Return Code (RC) in the header. The address and a list of variables that constitute the response follows.

Command	RC	<Address>[<Var><Var><Var>...]
1 byte	1 byte	0-n bytes

Notifications (Control includes *Transport=0*) contain one Command byte (=Notification) followed by application-specific payload:

Command	Notification Data
1 byte	0-n bytes

Command field

The Command field identifies how payload packet should be processed. The following commands are defined in OAP:

Command	Value	Sent by	Description
GET	0x01	Client	Retrieve values of one or more variable from the mote
PUT	0x02	Client	Update the values of one or more variables on the mote
POST	0x03	Client	Create an an application object with indicated values of variables
DELETE	0x04	Client	Delete an object specified by address and variable values
NOTIFICATION	0x05	Mote	Various mote notifications

Behavior of each command depends on the application and the payload contents, but follows general principles of [RESTful](#) architecture.

Return Code (RC) field

Return Code field indicates the result of processing an OAP request payload.

Return Code	Id	Description
OK	0	Request succeeded
RC_NOT_FOUND	1	Object not found
RC_NO_RESOURCES	2	No sufficient resources to complete request
RC_UNK_PARAM	3	Unknown parameter
RC_INV_VALUE	4	Invalid value
RC_INV_ADDR	5	Invalid address
RC_NO_SUPPORT	6	Not supported
RC_RD_ONLY	7	Variable is read-only
RC_WR_ONLY	8	Variable is write-only
RC_FEWER_BYTES	9	Fewer bytes than expected

RC_TOO_MANY_BYTES	10	More bytes than expected
RC_UNK_ERROR	11	Unknown error
RC_EXEC_SIZE	12	Command can't be executed



These OAP return codes should not be confused with the return codes returned by API commands.

10.1.4 Tag-Length-Value(TLV) Encoding

Application payload contains a variable-length list of Tag-Length-Value fields. The general format of TLV field is

Tag	Length	Value
1 byte	1 byte	<i>Length</i> bytes

- Tag is a numeric code which uniquely identifies the item represented by this encoding for a given context. A tag of 0xFF has a special meaning and denotes an encoded address.
- Len the size of the value field in bytes.
- Value is a variable sized set of bytes which contains data for this part of the message.

Data representation of values

Values encoded in TLV format have the following format:

Integers

Integer values are sent in big-endian order. Uint8 and int8 values are one byte. UINT16 and INT16 are 2 bytes, and UINT32 and INT32 are 4 bytes.

Character strings

All character strings are sent terminated with a '\0' byte ("null terminated")

Byte strings

Byte strings are sent as a stream of bytes

Fractional numbers

Fractional numbers are sent in fixed point notation. The length and point position is defined by the application.

Address representation

OAP addresses are ordered sets of one-byte numbers that identify location of various application variables. For example 3/2/1/6 is an address, and so is 1/1/1/25/1. The actual meaning of addresses is only meaningful in a context of an application.

Addresses are encoded in TLV format using a special tag value of 0xFF. The value part of the TLV is a byte string corresponding to the address, with each byte treated as separate number.

For example, an address of 1/4/0 would be represented by:

Tag = 0xFF, Length = 3, Value = 0x01, 0x04, 0x00

10.1.5 Using OAP to interact with an application

An application that supports OAP must map the variables that can be controlled and queried into a logical hierarchy, much like a RESTful HTTP-based APIs for many common websites arrange resources. It's easy to think about the hierarchy using the familiar URL notation.

Consider a simple device with 2 gpio pins and one analog channel. The application may represent these as 2 groups of variables: gpio (id=0) and adc (id=1). Resource gpio/0 (address 0/0) would refer to the first pin, gpio/1 (address 0/1) to the second, and adc/0 (address 1/0) to the one and only analog channel.

To allow querying and configuring of the device, we need variables for each of these groups.

For gpio, the variables could be the following (x below can take values of 0 or 1, depending on gpio):

Variable	Id	Address	Description
direction	0	0/x/0	0=input, 1=output
outval	1	0/x/1	0/1 for output pins
inval	2	0/x/2	0/1 for input pins

For adc, we could have the following variables

Variable	Id	Address	Description
enable	0	1/0/0	0=disabled, 1=enable
voltage	1	1/0/1	value read from the channel, in 8.8 fixed point notation (2 bytes)

Let's look at some examples of OAP payloads:

GET gpio/0 variables – Retrieve values of all variable for gpio/0 pin

We encode address of gpio/0 as a TLV with a special 'address' tag of 0xff, len=2, and value of 0x00 0x00.

Contents of the payload field in OAP request will be:

Command	Address
01	FF 02 00 00

The expected response payload includes all variables for gpio/0, i.e. direction(id=0), outval(id=1) and inval(id=2):

Command	RC	Address	Variable1	Variable2	Variable3
01	00	FF 02 00 00	00 01 01	01 01 01	02 01 00

PUT gpio/1 outval=1 – set output level of gpio/1 pin to high

We encode address of gpio/1 as address with tag 0xff, len=2 and value of 0x00 0x01. The variable outval has a tag of 0x01, length of 1, and value of 0x1.

Contents of the payload field in OAP request:

Command	Address	Variable1
02	FF 02 00 01	00 01 01

The expected response is:

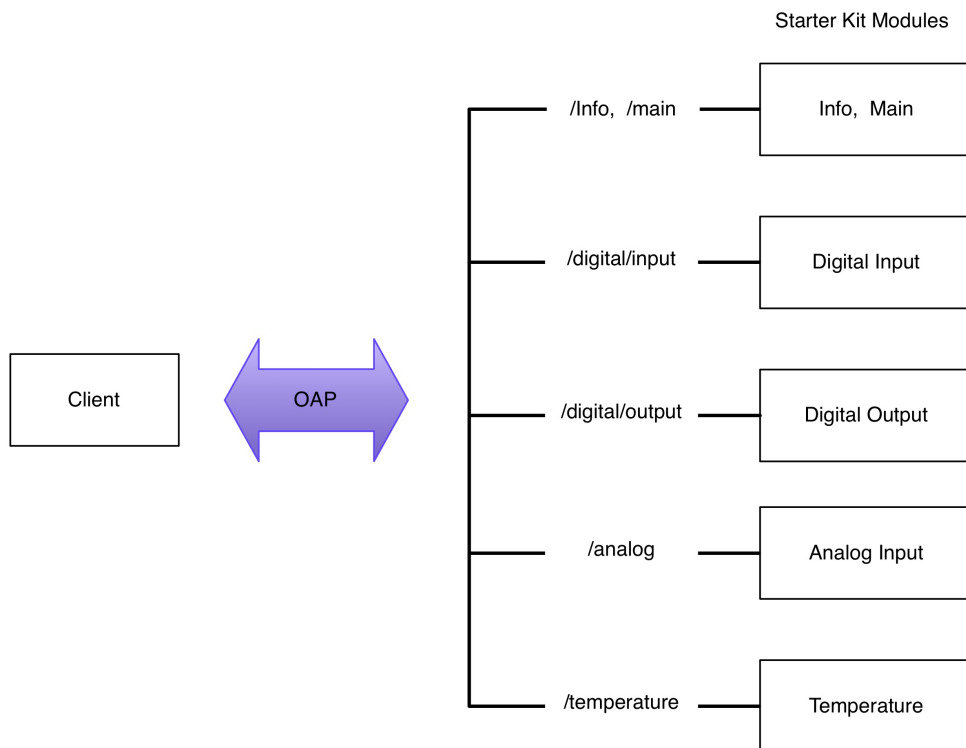
Command	RC	Address	Variable1
02	00	FF 02 00 01	00 01 01

Note that the variable that was set and the value it is set to is echoed in the response.

10.2 OAP in SmartMesh Motes

10.2.1 Introduction

The motes found in the SmartMesh IP Starter Kit ([DC9000](#)) and SmartMesh WirelessHART Starter Kit ([DC9007](#)) contain an application that, when operating in Master mode, demonstrates the capabilities of the LTC5800 platform. It includes several independent modules that may be accessed and controlled, either through Stargazer (for IP) or applications in the SmartMesh SDK.



The following table lists application modules implemented in each platform:

	Description	Support in SmartMesh IP	Support in SmartMesh WH
/info	Information	Yes	Yes
/main	Main	Yes	Yes
/digital/input	Digital Input	Yes	No
/digital/output	Digital Output	Yes	No
/analog	Analog Input	Yes	No
/temperature	Temperature	Yes	Yes
/pkgen	Packet Generator	Yes	Yes

Addressable Elements and Pinout

The addressable elements in the application, along with their element address identifiers (in parentheses) are listed here. The address for a particular pin is appended to its functional group address, so digital out D4 is addressed at /digital_out/D4 or 3/0.

- info (0)
- main (1)
- digital_in (2)
 - D0 (0)
 - D1 (1)
 - D2 (2)
 - D3 (3)
- digital_out (3)
 - D4 (0)
 - D5 (1)
 - INDICATOR_0 LED (2)
- analog (4)
 - A0 (0)
 - A1 (1)
 - A2 (2)
 - A3 (3)
- temperature (5)
- pkgen (254)

Signal Name	Address	LTC5800 Pin Name	LTC5800 Pin #	DC9003 Pin name	LTP5901/2 Pin Name	LTP5901/2 Pin #
D0	2/0	DP2	34	DP2	DP2 / GPIO21	26
D1	2/1	SPIS_MOSI	51	S_MOSI	GPIO26 / SPIS_MOSI	48
D2	2/2	IPCS_SSn	45	I_SSn	IPCS_SSn / GPIO3	39
D3	2/3	IPCS_SCK	44	I_SCK	IPCS_SCK / GPIO4	36
D4	3/0	IPCS_MOSI	42	I_MOSI	IPCS_MOSI / GPIO5	35
D5	3/1	IPCS_MISO	40	I_MISO	IPCS_MISO / GPIO6	33
Indicator	3/2	DP3	33	INDICATOR_0	DP3	25
A0	4/0	AI_0	15	AI_0	AI_0	10
A1	4/1	AI_1	16	AI_1	AI_1	8
A2	4/2	AI_2	18	AI_2	AI_2	7
A3	4/3	AI_3	17	AI_3	AI_3	9

/info

The **info** element contains information identifying the application. All devices implementing OAP should support variables in this element.

Variable	ID	Type	Access	Description
swRevMaj	0	INT8U	R	Major software revision of the application
swRevMin	1	INT8U	R	Minor software revision of the application
swRevPatch	2	INT8U	R	Patch number of the application
swRevBuild	3	INT16U	R	Build number of the application
appld	4	INT16U	R	Unique application ID. 0x0000-0x7FFF : reserved 0x8000-0xFFFF: available for use by customers SmartMesh Starter Kits = 0x0001
resetCounter	5	INT32U	R	Same as mote join counter.
changeCounter	6	INT32U	R	Number of configuration changes since last reset. Combination of resetCounter and changeCounter enables clients to determine if any changes occurred since last access.

/main

The **main** element contains addressing information such that the application can be configured to direct its data to a particular address and port. These should not be changed in the Starter Kit or the mote will not correctly interact with [Stargazer](#) or the [SmartMesh SDK](#).

Variable	ID	Type	Access	Default	Description
destAddr	0	INT8U[16]	r/w	FF02::02	IP address to which the mote should send data packets
destPort	1	INT16U	r/w	F0B9	UDP port to which the mote should send data packets

/digital_in

This element contains variables to access the digital input module. Individual pins may be accessed by appending a pin ID to the base address, e.g. /digital_in/2 is D2.

Variable	ID	Type	Access	Default	Description
enable	0	INT8U	r/w	disabled	0=disabled, 1=enabled
rate	1	INT32U	r/w	10,000	Sample rate, in milliseconds. Valid only for dataFormat is 'all' Min: 1000ms Max: 300,000ms
sampleCount	2	INT16U	r/w	1	Number of samples to accumulate in packet when dataFormat is 'all' If number specified does not fit in a packet, the mote will send packet when full.
dataFormat	3	INT8U	r/w	all	0='all' – accumulate and send all samples 1='on-change' – send packet if value changes; for rapid changes, updates will be limited to once a second. 2='on-high' - send packet if change from low to high detected; for rapid changes, updates will be limited to once a second. 3='on-low' - send packet if change from high to low is detected; for rapid changes, updates will be limited to once a second .
value	4	INT8U	r	n/a	Returns the value of the pin at time of read.

/digital_out

This element contains all variables to access the digital output module. Individual pins may be accessed by appending a pin ID to the base address, e.g. /digital_out/1 is D5.

Variable	ID	Type	Access	Default	Description
value	0	INT8U	w	n/a	Set the pin to desired value; 0=set pin to 0 1=set pin to 1 2=toggle pin once a second; Only valid for status LEDs

/analog

This element contains variables to access the ADC module. Individual channels may be accessed by appending an ID to the base address, e.g. /analog/0 is A0.

Variable	ID	Type	Access	Default	Description
enable	0	INT8U	r/w	disabled	0=disabled, 1=enabled
rate	1	INT32U	r/w	10,000	Sample rate, in milliseconds. Min: 1000ms Max: 300,000ms
sampleCount	2	INT8U	r/w	1	Indicates number of samples to accumulate in packet when dataFormat is 'all'. If number specified does not fit in a packet, the mote will send packet when full. Indicates number of samples to use for stats aggregation when dataFormat is 'stats'
dataFormat	3	INT8U	r/w	all	0='all' – accumulate and send all samples 1='stats' – send min/max/ave for each <i>sampleCount</i> samples
value	4	INT16U	r	n/a	Return the value of the channel at time of read, in mV

/temperature

This element contains variables for access to the internal temperature sensor module

Variable	ID	Type	Access	Default	Description
enable	0	INT8U	r/w	enabled	0=disabled, 1=enabled
rate	1	INT32U	r/w	30,000	Sample rate, in milliseconds. Min: 1000ms Max: 300,000ms
sampleCount	2	INT8U	r/w	1	Indicates number of samples to accumulate in packet when dataFormat is 'all'. If number specified does not fit in a packet, the mote will send packet when full. Indicates number of samples to use for stats aggregation when dataFormat is 'stats'
dataFormat	3	INT8U	r/w	all	0='all' – accumulate and send all samples 1='stats' – send min/max/ave for each <i>sampleCount</i> samples

Value	4	INT16S	r	n/a	Return current temperature, in 1/100ths of a °C (reported in °C in mote version <1.3)
-------	---	--------	---	-----	---

/pkgen

The packet generator module is used to send a specific number of packets (numPackets) at a specific rate (rate) with a specific packet size (packetSize). StartPID is used in the pkgen notification. Pkgen will start to generate notifications (type=pkgen) when it receives a PUT command. The client needs to specify all the variables since pkgen will not save them. Finally, echo will be used as an auto-increment variable. The client will use PUT/echo=value to start. Then every time the client calls GET/echo the value of echo will increment by 1. If the mote resets, pkgen will NOT restart sending notifications. To stop pkgen from sending notifications, send a PUT request with numPackets=0.

Variable	Id	Type	Access	Default	Description
echo	0	INT32U	r/w	0	Reply with echo received
numPackets	1	INT32U	w	0	Number of packets to send
rate	2	INT32U	w	10,000	Rate to generate packets in milliseconds.
packetSize	3	INT8U	w	80	Size of the packet to send.
startPID	4	INT32U	w	0	Starting Packet Id


10.2.2 Notifications

Sample/Report notification

Field	Type	Description
type	INT8U	0 = raw samples
channel	TLV	Source of data in address format, e.g. /digital_in/1
timestamp	UTC_TIME_L	Timestamp of first sample in this report
rate	INT32U	Time between samples, in milliseconds
numSamples	INT8U	Number of samples in this packet
sampleSize	INT8U	Size of each sample, in bits
samples[]	Bits	Samples, bit-aligned

Stats report (min/max/ave)

Field	Type	Description
type	INT8U	1 = stats report
channel	TLV	Source of data in address format, e.g. /analog/1
timestamp	UTC	Timestamp of start of stats window
rate	INT32U	Time between samples, in milliseconds
numSamples	INT8U	Window for stats collection
sampleSize	INT8U	sampleSize in bits (i.e. size of each metric)
stats	Bits	Min, Max, Ave (each of sampleSize)

 Analog channel values are sent in units of mVolts
 Temperature values are sent in units of 100th of a °C

Digital change notification

This notification is sent out on digital IO change

Field	Type	Description
type	INT8U	2 = digital change notification
Channel	TLV	Source channel
Timestamp	UTC	Timestamp when the change was detected
New value	INT8U	New value (0 or 1)

PkGen notification

This notification is sent when PkGen generates packets.

Field	Type	Description
type	INT8U	4 = PkGen notification
Channel	TLV	Source channel
Pid	INT32U	PacketId (Starting at startPID and incrementing)
StartPID	INT32U	StartPID
numPackets	INT32U	NumPackets
Payload	TLV	Should be the size set in packetSize and look like 00010203...

10.3 OAP Examples

These example encodings apply to the application found in the mote, as described in the [modules](#) section.

10.3.1 Turning on the INDICATOR_0 LED

Sending this packet to the mote will turn on it's indicator LED

OAP Header	Command ID	Address	Payload (Variable TLVs)
05 00	02	FF 02 03 02	00 01 01

OAP Header

- Control: 05 (acknowledged request, resync=1 used to establish a connection)
- ID: 00 (sequence = 0, session = 0)

OAP Payload

- Command: 02 (PUT)
- Address
 - Address TLV with length 2
 - 03 02 is /digital_out/INDICATOR_0
- Tag: 00 (the Value variable)
- Length: 01
- Value: 01 (set pin to 1)

10.3.2 Temperature Sample Notification

This shows temperature notifications coming from mote that is enabled to sample temperature.

OAP Header	Command ID	Notification type	Address	Timestamp	Rate	# samples	Sample size	Sample(s)
00 03	05	00	FF 01 05	00 00 00 00 53 16 60 93 00 04 e5 77	00 00 13 88	01	10	0ae0

OAP Header

- Control: 00 (unacknowledged request, normal sync)
- ID: 03 (sequence = 3, session = 0)

OAP Payload (Notification)

- Command: 05 (notification)
- Type: 00 (raw samples)
- Address: FF 01 05 (tag, length, value=5, i.e. temperature)
- UTC Timestamp: 00 00 00 00 53 16 60 93, 00 04 e5 77 (seconds into epoch, microseconds)
- Rate: 00 00 13 88 (5000 milliseconds)
- Number of samples: 01
- Sample size: 10 (16 bits)
- Samples: 0a e0 (2784 100ths of a °C)

10.3.3 Getting app info

OAP Header	Command ID	Address
05 00	01	FF 01 00

OAP Header

- Control: 05 (acknowledged request, resync=1 used to establish a connection)
- ID: 00 (sequence = 0, session = 0)

OAP Payload

- Command: 01 (GET)
- Address
 - Address TLV with length 1
 - 00 is info

Response Payload

OAP Header


- Control: 05 (acknowledged request, resync=1 used to establish a connection)
- ID: 00 (sequence = 0, session = x)
- RC: 00

OAP Payload

- Command: 01 (GET)
- Address
 - Address TLV with length 1
 - 00 is info
- swRevMajor, swRevMin, swRevPatch, swRevBuild, appld, resetCounter and changeCounter values

OAP Header	Address	swRevMajor	swRevMin	swRevPatch	swRevBuild	appld	resetCounter	changeCounter
07 x0 00	FF 01 00	00 01 01	01 01 00	02 01 10	03 01 03	04 02 0001	05 04 00000021	06 04 00000003

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

This documentation is protected by United States and international copyright and other intellectual and industrial property laws. It is solely owned by Linear Technology and its licensors and is distributed under a restrictive license. This product, or any portion thereof, may not be used, copied, modified, reverse assembled, reverse compiled, reverse engineered, distributed, or redistributed in any form by any means without the prior written authorization of Linear Technology.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g) (2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015 (b)(6/95) and DFAR 227.7202-3(a), and any and all similar and successor legislation and regulation.

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.

Linear Technology does not warrant or represent that any license, either express or implied, is granted under any Linear Technology patent right, copyright, mask work right, or other Linear Technology intellectual property right relating to any combination, machine, or process in which Linear Technology products or services are used. Information published by Linear Technology regarding third-party products or services does not constitute a license from Linear Technology to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from Linear Technology under the patents or other intellectual property of Linear Technology.

Dust Networks, Inc is a wholly owned subsidiary of Linear Technology Corporation.

© Linear Technology Corp. 2012-2014 All Rights Reserved.