SmartMesh IP Tools Guide

Table of Contents

1 About This Guide 5
1.1 Related Documents 5
1.2 Conventions Used 7
1.3 Revision History 8
2 Introduction 9
3 Installation 10
3.1 Setup 10
3.1.1 System Overview 10
3.1.2 Step 1 - Prepare the Hardware 11
3.1.3 Step 2a - Installing FTDI Serial Drivers 13
3.1.4 Step 2b - Installing Serial Mux 15
3.1.5 Step 2d - Installing Stargazer 17
3.2 Troubleshooting 19
3.2.1 Windows FTDI Driver installation 19
3.2.2 Linux FTDI Driver installation 24
3.2.3 Macintosh OS X FTDI Driver Installation 25
3.2.4 Trouble Connecting to Manager 26
3.2.5 Not Getting Notifications 27
3.2.6 Changing Network ID 28
3.2.7 Master/Slave 29
4 Serial Terminal Client 31
4.1 TeraTerm 31
4.2 PuTTY 31
4.3 minicom 32
4.4 Microsoft Windows HyperTerminal 32
5 Serial API Multiplexer (Serial Mux) 34
5.1 Overview 34
5.2 Serial Mux Configuration 35
5.2.1 Step 1: Edit the Serial Mux Configuration File 35
5.2.2 Step 2: Restart the Serial Mux Windows Service 37
5.2.3 Advanced Serial Mux Configuration 38
5.2.4 Serial Mux with Multiple Managers 39
5.3 Serial Mux Protocol 42
5.3.1 Basic Operation 42
5.3.2 Protocol 43
5.3.3 Connections 44
5.3.4 Info command 45
5.3.5 Subscriptions and Notifications 45
5.3.6 Disconnection 45

5.3.7 Serial Mux Definitions 46
6 Stargazer GUI 47
6.1 Upgrading Stargazer 47
6.1.1 Remove the Existing Application 47
6.2 Using Stargazer 49
6.2.1 Overview 49
6.2.2 Managing the Network 54
7 Interacting with a Network 73
7.1 Introduction 73
7.2 A First Network 73
7.2.1 Overview 73
7.2.2 Common Problems 78
7.3 Interacting with the Manager 80
7.3.1 Introduction 80
7.3.2 Common Problems 90
7.4 Interacting with a Mote 91
7.4.1 Overview 91
7.4.2 Common Problems 115
7.5 Advanced Topics 116
7.5.1 Exercise the API Programmatically 116
7.5.2 Log HDLC Frames 123
7.5.3 Upstream Communication 125
7.5.4 Downstream Communication 130
7.5.5 Internet Integration 135
8 Low-power Border Router 143
8.1 What is an LBR? 143
8.2 Documentation Organization 143
8.3 Overview 144
8.3.1 Goals of an LBR 144
8.3.2 Services 144
8.4 Installation 150
8.4.1 Requirements 150
8.4.2 Installation Steps 151
8.5 User Guide 156
8.5.1 Security Levels 156
8.5.2 User Account Types 156
8.5.3 Installing the LBR's Keying Material 157
8.5.4 Adding Users 159
8.5.5 Managing Users 165
8.5.6 Backup and Recovery 166
8.6 CLI Guide 167
8.6.1 add 167
8.6.2 backup 168

8.6.3 disconnect 169
8.6.4 help 170
8.6.5 loglevel 171
8.6.6 passwordremove 172
8.6.7 passwordset 173
8.6.8 publickeyremove 174
8.6.9 publickeyset 175
8.6.10 quit 176
8.6.11 remove 177
8.6.12 seclevel 178
8.6.13 status 179
8.6.14 users 180
8.6.15 version 181
9 On-chip Application Protocol 182
9.1 Protocol 182
9.1.1 Packet Format 182
9.1.2 Communication 183
9.1.3 OAP Payload 185
9.1.4 Tag-Length-Value(TLV) Encoding 187
9.1.5 Using OAP to interact with an application 188
9.2 OAP in SmartMesh Motes 190
9.2.1 Introduction 190
9.2.2 Notifications 198
9.3 OAP Examples 200
9.3.1 Turning on the INDICATOR_0 LED 200
9.3.2 Temperature Sample Notification 200
9.3.3 Digital Input Change Notification 201
9.3.4 Get app info 201
9.3.5 Get the settings of a Digital Input 202
9.3.6 Enable a Digital Input 203
9.3.7 Configure a Digital Input to report on change 204
10 Logging 205
10.1 Using the Logging Capabilities 205
10.2 Logfile format 205
10.3 Example logfile 205
10.3.1 Connecting to the manager 205
10.3.2 Issues the getNetworkInfo command 207
10.3.3 Disconnect 208
10.4 Implementation details 208
10.4.1 log statement in source code 209
10.4.2 log configuration file 209
10.5 Modifying logging in you application 210

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
6	11/06/2015	Moved SMSDK to Dustcloud.org
7	12/03/2015	Added SMSDK logging description
8	01/29/2016	Fixed lengths in Serial Mux examples
9	11/07/2016	Updated to support all manager options

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 connect to a mote or manager over a USB-to-serial link. Next, the user can interact with the Command Line Interface (CLI) of the mote and manager via a serial terminal client. Programmatic access to the mote and manager is done through the Application Programming Interface (API). A Serial Multiplexer (Mux) allows for multiple concurrent connections to the manager API - several reference/demo tools make use of this API:

- SmartMesh SDK (SMSDK) - a Python-based set of tools used to demonstrate various aspects of the mote and manager 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. Installation instructions and detailed descriptions of the sample apps found in the SMSDK can be found on the Dustcloud.org user portal.
- Stargazer GUI - visualize and interact with the network.
- Low-power Border Router (LBR) - send and receive data from an IPv6 host on your LAN or the internet.

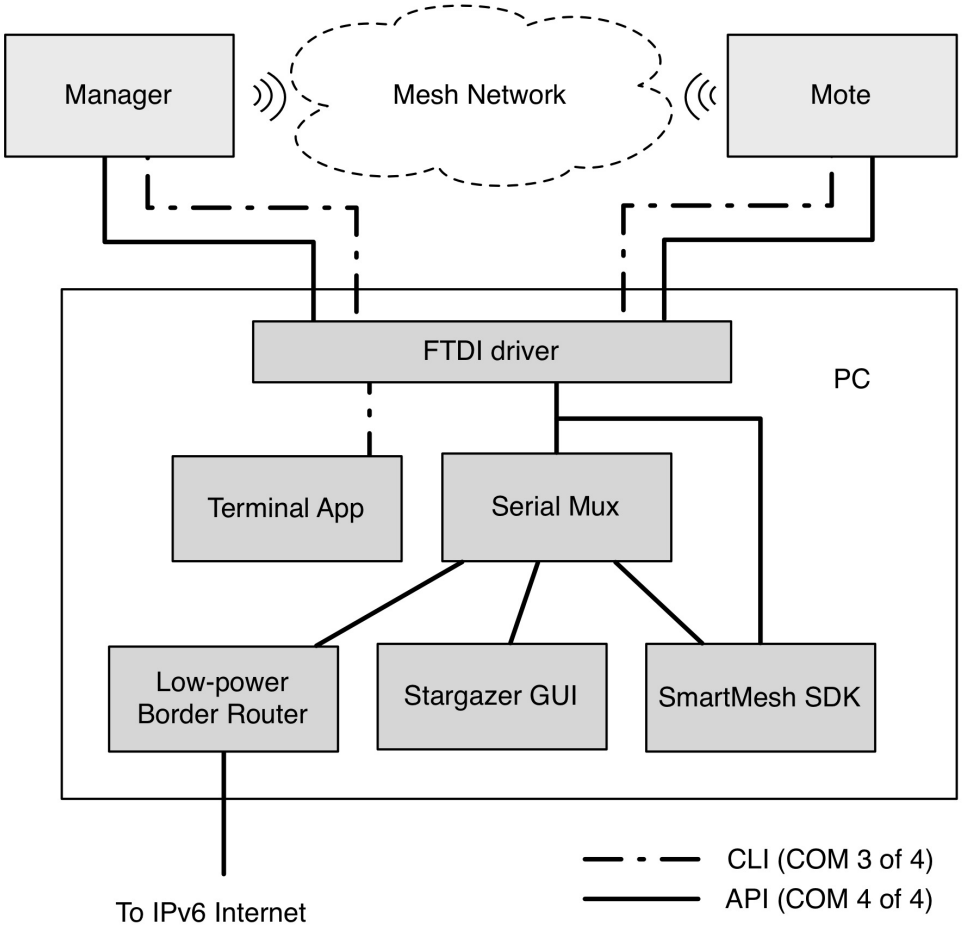


Figure 1 - Software tools for interacting with a SmartMesh IP network.

3 Installation

3.1 Setup

3.1.1 System Overview

Figure 2 shows the Evaluation Kit including a SmartMesh IP Manager (with interface board), five Eterna motes, and an additional interface board. There are different manager/mote combinations available, as shown in Table 1.

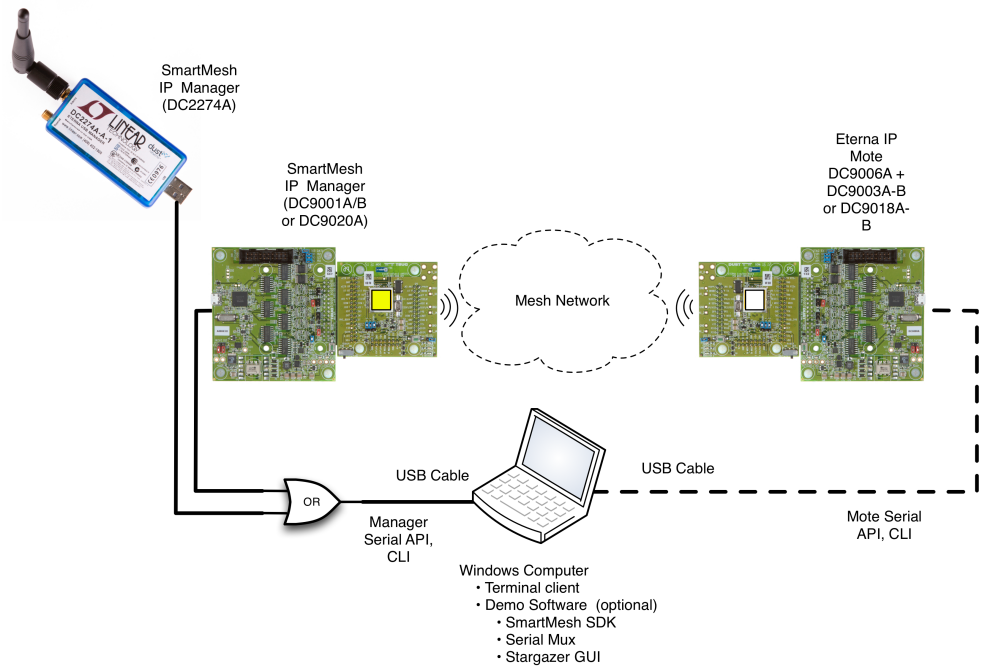


Figure 2: SmartMesh Evaluation Kit

Kit	Mote	Manager	Notes
DC9000A	DC9003A-B	DC9001A*	Eval/Dev Kit with 32 mote manager (discontinued)
DC9000B	DC9003A-B	DC9001B* or DC2274A-A	Eval/Dev kit with 100 mote manager
DC9021A	DC9018A-B	DC9020A* or DC2274A-A	RF Certified Eval/Dev kit with 100 mote manager

DC9021B	DC9018A-B	DC2274A-A DC2274A-B	RF Certified Eval/Dev kit with one embedded Manager (DC2274A-A) and one Access Point Mote - APM (DC2274A-B) for VManager

Table 1 - Kit configurations

*Orderable part number includes DC9006. Manager boards (**yellow** sticker) may be marked with a different sub-component part number (e.g. DC9011A) than the value in Table 1.

> 1) The interface boards for both the manager and mote are identical (DC9006).
>
> 2) The DC2274A-A Manager and DC2274A-B APM do not come with a DC9006 interface board, they connect directly via USB.

> The latest kit, DC90021B, contains motes with MMCX connectors and antennas. All previous evaluation kit motes shiped with chip antennas and the DC2274A-A manager has an external antenna. Additional RF Certified devices are available for order individually with MMCX connectors:
>
> - Mote - DC9018B-B (MMCX connector)
> - Manager/APM - DC9020B (MMCX connector), DC2274A-A and DC2274A-B include an antenna.

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: Stargazer
  - d: Python (if not already installed) and the SmartMesh SDK. Instructions for Python and SMSDK installation are found on DustCloud.org.

You will then:

1. Use APIExplorer to establish a PC to Manager (DC9001) connection
2. Use APIExplorer to stablish 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

3.1.2 Step 1 - Prepare the Hardware

The DC2274A manager is plugged directly into a USB port for power and communication - a blue LED will light showing it has power. Other managers (see table 1) and all motes require a DC9006 interface board to communicate via USB.

You can connect a DC9006 interface board to a mote or manager using the board-to-board connector, as shown in the figure:

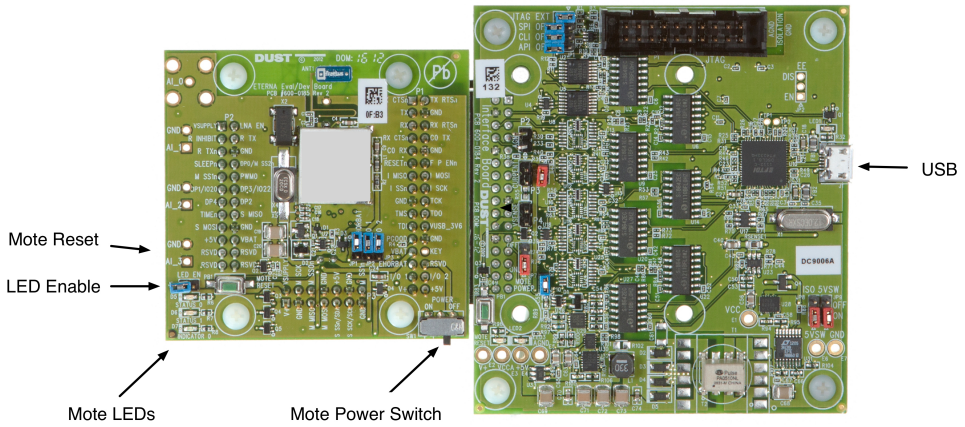


Figure 1 - DC9003 board (left) connected to DC9006 board (right)

- Turn the slide switch marked "Power" on the DC9003 boards to ON.
  - On the manager (marked with a yellow sticker for some hardware types), 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.

3.1.3 Step 2a - Installing FTDI Serial Drivers

Installing Serial Drivers

Most modern OSes come with FTDI drivers pre-installed, but you may have to install them manually if they don't configure automatically when you plug in a device. You can check this by plugging in your manager - four virtual COM ports should automatically be added. These can be viewed using the Windows Device Manager (Control Panel -> System -> Hardware -> Device Manager -> Ports). See the troubleshooting section if the driver doesn't configure automatically.

Driver installation has three steps:

- Download FTDI driver software (if needed)
- Connect a manager (DC9001) and run through driver setup
- Connect a mote (DC9006 + DC9003A-B) and run through driver setup

> ⚠ 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 when using an API or CLI client

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.

3.1.4 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

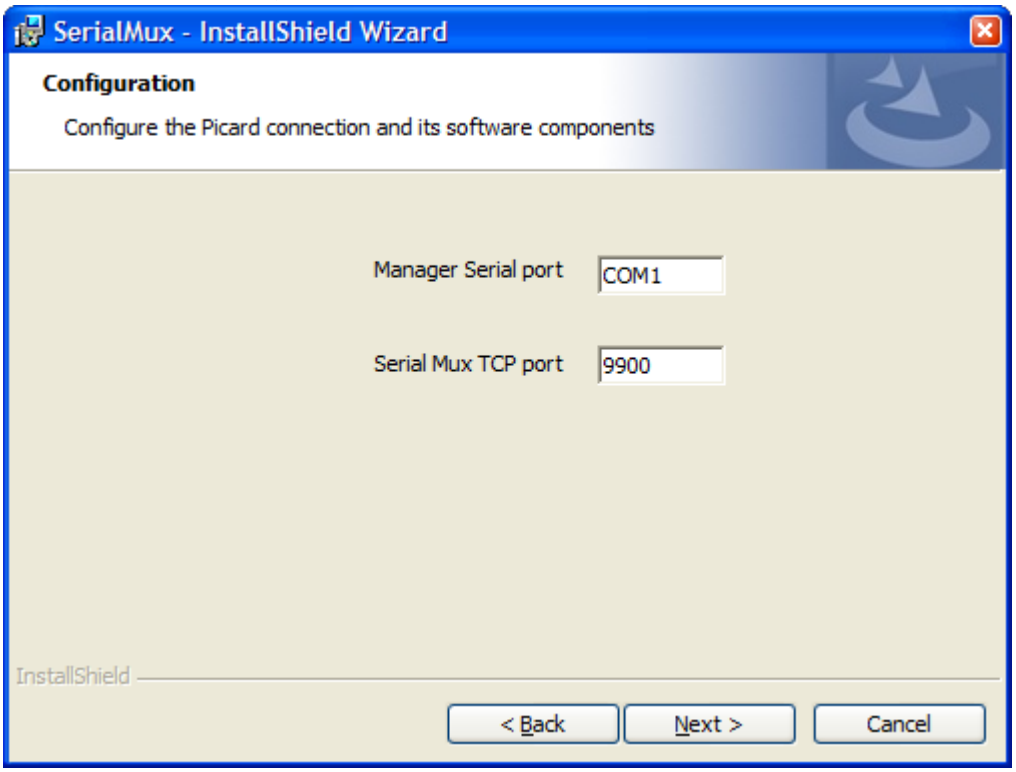
The Serial Mux software is distributed in a zip archive file named `SerialMuxInstaller` with the Serial Mux version appended. Unless you know that you need a particular version for compatibility, choose the latest version.

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.

4. 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).

3.1.5 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`.

3.2 Troubleshooting

3.2.1 Windows FTDI Driver installation

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.

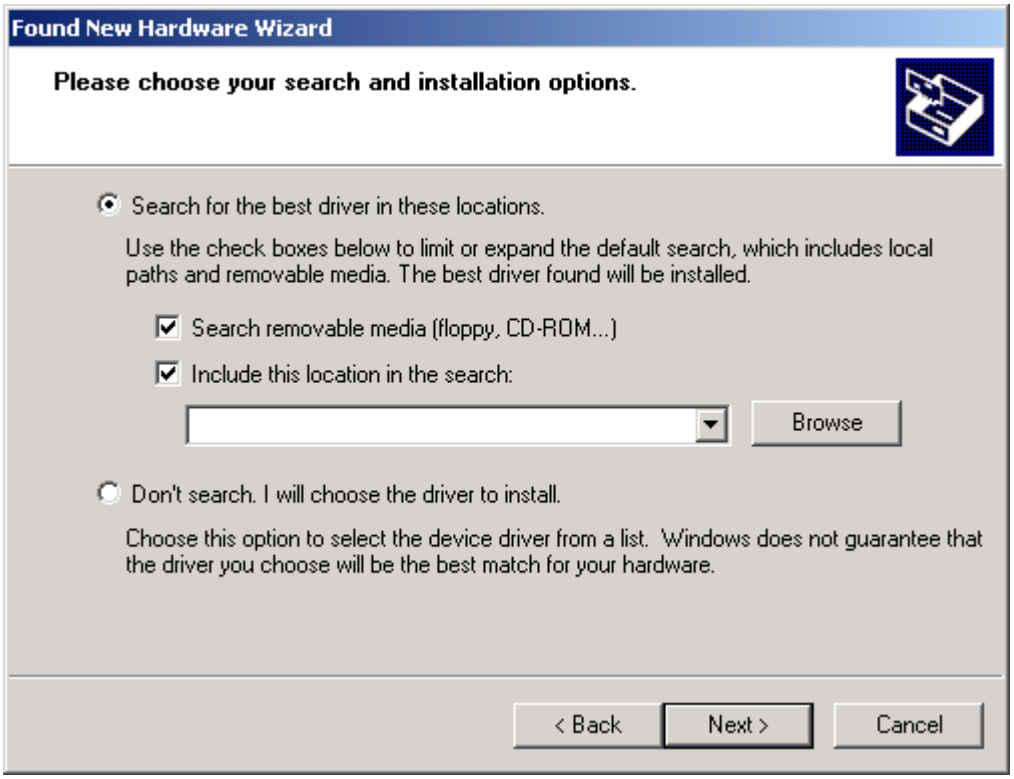
Windows Driver Installation

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.**

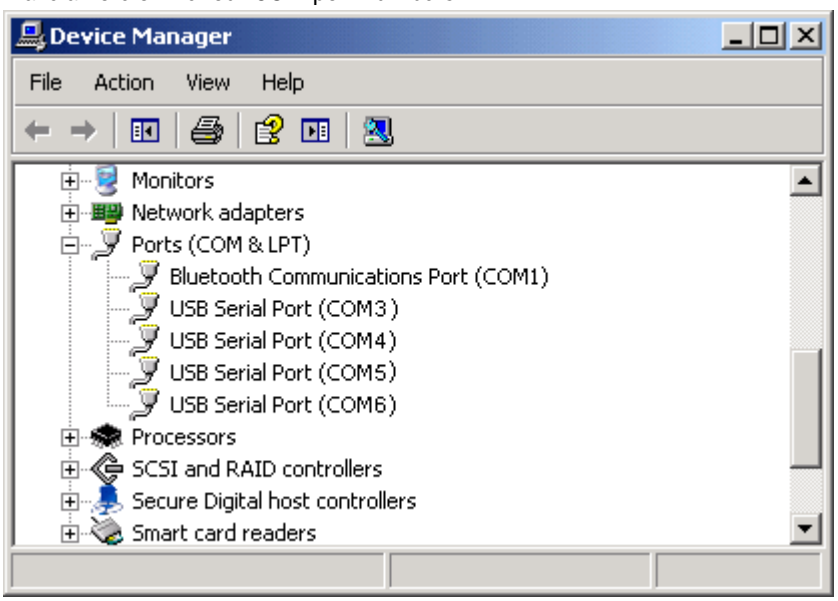
3. Select the box to **Include this location in the search**. Then, use the **Browse** button to navigate to your desktop, and click **Next**.



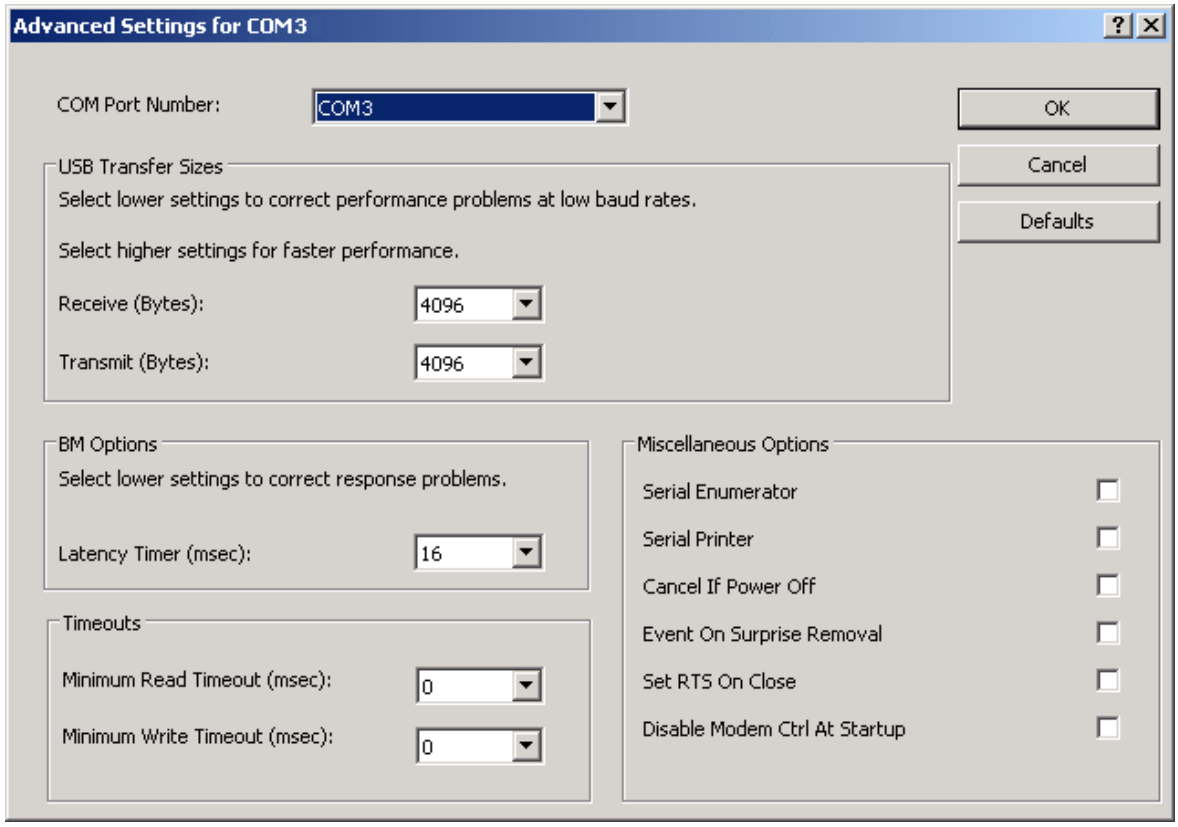
4. After the Wizard installs the software, click **Finish**.
5. When the Found New Hardware Wizard reappears, repeat steps 2 through 4 to continue the installation. Repeat these steps each time the Wizard appears.

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.



3.2.2 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]
```

3.2.3 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.

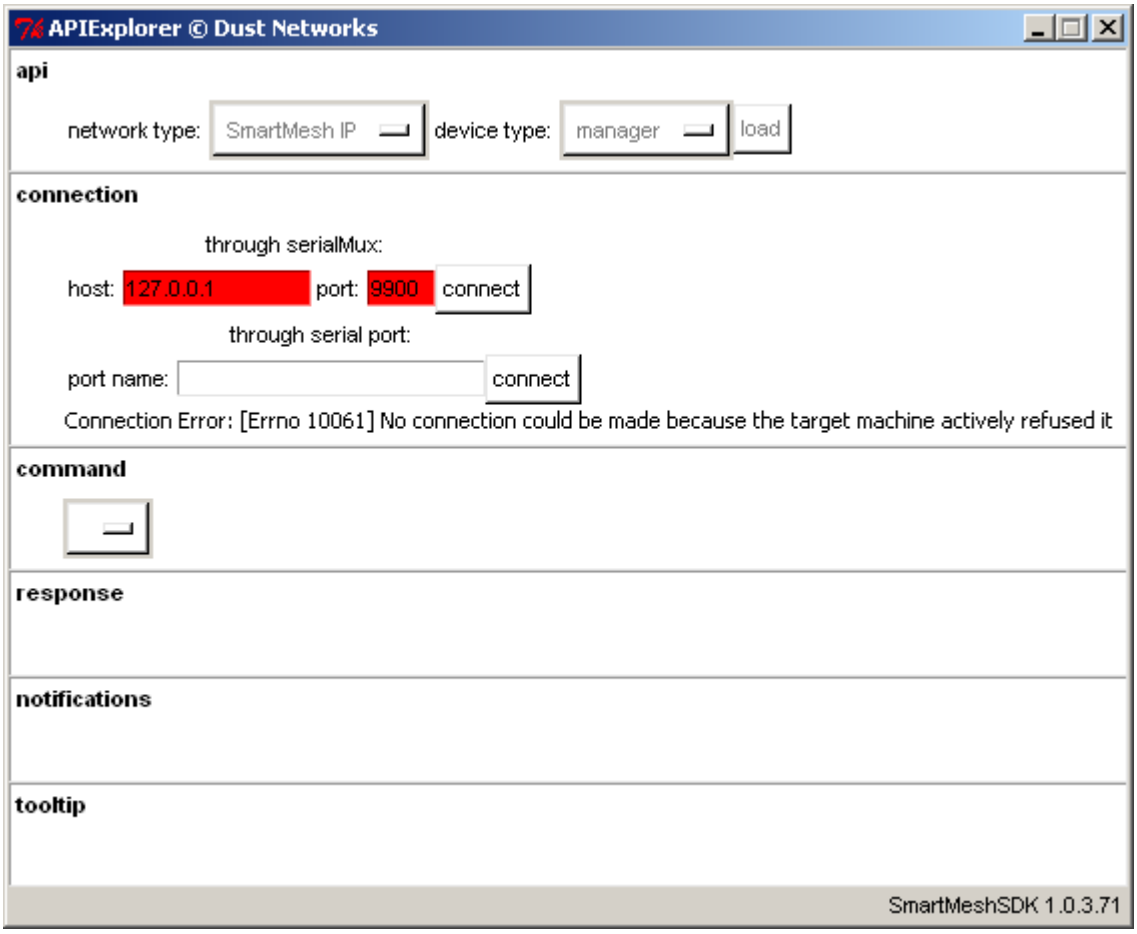
The `screen` program (provided with OS X) can be used as a serial terminal to connect to the CLI port.

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

> help
...
```

3.2.4 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

3.2.5 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.

3.2.6 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.

3.2.7 Master/Slave

Mode Behavior

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

- **Master** - a demo mode enabled on the motes in Starter kits. In this mode, the mote runs an application that generates sample data and controls joining. The mote API is disabled in **master** mode.
- **Slave** - the default mode for LTC58xx and LTP59xx motes. The mote expects 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 or an external microcontroller to connect to it.

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

> 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 in order to simplify external microcontroller logic. Do not use the *autojoin* parameter with a mote in **master** mode, as it may become unresponsive in some revisions of software.

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 light - this should not be mistaken for a dead battery.

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

Switching To Slave Mode

By default, motes in starter kits (DC9000 & DC9021 and DC9007) and are configured for **master** mode. To read the current configuration, connect the mote to a computer via a USB cable and use the `get` mote CLI command. To configure the mote for **slave** mode, use the `set` mote CLI command:

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

```
> get mode
master
```

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

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

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

Switching To Master Mode

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

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

```
> get mode
slave
```

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

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

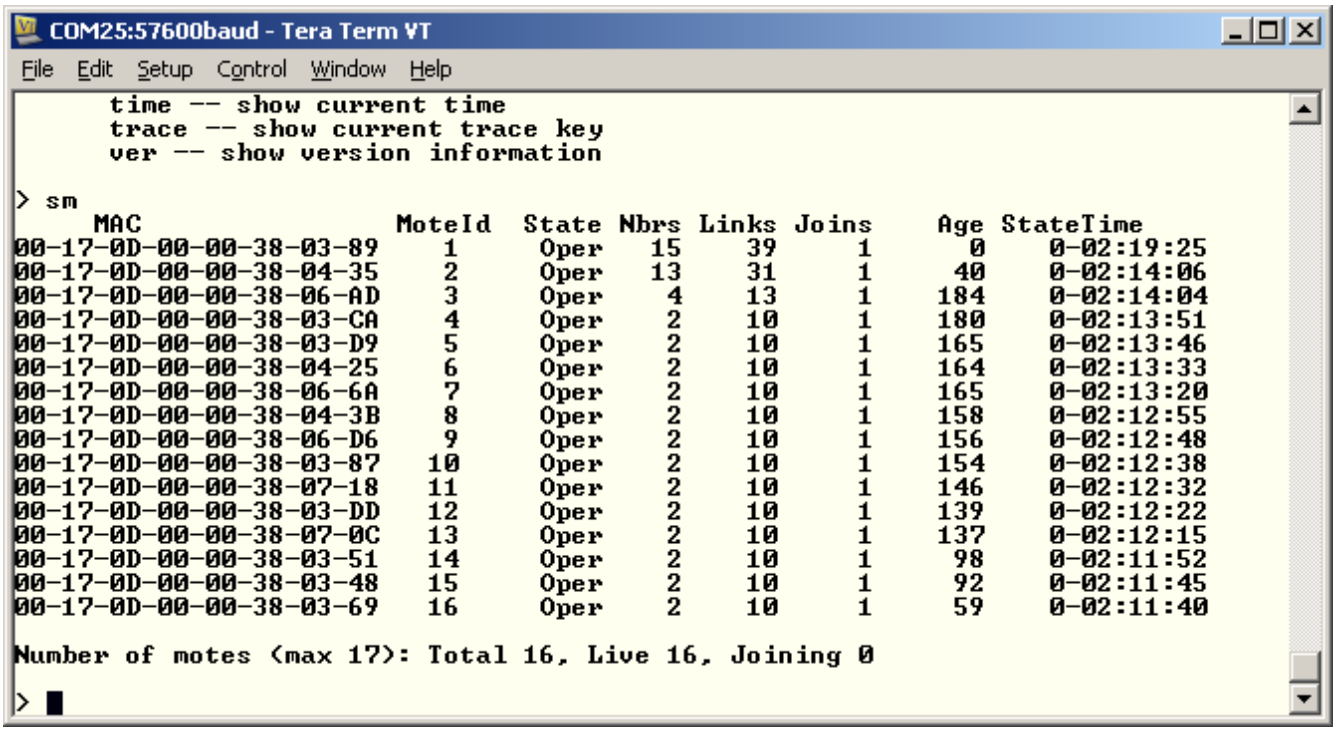
You must reset the mote for the `set mode` command to take effect. Once set, the mode persists through reset.

4 Serial Terminal Client

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

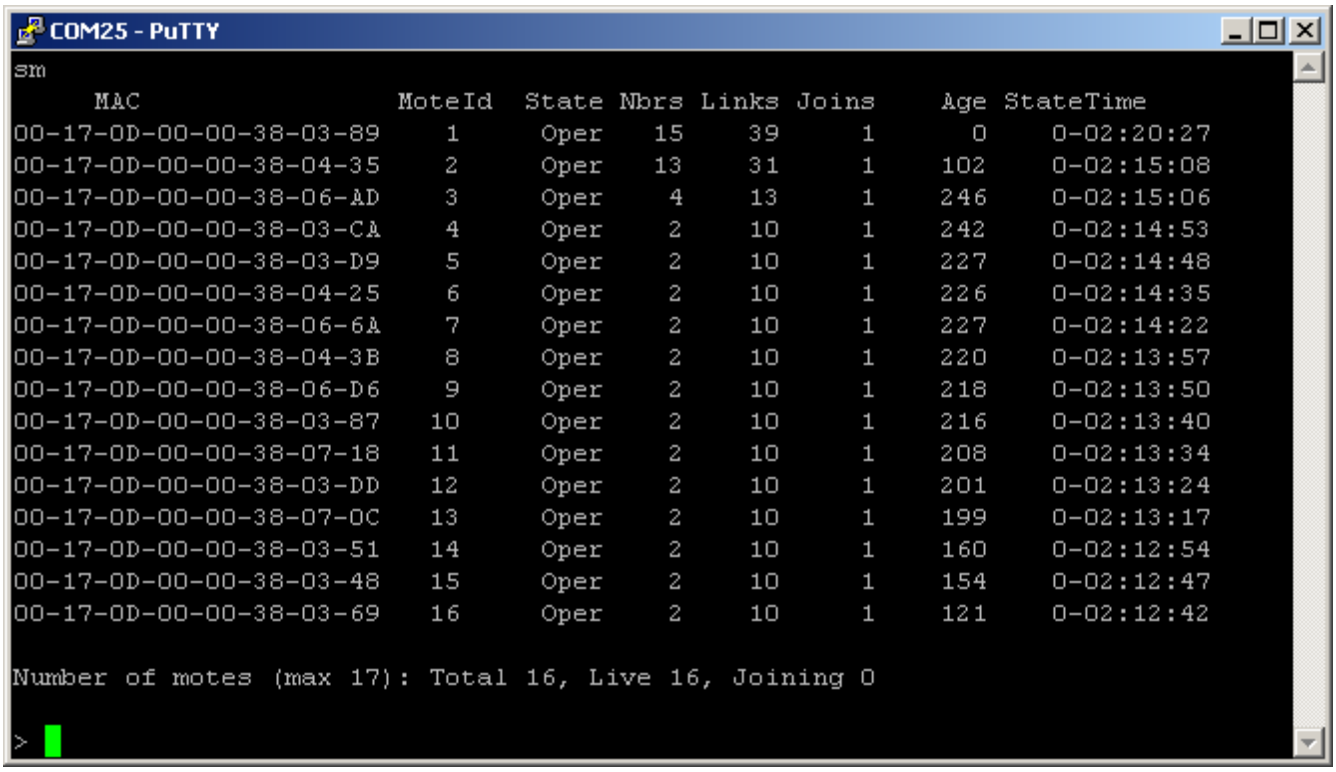
4.1 TeraTerm

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



4.2 PuTTY

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



4.3 minicom

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

4.4 Microsoft Windows HyperTerminal

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

⚠ Not installed in Windows Vista or Windows 7

poipoi - HyperTerminal

File Edit View Call Transfer Help

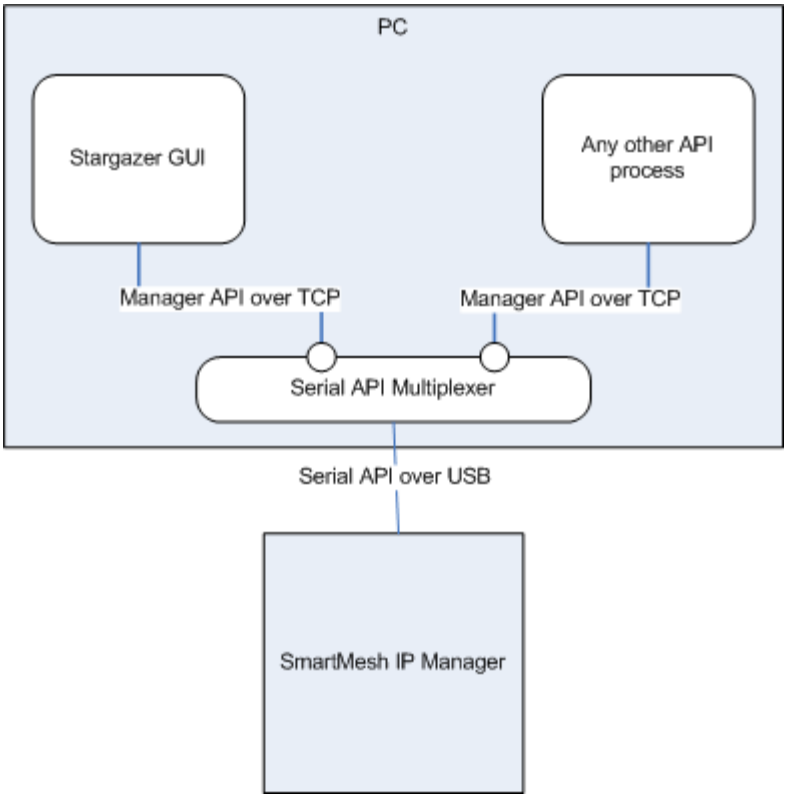
```
>
>
> sm
     MAC                 MoteId  State Nbrs Links Joins    Age StateTime
00-17-0D-00-00-38-03-89    1      Oper   15    39     1      0    0-02:22:56
00-17-0D-00-00-38-04-35    2      Oper   13    31     1    251    0-02:17:37
00-17-0D-00-00-38-06-AD    3      Oper    4    13     1    395    0-02:17:35
00-17-0D-00-00-38-03-CA    4      Oper    2    10     1    391    0-02:17:22
00-17-0D-00-00-38-03-D9    5      Oper    2    10     1    376    0-02:17:17
00-17-0D-00-00-38-04-25    6      Oper    2    10     1    375    0-02:17:04
00-17-0D-00-00-38-06-6A    7      Oper    2    10     1    376    0-02:16:51
00-17-0D-00-00-38-04-3B    8      Oper    2    10     1    369    0-02:16:26
00-17-0D-00-00-38-06-D6    9      Oper    2    10     1    367    0-02:16:19
00-17-0D-00-00-38-03-87   10      Oper    2    10     1    365    0-02:16:09
00-17-0D-00-00-38-07-18   11      Oper    2    10     1    357    0-02:16:03
00-17-0D-00-00-38-03-DD   12      Oper    2    10     1    350    0-02:15:53
00-17-0D-00-00-38-07-0C   13      Oper    2    10     1    348    0-02:15:46
00-17-0D-00-00-38-03-51   14      Oper    2    10     1    309    0-02:15:23
00-17-0D-00-00-38-03-48   15      Oper    2    10     1    303    0-02:15:16
00-17-0D-00-00-38-03-69   16      Oper    2    10     1    270    0-02:15:11

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

> _
```

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

5 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:

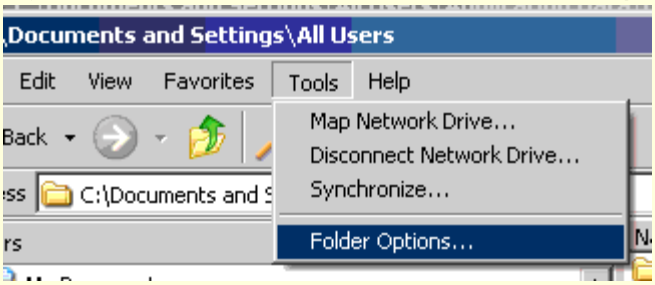
```
C:\Documents and Settings\All Users\Application Data\Dust Networks\SerialMux\Default\serial_mux.cfg
```

On Windows 7, the configuration file is located at:

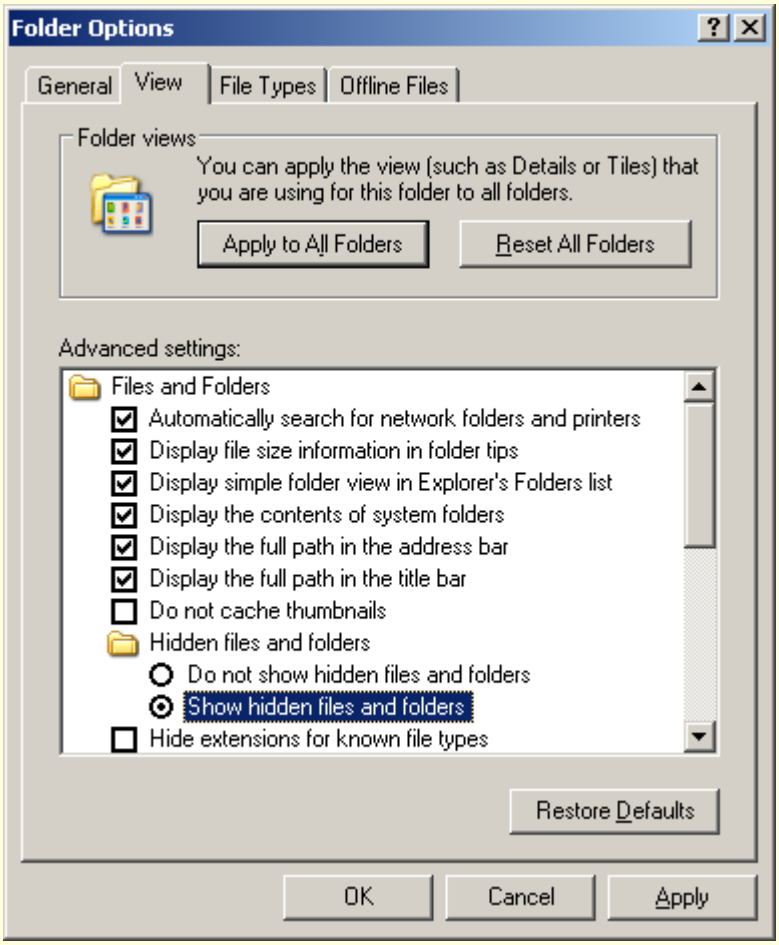
```
C:\ProgramData\Dust Networks\SerialMux\Default\serial_mux.cfg
```

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

1. in your Windows Browser 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.

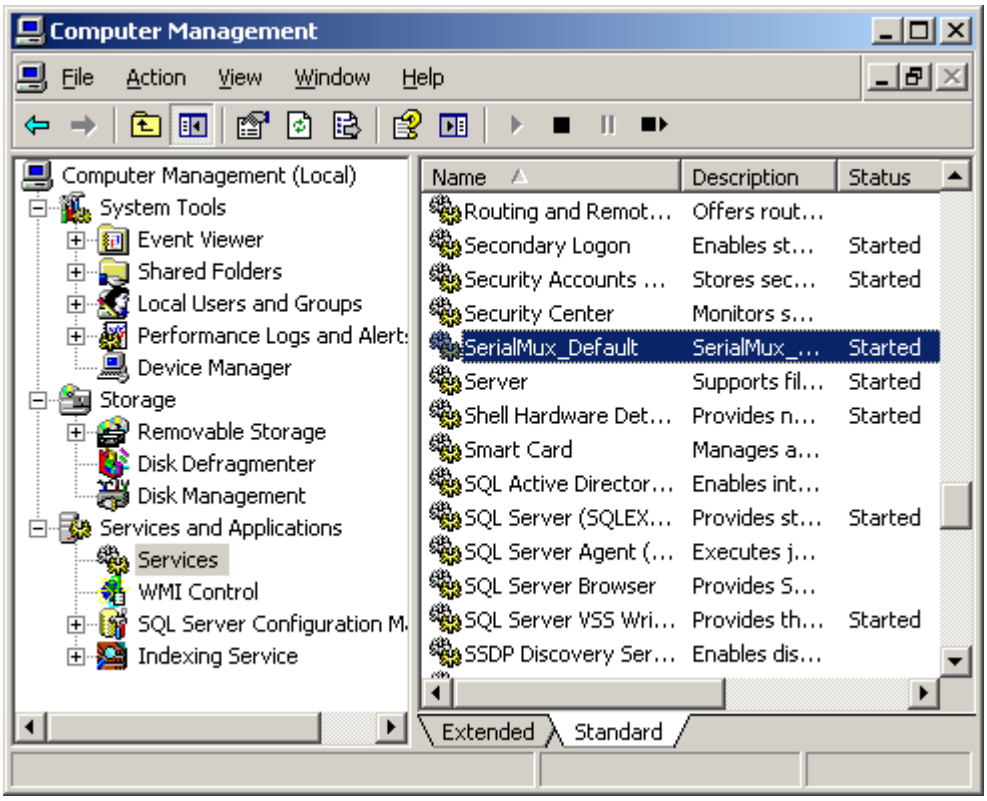
5.2.2 Step 2: Restart the Serial Mux Windows Service

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.

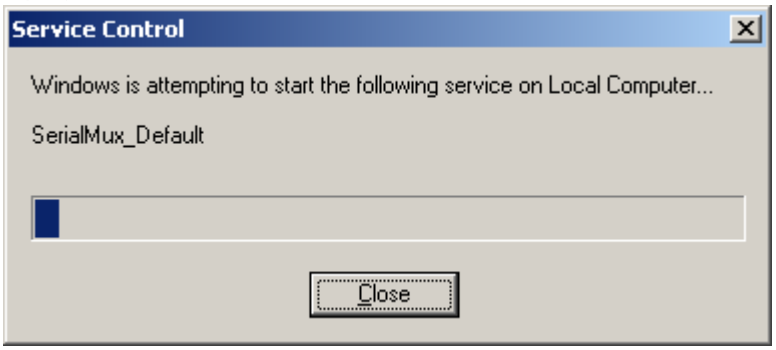
1. In your **Start** Menu, select **Settings > Control Panel**
2. Select **Administrative Tools**

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. `trace`, `info`, `warning`, `error` 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

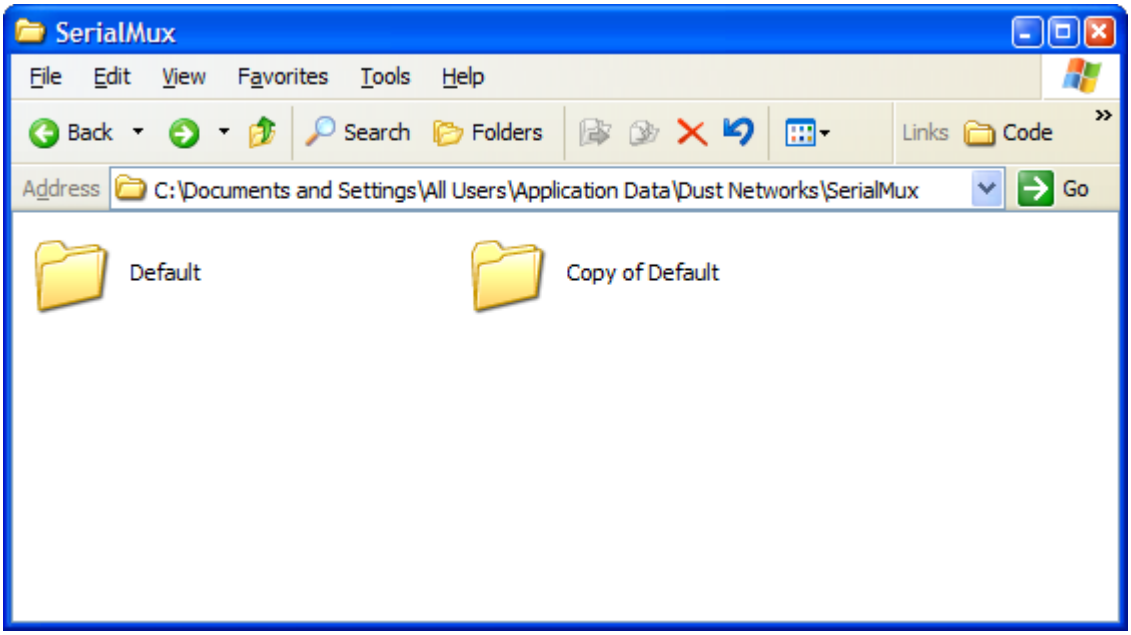
The standard installation of the Serial Mux supports a connection to one Manager. In order to support multiple Managers, multiple copies of the Serial Mux service must be created, one for each Manager.

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

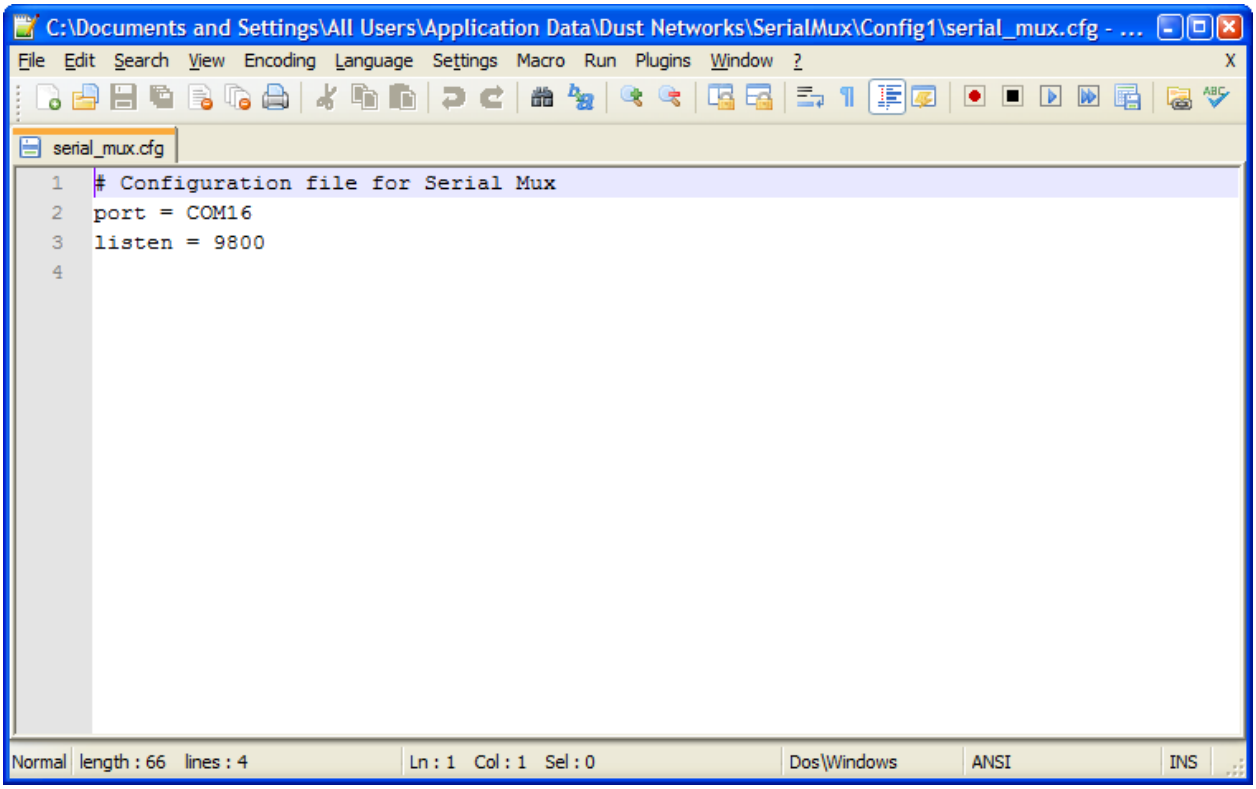
Creating a Serial Mux Configuration

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 cannot 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	reserved	commandType	Hello:version	Hello:authentication
A740A0F5	000C	0000	01	04	3031323334353637

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

headerToken	length	reserved	commandType	response code	Hello Response:version
A740A0F5	0005	0000	01	00	04

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

headerToken	length	reserved	commandType
A740A0F5	0003	0000	40

The Serial Mux responds with:

headerToken	length	reserved	commandType	response code	getNetworkInfo response
A740A0F5	0022	0000	40	00	0000021C520001644800000 8FC00FE 8000000000000000170D00003001C7

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

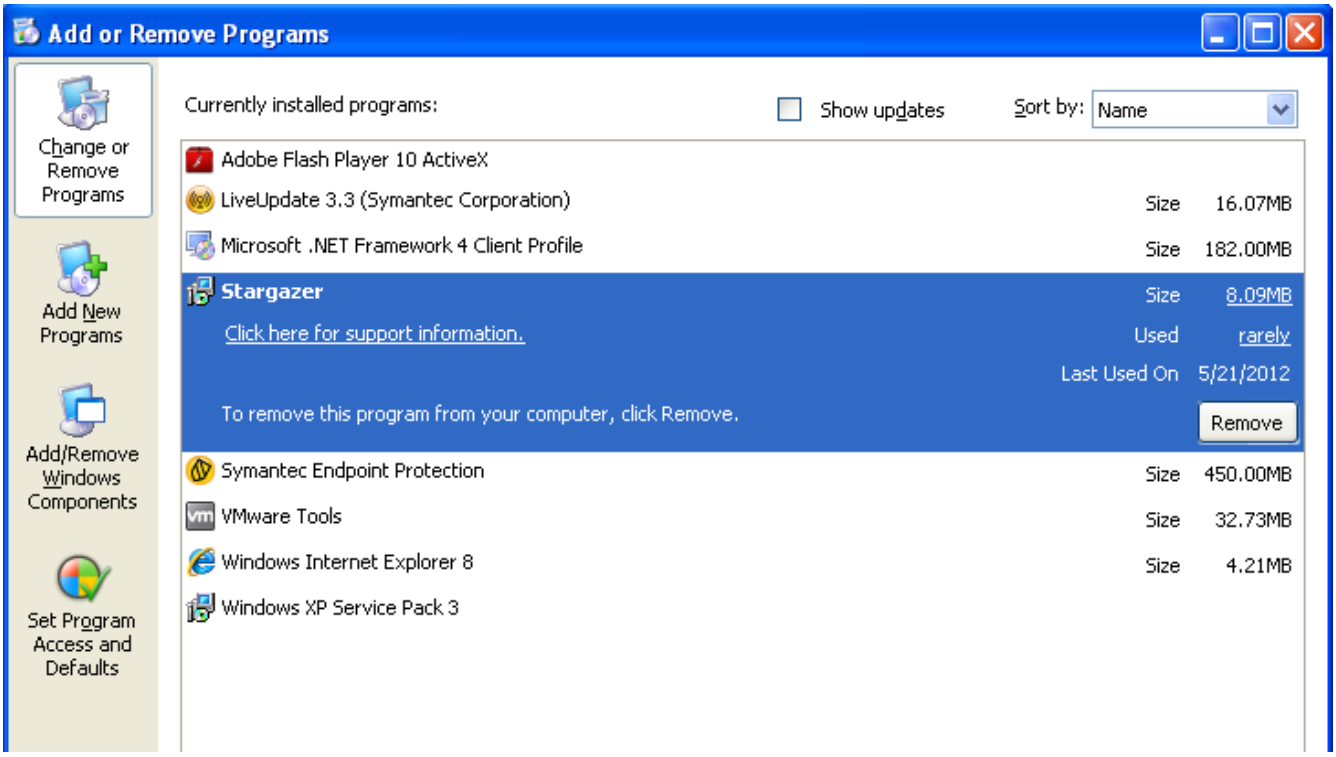
6 Stargazer GUI

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.

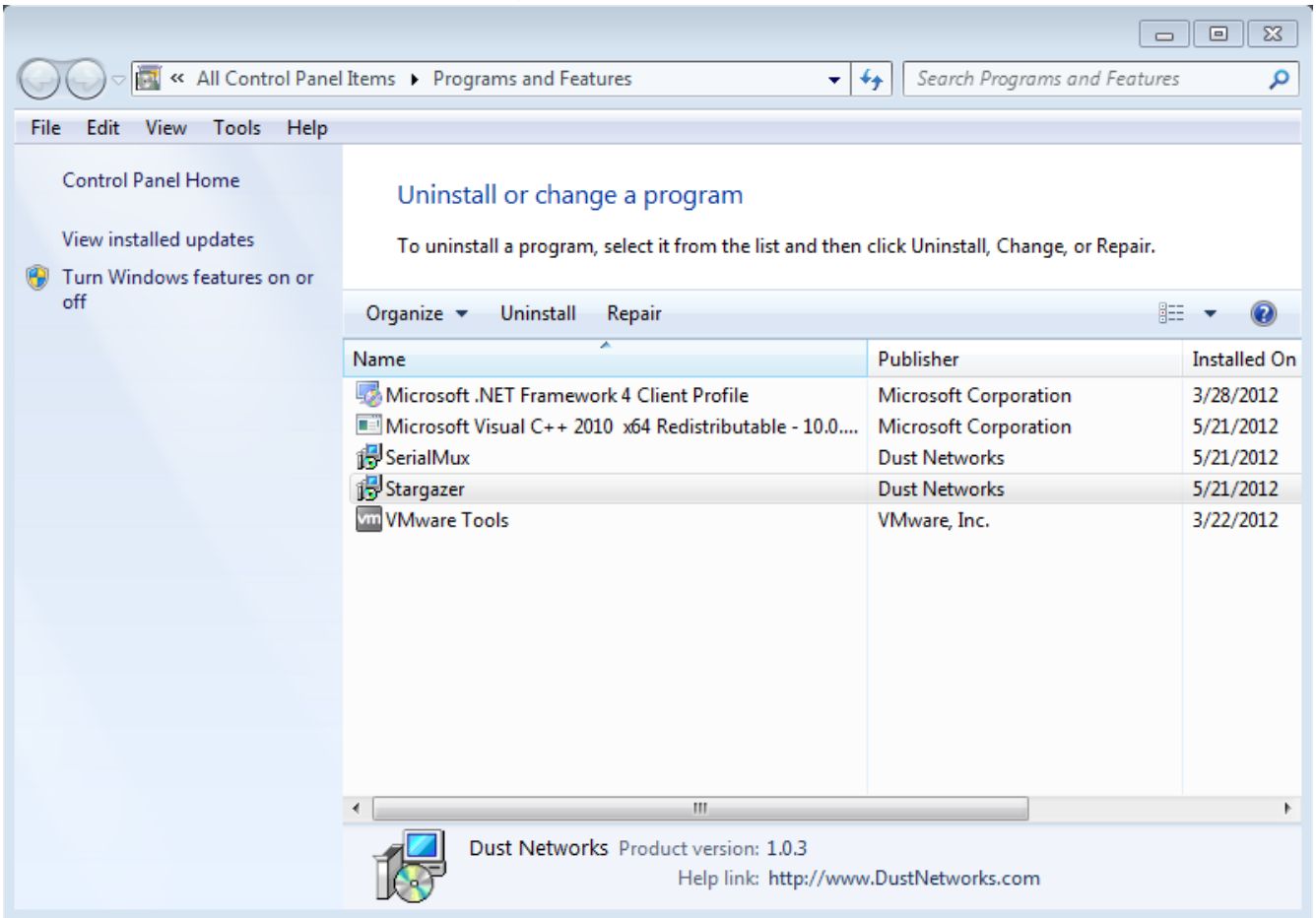
6.1.1 Remove the Existing Application

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.

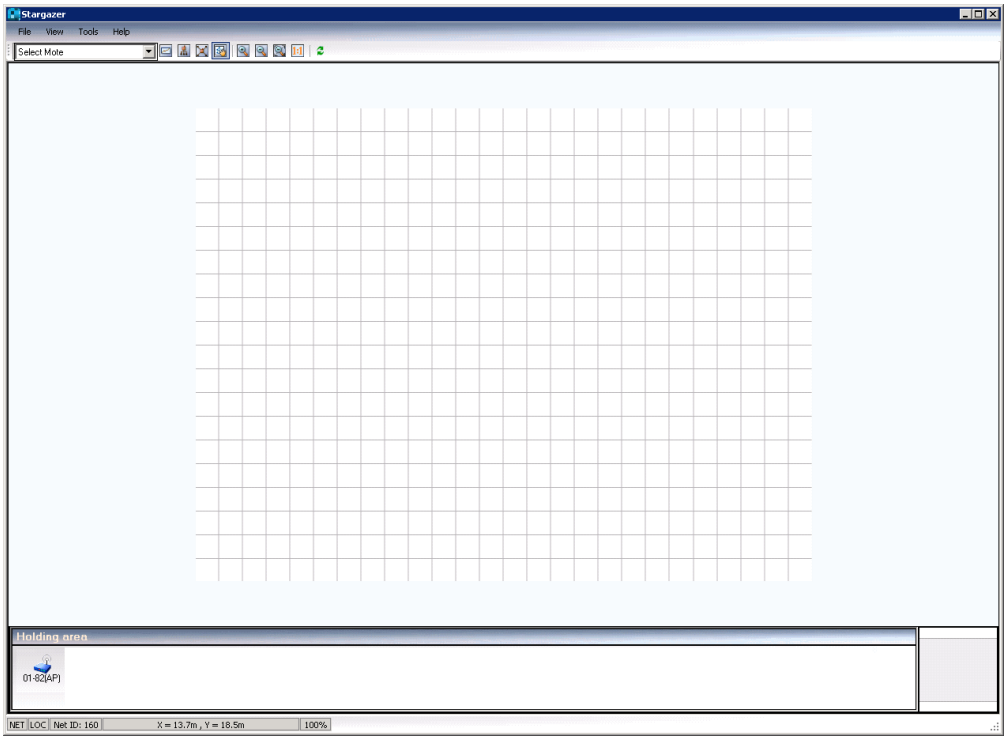
6.2 Using Stargazer

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 & DC9021) 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.

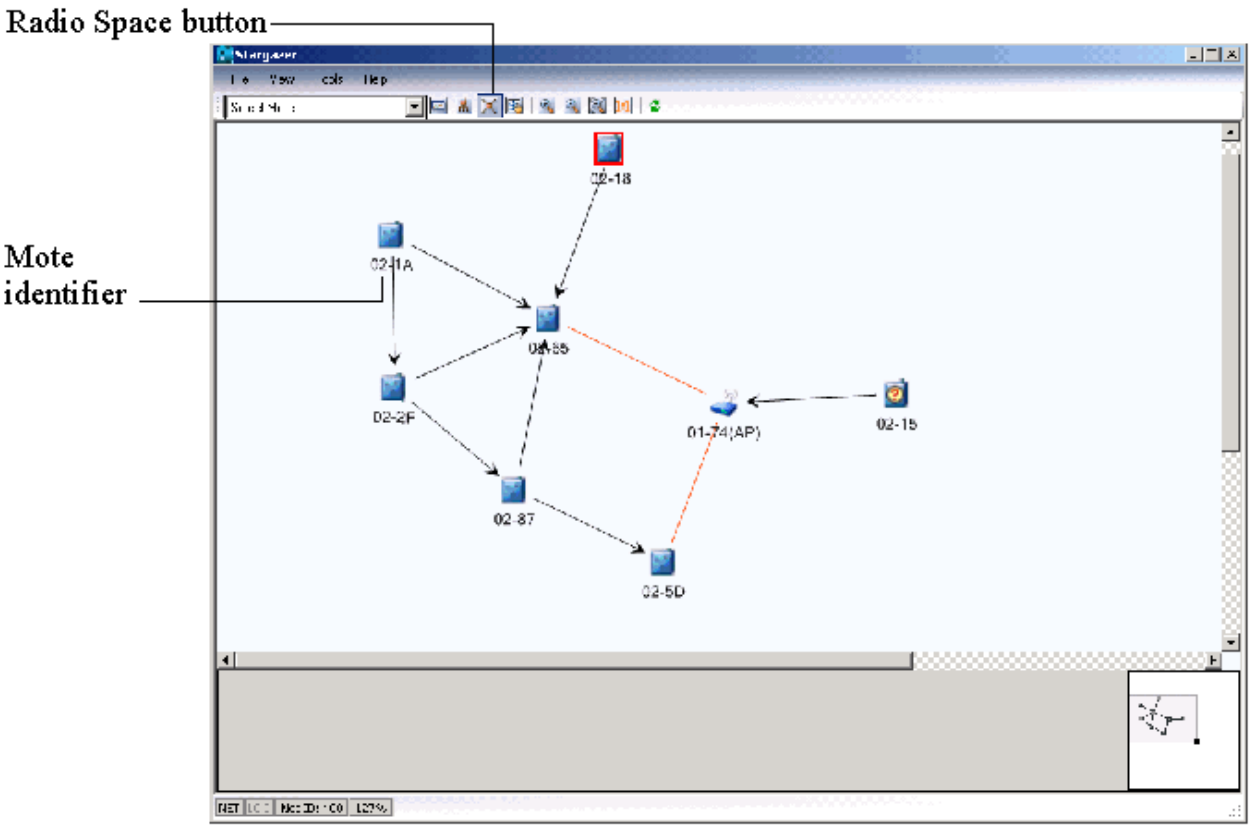
The moment you turn on a mote, it begins searching for nearby motes and starts to join the network. Within minutes, the motes form a reliable multichannel mesh network.

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.



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

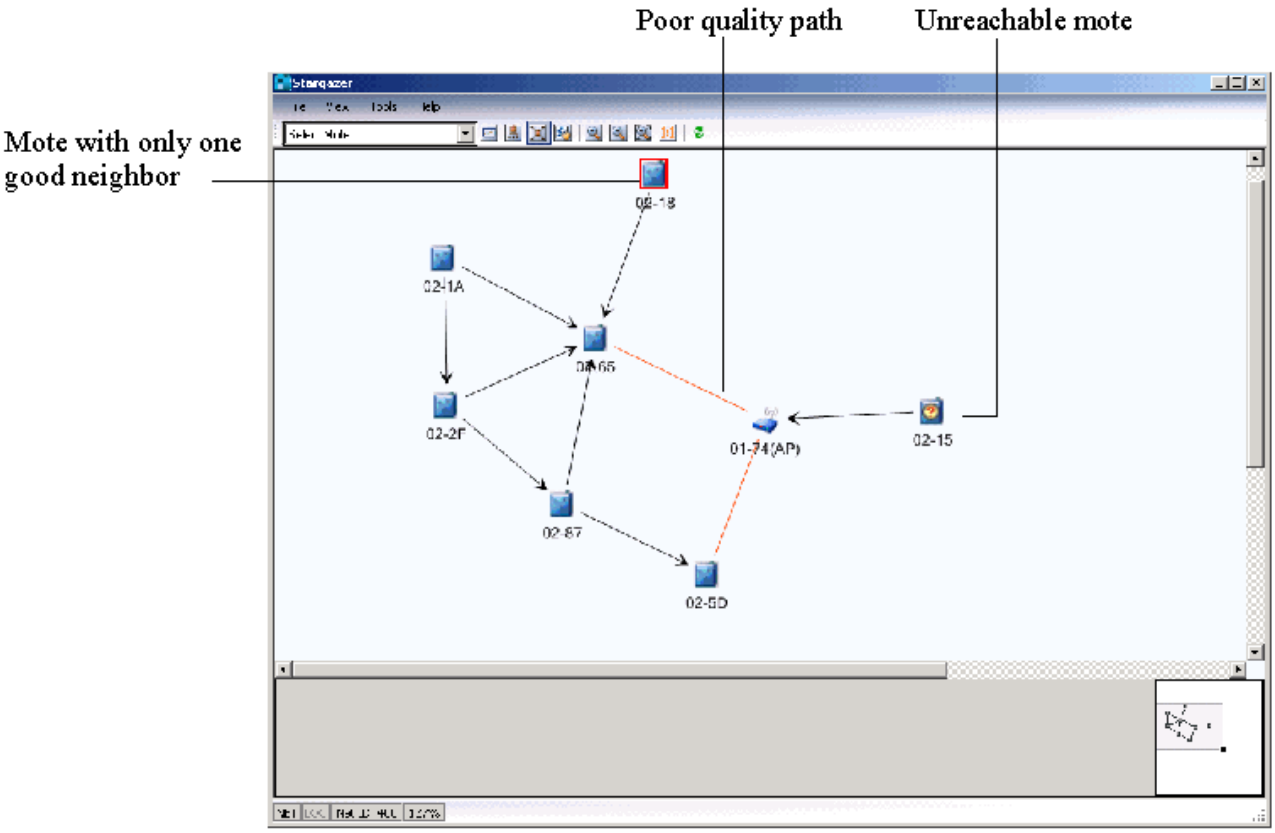
⚠ LEDs are only activated when the mote is configured in Master mode.

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.

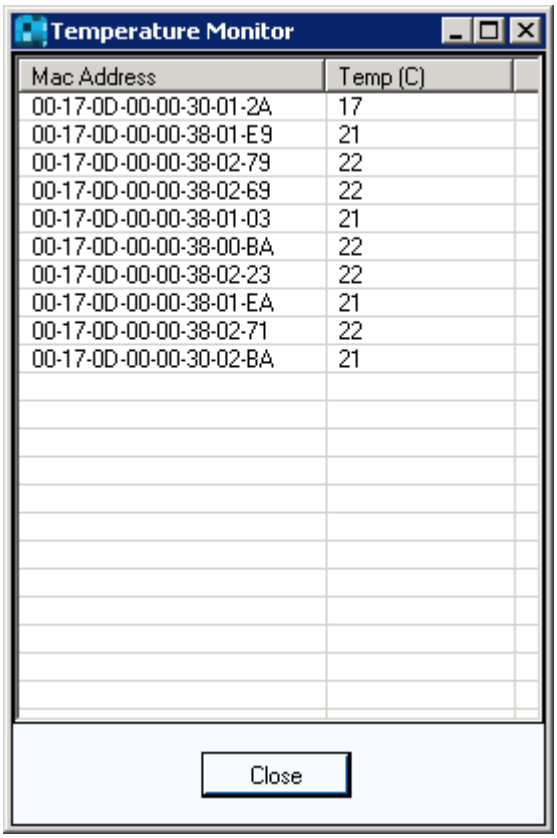


If connectivity problems persist for any mote after the network has been running for an hour, you can manually intervene by trying one of the following:

- 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.



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:

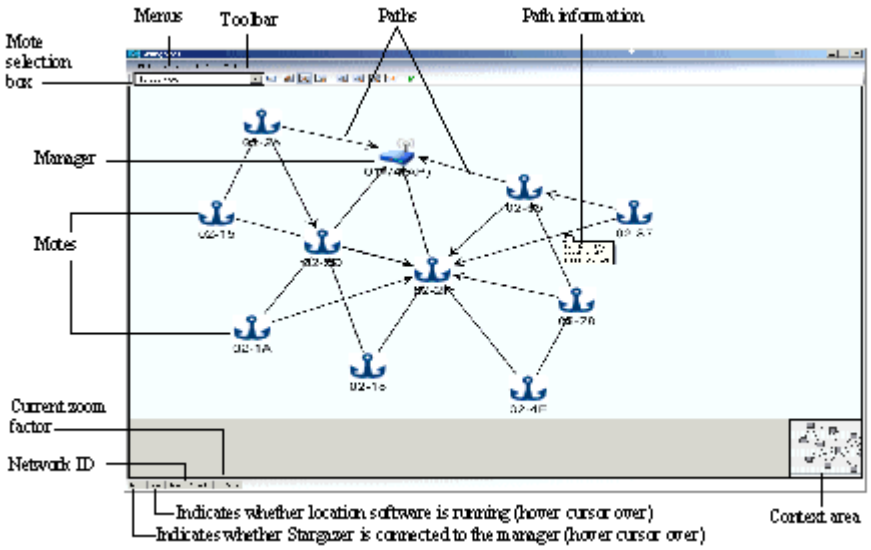
- *File* —Sets user preferences, prints information, and saves a snapshot of mote states and statistics.
- *View* —Displays the network in different layouts, zooms in or out.

- *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

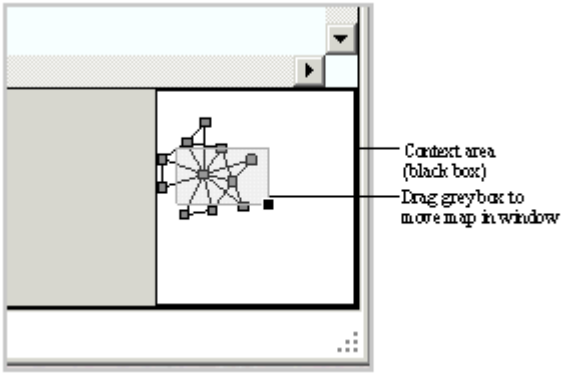
The following icons appear in all topology views:

	SmartMesh IP Manager — The manager controls the network and publishes data received from the wireless mesh network.
or	**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.

	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.
or	**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.
	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.

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.

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

	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

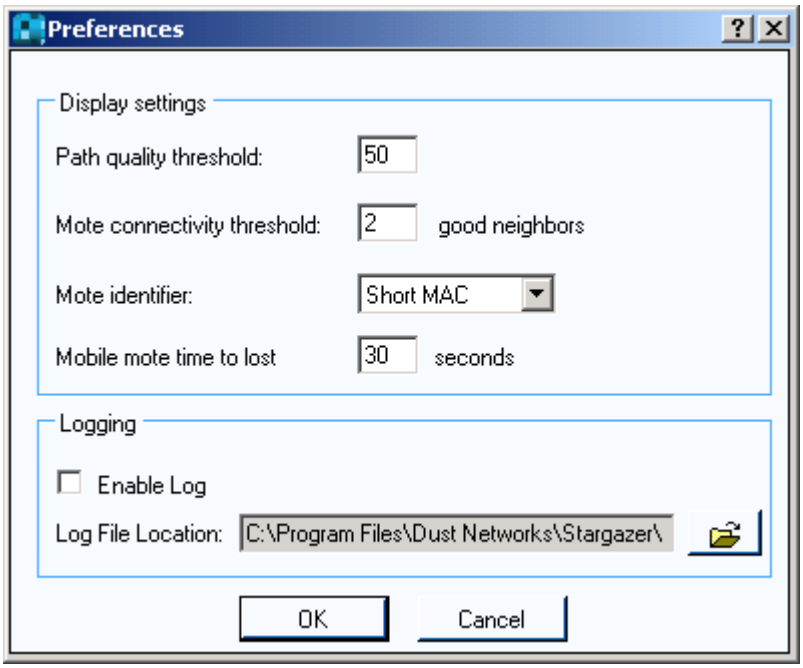
With a few clicks, you can display more detailed network information about a selected mote or path. For example, you can:

- 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 >= 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 *good neighbors* 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: • *Long MAC*—Displays the mote's 8-byte EUI address. The MAC address is the unique address assigned to the mote at the factory. For example: *00-17-0D-00-00-38-01-74* • *Short MAC*—Displays the last 2 bytes of the mote's MAC address. For example: *01-74* • *Mote ID*—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.

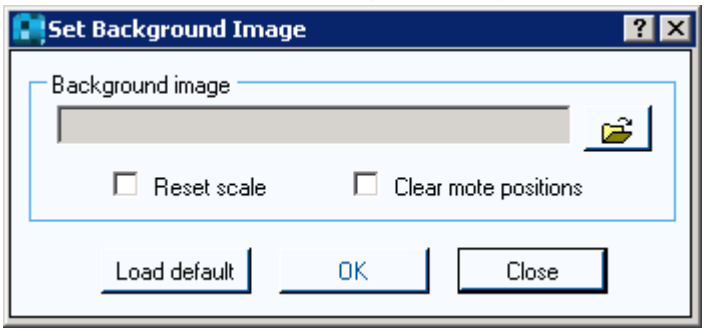
Log file location	Specifies the location of the log.

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:

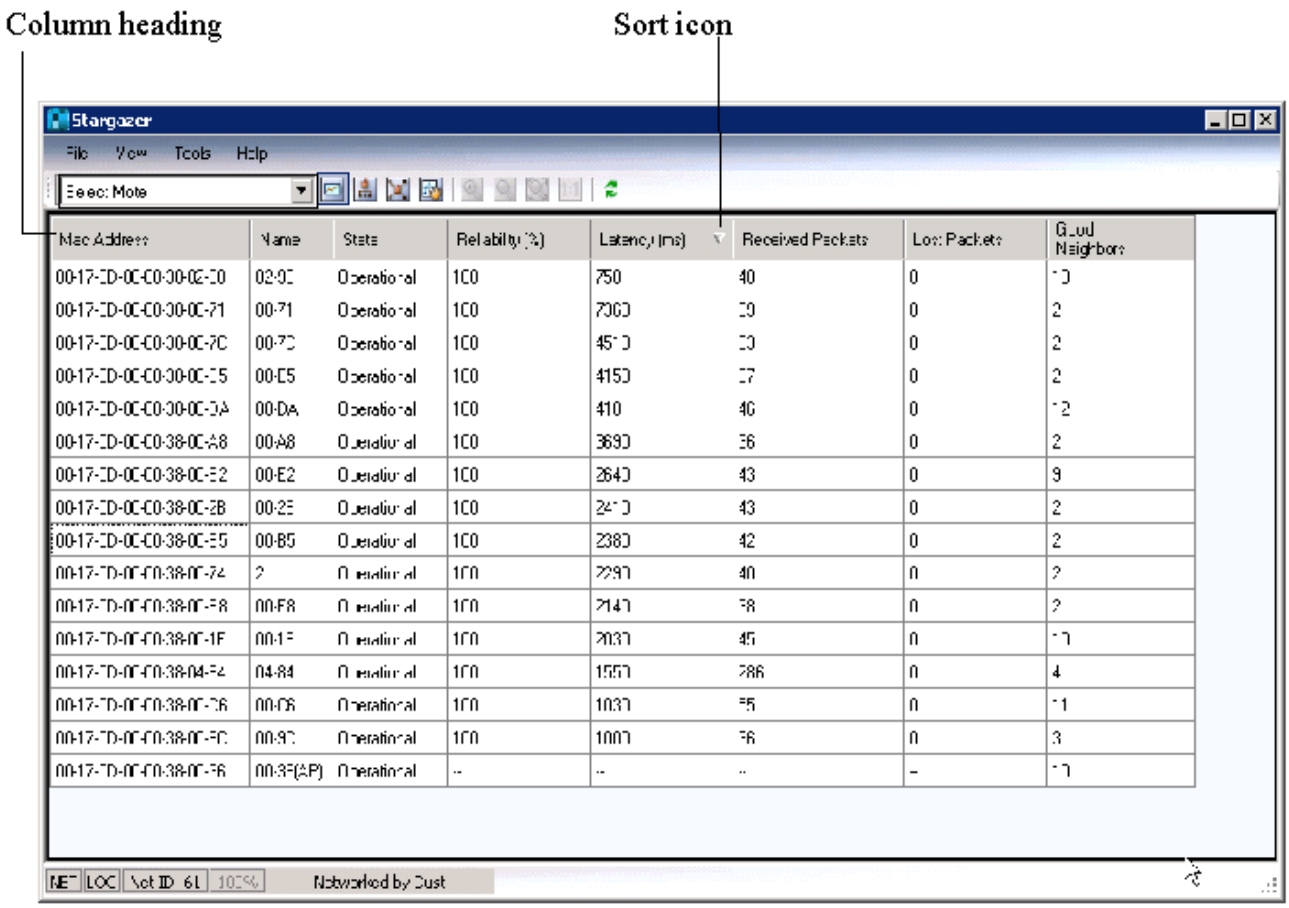


3. Click the **Browse** button, navigate to where the site map image is stored on your computer, and click **Open**.
4. Click **OK**.The site map appears in the Manual view in Stargazer.

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.



Mote Table View

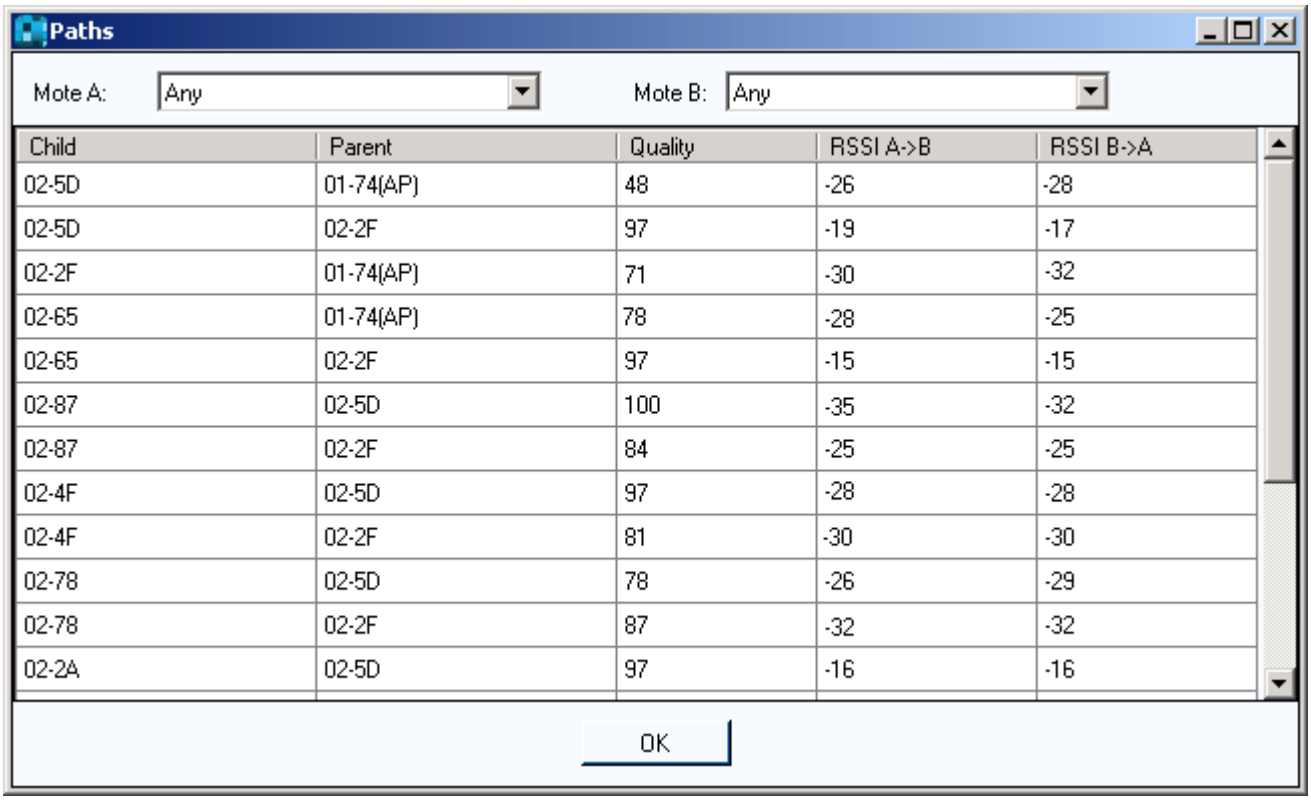
The mote table displays the following information:

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: • *Negotiating* indicates the mote is attempting to join the network. • *Operational* indicates the mote is connected to the network and reporting data. • *Lost* 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

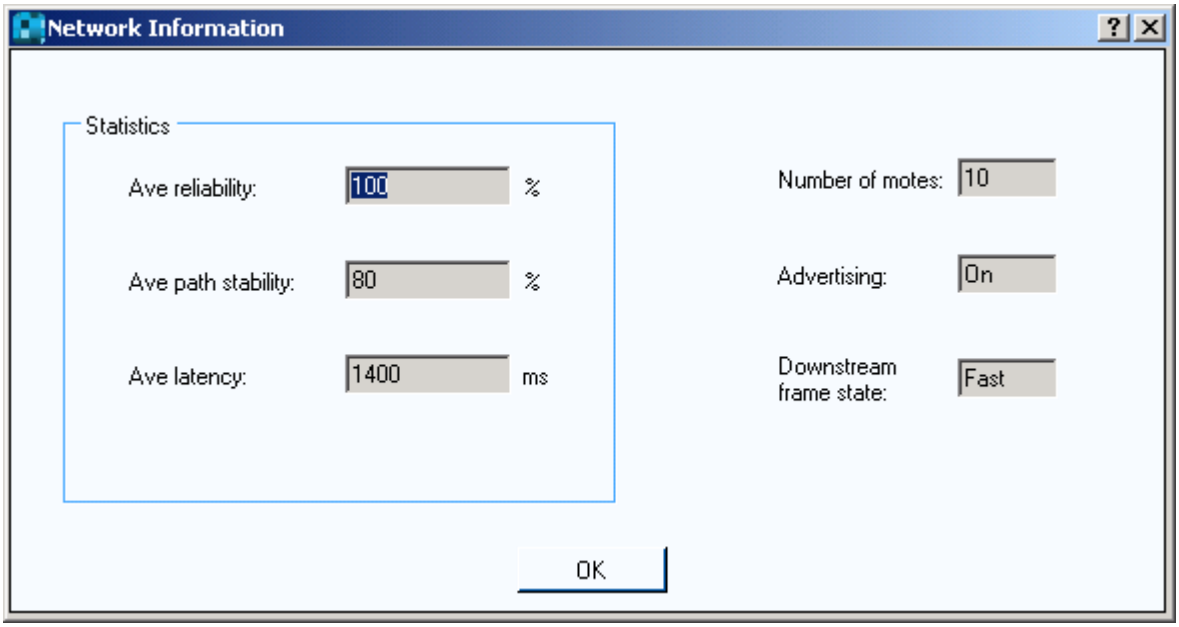
The Paths window displays the following information.

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 ≥ 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

☑ Request/Response ☑ Notifications ☐ 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	1404003D240257000232800 0170D000038022FF0B9F0B900030500FF010500...
5/26/2011 6:50:24 PM	RX	02-4F	94	1404003D240258000138800 0170D000038024FF0B9F0B900020500FF010500...
5/26/2011 6:50:25 PM	RX	02-18	94	1404003D240259000288700 0170D0000380218F0B9F0B900010500FF010500...
5/26/2011 6:50:28 PM	RX	02-5D	94	1404003D24025C000501400 0170D000038025DF0B9F0B900040500FF010500...
5/26/2011 6:50:29 PM	RX	02-1A	94	1404003D24025D000927C000170D000038021AF0B9F0B900010500FF010500...

Traffic Monitor Window

The Traffic Monitor window displays the following information.

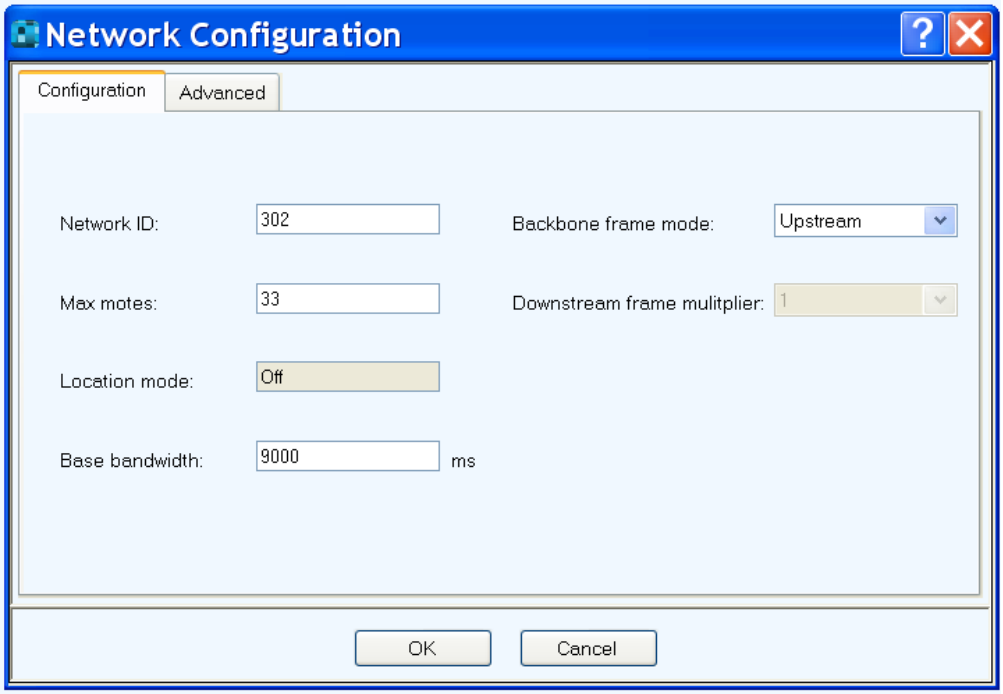
Field	Description
Request/Response check box	When selected, the Traffic Monitor displays request/response packets exchanged between Stargazer and the manager.
Notifications check box	When selected, the Traffic Monitor displays response packets and data packets received by Stargazer from motes.
Scroll 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),

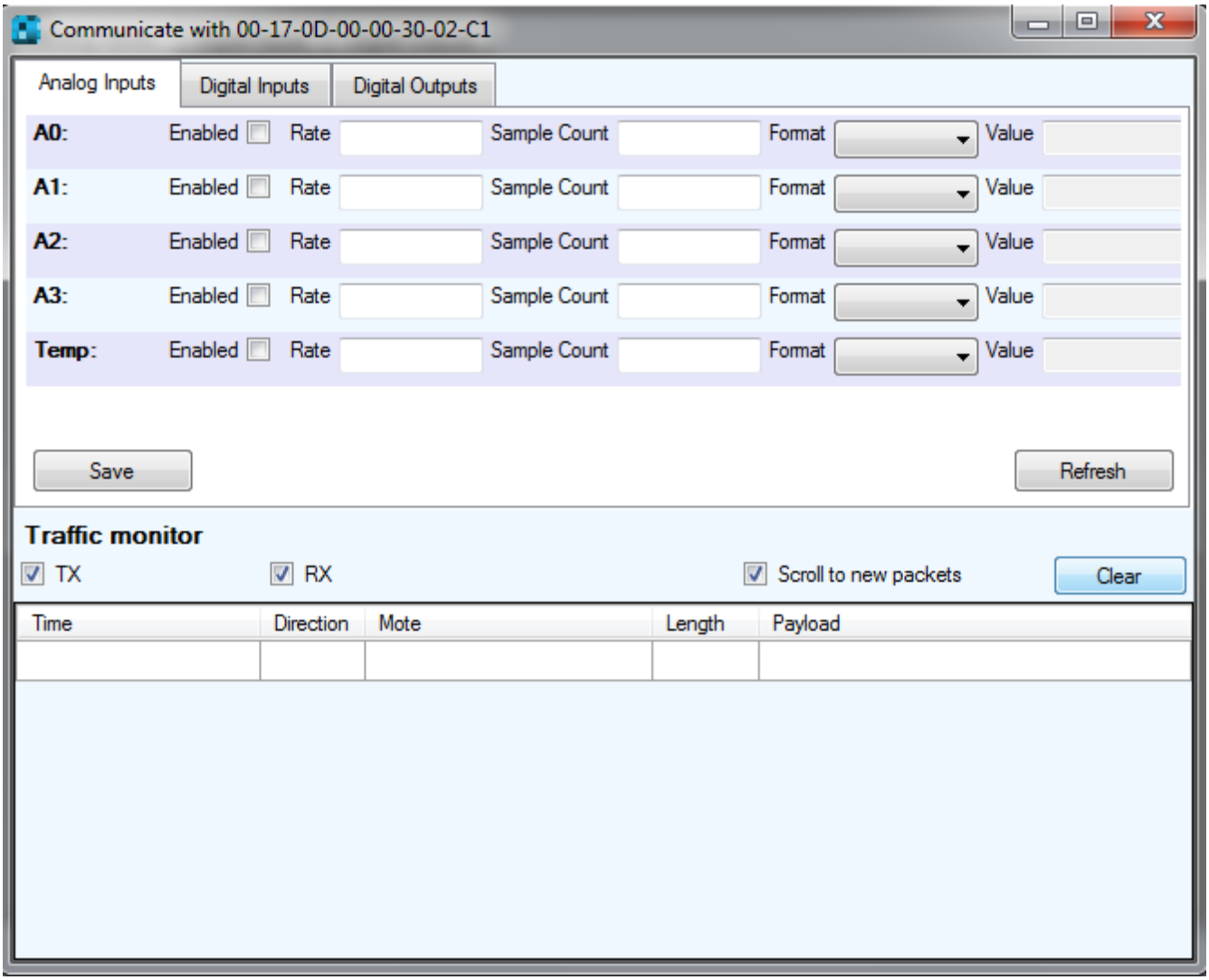
- Digital Inputs (4 channels),
- and Digital Outputs (3 channels).

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.

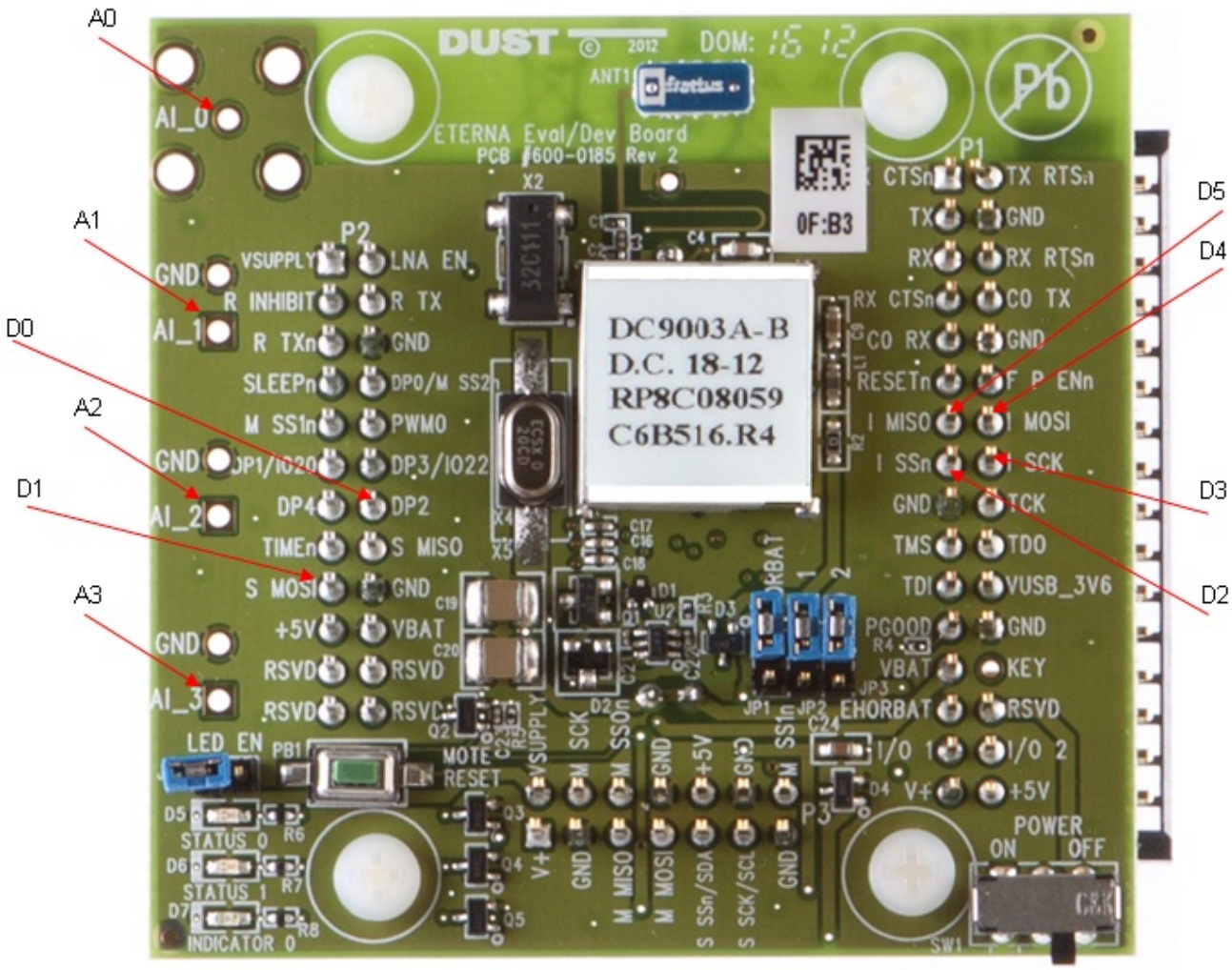
Digital inputs (D0-D3) have internal pull-ups enabled when the mote is in **master** mode.

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 cannot 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.



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. • 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 it the sample count has been reached.
Format	Defines how data is reported in the packet: • *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. • 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 x 1.8 = 1.11 V) Note: If there is nothing connected to the pin, it will float and the value can be anything within the valid range.
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. • 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 it the sample count has been reached.
Format	Defines how data is reported in the packet: • *All* includes all data samples (samples that do not fit in the packet are dropped) • *On change* provides a sample when the input changes from 0 to 1 or vice versa • *On rising* provides a sample whenever a rising edge (0 to 1 transition) is seen • *On falling* provides a sample whenever a falling edge (1 to 0 transition) is seen

Value	Either a 1 or 0. See the LTC5800-IPM Datasheet for minimum I/O levels. Note: If there is nothing connected to the pin, it will float and the value can be anything within the valid range.
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: • *No change* - 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 Interacting with a Network

7.1 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 esend 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.

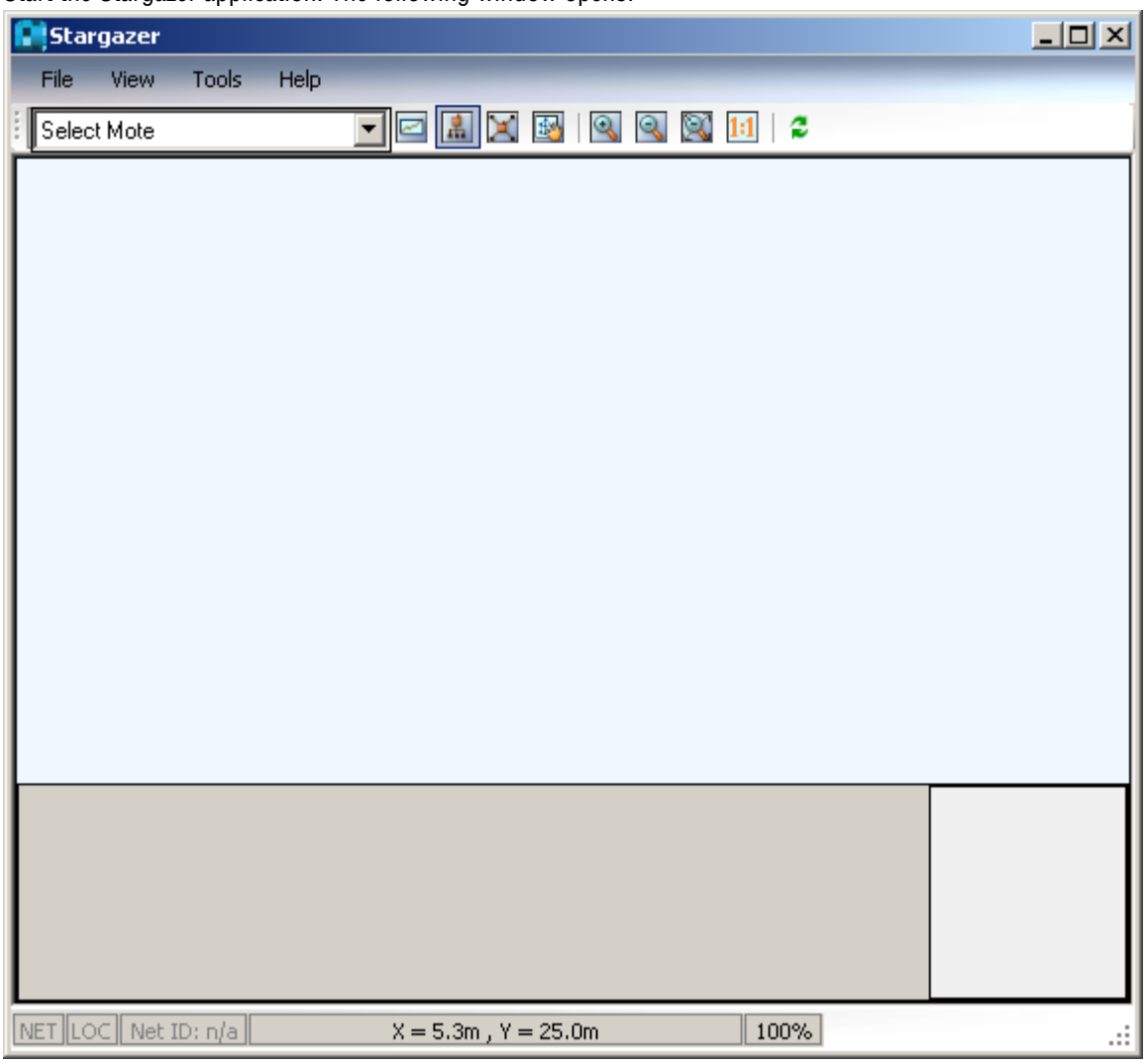
7.2 A First Network

7.2.1 Overview

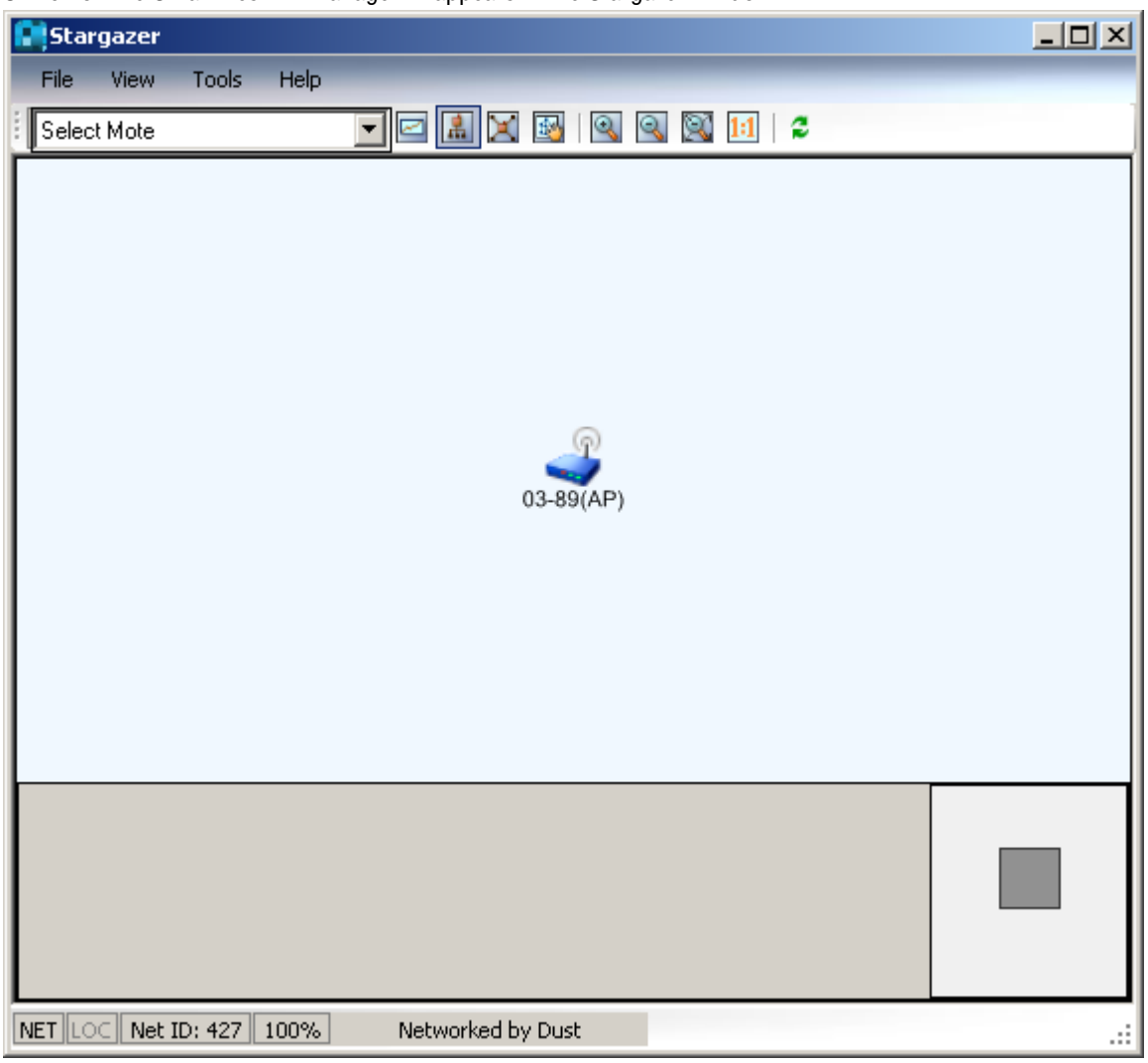
In this tutorial, you will use the Stargazer GUI and a SmartMesh IP Manager to watch a mesh network form. It assumes that you have previously followed the installation steps outlined earlier in this document.

Building the network

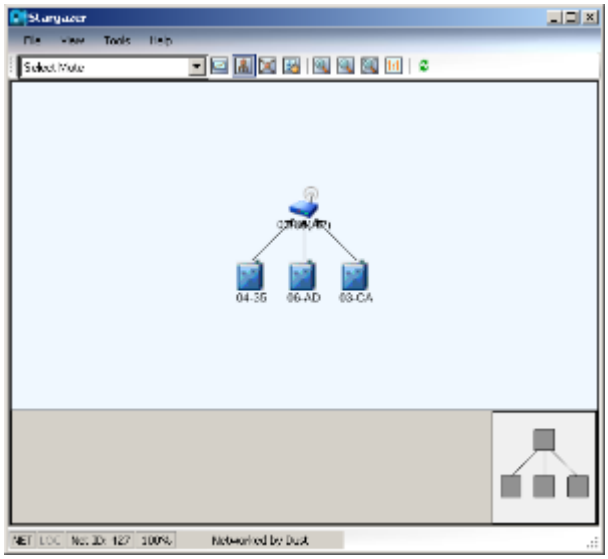
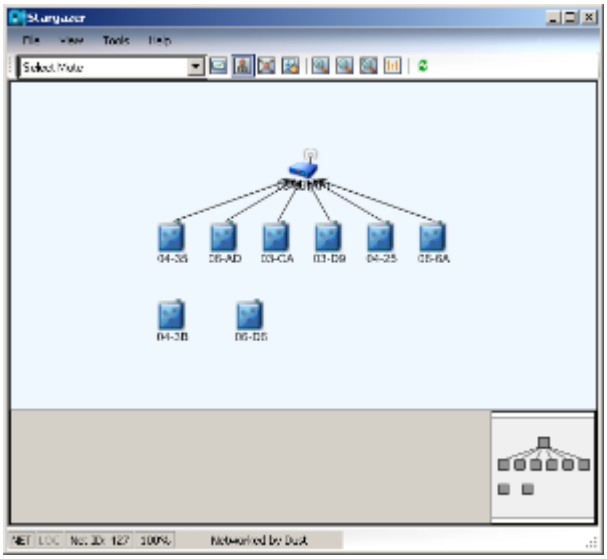
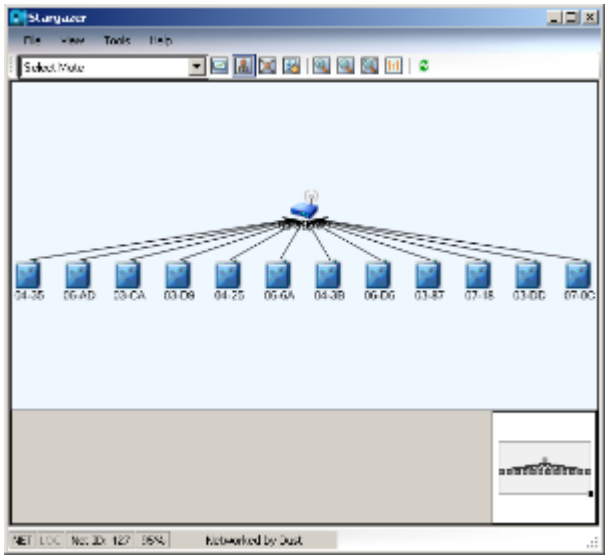
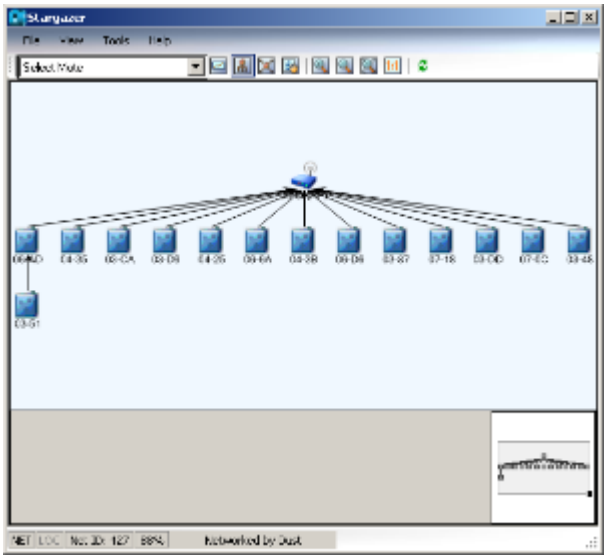
1. Switch off all the motes and manager.
2. Connect the SmartMesh IP Manager to you computer.
3. Start the Stargazer application. The following window opens:



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:

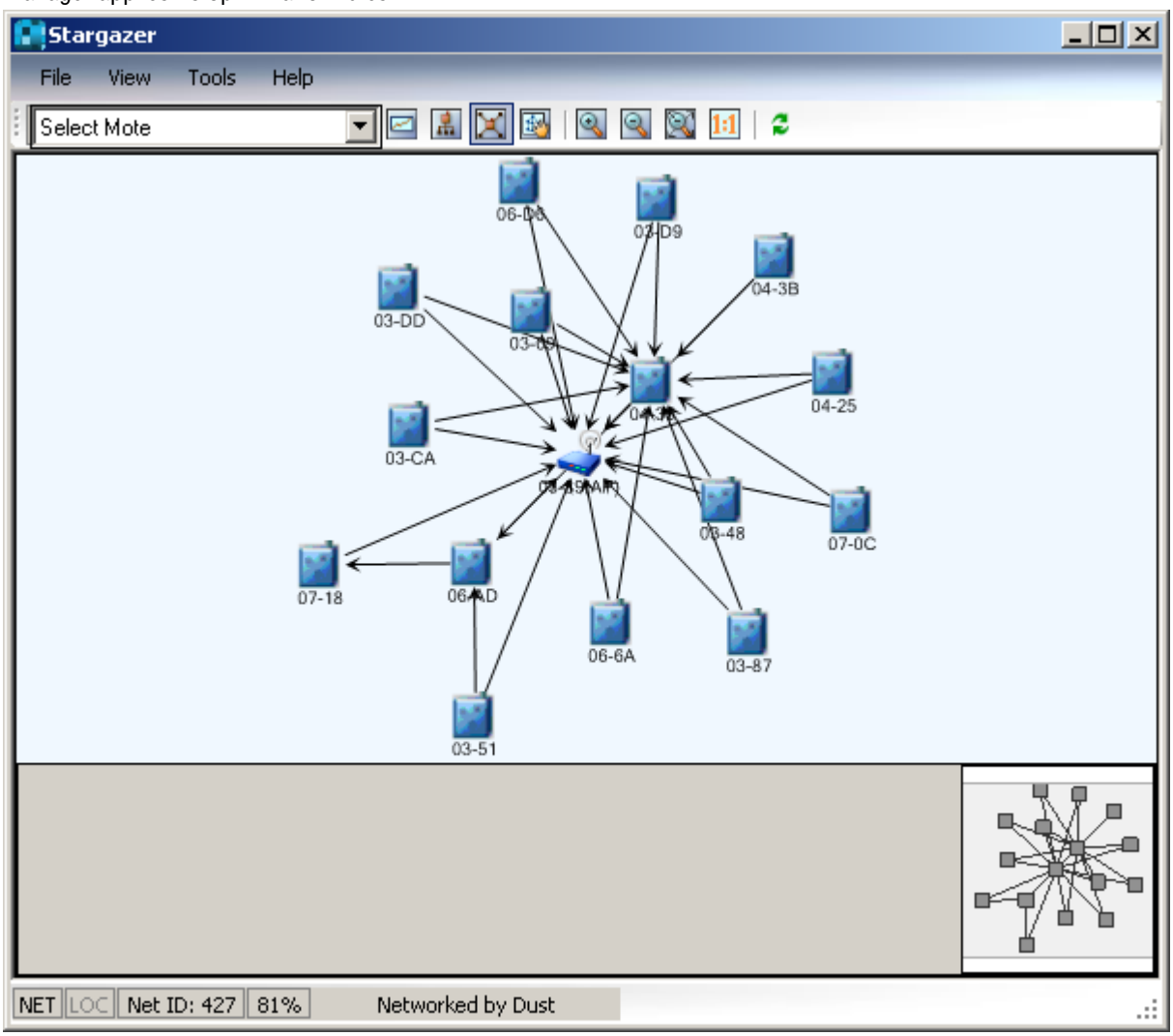
3 motes	8 motes
	
12 motes	**14 motes**
	

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

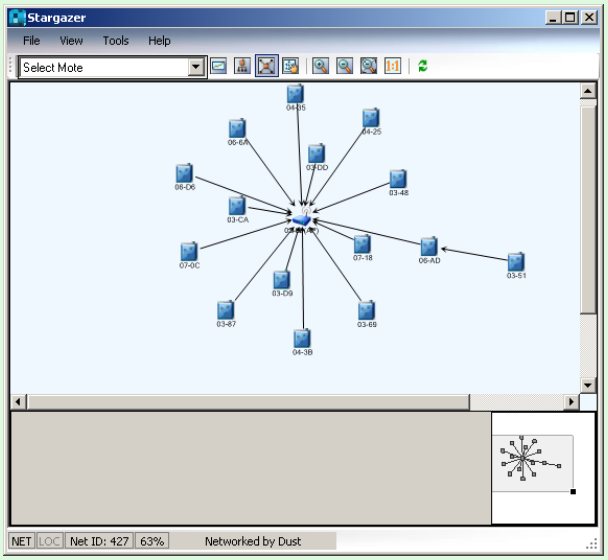
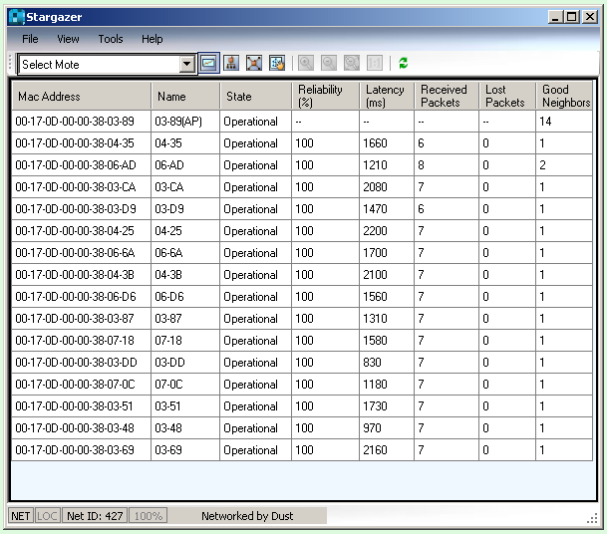
```
> set mode master
> reset
```

> ⚠ Motes in the starter kit (DC9000 & DC9021) 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.

6. 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.

7.2.2 Common Problems

No motes 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.

7.3 Interacting with the Manager

7.3.1 Introduction

In this step, you will interact with the SmartMesh IP Manager that was supplied with your kit (see table 1) 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:
   login
   logout
```

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

```
> login user
```

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. Once logged in, you can use the `help` command to see additional commands - these are discussed in detail in the SmartMesh IP Manager CLI Guide:

```
> help
help <command>
Commands:
   mset
   mget
   ...
```

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.

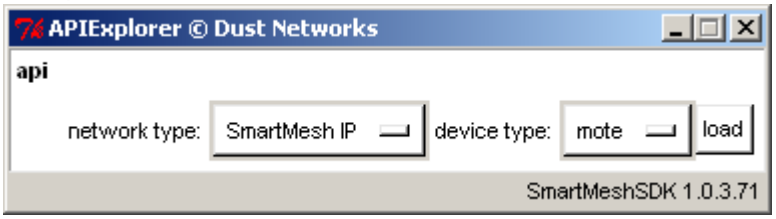
Installation instructions and detailed descriptions of the various sample applications in the SMSDK can be found on dustcloud.org.

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.

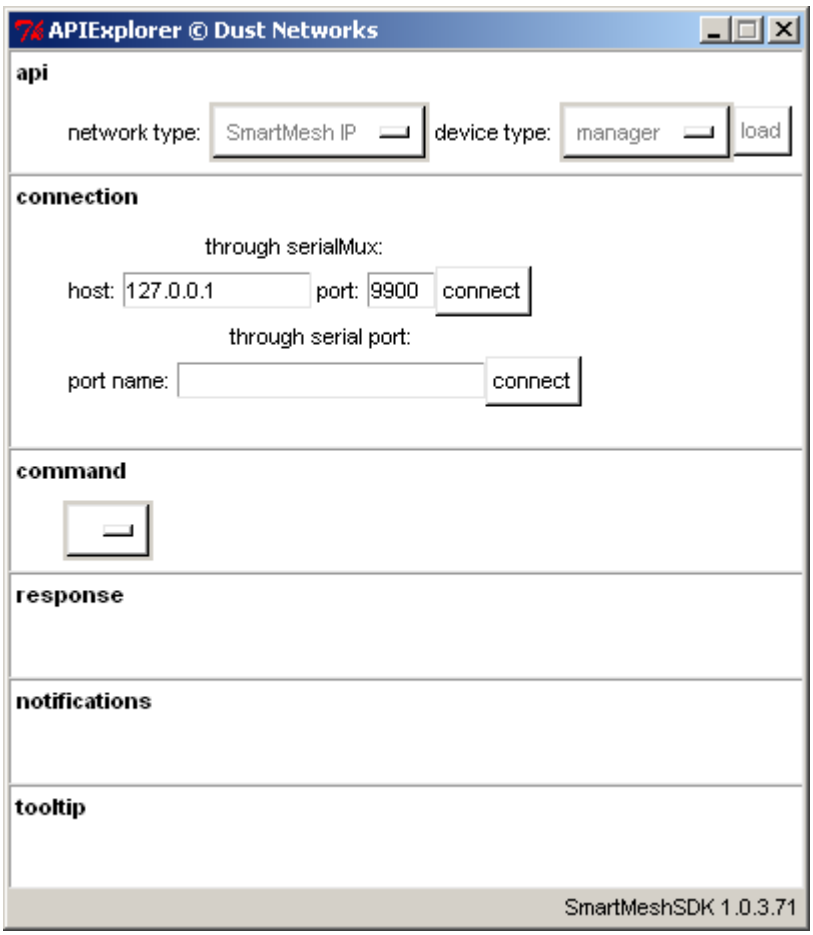
Connect to the Manager

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.

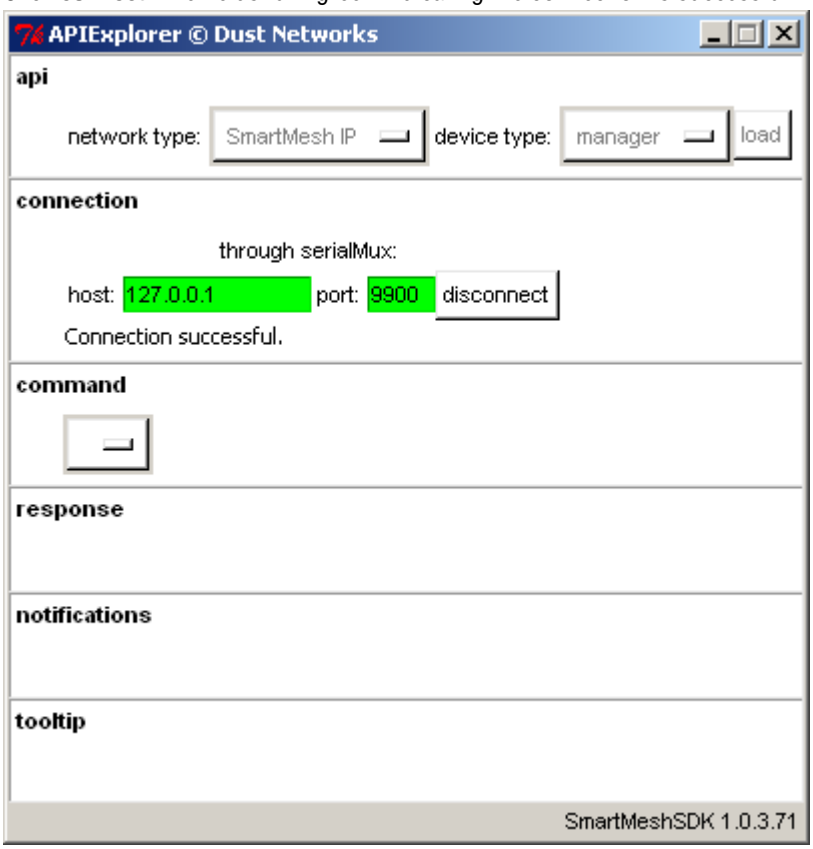


5. In the **connection** frame, you are presented with two options to connect to the SmartMesh IP Manager; we will connect through the SerialMux.

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

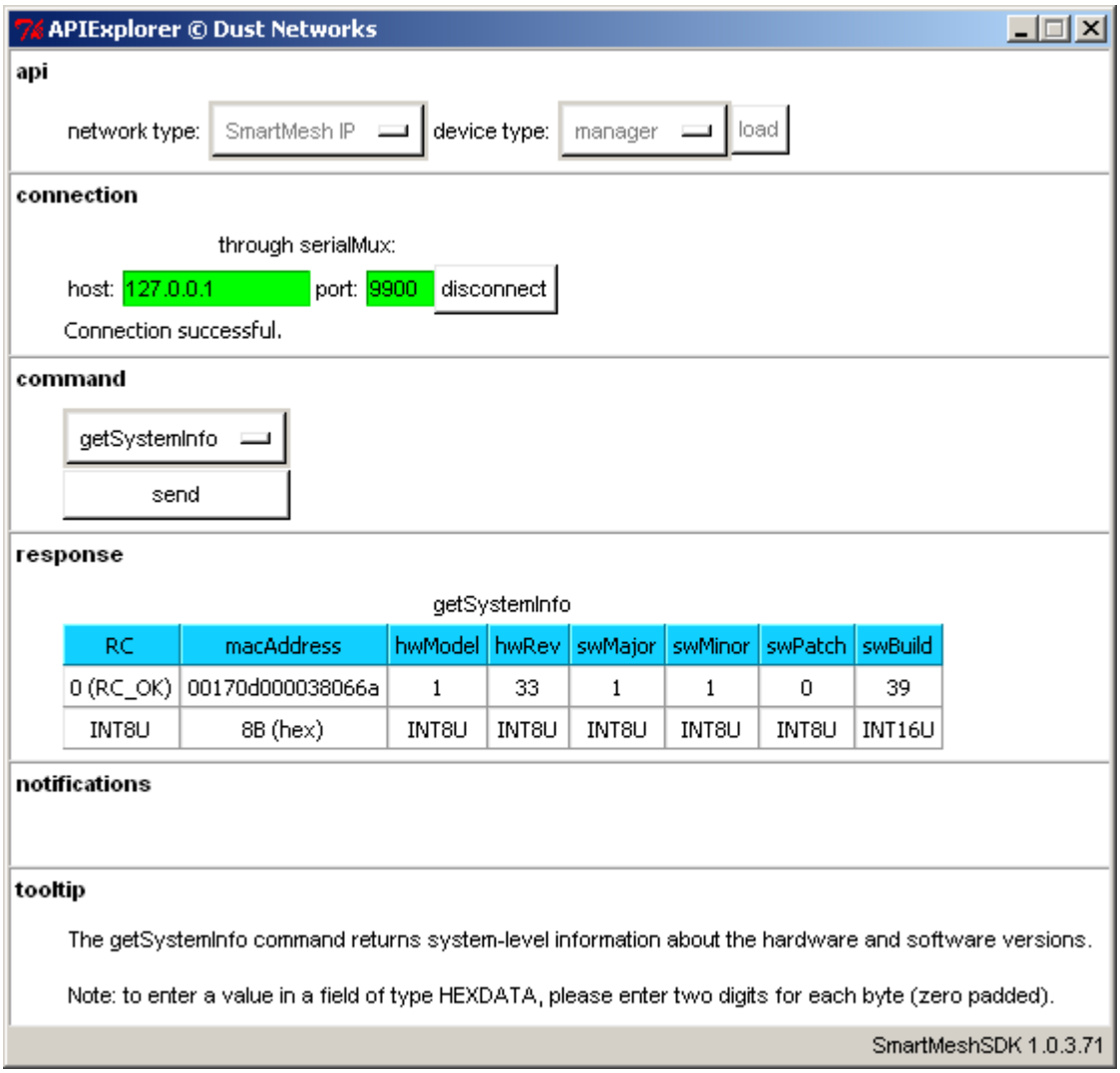
   - `host`: `127.0.0.1`
   - `port`: 9900

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



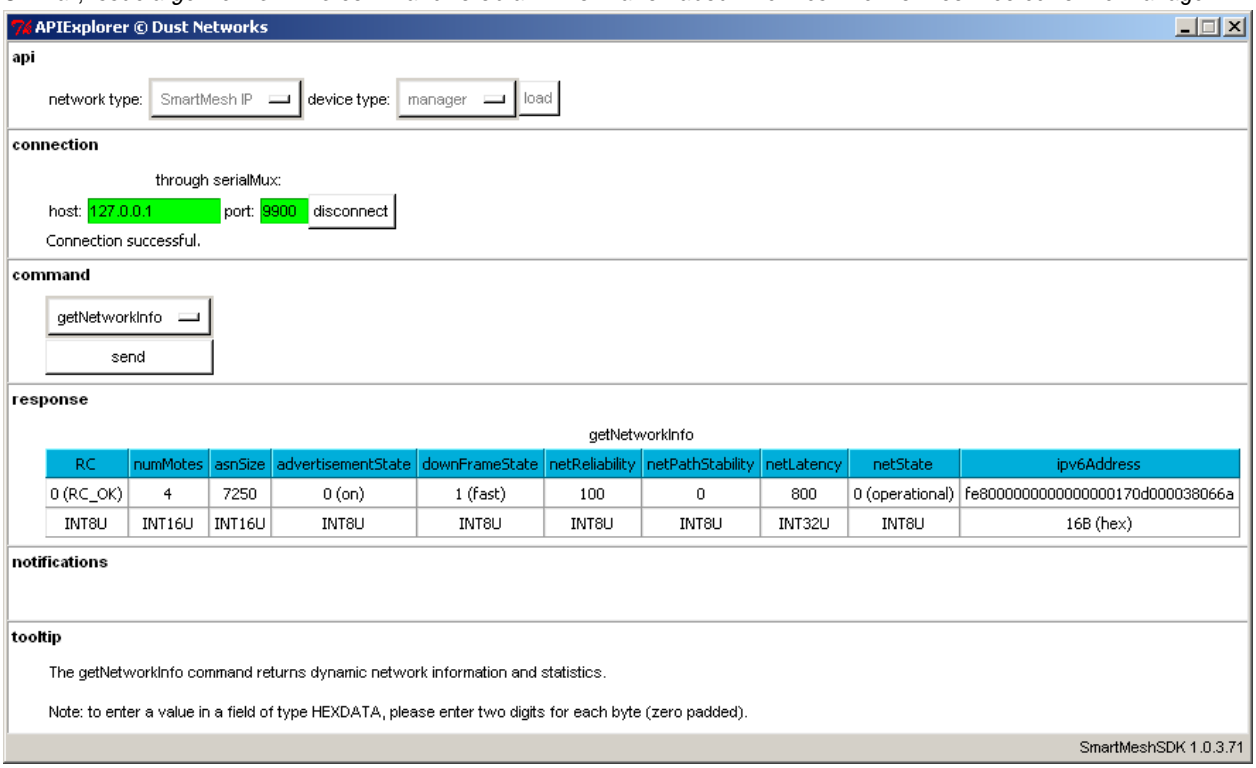
Obtain Information about the Manager and the Network

1. 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.



2. 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.



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

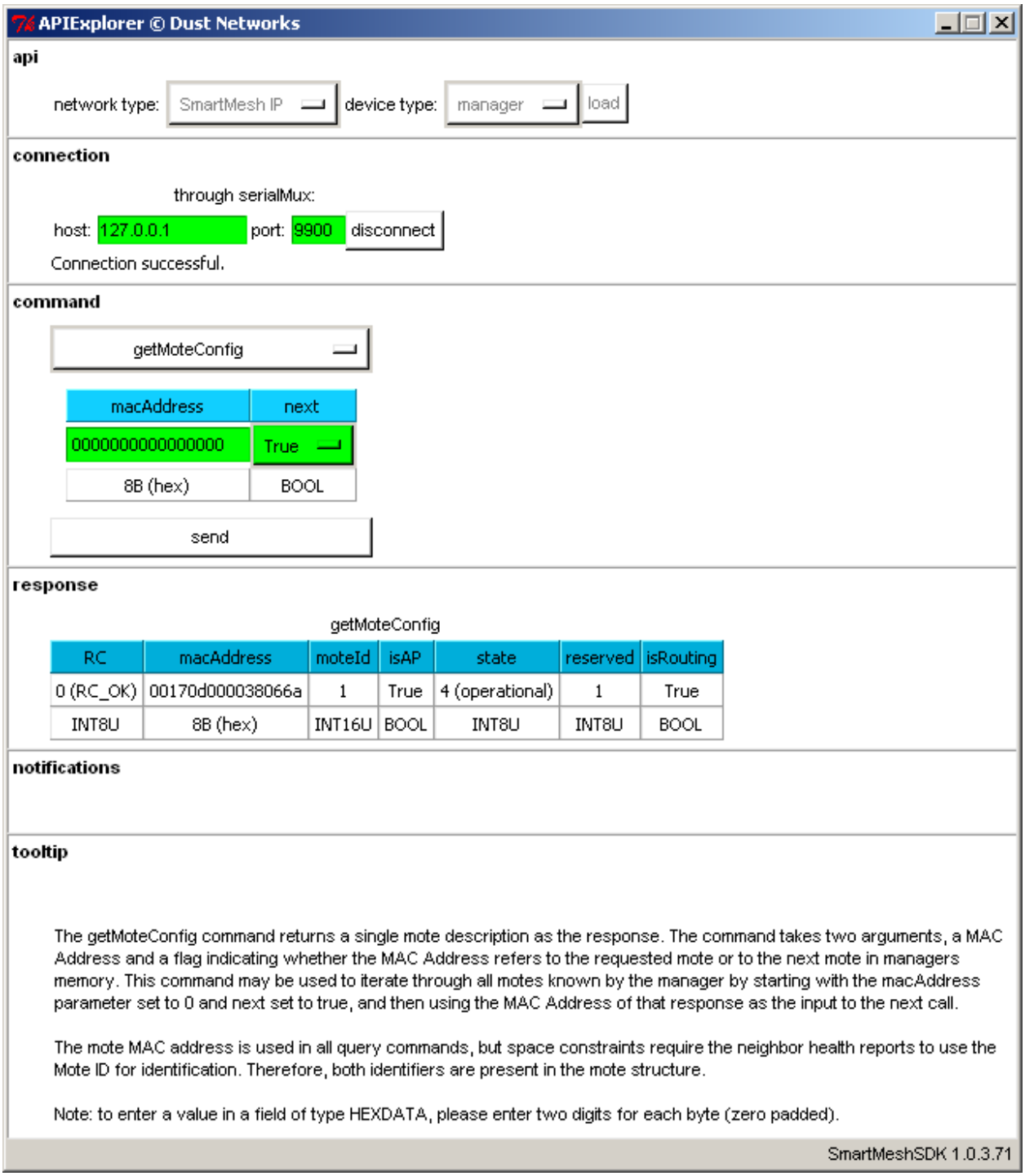
Consult the SmartMesh IP Manager API Guide for details about all commands and fields.

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`).



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 .

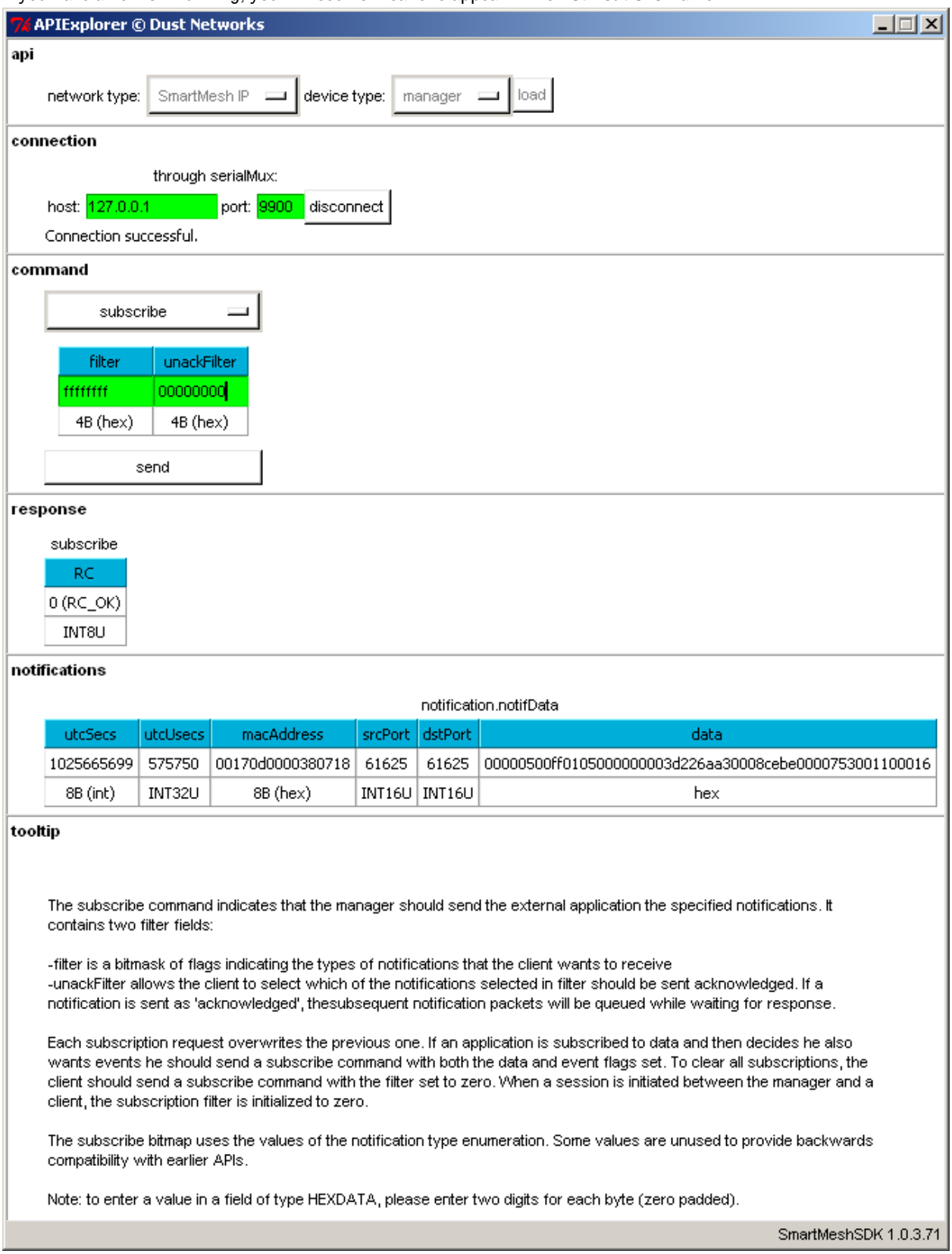
By default, the manager will *not* send you notifications; you need to use the *subscribe* command to specify which types of notifications you'd like to receive.

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.



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

7.3.2 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 possibe 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.

7.4 Interacting with a Mote

7.4.1 Overview

In this step, you will interact with the SmartMesh IP Mote that was supplied with your kit (DC9003A-B, DC9018A-B, or DC9018B-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.

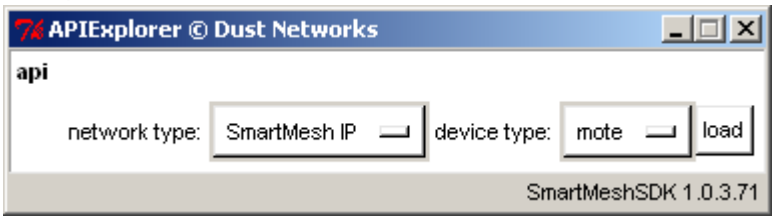
Connect to the Device

1. Make sure your SmartMesh IP Mote is connected to your computer.
   Make sure your SmartMesh IP Mote is operating in **slave**mode by issuing the following command on the mote CLI:

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

   > ⚠ Motes in the starter kit (DC9000 & DC9021) 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.

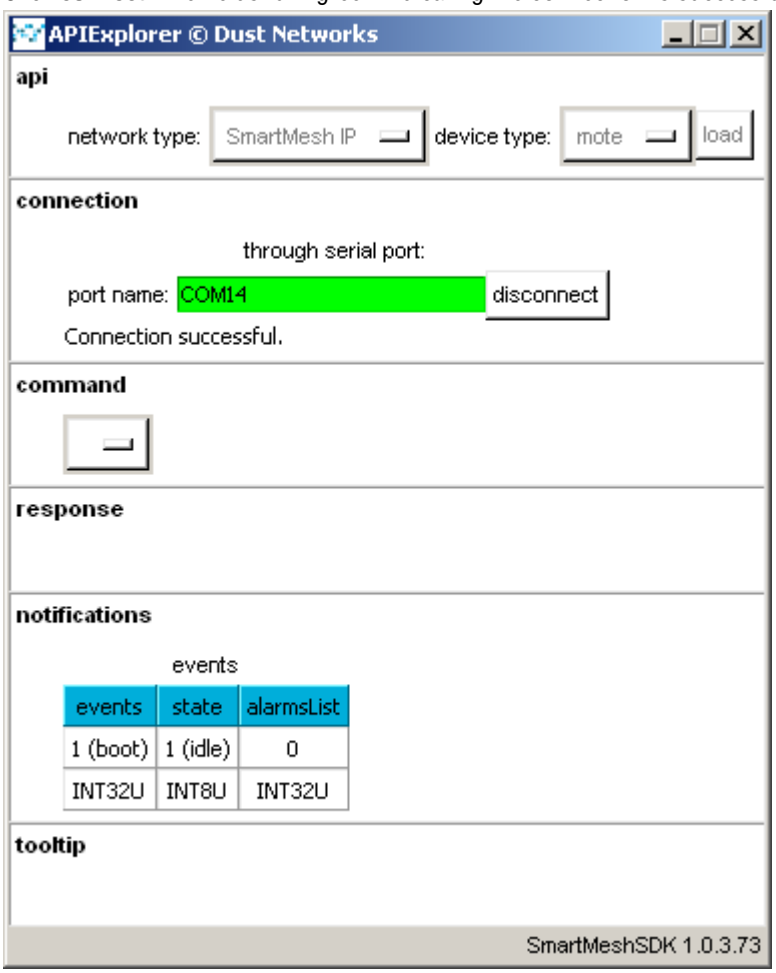
   

3. Tell the application you want to connect to a SmartMesh IP Mote by selected the following:
   - `network type`: `SmartMeshIP`
   - `device type`: `mote`

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.



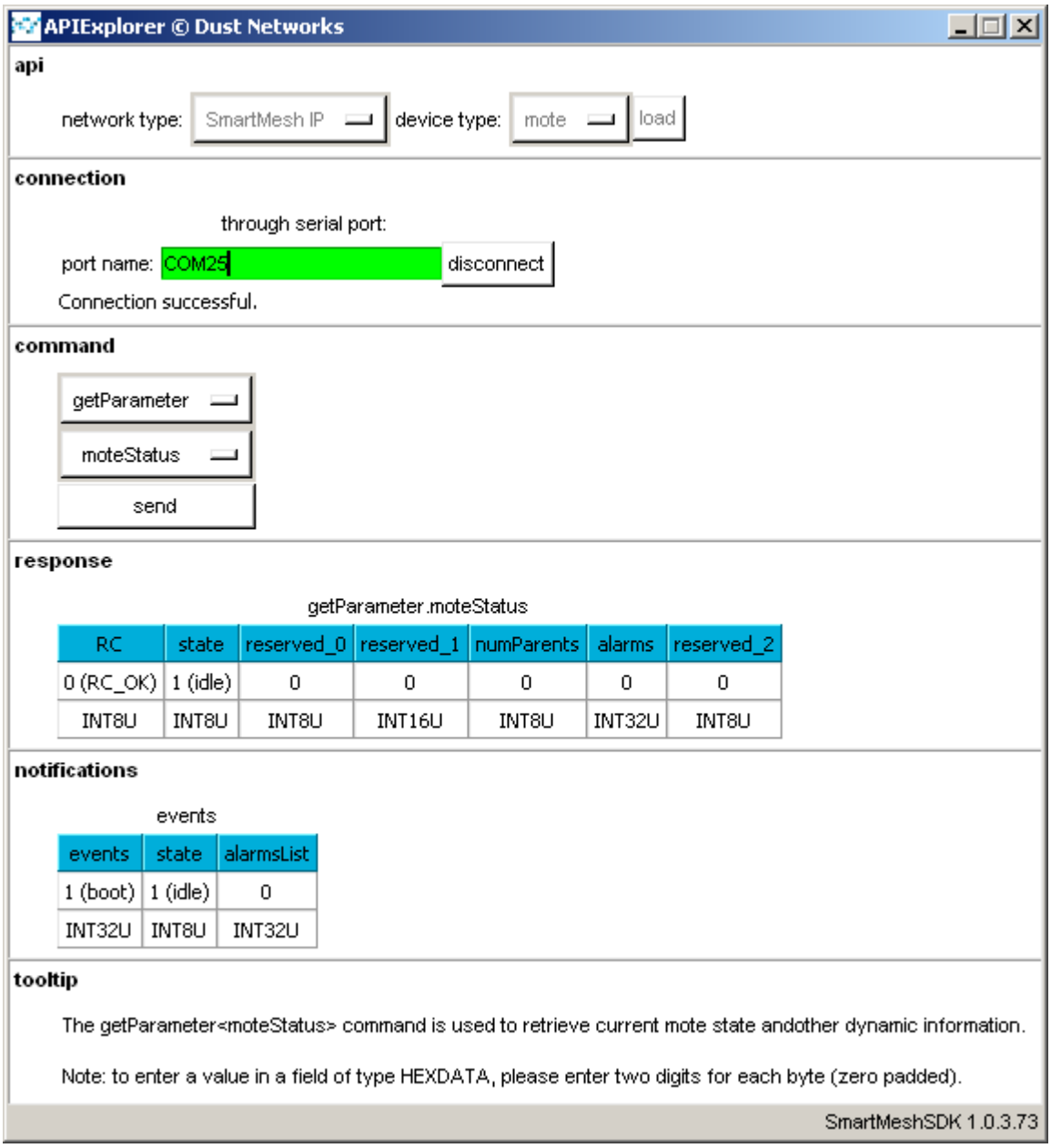
Getting Notifications from the Mote

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 APIExplorer application. This explains the boot mote event displayed in the screen shot above.

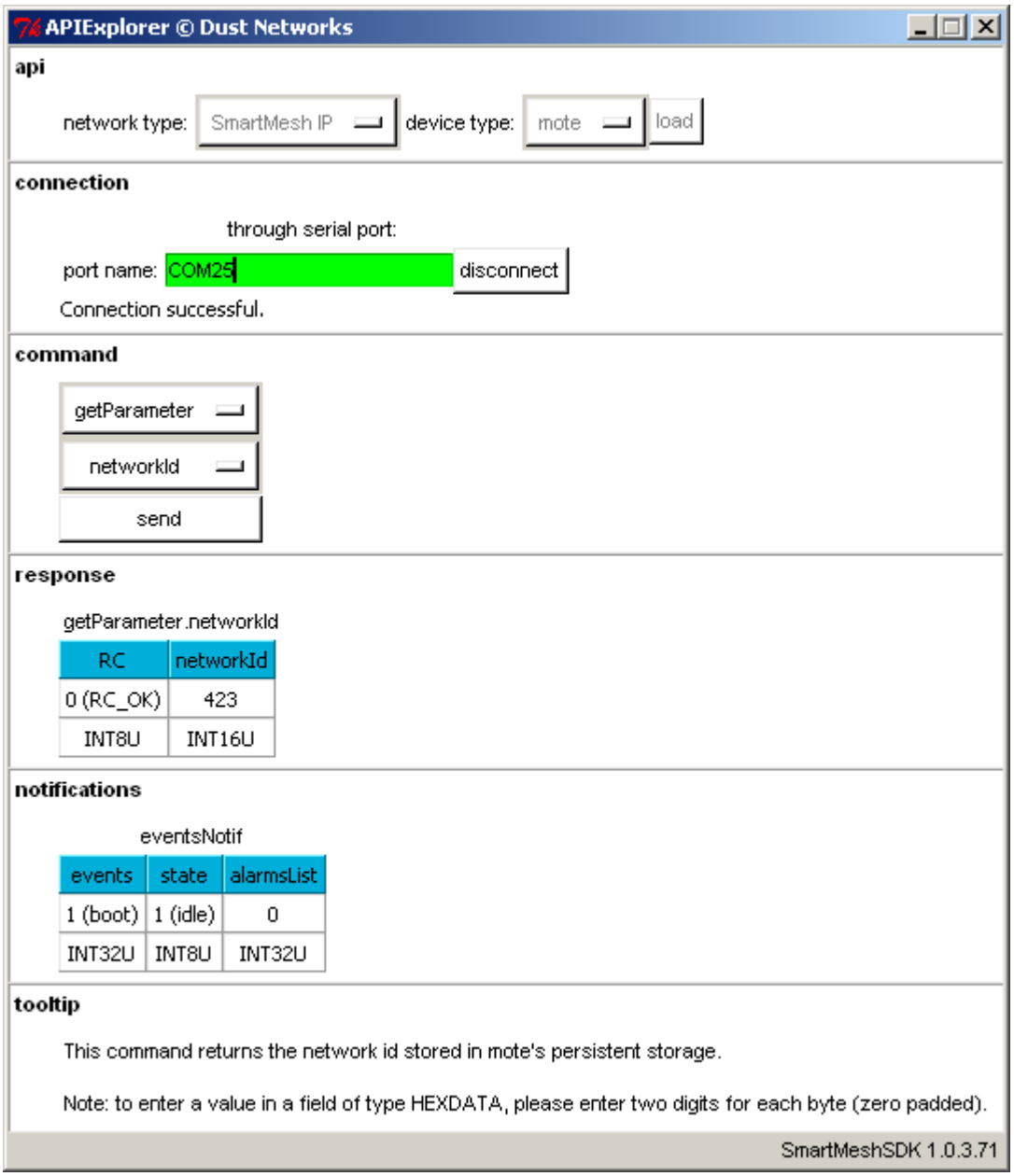
Obtain Information About the Mote

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



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

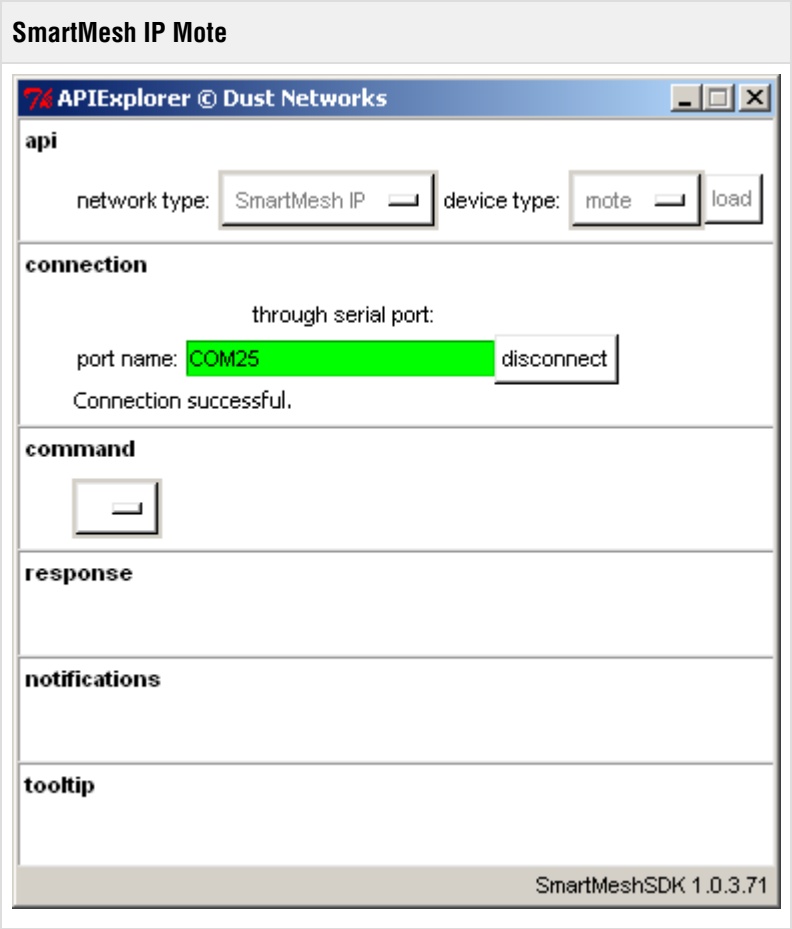
3. Similarly, issue a *getParameter.networkId* command to verify that your SmartMesh IP Mote is configured with the correct network ID.



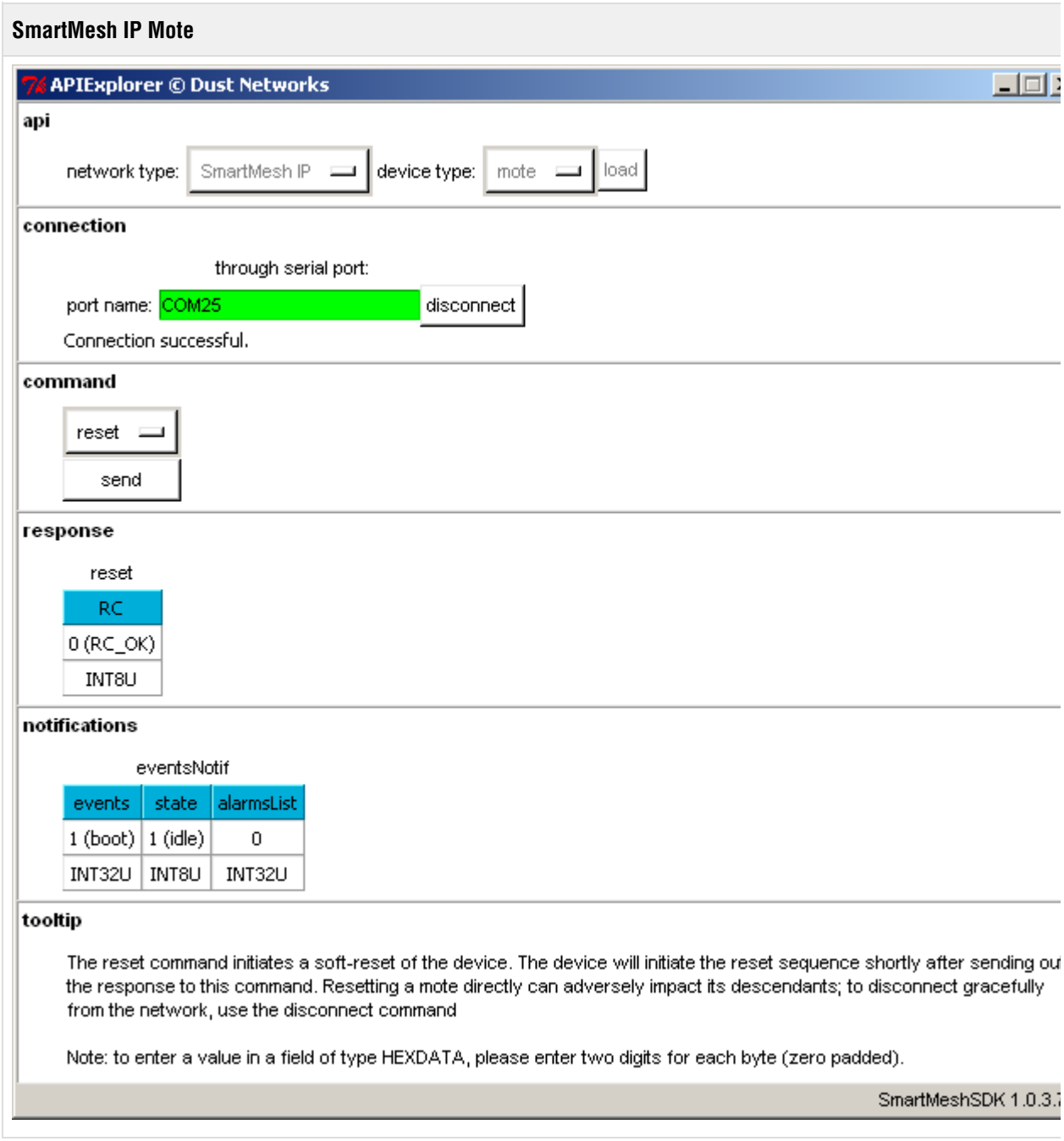
Have the Mote Join the Network

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.

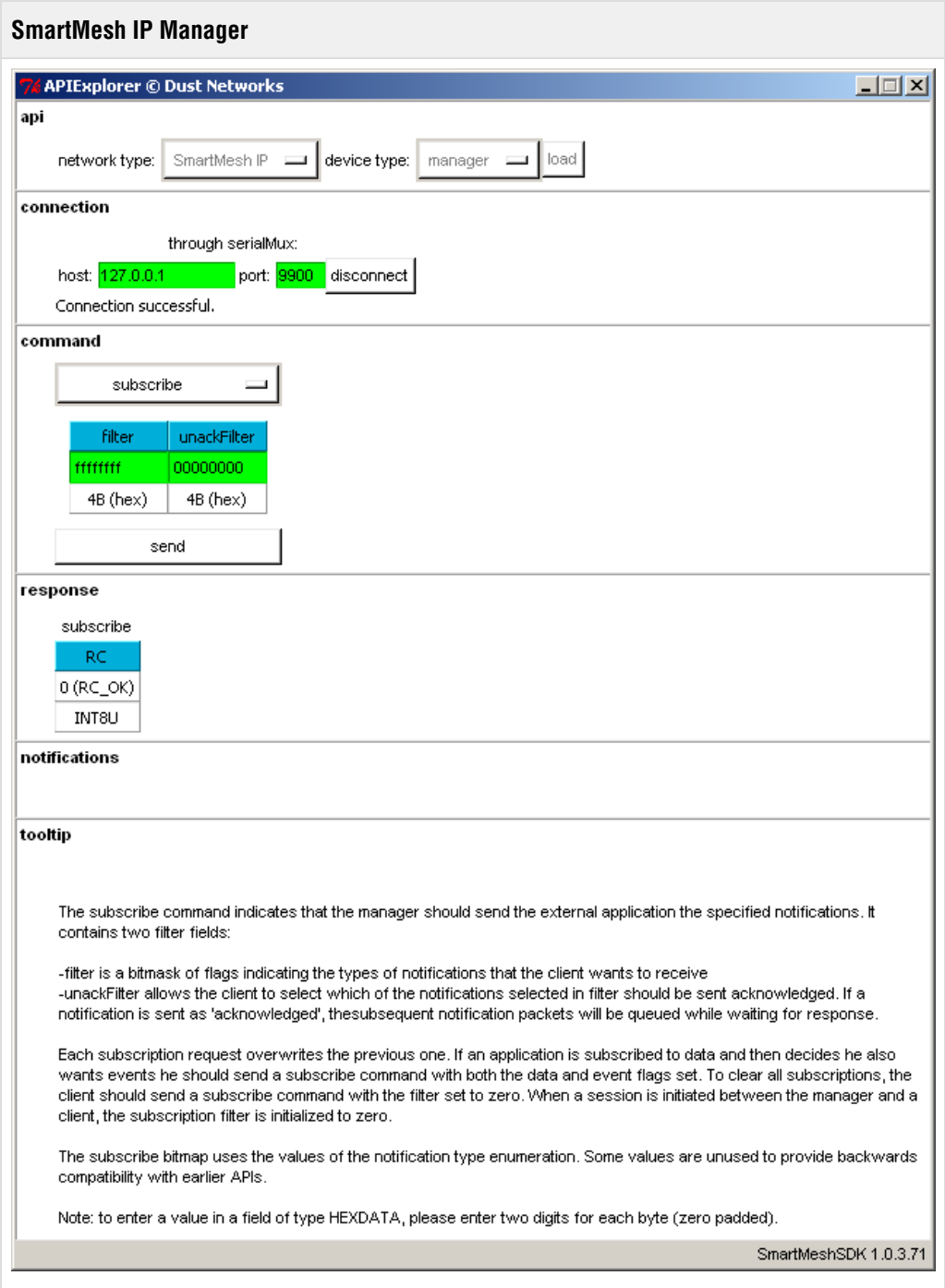
3. For easier reading, we recommend that you display the two APIExplorer windows side-by-side.



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.



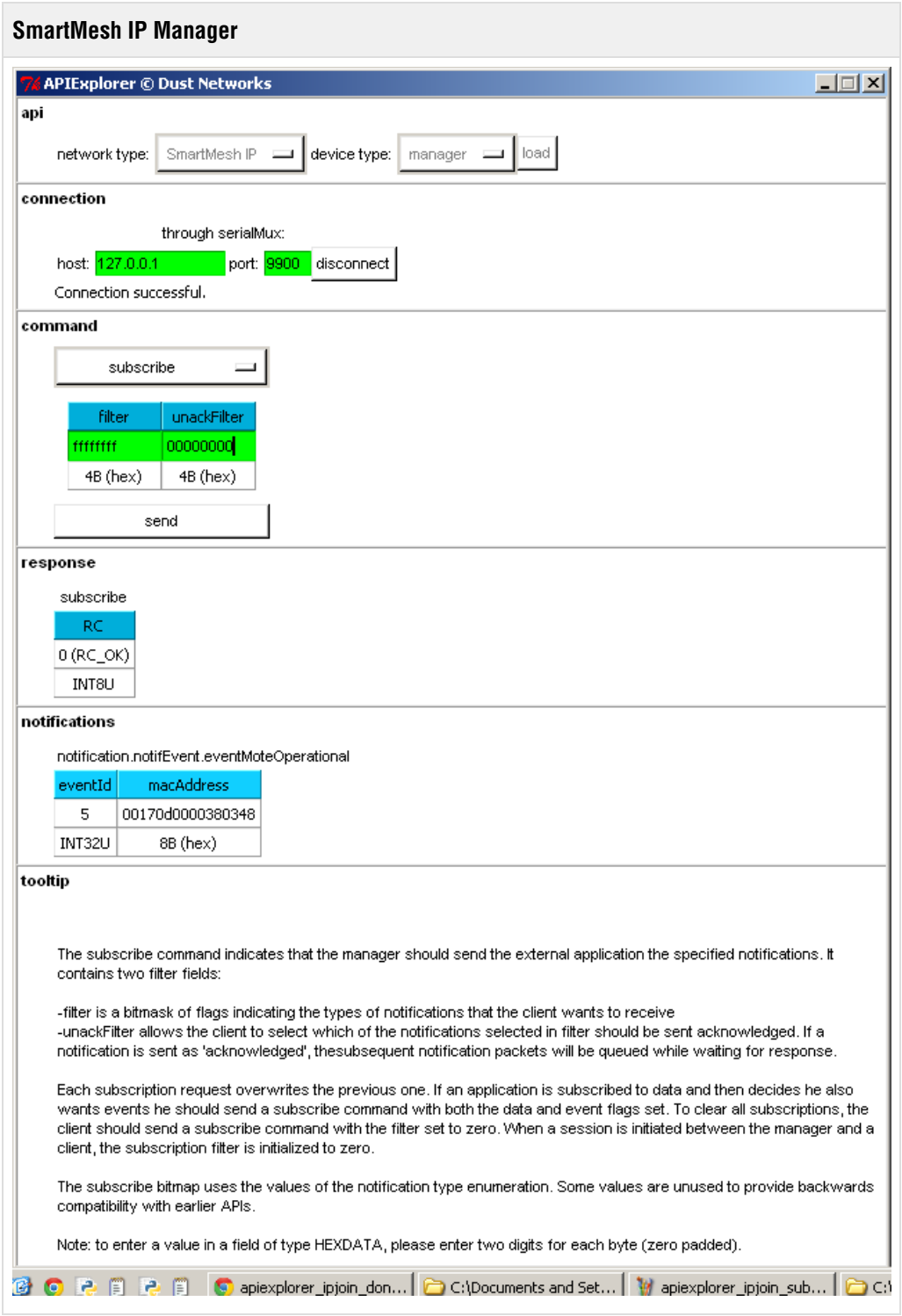
5. At the SmartMesh IP Manager, *subscribe* to all notifications. If you have a network running, you may receive notifications from time to time.



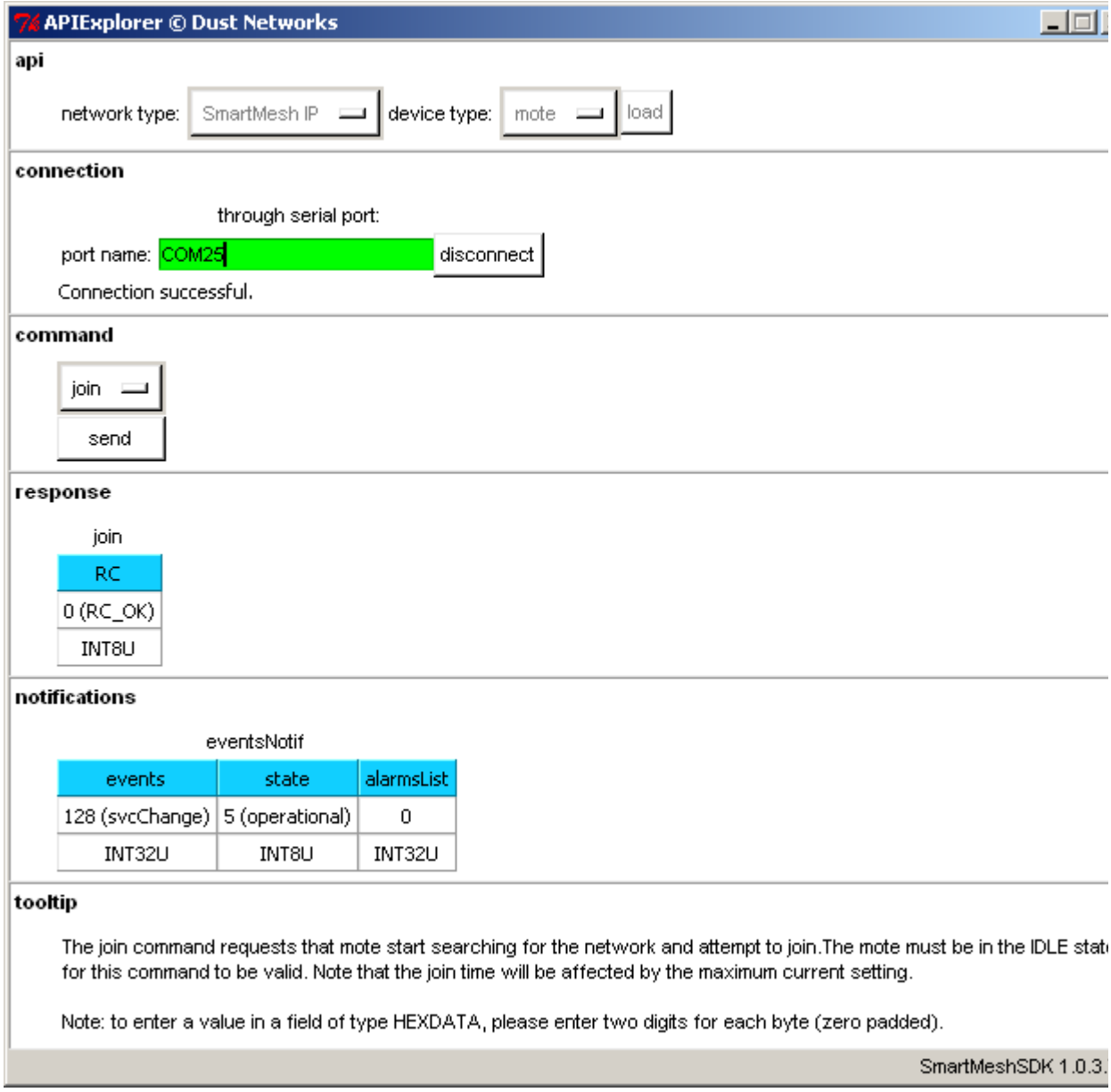
6. At the SmartMesh IP Mote, call the *join* command. This causes it to join the network, which will trigger the following notifications on both the SmartMesh IP Mote and the SmartMesh IP Manager:

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 Mote

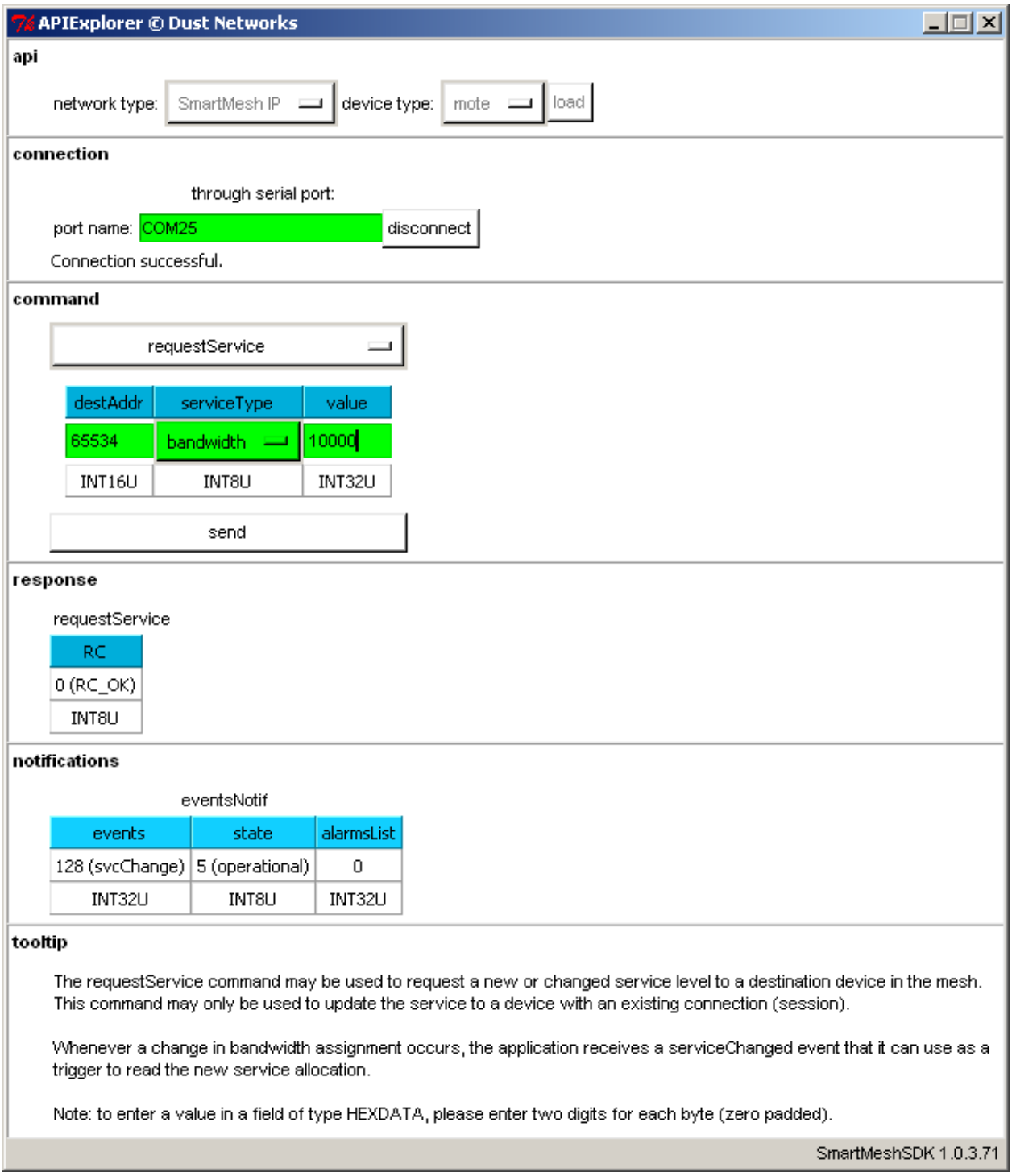


Have the Mote Request a Service

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

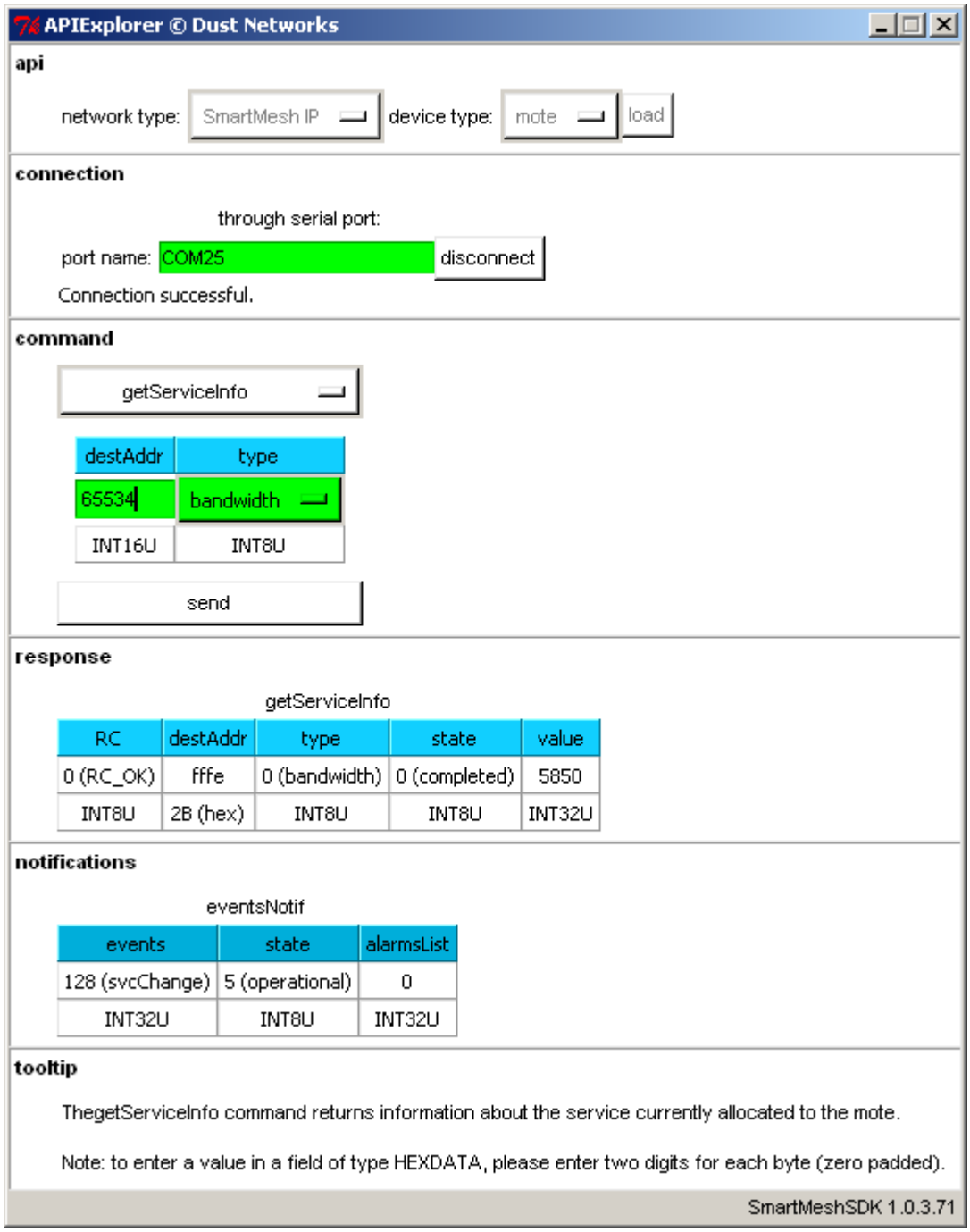
1. At the SmartMesh IP Mote, issue a *requestService* command to request bandwidth to send one packet every 10s to the SmartMesh IP Manager:
    - **destAddr** to `65534` (the well-known address of the SmartMesh IP Manager)
    - **serviceType** is `bandwidth`
    - **value** is `10000` (the period between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

2. 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.



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

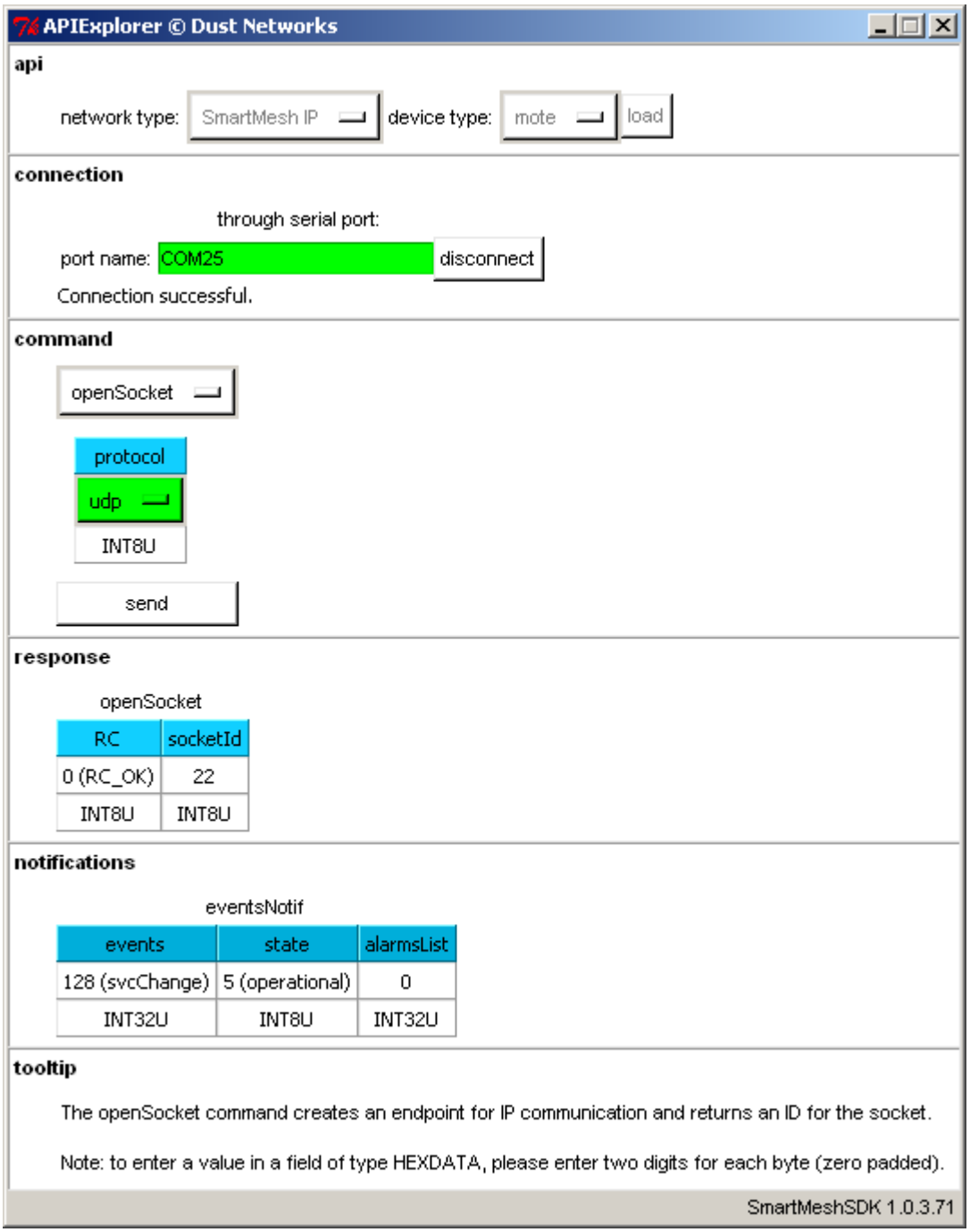


The bandwidth allocated is expressed in time between transmits, *i.e.* a small number indicates a larger bandwidth. The SmartMesh IP Manager uses a safety margin when allocating bandwidth, so the period really allocated is smaller that the one requested.

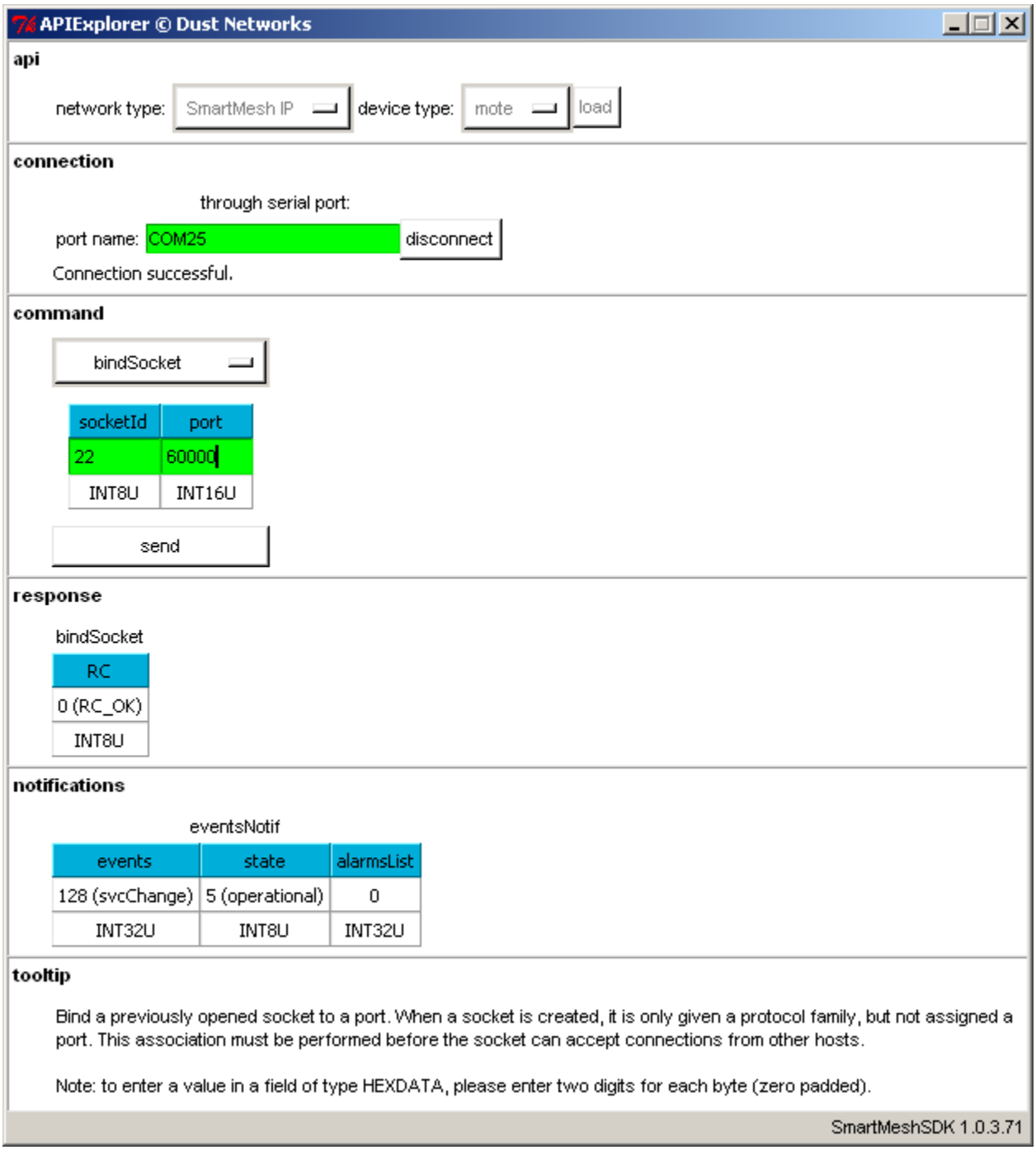
Have the Mote Prepare a UDP Socket

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**.



2. 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`).



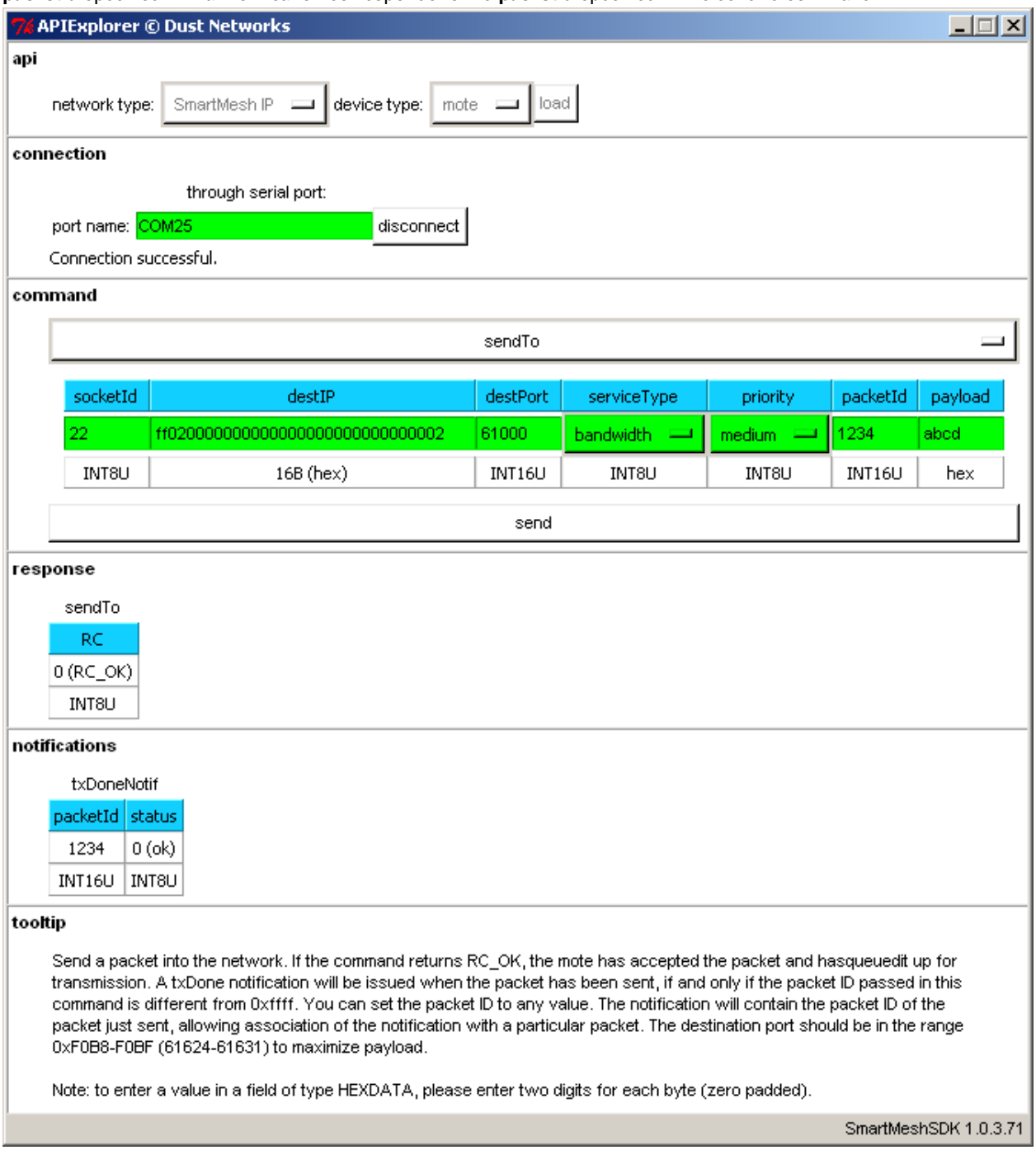
> ⚠ You cannot send data without opening and binding a socket. This is because the socket is used to identify the UDP source port of the data you are sending.

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 `ff0200000000000000000000000000002`.

2. 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.



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

APIExplorer © Dust Networks

api

network type: SmartMesh IP device type: manager load

connection

through serialMux:

host: 127.0.0.1 port: 9900 disconnect

Connection successful.

command

subscribe

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

send

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', thesubsequent 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

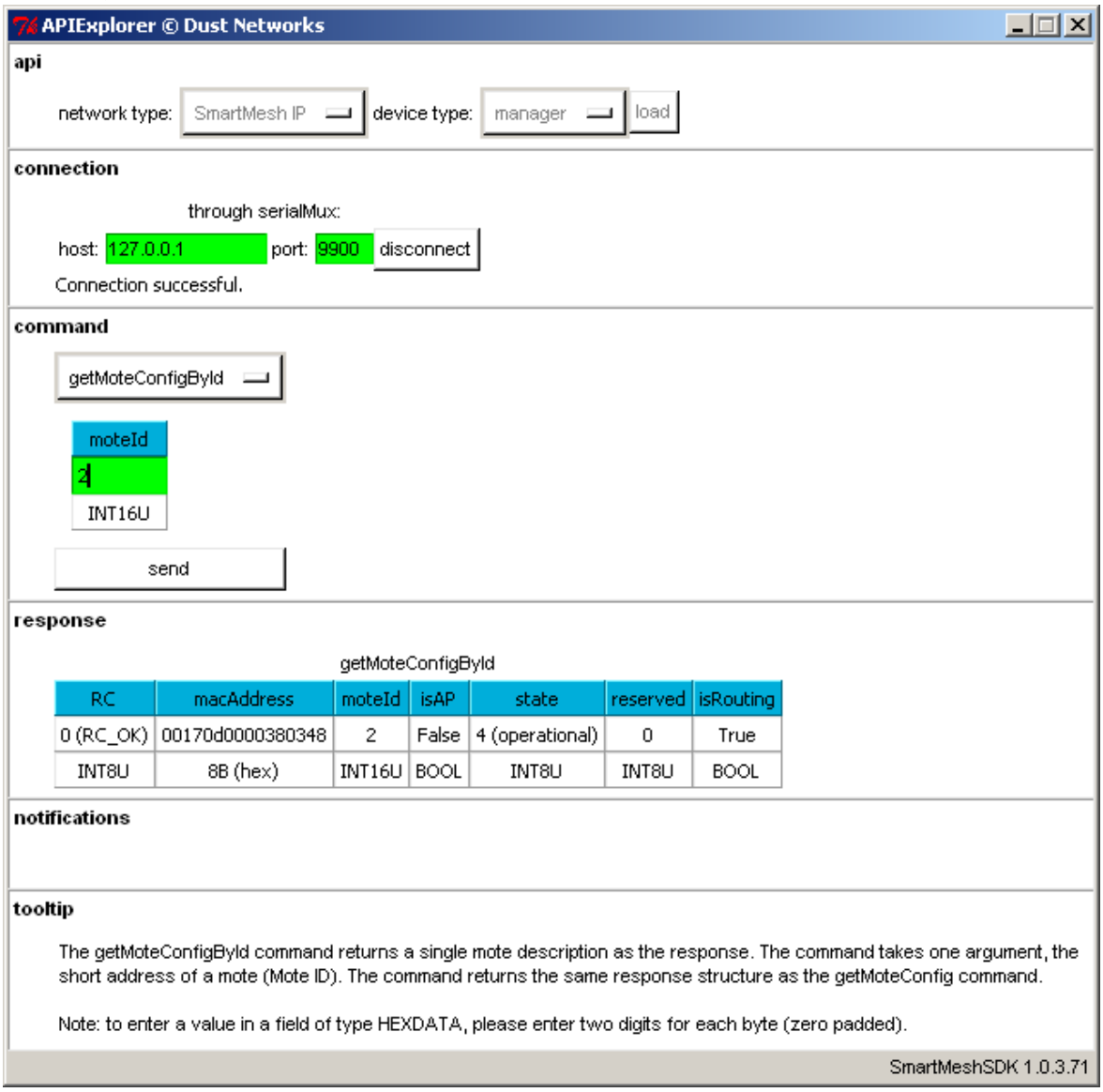
Log into the CLI interface of the Manager as described in Interacting With the Manager. Send a ping command to the mote 2. The mote's ID can be found by using the `sm` command as shown earlier. The Mote with ID=2 will respond with the round trip delay time, temperature and supply voltage.

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

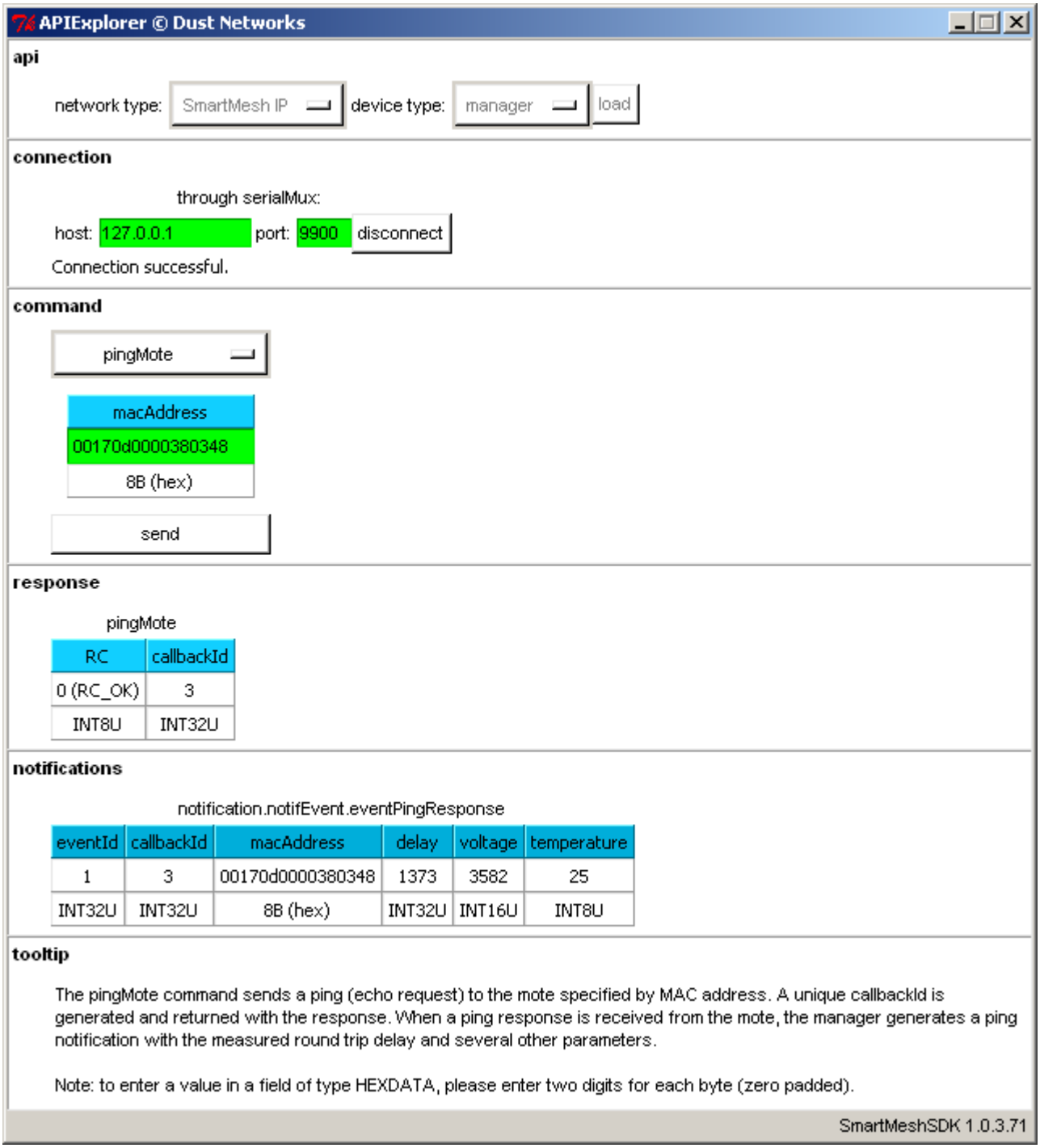
Via API

> ⚠ 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.



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.



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

7.4.2 Common Problems

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

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

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

The application won't connect to my mote

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

The table below details the serial setting for 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.

7.5 Advanced Topics

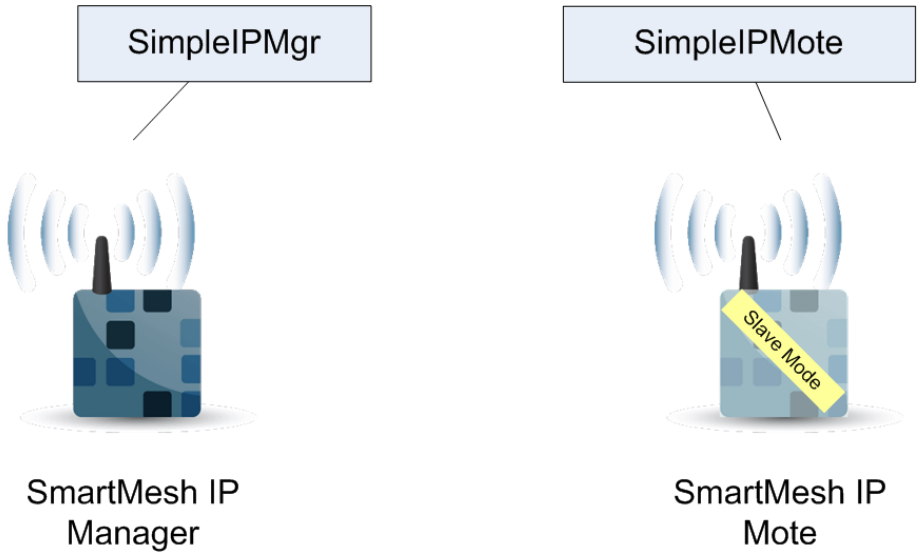
7.5.1 Exercise the API Programmatically

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, numMotes=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', 'cl
oseSocket', '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 ando
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.
```

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 possibe 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.

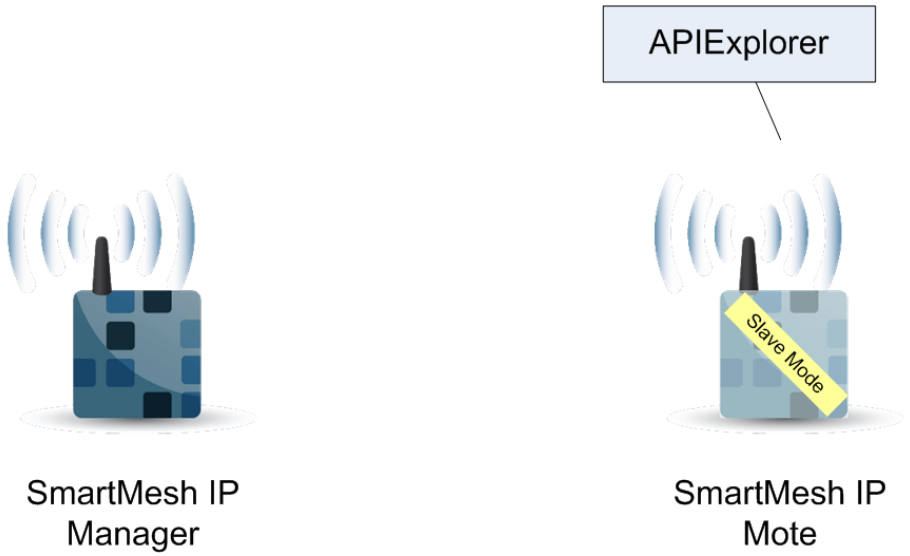
To change the configuration of your SerialMux, follow the Serial Mux Configuration guide.

7.5.2 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.

7.5.3 Upstream Communication

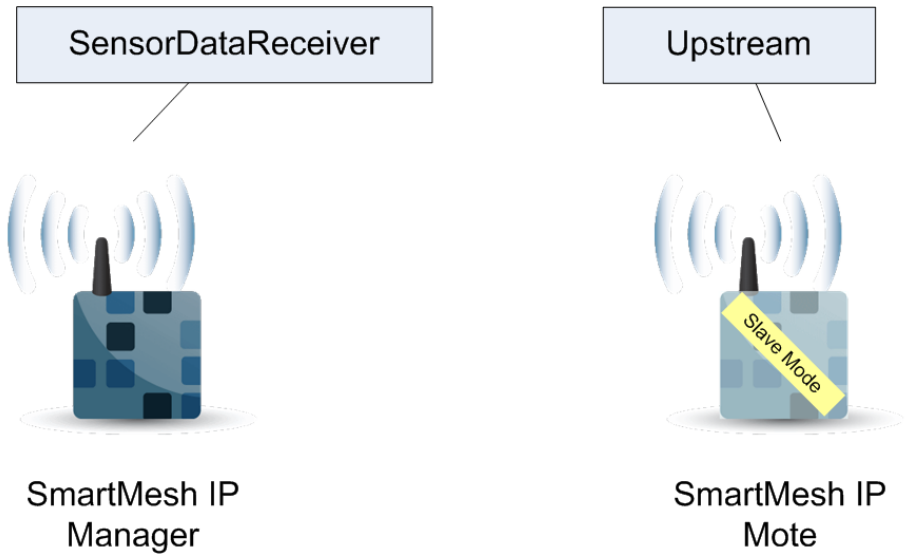
Definition

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

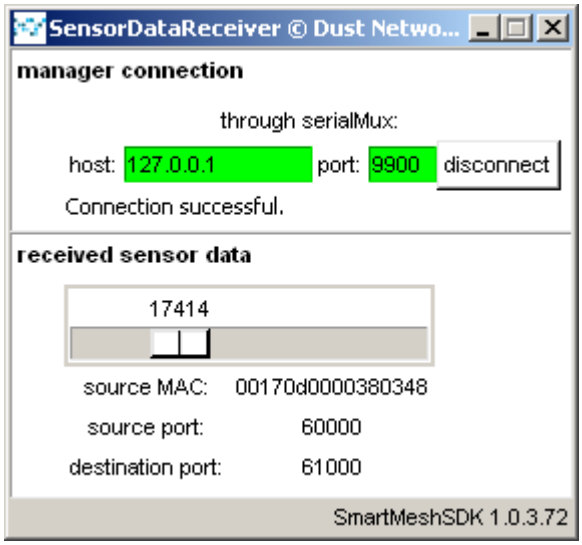


Steps

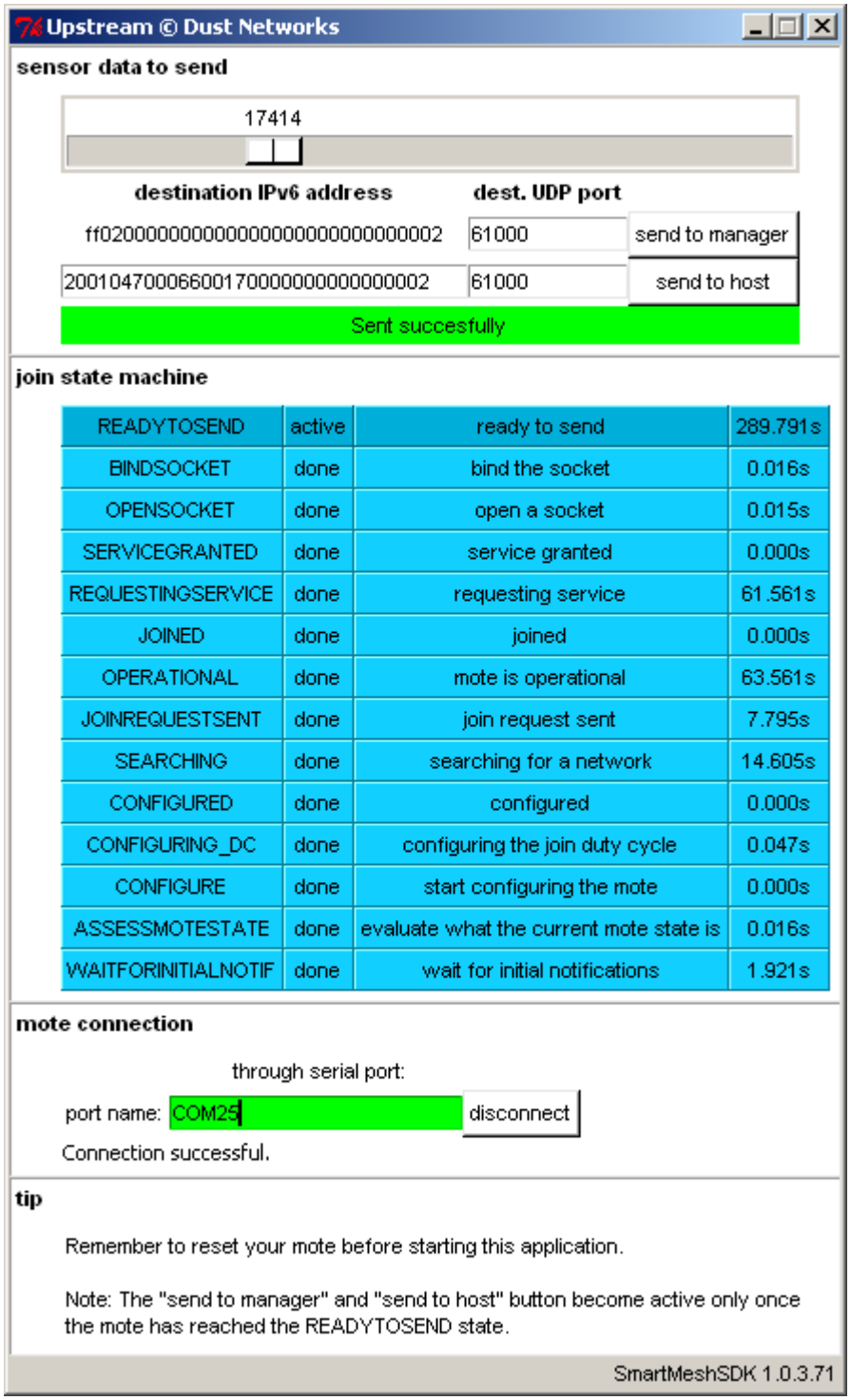
The UpStream application can be seen as a programmatic version of the steps followed in the Basic Walk-Through using the APIExplorer.

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



at theSmartMesh IP Mote



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

7.5.4 Downstream Communication

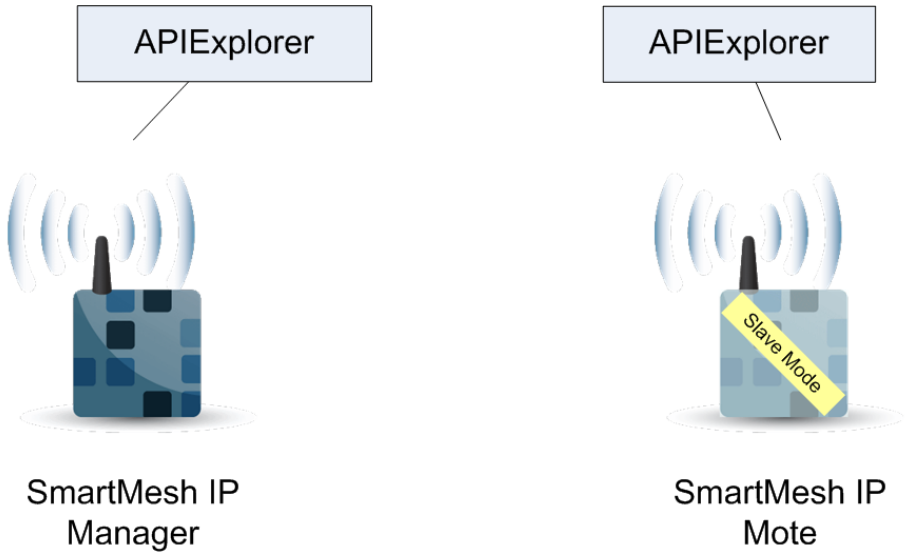
Definition

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



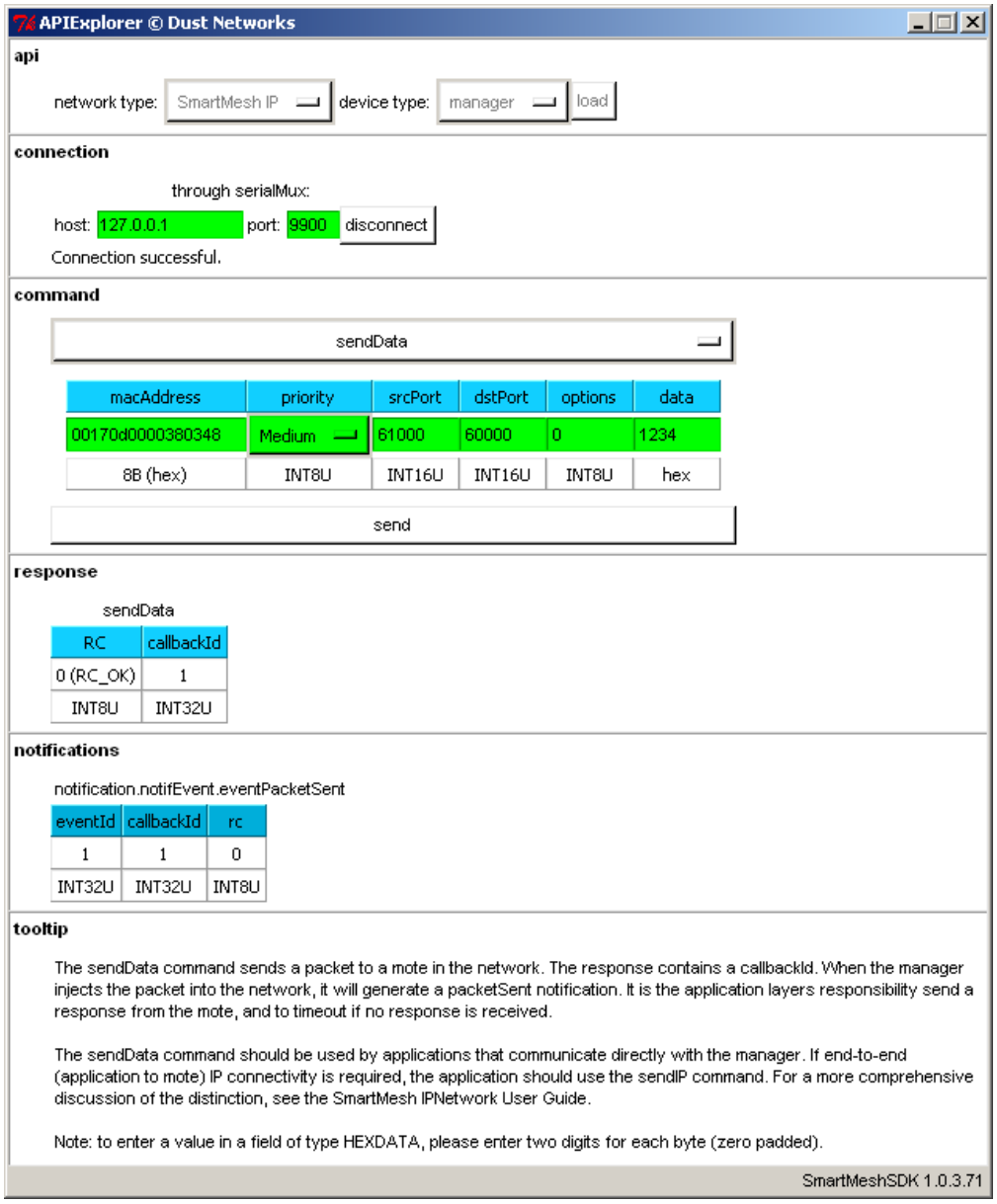
Steps

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.

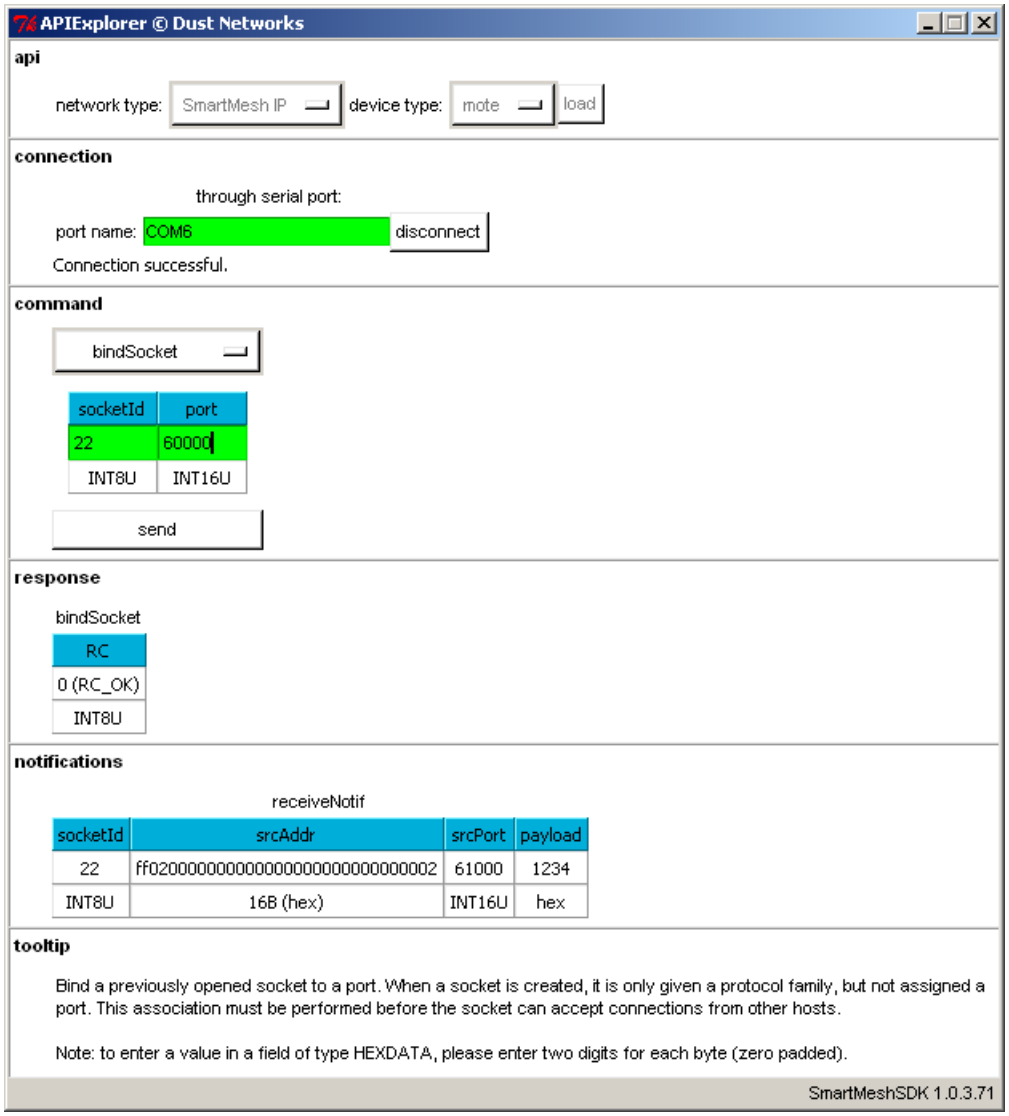
4. In that second APIExplorer, issue a *sendData* command and fill in the following data:
    - The **macAddress** field should contain the MAC address as returned by the SmartMesh IP Mote.
    - **priority:**`Medium`. Without any other traffic in the network, this priority does not change packet propagation.
    - **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



at the SmartMesh IP Mote



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 possibe 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

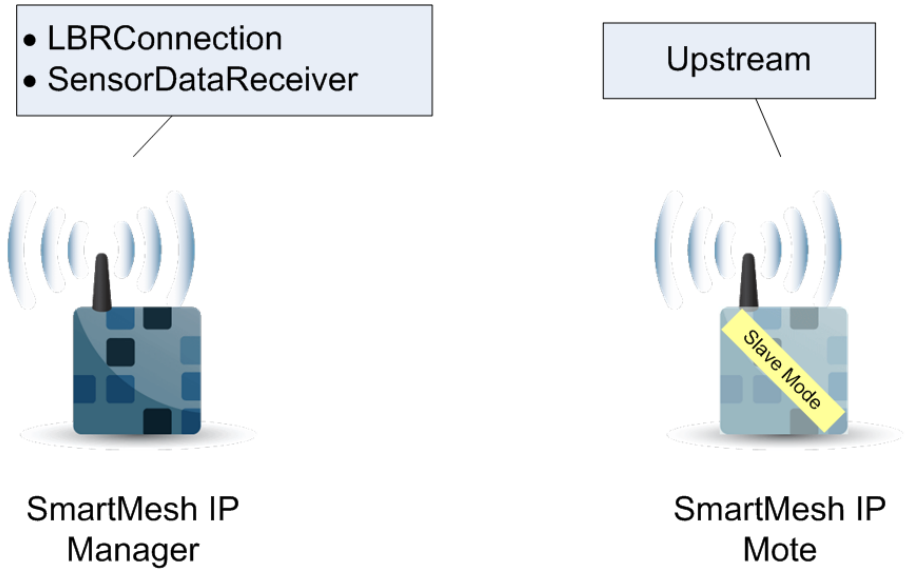
7.5.5 Internet Integration

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



Steps

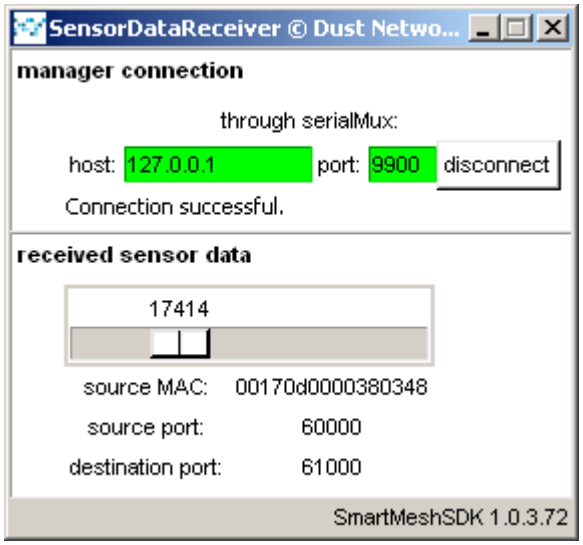
You will be connecting two different applications to your SmartMesh IP Manager at the same time. You therefore need to connect through the SerialMux.

Send data to the manager

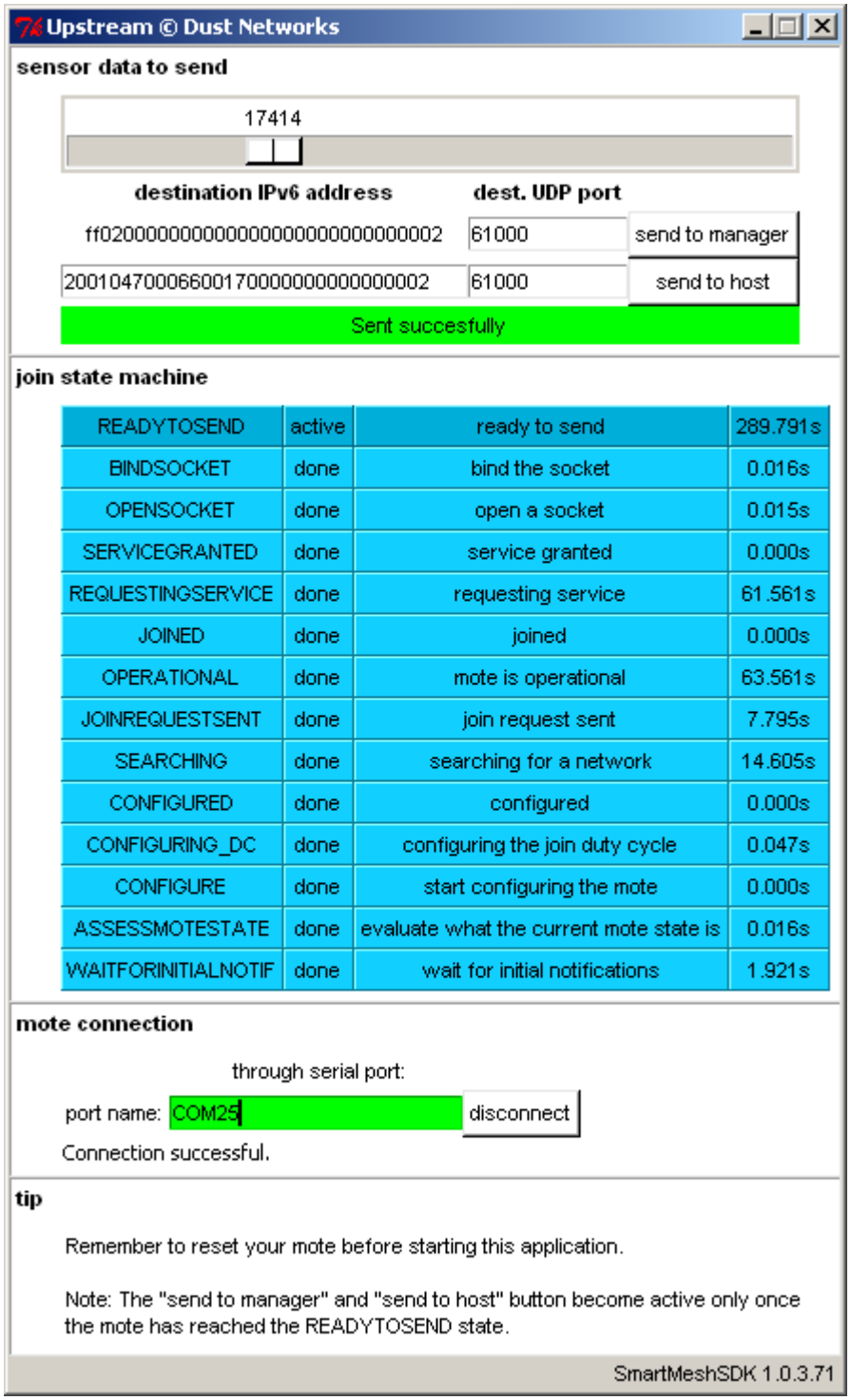
The first steps described below are the same as the steps from the Upstream Communication tutorial.

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



at theSmartMesh IP Mote

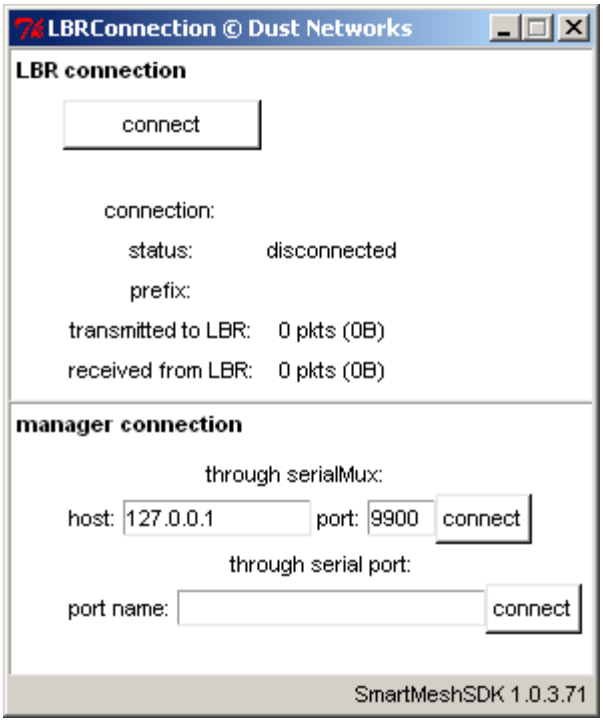


The "send to host" button allows you to send data to any IP address. This functionality requires routing through an LBR and is covered in a separate section.

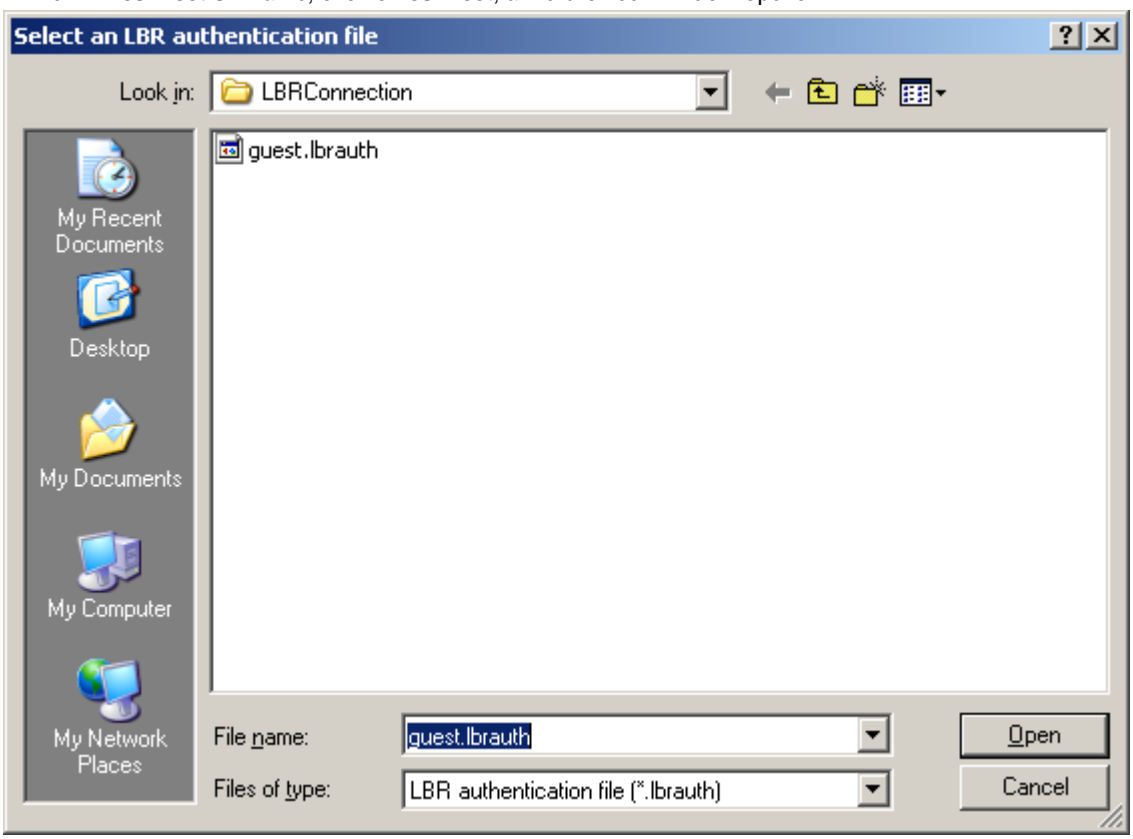
Connect the manager to the LBR

> 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.

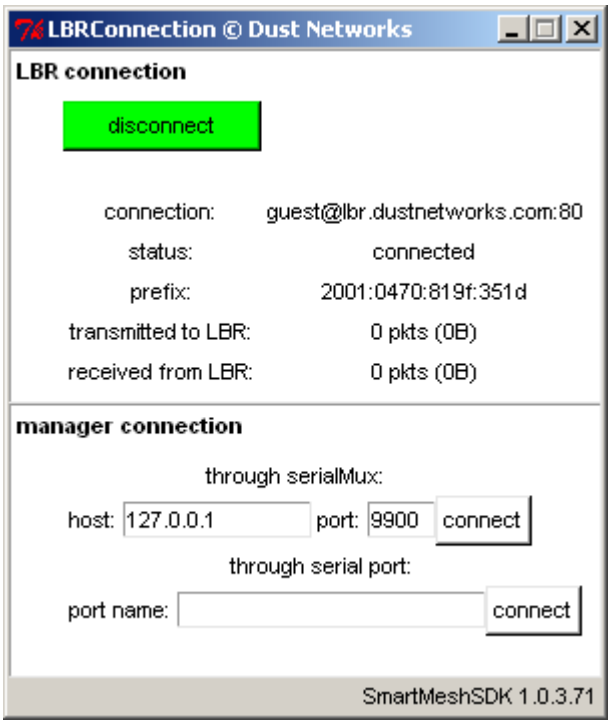


3. In the **LBR connection** frame, click on **connect**, a file browser window opens.

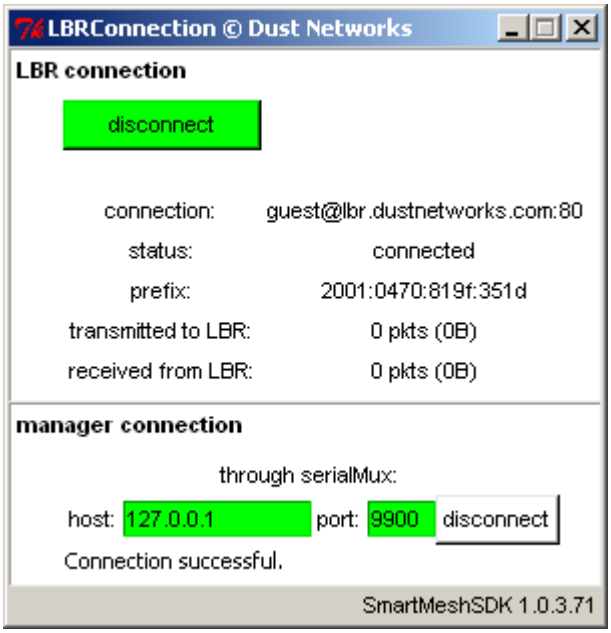


4. The `guest.lbrauth` file is a sample - you will need to replace its contents with the IPv4 address of your LBR instance, and click **open**. This file contains the address of the LBR, and the credentials you need to authenticate to it.

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



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



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.

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

8 Low-power Border Router

8.1 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 IPnetwork 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

8.2 Documentation Organization

- Overview.
- 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.

8.3 Overview

8.3.1 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.

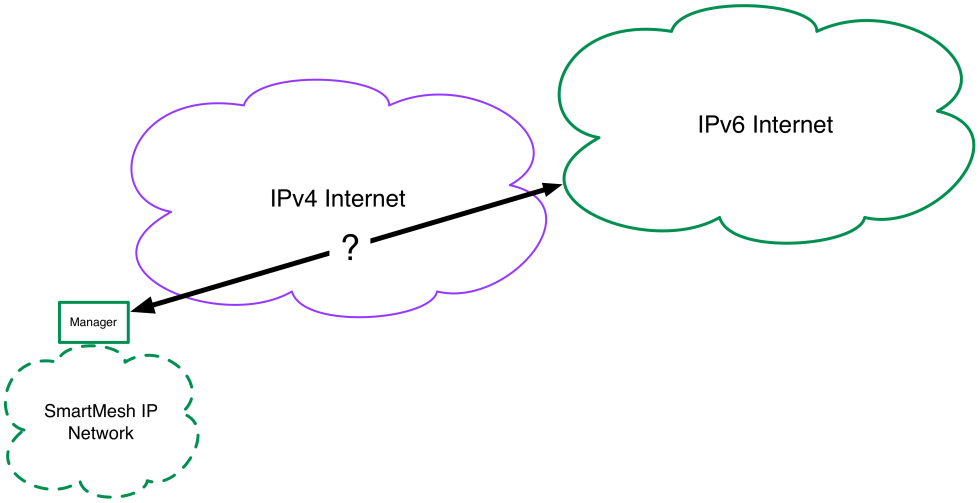
> The 6LoWPAN standard and the concept of the LBR have been developed with the Internet Engineering Task Force, the standardization body behind all Internet-related standards.

8.3.2 Services

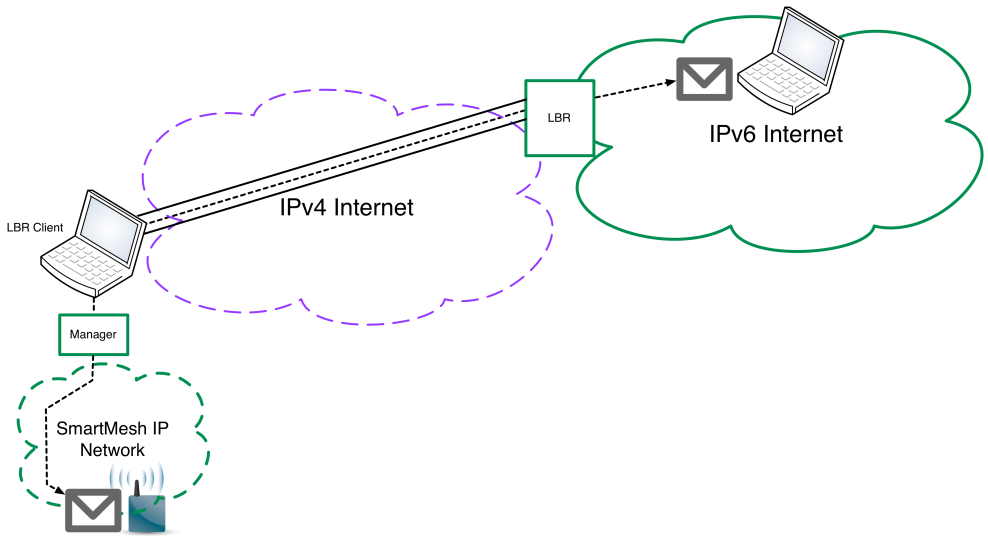
The LBR performs three functions:

- **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.

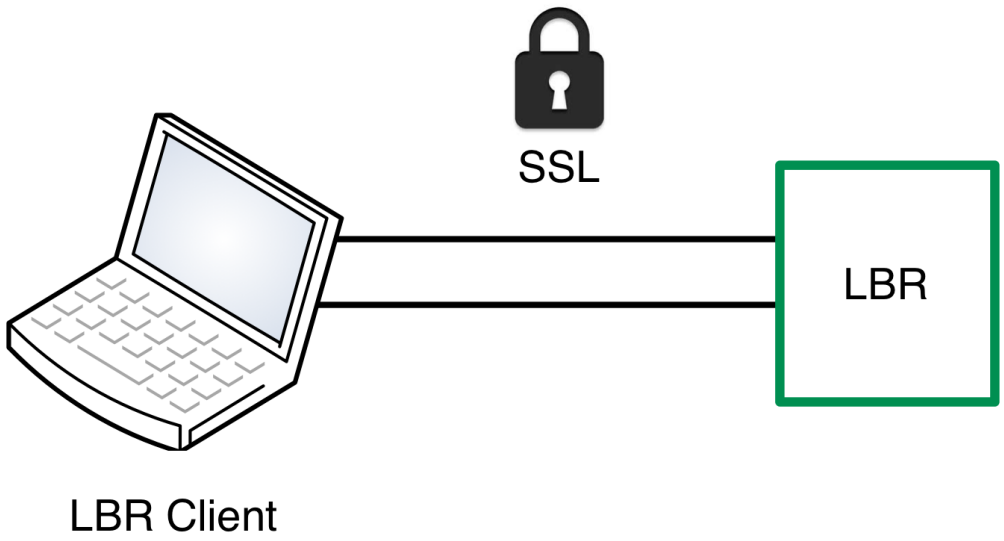
Connectivity



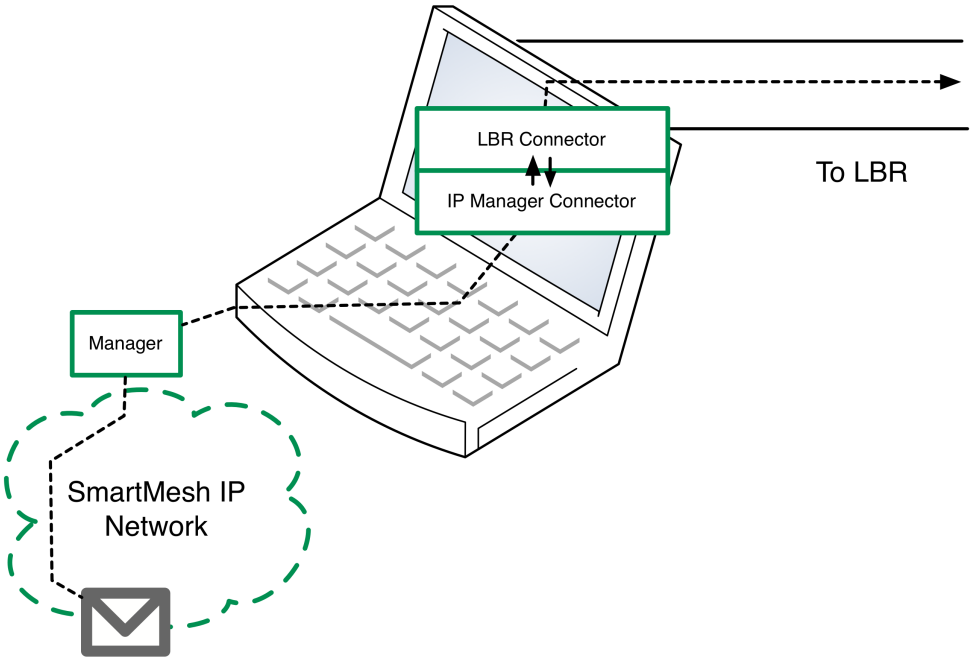
Today, there are two versions of the Internet: the older IPv4 version and the new IPv6 version. While a SmartMesh IP network is IPv6-ready, most of the current Internet still run on the older IPv4 version. The LBR understands both version, allowing for an SmartMesh IP Manager to connect to it even from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

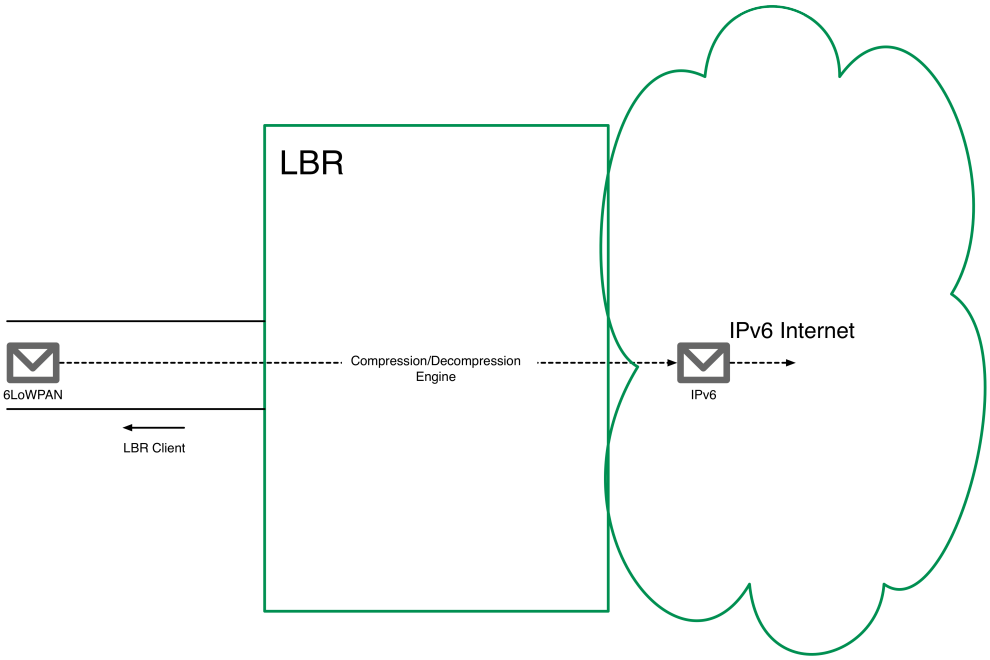


The LBR client is a computer program running on the computer physically connected to the SmartMesh IP Manager. It consists of these components:

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

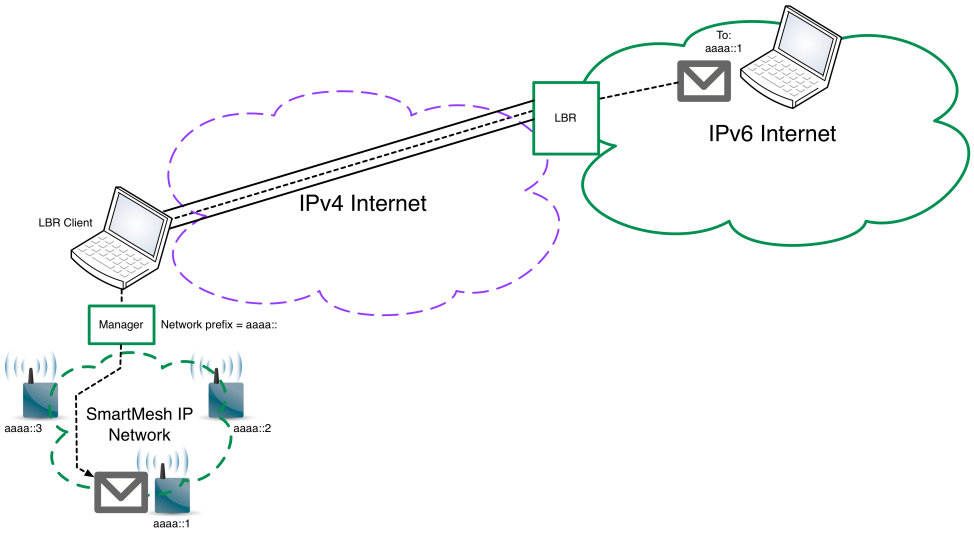
The LBR client program is part of the SmartMesh SDK.

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.

8.4 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.

8.4.1 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
`ping6`	a utility which send/receives ICMPv6 echo requests and responses
`ifconfig`	a utility to configure the IPv6 addresses of the `tun` interfaces the LBR manages
`route`	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

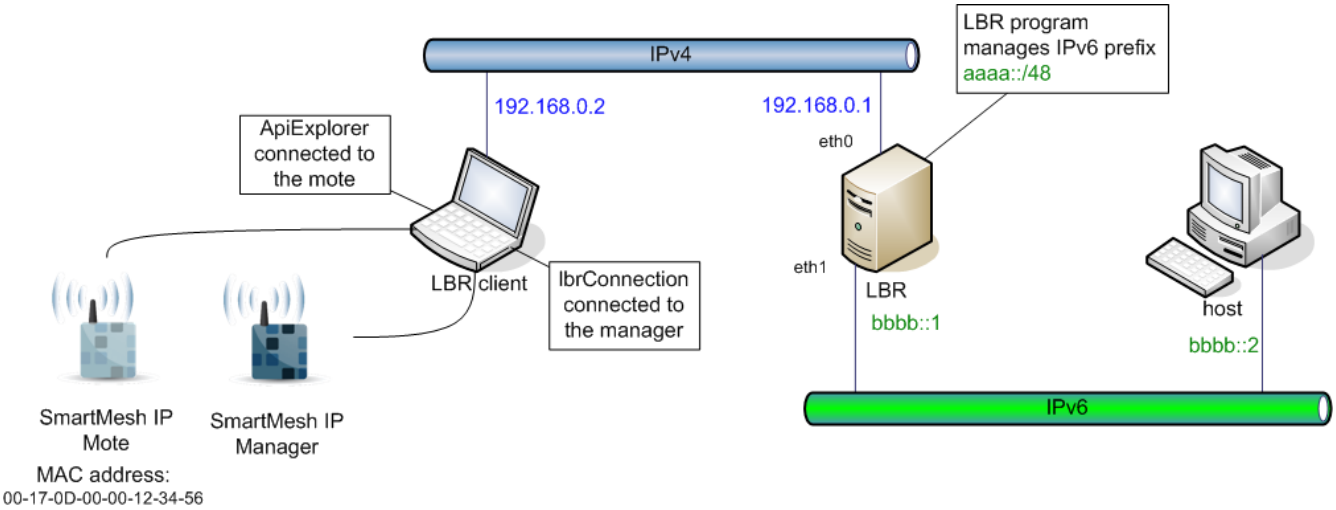
This /48 must be routed to your LBR, i.e. all traffic going to an address pertaining to that prefix should end up at your LBR.

8.4.2 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 Integrationtutorial to send following:
    - destination address (**destIP** field): `bbbb0000000000000000000000000002`
    - 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 IPnetwork'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 theSmartMesh 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`: `bbbb0000000000000000000000000002`, 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.

8.5 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.

8.5.1 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 estalished.
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.

8.5.2 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

8.5.3 Installing the LBR's Keying Material

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

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+4vew7eMA0GCSqGSIb3DQEBBQUAMEUxCzAJBgNV
BAYTAkFVMRMwEQYDVQQIEwpTb21lLVN0YXRlMSEwHwYDVQQKExhJbnRlcm5ldCBX
aWRnaXRzIFB0eSBMdGQwHhcNMTIwNDE2MTg1MDA2WhcNMTMwNDE2MTg1MDA2WjBF
MQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEhMB8GA1UEChMYSW50
ZXJuZXQgV2lkZ2l0cyBQdHkgTHRkMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQDd/jVNuPdfo9WeocR1kEcWXnVux3aZoPtFKfPqnfEBi5502GcPu1rS6ZyT1FU4
oGLrGHE/nZJS5ZvT3I8E+mDmTVQLtouf7I9MAoFZPIwVGAnQ7u+x9q3UOl7f7nQl
dYn+KRtLcZkevOBWBiD1B8ps5Pev36SWsx+FTiJ+GrcpkwIDAQABo4GnMIGkMB0G
A1UdDgQWBBSO2j8DzCjYEy3R3+D8k75PtbzfOjB1BgNVHSMEbjBsgBSO2j8DzCjY
Ey3R3+D8k75PtbzfOqFJpEcwRTELMAkGA1UEBhMCQVUxEzARBgNVBAgTClNvbWUt
U3RhdGUxITAfBgNVBAoTGEludGVybmV0IFdpZGdpdHMgUHR5IEx0ZIIJAPMne+4v
ew7eMAwGA1UdEwQFMAMBAf8wDQYJKoZIhvcNAQEFBQADgYEAvHT0PRtvCbESrLka
9omcDyfNs2su3ciaPgQFZHcp+QAlHjYmsvkcYYJhovZ9kqGMHsXuLFZKgiy+J2w5
Y70vJ2lOBxYM6sTFvdTP9/JkgRYIOFWiTfHXCfTHv54YH3T8R892R2hGBAujl3cx
x9NS9GArXCJJ4Gy4KAca8TfK2Ho=
-----END CERTIFICATE-----
```

```
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQDd/jVNuPdfo9WeocR1kEcWXnVux3aZoPtFKfPqnfEBi5502GcP
u1rS6ZyT1FU4oGLrGHE/nZJS5ZvT3I8E+mDmTVQLtouf7I9MAoFZPIwVGAnQ7u+x
9q3UOl7f7nQldYn+KRtLcZkevOBWBiD1B8ps5Pev36SWsx+FTiJ+GrcpkwIDAQAB
AoGAdD3hkYIyXm0+taMFaX4UCz2JBmoBy25FRLE0HP15LpL6dTq/tLgpVdmn+Kyt
t0ocofgZjPMopKnAkA6lATlONTA9SDsUdwhMSMi5+7xM6SEGPV/OCVzn1eTxwGue
Q41ZD8okADXjqLY/vzEiYeDrmcx9FoULbQxWUPhT9yTO6VECQQDz3e2F5K3Sy/vt
ZIenXTKeWGqHyy30i2e5W0J8oOYJqPv5PjPbN1r7rNrKZBtdvyPcYQQHyv8rhZzp
WWYdZSCbAkEA6QmtiyCo8Y1LnSrPV7lYCxFl4WB4I9+6CxsxYsQeLPrz8dBoYtAX
bNdyIDENaTjHETSeahftAavvnCsCIFn+aQJAB/HH5h/ABej9SQuIW8xudLgeqFPX
KGtOMrylWtgHBnOJ2eHL4K1Z+m70JbnDJneunGRQtExJqcpNhVCTQgvkVwJAA99O
S+6KACGJ7R2l/l4tEVoDqGAS/v2buM2F349EtRiibxU4dtPgX8Wgtto5z9LKtAV8
0GR/YrS5sY2hZmo4aQJBAIIhoPi2zShLLM8N9/px/ljqs5M7ElAMJCGCy8Y5c1nv
SnhSy08IFoXquKJ1BAKtBDv6Li4nX8tCCeaGsRWyRsM=
-----END RSA PRIVATE KEY-----
```


8.5.4 Adding Users

.lbrauth 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
seclevel = 0
```

Regular User with security level 0

This section indicates how to create the following user:

username	subprefix	security level	credentials
`user_seclevel_0`	`ab00`	0 (none)	*none*

Configure the LBR

On the LBR, enter the following commands to add the new user:

```
> add user_seclevel_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 `servercert.pem` 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-----
MIICsDCCAhmgAwIBAgIJAOBABlW4RBllMA0GCSqGSIb3DQEBBQUAMEUxCzAJBgNV
BAYTAkFVMRMwEQYDVQQIEwpTb21lLVN0YXRlMSEwHwYDVQQKExhJbnRlcm5ldCBX
aWRnaXRzIFB0eSBMdGQwHhcNMTIwNDE2MTkzMzAyWhcNMTMwNDE2MTkzMzAyWjBF
MQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEhMB8GA1UEChMYSW50
ZXJuZXQgV2lkZ2l0cyBQdHkgTHRkMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQCtR2w+MFGgBpp5iBwzNxUYhB8uoLsXMfrq9aofSgsYlHER0oP67m/qkD/0DMsO
xL0nrF9Q6m2oaNLMGD6sLsi4LmGWcVHmEw24sz406VgV+Pbp6Xc4xJ8jzLOUMSgc
NUu2kjOj9Vx7TD11XRvgd1OY6FuKZXVxtA0LxOjYF3U89wIDAQABo4GnMIGkMB0G
A1UdDgQWBBQoU/QBawYk8hja8V64ldZtpGmvKjB1BgNVHSMEbjBsgBQoU/QBawYk
8hja8V64ldZtpGmvKqFJpEcwRTELMAkGA1UEBhMCQVUxEzARBgNVBAgTClNvbWUt
U3RhdGUxITAfBgNVBAoTGEludGVybmV0IFdpZGdpdHMgUHR5IEx0ZIIJAOBABlW4
RBllMAwGA1UdEwQFMAMBAf8wDQYJKoZIhvcNAQEFBQADgYEAXjpyhcAC/Z78qyJr
KRfpirn+/376gXi3xpg2KJO2SexcFmq7tnucpmrpbrdGhENy6YmxCcEk4fOFYMLA
udFlrfX1IEXqv1wHggvxd+N5+UNvX1lxCrfi1O8Z7PaZKxb1tpNMTmI3i0NSzl62
l+3tow9UXSKf37j3ldLgXmh7pCk=
-----END CERTIFICATE-----
```



```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQCtR2w+MFGgBpp5iBwzNxUYhB8uoLsXMfrq9aofSgsYlHER0oP6
7m/qkD/0DMsOxL0nrF9Q6m2oaNLMGD6sLsi4LmGWcVHmEw24sz406VgV+Pbp6Xc4
xJ8jzLOUMSgcNUu2kjOj9Vx7TD11XRvgd1OY6FuKZXVxtA0LxOjYF3U89wIDAQAB
AoGAK5QE0vcP8DD49IuYnADEW3AO74kYxFKbii/SYyAZ/kqGzTamXptMpi81BkmZ
X9N2xt2A8zah8XK7YPzP9jml3NIzShiZLNdrsdFCijAueXmWH/fFCzihN1Uwsm8/
8DxCAOP763y/SebuCVWXKOm7JvbGNnh0teexsWN7RNabdYECQQDcA5nPc6vI7LBx
3DILzWG69BfWQuux62C6k4IdQGv3ye3d13lCI5y3IKw+kECQFynigjtVyLarjmUD
7YEJOa4hAkEAyZ7v/ayQk55bZkAg9JTGMkZkY5UsxXhGT4npgj0/Sa4wHg8S3Q22
EkVh5hgWzZtqzJw/lLRN/diQvf1KyYOYFwJAfgrODgvdSSFXwBL+1MYXjAwUr9ns
vyPyaVDiRLHIaM9VJzcvL5XJTRw5sSng4utyQmKLbYysIcJU2pgwyUEzIQJBAMfh
LFTVXeMqq7vbuZAfablvtZhPzDncOgAiXf+Czpox+HvkP7QsIqNMa3ibywd8m01L
XXEDqwoMR7pCQIE0V3MCQQDBrvQ8izzTmWRPXTh0s2iKNUmuuKsg0Gv1QWOjyqzl
hl5s+QQ/tP3VS8ITzWIkkiF8SJDj5KymYUTftM41veh0
-----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:

```
MIICsDCCAhmgAwIBAgIJAOBABlW4RBllMA0GCSqGSIb3DQEBBQUAMEUxCzAJBgNVBAYTAkFVMRMwEQYDVQ... <truncated>
```

```
MIICXQIBAAKBgQCtR2w+MFGgBpp5iBwzNxUYhB8uoLsXMfrq9aofSgsYlHER0oP67m/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 MIICsDCCAhmgAwIBAgIJAOBABlW4RBllMA0GCSqGSIb3DQEBBQU... <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+MFGgBpp5iBwzNxUYhB8uoLsXMfrq9aofSgsYl... <truncated>

# Client's public key
clientpublickey = MIICsDCCAhmgAwIBAgIJAOBABlW4RBllMA0GCSqGSIb3DQEBBQUAMEUxCz... <truncated>
# LBR's public key
lbrpublickey = MIICsDCCAhmgAwIBAgIJAPMne+4vew7eMA0GCSqGSIb3DQEBBQUAMEUxCzAJB... <truncated>
```

8.5.5 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.

8.5.6 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).

8.6 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:

8.6.1 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

8.6.2 backup

Description

Backup the user database to the `userDB.pkl` file.

Syntax

backup

Parameters

Parameter	Description

Example

backup

8.6.3 disconnect

Description

Disconnect a user currently connected.

Syntax

disconnect <username>

Parameters

Parameter	Description
username	The username of the user to disconnect

Example

disconnect myuser

8.6.4 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
seclevel (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
```

8.6.5 loglevel

Description

Change the loglevel for a given user.

Syntax

loglevel <username> <loglevel>

Parameters

Parameter	Description
username	The username of interest, or `all` to apply to all users
loglevel	The loglevel to set; the options are `debug`, `info`, `warning`, `error` or `critical`

Example

8.6.6 passwordremove

Description

Remove the password associated with a user.

Syntax

passwordremove <username>

Parameters

Parameter	Description
username	The username of interest

Example

8.6.7 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

8.6.8 publickeyremove

Description

Remove the public key associated with a user.

Syntax

publickeyremove <username>

Parameters

Parameter	Description
username	The username of interest

Example

8.6.9 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

8.6.10 quit

Description

Exit the LBR application.

Syntax

quit

Parameters

Parameter	Description

Example

8.6.11 remove

Description

Remove a user from the user database.

Syntax

remove <username>

Parameters

Parameter	Description
username	The username of interest

Example

8.6.12 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

8.6.13 status

Description

Print the status of the LBR.

Syntax

status

Parameters

Parameter	Description

Example

8.6.14 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

8.6.15 version

Description

Prints the version of the LBR.

Syntax

version

Parameters

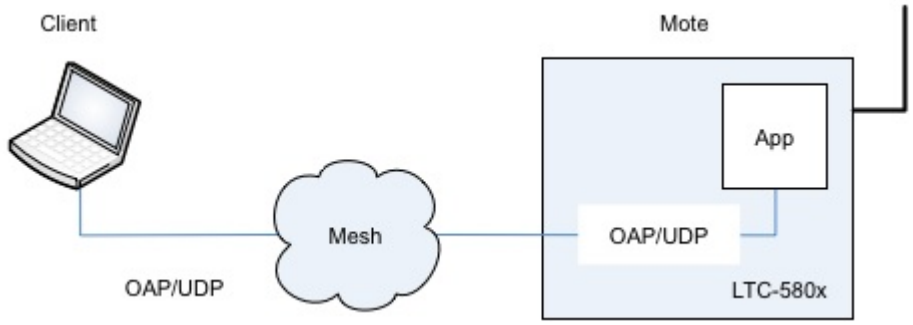
Parameter	Description

Example

9 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 & DC9021 and DC9007).

9.1 Protocol

9.1.1 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

9.1.2 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.

Mote: 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.

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.

9.1.3 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.

9.1.4 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	*Length* 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. INT8U and INT8S values are one byte. INT16U and INT16S are 2 bytes, and INT32U and INT32S 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

9.1.5 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:

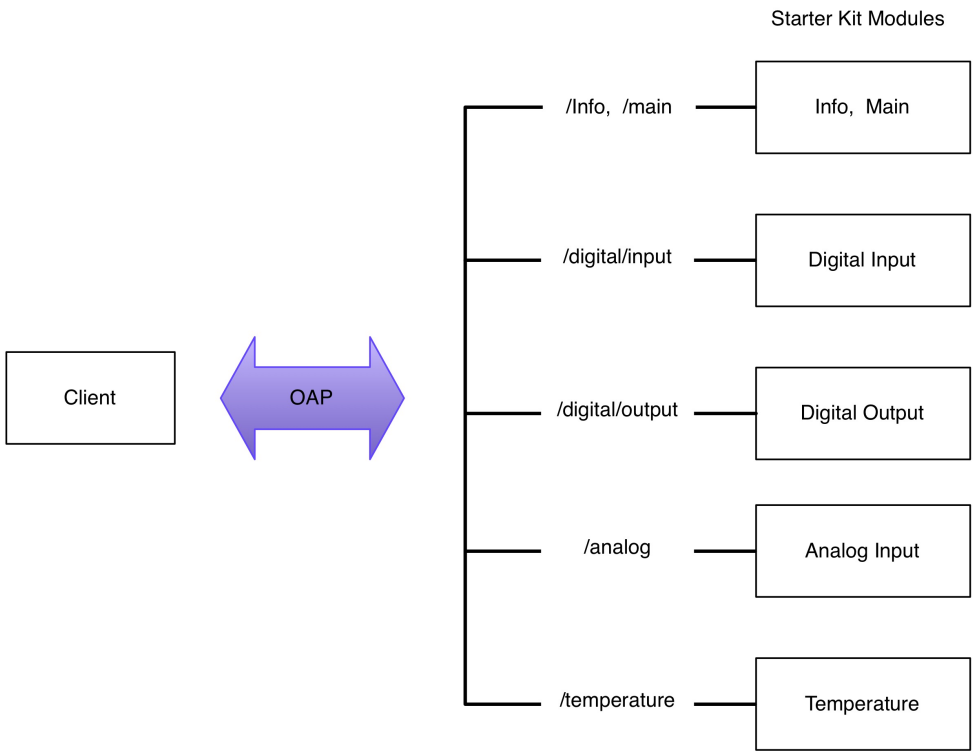
Command	RC	Address	Variable1
02	00	FF 02 00 01	00 01 01

Note that the variable that was set and the value it is set to is echoed in the response.

9.2 OAP in SmartMesh Motes

9.2.1 Introduction

The motes found in the SmartMesh IP Starter Kit (DC9000 & DC9021) 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
appId	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 *sampleCount* 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 *sampleCount* 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

9.2.2 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_TIME_L	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_TIME_L	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…

9.3 OAP Examples

These example encodings apply to the application found in the mote, as described in the modules section.

9.3.1 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)

9.3.2 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)

9.3.3 Digital Input Change Notification

This shows digital pin change notifications coming from mote that is configured to report on change

OAP Header	Command ID	Notification type	Address	Timestamp	Value
00 02	05	02	FF 02 02 00	00 00 00 00 3d 22 70 f8 00 00 c5 ce	01

OAP Header

- Control: 00 (unacknowledged request, normal sync)
- ID: 02 (sequence = 2, session = 0)

OAP Payload (Notification)

- Command: 05 (notification)
- Type: 02 (digital notification)
- Address: FF 02 02 00 (02 00 is /digital_in/D0)
- UTC Timestamp: 00 00 00 00 3d 22 70 f8 00 00 c5 ce
- Value: 01

9.3.4 Get 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, appId, resetCounter and changeCounter values

OAP Header	Address	swRevMajor	swRevMin	swRevPatch	swRevBuild	appId	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

9.3.5 Get the settings of a Digital Input

Sending this packet to the mote will return the configuration of digital input D0 (pin DP2 of the DC9003 board)

OAP Header	Command ID	Address	Payload (Variable TLVs)
05 00	01	FF 02 02 00	(none)

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 2
    - 02 00 is /digital_in/D0

The response from the mote is:

OAP Header	Command ID	RC	Address	Payload (Variable TLVs)
07 10	01	00	FF 02 02 00	00 01 00 01 04 00 00 27 10 02 02 00 01 03 01 00 04 00

OAP Header

- Control: 07 (acknowledged response, resync=1 used to establish a connection)
- ID: 00 (sequence = 0, session = 1)

OAP Payload

- Command: 01 (GET)
- RC: 00 (OK)
- Address
    - Address TLV with length 2
    - 02 00 is /digital_in/D0
    - Field ID 00 (enable/disable) with length 1
    - Value = 00 (disabled)
    - Field ID 01 (rate) with length 4
    - Value = 00002710 (10,000ms)
    - Field ID 02 (sample count) with length 2
    - Value = 0001 (one sample per report)
    - Field ID 03 (data format) with length 1
    - Value = 00 (send all)
    - Field ID 04 (value) with length 0 (because it is disabled)

9.3.6 Enable a Digital Input

Sending this packet to the mote will enable digital input D0 (pin DP2 of the DC9003 board)

OAP Header	Command ID	Address	Payload (Variable TLVs)
05 00	02	FF 02 02 00	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
    - 02 00 is /digital_in/D0
    - Field ID 00 with length 1
    - set to value 1 (enable)

The mote will start sending reports on this digital input as that channel is configured. No polling is required to get the value.

9.3.7 Configure a Digital Input to report on change

Sending this packet to the mote will cause it to send a data notification whenever digital input D0 (pin DP2 of the DC9003 board) changes state (low to high or high to low)

OAP Header	Command ID	Address	Payload (Variable TLVs)
05 00	02	FF 02 02 00	00 01 01 03 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
    - 02 00 is /digital_in/D0
    - Field ID 00 with length 1
    - set to value 1 (enable)
    - Field ID 03 with length 1
    - set to value 1 (send on change)

The mote will send a data report any time the digital input transitions from low to high or from high to low. Note, this is only usable for very low frequency changes. Like a door open/close, or a pushbutton.

10 Logging

10.1 Using the Logging Capabilities

The SmartMesh SDK has advanced logging capabilities. All sample applications log activity in a log file. For example, when launching `APIExplorer.py`, a file `APIExplorer.log` is created in the same directory. This file contains information about the activity of the sample application.

10.2 Logfile format

Each entry in the logfile follows the format:

```
<timestamp> [<component>:<loglevel>] <message>
```

Where:

- `<timestamp>` is the date and time the line was printed to the logfile. For example, "`2014-10-27 13:02:04,431`".
- `<component>` is the Python component which generates this message. The component name is usually the same as the name of the Python source file in which the log statement appears.
- `<loglevel>` is the "seriousness" of the event. Possible values are `DEBUG`, `INFO`, `WARNING`, `ERROR` and `CRITICAL`.
- `<message>` is an arbitraty string, which can span over multiple lines.

10.3 Example logfile

To illustrate this capability, let's use `APIExplorer.py` to connect to a SmartMesh IP Manager, issue the `getNetworkInfo` command, and disconnect. This section shows the contents of the `APIExplorer.log` file generated by that transaction.

10.3.1 Connecting to the manager

First, we connect the `APIExplorer.py` to the SmartMesh IP Manager on serial port `COM33`. As shown in the logs below, this causes the `SerialConnector` module to issue a `hello` command.

The command is serialize by the `ByteArraySerializer` into command ID `1` and bytes `04-00-00`.

```
2014-10-27 13:02:04,431 [SerialConnector:INFO] creating object
2014-10-27 13:02:04,431 [Hdlc:INFO] Creating object
2014-10-27 13:02:04,463 [Hdlc:INFO] opened port COM33
2014-10-27 13:02:04,463 [SerialConnector:INFO] hdlc notification: connection state=True
2014-10-27 13:02:04,463 [Hdlc:INFO] thread started
2014-10-27 13:02:04,463 [ByteArraySerializer:DEBUG] serialize ...
- commandArray:     ['hello']
- fieldsToFill:     {'cliSeqNo': 0, 'version': 4, 'mode': 0}
2014-10-27 13:02:04,463 [ByteArraySerializer:DEBUG] ... serialize into
- cmdId:            1
- byteArray:        04-00-00
```

The `SerialConnector` then receives the command and uses the `Crc` module to calculate the CRC.

```
2014-10-27 13:02:04,463 [SerialConnector:DEBUG] _sendInternal cmdId=1 retry=0 isResponse=False
serializedFields=[4, 0, 0]
2014-10-27 13:02:04,463 [SerialConnector:DEBUG] ---------- pcToMote DATA (1) ---------->
2014-10-27 13:02:04,463 [Crc:DEBUG] calculating for data=00-01-01-03-04-00-00
2014-10-27 13:02:04,463 [Crc:DEBUG] fcs=0xb3c5
```

The `Hldc` module sends the bytes over the serial port to the SmartMesh IP Manager.

```
2014-10-27 13:02:04,463 [Hdlc:DEBUG]
packetToSend:
 - payload: 00 01 01 03 04 00 00
 - fcs:     b3 c5
 - valid:   True
```

When receiving the `hello` command, the SmartMesh IP Manager replies with a `hello_response`. First, the bytes are received by the `Crc` and `Hdlc` modules which verifies the bytes form a correctly formatted HDLC frame:

```
2014-10-27 13:02:04,463 [Crc:DEBUG] calculating for data=00-02-00-05-00-04-00-00-00
2014-10-27 13:02:04,463 [Crc:DEBUG] fcs=0xe3dc
2014-10-27 13:02:04,463 [Hdlc:DEBUG]
receivedFrame:
 - payload: 00 02 00 05 00 04 00 00 00
 - fcs:     e3 dc
 - valid:   True
```

The `SerialConnector` module the parses the header of the serial packet to determine the command ID, the length, whether it is a response, the packet ID, and the payload bytes:

```
2014-10-27 13:02:04,463 [SerialConnector:DEBUG] cmdId=2 length=5 isResponse=False packetId=0
payload=[0, 4, 0, 0, 0]
2014-10-27 13:02:04,463 [SerialConnector:DEBUG] <--------- moteToPc DATA (0) ----------
2014-10-27 13:02:04,463 [SerialConnector:DEBUG] no ack needed
```

The `ByteArraySerialzer` parses the payload bytes into the command `hello_response`:

```
2014-10-27 13:02:04,463 [ByteArraySerializer:DEBUG] deserialize ...
- type:             command
- id:               2
- byteArray:        00-04-00-00-00
2014-10-27 13:02:04,479 [ByteArraySerializer:DEBUG] ... deserialized into
- nameArray:        ['hello_response']
- returnFields:     {'cliSeqNo': 0, 'version': 4, 'successCode': 0, 'mgrSeqNo': 0, 'mode': 0}
```

10.3.2 Issues the getNetworkInfo command

Then, we issue a `getNetworkInfo` command.

The same interaction between the `ByteArraySerializer`, `Hdlc` and `Crc` module happens as with the `hello` packet above to send the correct bytes over the serial port to the SmartMesh IP Manager:

```
2014-10-27 13:02:09,526 [ByteArraySerializer:DEBUG] serialize ...
- commandArray:     ['getNetworkInfo']
- fieldsToFill:     {}
2014-10-27 13:02:09,526 [ByteArraySerializer:DEBUG] ... serialize into
- cmdId:            64
- byteArray:
2014-10-27 13:02:09,526 [SerialConnector:DEBUG] _sendInternal cmdId=64 retry=0 isResponse=False
serializedFields=[]
2014-10-27 13:02:09,526 [SerialConnector:DEBUG] ---------- pcToMote DATA (1) ---------->
2014-10-27 13:02:09,526 [Crc:DEBUG] calculating for data=02-40-01-00
2014-10-27 13:02:09,526 [Crc:DEBUG] fcs=0x 6da
2014-10-27 13:02:09,526 [Hdlc:DEBUG]
packetToSend:
 - payload: 02 40 01 00
 - fcs:     06 da
 - valid:   True
2014-10-27 13:02:09,542 [Crc:DEBUG] calculating for
data=03-40-01-1e-00-00-09-1c-52-00-01-64-00-00-00-15-18-00-fe-80-00-00-00-00-00-00-00-17-0d-00-00-3f-f
13:02:09,542 [Crc:DEBUG] fcs=0x943f

2014-10-27 13:02:10,917 [Hdlc:INFO] disconnect
2014-10-27 13:02:10,917 [SerialConnector:INFO] hdlc notification: connection state=False
2014-10-27 13:02:10,917 [Hdlc:INFO] thread ended
```

The SmartMesh IP Manager answers with the response to the `getNetworkInfo` command. This response is parsed by the `Hdlc`, `SerialConnector` and `ByteArraySerializer` modules, as the `hello_response` packets above.

```
2014-10-27 13:02:09,542 [Hdlc:DEBUG]
receivedFrame:
 - payload: 03 40 01 1e 00 00 09 1c 52 00 01 64 00 00 00 15 18 00 fe 80 00 00 00 00 00 00 00 17 0d
00 00 3f f8 1e
 - fcs:     94 3f
 - valid:   True
2014-10-27 13:02:09,542 [SerialConnector:DEBUG] cmdId=64 length=30 isResponse=True packetId=1
payload=[0, 0, 9, 28, 82, 0, 1, 100, 0, 0, 0, 21, 24, 0, 254, 128, 0, 0, 0, 0, 0, 0, 0, 23, 13, 0,
0, 63, 248, 30]
2014-10-27 13:02:09,542 [SerialConnector:DEBUG] <--------- pcToMote ACK (1) after 0.016 --------
2014-10-27 13:02:09,542 [SerialConnector:DEBUG] no ack needed
2014-10-27 13:02:09,542 [ByteArraySerializer:DEBUG] deserialize ...
- type:             command
- id:               64
- byteArray:
00-00-09-1c-52-00-01-64-00-00-00-15-18-00-fe-80-00-00-00-00-00-00-00-17-0d-00-00-3f-f8-1e
2014-10-27 13:02:09,542 [ByteArraySerializer:DEBUG] ... deserialized into
- nameArray:        ['getNetworkInfo']
- returnFields:     {'numLostPackets': None, 'advertisementState': 0, 'ipv6Address': [254, 128, 0,
0, 0, 0, 0, 0, 0, 23, 13, 0, 0, 63, 248, 30], 'asnSize': 7250, 'numMotes': 9, 'numArrivedPackets':
None, 'netLatency': 5400, 'netState': 0, 'netPathStability': 0, 'downFrameState': 1, 'maxNumbHops':
None, 'RC': 0, 'netReliability': 100}
```

10.3.3 Disconnect

```
2014-10-27 13:02:10,917 [Hdlc:INFO] disconnect
2014-10-27 13:02:10,917 [SerialConnector:INFO] hdlc notification: connection state=False
2014-10-27 13:02:10,917 [Hdlc:INFO] thread ended
```

Disconnecting just consist in deleting the Python object. Deleting the `Hdlc` instance will cause it to close the serial port.

10.4 Implementation details

The SmartMesh SDK's logging capabilities is built around Python's built-in `logging` module.

> For details about Python's `logging` module, see https://docs.python.org/2/library/logging.html.

Logging consists in two steps:

- sprinkle the source code with log statements which generate log event by different logging modules and loglevels.

- configure which combinations of logging modules and loglevels should be printed into the logfile.

These steps are detailed in the following subsections.

10.4.1 log statement in source code

Most files in the source code contain the following header:

```
import logging
class NullHandler(logging.Handler):
    def emit(self, record):
        pass
log = logging.getLogger('SerialConnector')
log.setLevel(logging.ERROR)
log.addHandler(NullHandler())
```

This creates a object `log`, local to that file. Once this object is called, the following functions can be used:

`log.debug()`	creates an entry of loglevel `DEBUG`
`log.info()`	creates an entry of loglevel `INFO`
`log.warning()`	creates an entry of loglevel `WARNING`
`log.error()`	creates an entry of loglevel `ERROR`
`log.critical()`	creates an entry of loglevel `CRITICAL`

When creating the `log` object in the header, it is associated the name of the log module, in this case "`SerialConnector`".

10.4.2 log configuration file

The `src/SmartMeshSDK-x.x.x.x/bin/logging.conf` file indicates what combinations of logging modules and loglevels should be printed into the logfile.

It contains for example:

```
[logger_SerialConnector]
level         = DEBUG
handlers      = std
propagate     = 0
qualname      = SerialConnector
```

Which tells the `logging` to print all debug statements issued by the `SerialConnector` module at loglevel `DEBUG` and above to the file.

> The same `logging.conf` file is used by all sample applications in the SmartMesh SDK.

10.5 Modifying logging in you application

The logging framework present in the SmartMesh SDK sample applications is designed to be flexible. To configure it, you only need to edit the `logging.conf` file. This allows you to:

- remove logging entirely from certain modules
- change the loglevel of certain modules

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. 2012-2016 All Rights Reserved.