SmartMesh IP ツール・ガイド

目次

1 本書について

1.1 関連資料

SmartMesh IP ネットワーク向けに以下の資料が提供されています。

スタータ・キットのクイック・ガイド

- SmartMesh IP Easy Start Guide -基本的なインストールとネットワークの動作確認テストについて説明しています。
- SmartMesh IP Tools Guide - 「インストール」セクションではシリアル・ドライバのインストール手順について説明しており、Easy Start Guide やその他のチュートリアルで使用されるサンプル・プログラムも含まれています。

ユーザー・ガイド

- SmartMesh IP ユーザー・ガイド - ネットワーク概念についての説明と、モートおよびマネージャの API を使用して特定のタスク(データ送信や統計情報の収集など)を実行する方法について説明します。この資料は、API ガイドを使用するための予備知識を提供します。

デバイスの対話操作用インタフェース

- SmartMesh IP Manager CLI Guide - CLI は、クライアントの開発中やトラブルシューティングのためにユーザーがマネージャとやり取りするために使用します。このドキュメントは、CLI の接続とそのコマンド・セットについて説明しています。
- SmartMesh IP Manager API Guide - API は、プログラムを使用してマネージャとやり取りするために使用します。このドキュメントは、API の接続とそのコマンド・セットについて説明しています。
- SmartMesh IP Mote CLI Guide - CLI は、センサ・アプリケーションの開発中やトラブルシューティングのためにユーザーがモートとやり取りするために使用します。このドキュメントは、CLI の接続とそのコマンド・セットについて説明しています。
- SmartMesh IP Mote API Guide - API は、プログラムを使用してモートとやり取りするために使用します。このドキュメントは、API の接続とそのコマンド・セットについて説明しています。

ソフトウェア開発ツール

- SmartMesh IP ツール・ガイド - SmartMesh SDK に含まれる各種の評価および開発サポート・ツールについて説明しています。モートおよびマネージャ API の実行やネットワークの視覚化を行うツールが含まれます。

アプリケーション・ノート

- SmartMesh IP Application Notes - SmartMesh IP ネットワーク固有の各種トピックと、SmartMesh ネットワーク全般に当てはまるトピックを含みます。

新規設計の開始時に役立つ資料

- LTC5800-IPM SoC またはそのモジュールのデータシート。
- LTC5800-IPR SoC またはその組み込みマネージャのデータシート。
- モート/マネージャ SoC 用またはモジュール用のハードウェア統合ガイド - 設計への SoC またはモジュールの統合に関するベスト・プラクティスを提供しています。
- 組み込みマネージャ用のハードウェア統合ガイド - 設計への組み込みマネージャの統合に関するベスト・プラクティスを提供しています。
- Board Specific Integration Guide - SoC モートおよびマネージャに対するデフォルトの I/O 設定方法と、「ヒューズ表」を使用した水晶振動子の補正情報について説明しています。

- Hardware Integration Application Notes - SoC 設計チェックリスト、アンテナ選定ガイドなどを含みます。
- ESP Programmer Guide - DC9010 Programmer Board と、デバイスへのファームウェアのロードに使用する ESP ソフトウェアのガイドです。
- ESP ソフトウェア - モートまたはモジュールにファームウェア・イメージをプログラミングするために使用します。
- Fuse Table ソフトウェア - Board Specific Configuration Guide で説明されているヒューズ表を作成するために使用します。

その他の役立つ資料

- SmartMesh IP ユーザー・ガイド http://www.linear-tech.co.jp/docs/41880 には、SmartMesh 関連の資料で使用されているワイヤレス・ネットワーク用語の解説が含まれています。
- よくある質問の一覧。

1.2 表記規則

本書では、以下の表記規則を使用します。

*コンピュータ・タイプ*は、URL の指定など、ユーザーが入力する情報を示します。

太字は、ボタン、フィールド、メニュー・コマンド、デバイス・ステート、モードを示します。

*斜体*は、新しい用語や API とそのパラメータを示します。

ヒントは、製品に関して役立つ情報を提供します。

情報テキストは、背景や前後関係の理解に役立つ追加情報を提供します。

注記は、概念についてより詳しい説明を提供します。

警告! 警告は、データ損失やハードウェアまたはユーザーへの物理的な損害を引き起こす可能性のある動作をユーザーに知らせます。

```
コード・ブロックは、コード例を示します。
```

1.3 改訂履歴

改訂番号	日付	説明
1	2013/03/18	初期リリース
2	2013/09/18	OAP ピンへの LTP5901/2 の追加
3	2013/10/22	新規 SMSDK アプリケーションの追加、その他の軽微な修正
4	2014/04/04	HRListener サンプル・アプリケーションの文書化
5	2014/10/28	OAP でのアナログ入力ユニットの明確化、その他の軽微な修正

2 はじめに

本書では、SmartMesh IP ネットワークとのやり取りに使用できる各種ツールのインストールと使用法について説明します。各種ソフトウェア・ツールの関係を図 1 に示します。最も低いレベルにある FTDI ドライバを通じて、コンピュータは、USB-シリアル・リンク経由でモートおよびマネージャの API や CLI と相互作用することができます。ユーザーはシリアル・ターミナル・クライアントを介して、モートおよびマネージャの CLI とやり取りできます。シリアル マルチプレクサ（Mux）は、マネージャ API に対して複数の同時接続を可能にします。この API は、いくつかのツールで使用されています。

- SmartMesh SDK は Python ベースのツール・セットであり、モートおよびマネージャ API をさまざまな側面からデモンストレーションするために使用されます。この SDK はモートの API に直接接続でき、マネージャ API には、直接または SerialMux 経由で接続することができます。
- Stargazer GUI は、ネットワークの視覚化と相互作用を可能にします。
- Low-power Border Router（LBR）を使用すると、LAN またはインターネット上の IPv6 ホストとデータを送受信できます。

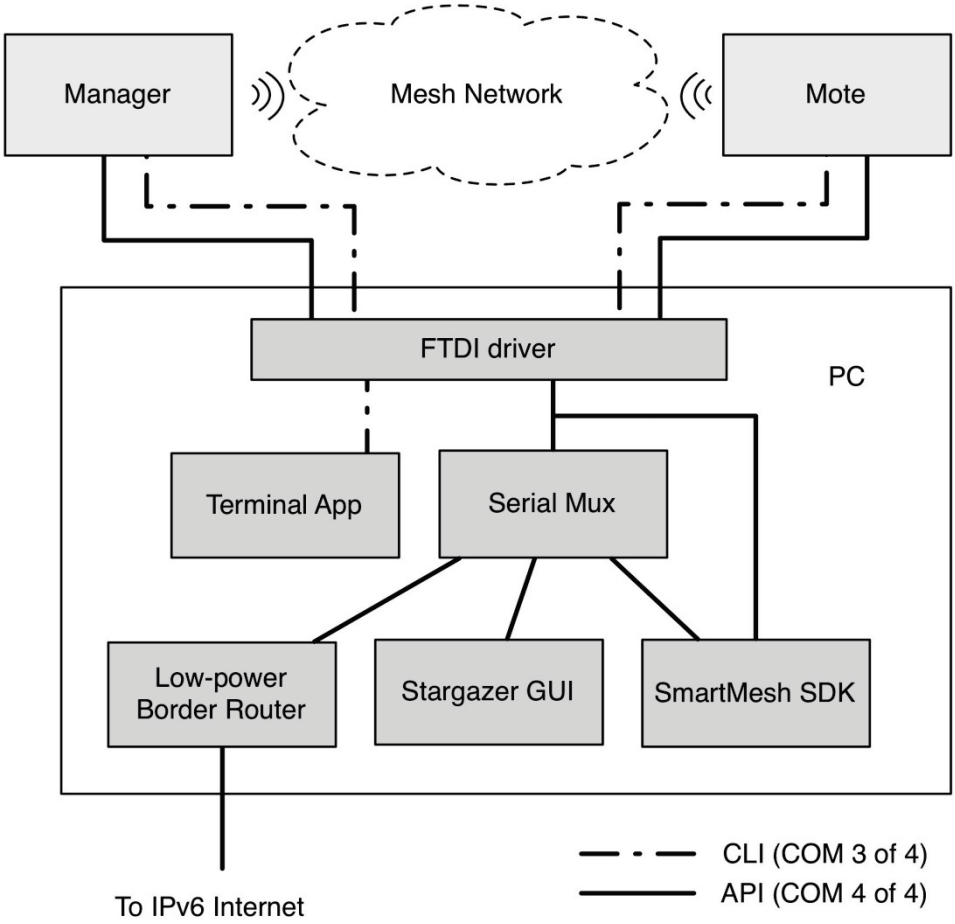


図 1 - SmartMesh IP ネットワークとやり取りするためのソフトウェア・ツール

3 インストール

3.1 作業を始める前に

3.1.1 前提条件

「マネージャの対話操作」および「モートの対話操作」チュートリアルで、マネージャおよびモートとやり取りするために必要なハードウェアは以下のとおりです。

数量	概要
1	Microsoft Windows（XP または 7）を実行しているコンピュータ、空き USB ポート 2 個
2	DC9006 Eterna インタフェース・カード
2	DC9018 Eterna 評価/開発ボード ● 黄色のラベルが付いたボードは、SmartMesh IP マネージャ（DC9018A-A）としてプログラミングされています。 ● 白のラベルが付いたボードは、SmartMesh IP モート（DC9018A-B）としてプログラミングされています。 ⚠ DC9006 Eterna インタフェース・ボードと DC9018A-A マネージャを組み合わせたものが、DC9001 です。
2	コンピュータを DC9006 に接続するためのマイクロ USB ケーブル

「マネージャの対話操作」および「モートの対話操作」チュートリアルで、マネージャおよびモートとやり取りするために必要なソフトウェアは以下のとおりです。

名前	概要
FTDI ドライバ	DC9006 ボードとの通信用 USB-シリアル・ドライバ
シリアル・ターミナル・クライアント	デバイスのコマンド・ライン・インタフェース（CLI）とやり取りするためのソフトウェア
SerialMux	複 APIExplorer や Stargazer など数のクライアントをマネージャに接続するために使用するソフトウェア
SmartMesh SDK	デバイスのアプリケーション・プログラミング・インタフェース（API）とやり取りするためのソフトウェア開発キット

「初期のネットワーク」チュートリアルを実行するために必要なハードウェアは以下のとおりです。

数量	概要
1	SmartMesh IP スタータ・キット（DC9021A） DC9018A-B（SmartMesh IP モート）が 5 つ含まれます。 マネージャに外付け RAM が装着されているかどうかによって、最大で 32 または 100 のモートを使用できます。

「初期のネットワーク」チュートリアルを実行するために必要なソフトウェアは以下のとおりです。

名前	概要
Stargazer	ネットワークの視覚化とモートの対話操作に使用される GUI アプリケーション

3.2 セットアップ

3.2.1 システム概要

図 1 に、評価/開発キットのコンポーネントを示します。このキットにはマネージャと 5 つのモートが含まれています。「ネットワークの対話操作」_bookmark84 セクションでは、マネージャにモートを接続して、メッセージをやり取りします。

> ⚠ 注：SmartMesh IP マネージャの外観は、SmartMesh IP モートとまったく同じです。黄色のラベル（DC9018A-A）が付いているものがマネージャです。インタフェース・ボードは、マネージャ用もモート用も同一（DC9006）です。DC9001 は、マネージャとインタフェース・ボード（DC9018A-A と DC9001）を組み合わせたものです。

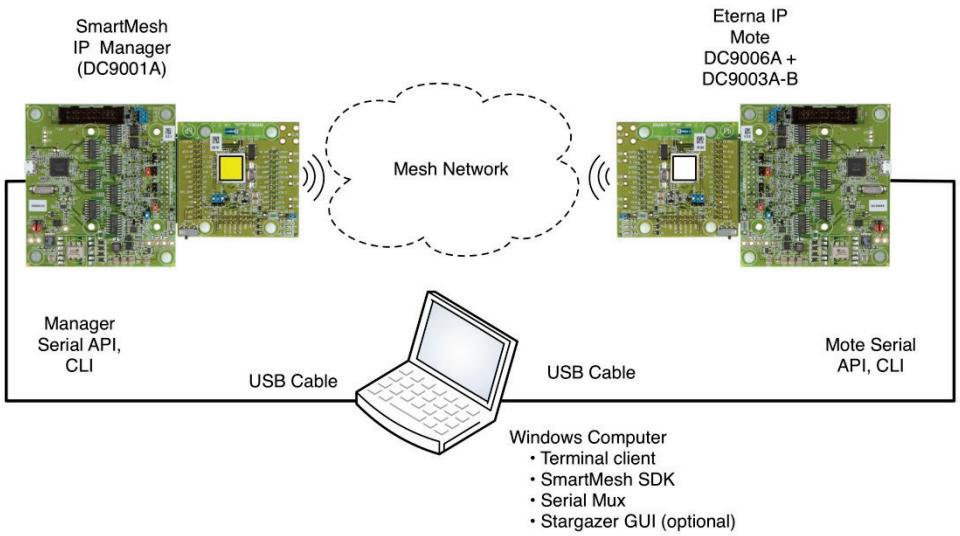


図 1 - 評価キットのコンポーネント

データを送信する前に、いくつかのソフトウェアをインストールし、モートとマネージャに接続できることを確認する必要があります。以下の手順に従います。

- ステップ 1：ハードウェアの準備
- ステップ 2：PC へのソフトウェアのインストール
    - a：FTDI シリアル・ドライバ（および、必要な場合はカスタムのターミナル・クライアント）
    - b：SerialMux
    - c：Python（まだインストールされていない場合のみ）および SmartMesh SDK

続いて、「基本的な通信」セクションに進みます。

1. APIExplorer を使用して、PC からマネージャ（DC9001）への接続を確立
2. APIExplorer を使用して、PC からモート（DC9018A-B）への接続を確立
3. モートを無線でマネージャに接続
4. マネージャとモート間の双方向でメッセージを送信

3.2.2 ステップ 1 - ハードウェアの準備

DC9018A-A マネージャおよび DC9018A-B モートと通信するためには、下図のようにボード間コネクタを使用して、DC9006 インタフェース・ボードをそれぞれに接続する必要があります。

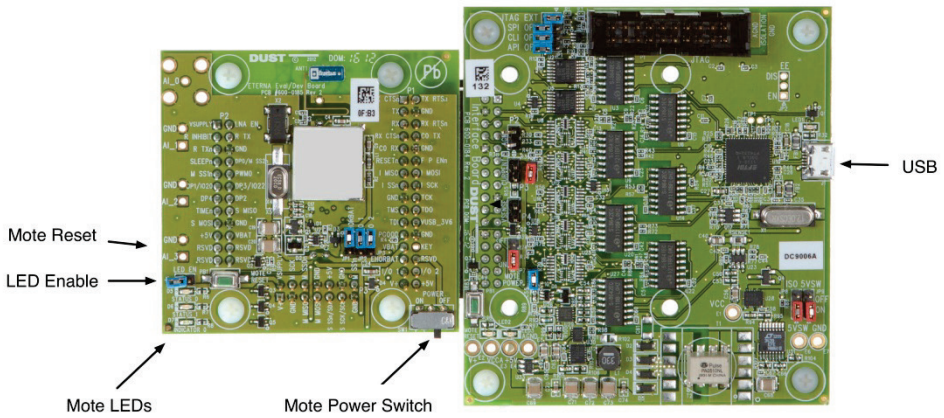


図 1 - DC9018 ボード（左）と DC9006 ボード（右）の接続

- 「Power」と書かれた DC9018 ボードのスライド・スイッチをオンにします。
    - マネージャ（黄色のラベル）で、青色 LED が点灯します。点灯しない場合は、LED を有効にするジャンパ接続（LED_EN）をチェックします。
    - モート（白のラベル）で、いずれかの黄色 LED（STATUS_0）が点滅を始め、モートがネットワークを検索していることが示されます。点滅しない場合は、LED を有効にするジャンパ接続（LED_EN）をチェックします。
- キットに付属しているマイクロ USB ケーブルを、それぞれの DC9006 に接続します。シリアル・ドライバをインストールするまでは、マイクロ USB ケーブルをコンピュータに接続しないでください。

⚠ DC9006 ボードとコンピュータに接続すると、電源スイッチがオフになっているにもかかわらず、(LED によって)モートおよびマネージャが動作しているように見える場合があります。各デバイスに 4 つの COM ポートが確認できますが、信頼性のある通信は実行できません。正しく動作させるためには、全てのボードで電源スイッチがオンになっていることを確認します。

⚠ DC9018 ボードは、LED_EN ジャンパが短絡した状態で出荷されます。これは、接続したときにステータス LED を確認できるようにするためです。これらの LED がネットワーク内にあるとき、モートの 100 倍以上の電力を消費するため、長期的な評価や電力計測では LED_EN ジャンパを取り外すことをお勧めします。

3.2.3 ステップ 2a - FTDI シリアル・ドライバのインストール

シリアル・ドライバのインストール

ドライバのインストールには、以下の 3 つのステップがあります。

- FTDI ドライバ・ソフトウェアのダウンロード
- マネージャ(DC9001)の接続とドライバのセットアップ
- モート(DC9006 + DC9018A-B)の接続とドライバのセットアップ

各デバイスは、シリアル接続を使用して USB 経由でコンピュータと通信します。デバイスをコンピュータに接続すると、ドライバをインストールするかどうかを確認するメッセージが表示されます。このデバイスが使用するシリアル・チップセットは Future Technology Devices International(FTDI)社製であり、さまざまなデバイスに搭載されているため、いずれかのバージョンの FTDI ドライバが、コンピュータ上にすでにインストールされている可能性があります。

ドライバがインストールされていない場合、オペレーティング・システムに適したバージョンを、http://www.ftdichip.com/Drivers/VCP.htm からダウンロードします。ダウンロードしたドライバ・ファイルは、コンピュータのデスクトップに保存することをお勧めします。

✓ ドライバをインストールしたら、コンピュータにデバイスを再接続するときは毎回同じ USB ポートを使用します。別の USB ポートに接続すると、そのたびに以下の手順を繰り返す必要があります。

Windows ドライバのインストール

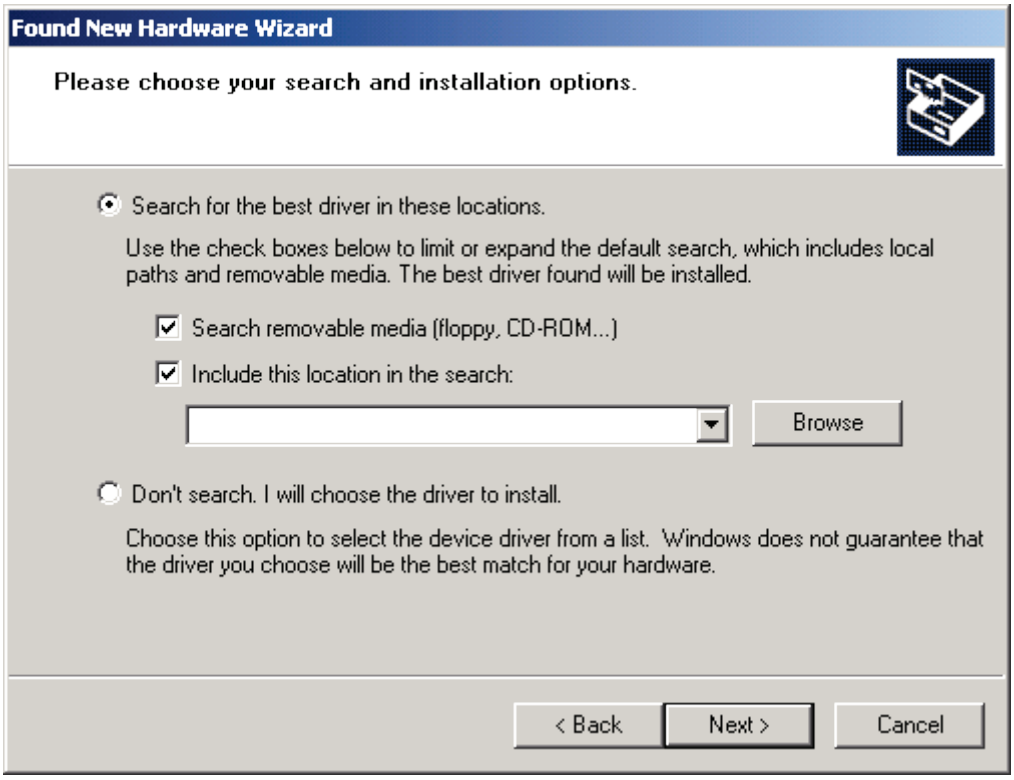
以下の手順に従って、Windows へのインストールを完了します。

1. デバイス(マネージャまたはモート)とコンピュータ間に USB ケーブルを接続します。Found New Hardware Wizard が表示されたら、ステップ 2 に進みます。
   Found New Hardware Wizard が表示されない場合は、以下を実行します。

1. ポートが機能しており、デバイスが正しく接続されていることを確認します。それでもウィザードが表示されない場合は、Windows Device Manager を開いて、デバイスが Windows に認識されているかどうかを確認します。
2. デバイス「Dust Interface Board」が不明なデバイス（黄色アイコン）として表示されている場合、デバイスを右クリックして **Update Driver** を選択します。Found New Hardware Wizard が表示されます。
3. ステップ 2 に進みます。

2. ウィザードで、**Install from a list or specific location** オプションを選択し、**Next** をクリックします。

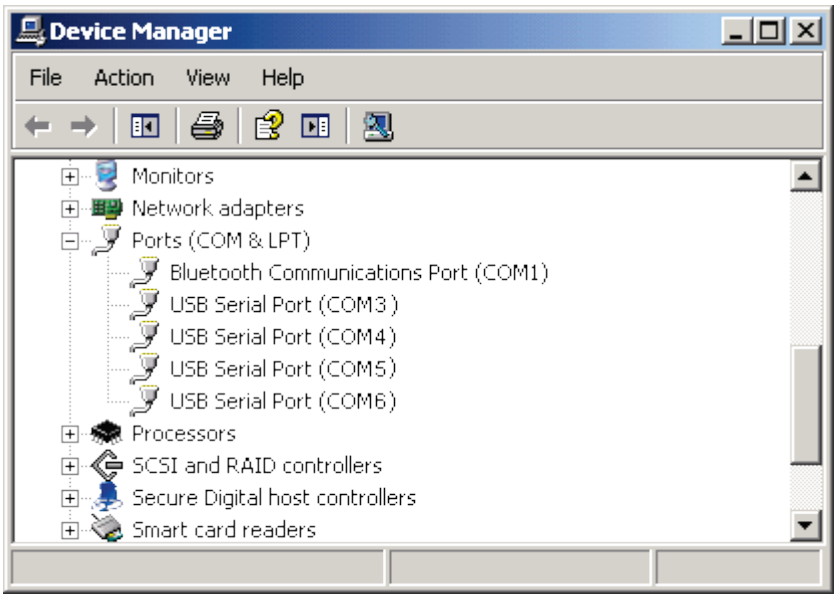
3. **Include this location in the search** チェック・ボックスを選択します。**Browse** ボタンを使用してデスクトップに移動し、**Next** をクリックします。



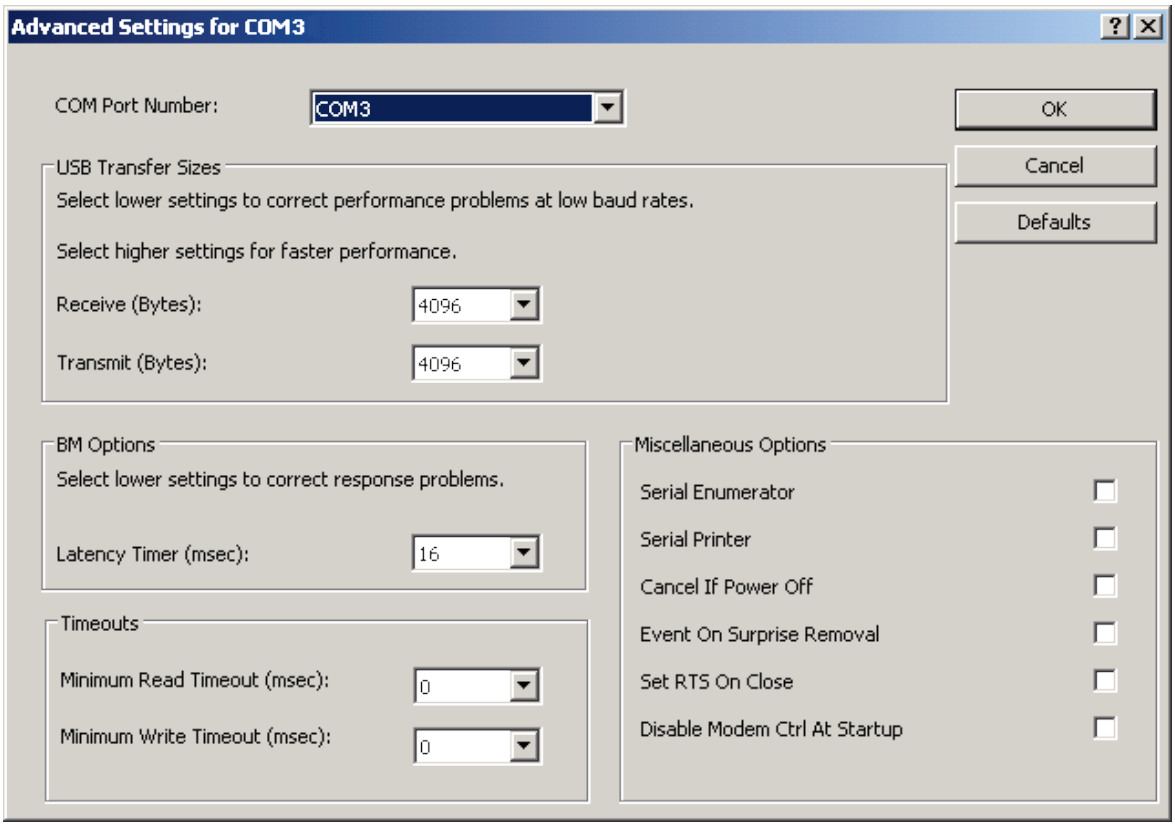
4. ウィザードによってソフトウェアがインストールされたら、**Finish** をクリックします。
5. もう一度 Found New Hardware Wizard が表示されたら、ステップ 2 から 4 を繰り返して、インストールを続行します。ウィザードが表示されるたびに、ステップ 2 から 4 を繰り返します。

⚠ Windows では、USB ポートのインストールとマッピングが完了するまでに、ウィザードの処理を最大で 8 回繰り返すように指示される場合があります。これは、マネージャが合計 4 つの仮想シリアル・ポートを制御用の USB デバイスとともにインストールするためです。

6. USB ポートのインストールとマッピングが完了したら、Device Manager を開いて、仮想シリアル・ポートに割り当てられた COM ポートの番号を確認します。
   1. Start メニューからを **Control Panel** を選択します。
   2. *System* フォルダを開きます
   3. **Hardware** タブをクリックし、Device Manager をクリックします。
   4. **Ports** を開き、COM ポートを表示します。
      Device Manager に、新しい COM ポートが 4 つ表示されます。
   5. 4 つの COM ポート番号を書き留めます。



7. Advanced Settings で、それぞれの COM ポートに対して以下を設定します。
    1. COM ポートを右クリックし、**Properties** をクリックします。
    2. **Port Settings** タブをクリックし、**Advanced** をクリックします。
    3. **Serial Enumerator** オプションの選択を解除して、**OK** をクリックします。
    4. **OK** をクリックして、Device Manager に戻ります。
    5. 4 つの COM ポートに対して上記ステップを繰り返します。全て終了したら、Device Manager を閉じます。



⚠ マネージャに使用する物理的な USB ポートを記録して、今後のマネージャの接続には常に同じポートを使用します。モートに対しても同様です。こうすることで、COM ポートの割り当てを維持することができます。

シリアル・ポートの機能

DC9006 をインストールすると、コンピュータ上に 4 つのシリアル・ポートが作成されます。以下の 2 種類のシリアル・ポート経由で各デバイスと相互作用することができます。

- コマンド・ライン・インタフェース（CLI）シリアル・ポート。
  このポートに接続するには、サードパーティのシリアル・ターミナル・ソフトウェアを使用します。
- アプリケーション・プログラミング・インタフェース（API）シリアル・ポート。
  このポートに接続するには、SmartMesh SDK ベースのアプリケーションまたは Stargazer を使用します。

以下の表に、シリアル・ポートのマッピングとその設定を示します。インストールによって作成されたポートが、COM17、COM18、COM19、COM20 である場合、以下の表で「3 番目」と記載したポートは COM19 になり、「4 番目」は COM20 になります。

デバイス	シリアル・ポート番号	用途	ボーレート	データ・ビット	パリティ	ストップ・ビット
SmartMesh IP マネージャ	3 番目*	CLI	9600	8	N	1
	4 番目*	API	115200**	8**	N**	1**
SmartMesh IP モート	3 番目*	CLI	9600	8	N	1
	4 番目*	API	115200**	8**	N**	1**

* FTDI ドライバによって作成されたシリアル・ポート
** デフォルト値

ここで、コンピュータに接続するデバイスの API ポート番号と CLI ポート番号を書き留めておくことをお勧めします。これらの番号は、本書全体を通じて使用します。

Windows 7 へのインストールでは、シリアル・ポート番号が順番に付与されないケースが確認されており、CLI ポートと API ポートがそれぞれ 3 番目と 4 番目にはならない可能性があります。このような場合、APIExplorer（SmartMesh SDK に含まれる）を使用して各ポートを調べると、API ポートが見つかります。また、CLI ポートを見つけるには、ターミナル・プログラムを使用します。

ターミナル・クライアント

Windows XP のデフォルト・シリアル・クライアントはハイパーターミナルであり、これを使用してマネージャおよびモートの CLI と通信することができます。Windows に含まれていない場合や、好みに応じて、別のターミナル・クライアントをインストールすることもできます。

3.2.4 ステップ 2b - SerialMux のインストール

概要

マネージャの API 専用シリアル・ポートは 1 つだけなので、通常、同時に接続できるクライアントは 1 つだけです。シリアル API マルチプレクサ（SerialMux）は、マネージャの API ポートに対する複数の同時接続（Stargazer GUI と SmartMesh SDK に含まれる APIExplorer など）を可能にする Windows サービスです。

インストール

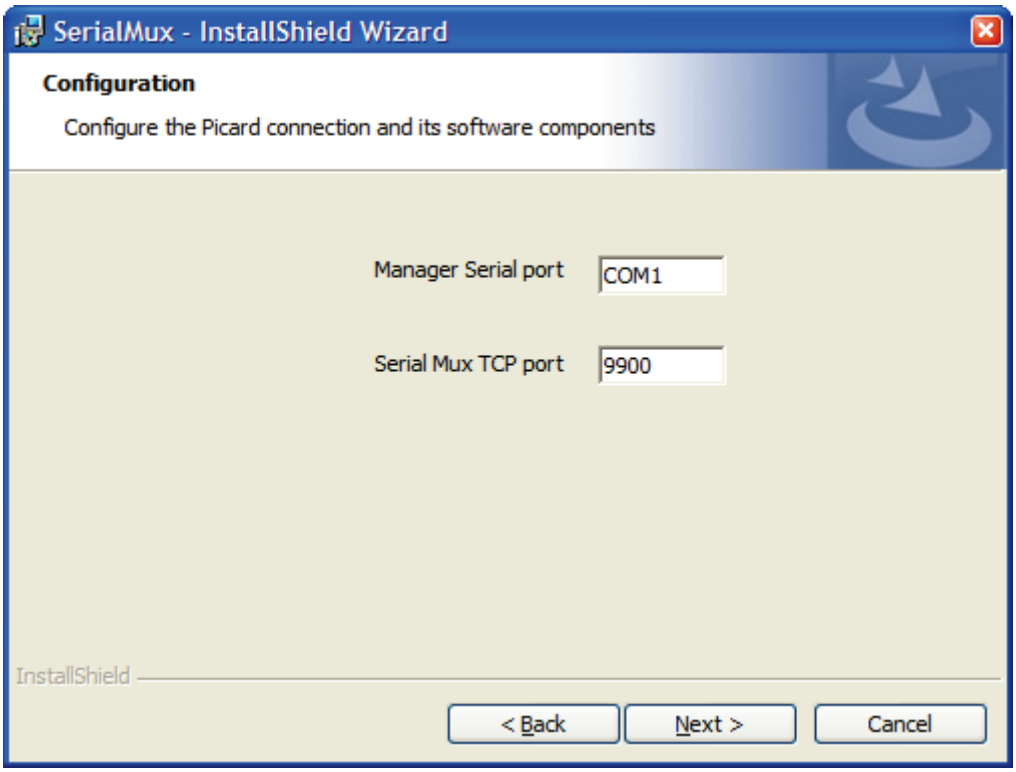
SerialMux ソフトウェアは zip アーカイブ形式で配布されており、*SerialMuxInstaller* に SerialMux のバージョンが付加された名前が付けられています。互換性の関係で特定のバージョンが必要である場合以外は、最新バージョンを選びます。

マネージャに接続したコンピュータに、SerialMux をインストールします。SerialMux に必要なランタイム・コンポーネントがコンピュータ上にない場合は、Windows インストーラによって自動的にインストールされます。インストーラには、使用許諾条件を表示したプロンプトが含まれています。インストールを続行するには条件に同意する必要があります。

SerialMux のインストール（Windows）

1. インストーラの zip アーカイブを解凍します。
2. SerialMux インストーラ *(setup.exe)* を起動します。
3. インストール・ウィザードに従って、SerialMux をインストールします。デスクトップにショートカットが追加されます。SerialMux を実行するには、VC++ 2010 ランタイム・ライブラリが必要です。このライブラリがシステムにインストールされていない場合、インストールの許可が求められます。SerialMux を正しく動作させるためには、このランタイム・ライブラリのインストールを許可する必要があります。

4. Configuration 画面が表示されたら、マネージャに接続したときにマネージャのシリアル API に割り当てられたシリアル・ポート(Windows 上の COM ポート)を入力します。



> ⚠ API シリアル・ポートは、FTDI ドライバのインストール時にインストールされた 4 つのデバイスのうち、4 番目のものになります。例えば、COM3、COM4、COM5、COM6 というシリアル・ポートがインストールされた場合、API ポートは 4 番目の COM6 になります。

COM ポートの再構成

以下のいずれかに当てはまる場合、

- 正しい COM ポートを記録していない
- USB ポート間でマネージャを移動した
- SerialMux をインストールする前に、Stargazer を起動した

Serial Mux Configurator(現在は SmartMesh SDK に含まれる)を使用すると、SerialMux が使用するシリアル・ポートを再構成することができます。

詳しくは、「シリアル API マルチプレクサ(SerialMux)」セクションを参照してください。

3.2.5 ステップ 2c - SmartMesh SDK のインストール

概要

SmartMesh SDK は Python ベースのツール・セットであり、モートおよびマネージャの API をさまざまな側面からデモンストレーションするために使用されます。基本インストールを実行すると、全てのサンプル・アプリケーションを Windows 実行可能ファイルとして実行できるようになります。Python 用の詳細インストール手順は、サンプル・アプリケーションを変更する開発者向けです。

インストール

SmartMesh SDK のダウンロード

ダウンロード・ファイルは、Linear Design Tools サイトの「Dust Networks」部分にあります。SmartMesh SDK のダウンロード・リンクは、「Software Utilities」セクションに含まれています。このリンクをクリックすると、最新バージョンの SmartMesh SDK がダウンロードされます。

SmartMesh SDK 全体が、*SmartMeshSDK-full-X.X.X.X.zip* ファイルに含まれています。

- 最新版の SmartMeshSDK zip ファイルの *SmartMeshSDK-full-X.X.X.X.zip* をダウンロードします。
- ファイルを解凍します。同じ名前のフォルダが作成され、4 つのサブ・フォルダ（*-/api*、*/doc*、*/src*、*/win*）が作成されます。
    - スタンドアロン・アプリケーションは、*/win* に含まれています。
- その他のインストールは不要です。*SmartMeshSDK-X.X.X.X/*フォルダは、どこでも使いやすい場所に移動することができます。

SmartMesh SDK の内容と使用法について、詳しくは SmartMesh IP ツール・ガイドまたは SmartMesh WirelessHART Tools Guide を参照してください。

開発者向けの詳細手順

ソース・コードを実行するためのインストール

SmartMesh SDK サンプル・アプリケーションをソース・コードから直接実行するには、Python と追加ライブラリをインストールする必要があります。この作業が必要になるのは、サンプル・アプリケーションの動作を変更する場合です。また、Windows 実行可能ファイルを直接実行できないシステムでアプリケーションを実行する場合も、この作業が必要です。

Python のインストール手順は、Windows の 32 ビット・バージョンと 64 ビット・バージョンで異なります。以下の適切な手順に従ってください。

Windows XP または 32 ビット Windows 7 へのインストール

Python 2.7 のインストール

1. http://python.org/download/から、最新の Windows 向け Python 2.7 インストーラをダウンロードします。
   本書の執筆時点での最新インストーラは、*python-2.7.2.msi* でした。
2. インストーラをダブルクリックし、全てデフォルト設定のままでインストール手順に従います。

PySerial のインストール

PySerial は、Python にシリアル・ポートのサポートを追加します。

1. http://sourceforge.net/projects/pyserial/files/pyserial/から、Python 2.7 向けの PySerial をダウンロードします。
   本書の執筆時点での最新バージョンは、pyserial-2.5 でした。

> ⚠ 必ず Python 2.7 向けの PySerial（例：*pyserial-2.5.win32.exe*）をインストールします。
>
> Python 3.0 向けの PySerial（*pyserial-py3k-2.5.win32.exe*）をインストールしないでください。

2. インストーラをダブルクリックし、全てデフォルト設定のままでインストール手順に従います。

PyWin32 のインストール

このステップは省略可能です。ソースから Serial Mux Configurator を実行しない場合、この作業を実行する必要はありません。Serial Mux Configurator は、実行可能ファイルから実行することができます。

PyWin32 は、Windows 固有のサポートを Python に追加します。SmartMesh SDK に含まれるアプリケーションのほとんどはプラットフォームに依存しませんが、Serial Mux Configurator などのアプリケーションには、プラットフォーム固有の要件があります。PyWin32 のインストーラは、各種の Python バージョンによって異なります。

1. PyWin32 インストーラをダウンロードします。
   Pywin32 ダウンロード・ページ（http://sourceforge.net/projects/pywin32/files/pywin32/）で、使用する Python バージョンと、32 ビットまたは 64 ビットのいずれのバージョンを実行するかに応じて、適切な Pywin32 インストーラを見つけます。
   本書の執筆時点では、Python 2.7 向けの最新 32 ビット・ビルドは、pywin32 build 217 でした。
2. インストーラをダブルクリックし、全てデフォルト設定のままでインストール手順に従います。

64 ビット Windows 7 へのインストール

何らかの理由で 64 ビットの Python と PySerial を実行する必要がある場合を除いて、通常は上記のインストール手順に従って、32 ビット・バージョンを実行することをお勧めします。

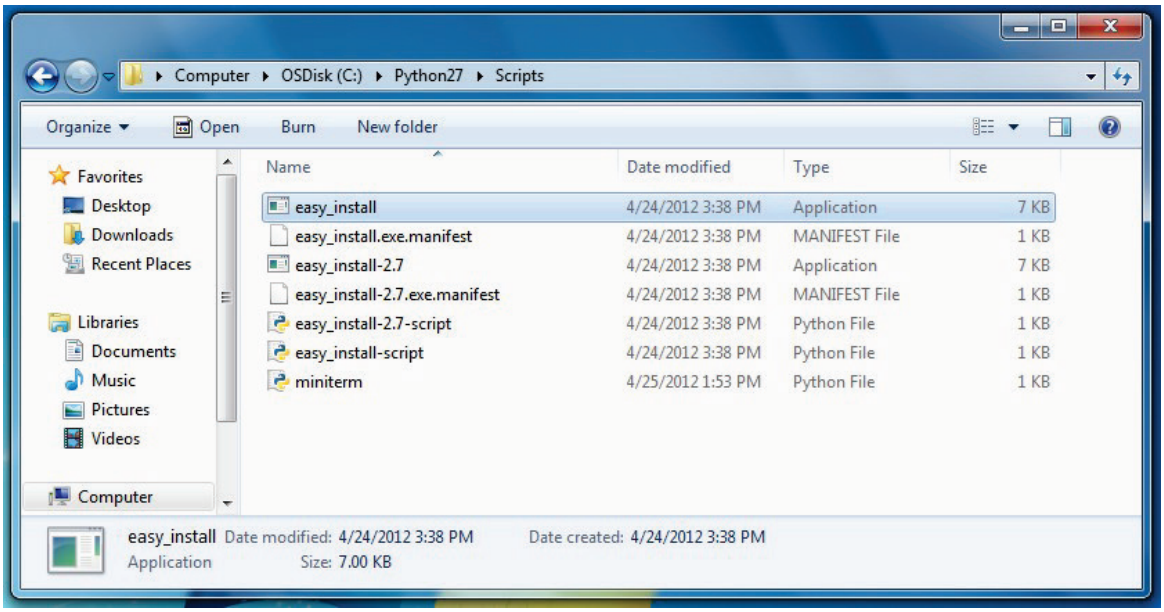
Python 2.7 のインストール

1. http://python.org/download/から、最新の Windows X86-64 向け Python 2.7 インストーラをダウンロードします。本書の執筆時点での最新インストーラは、*python-2.7.2.amd64.msi* でした。
2. インストーラをダブルクリックし、全てデフォルト設定のままでインストール手順に従います。

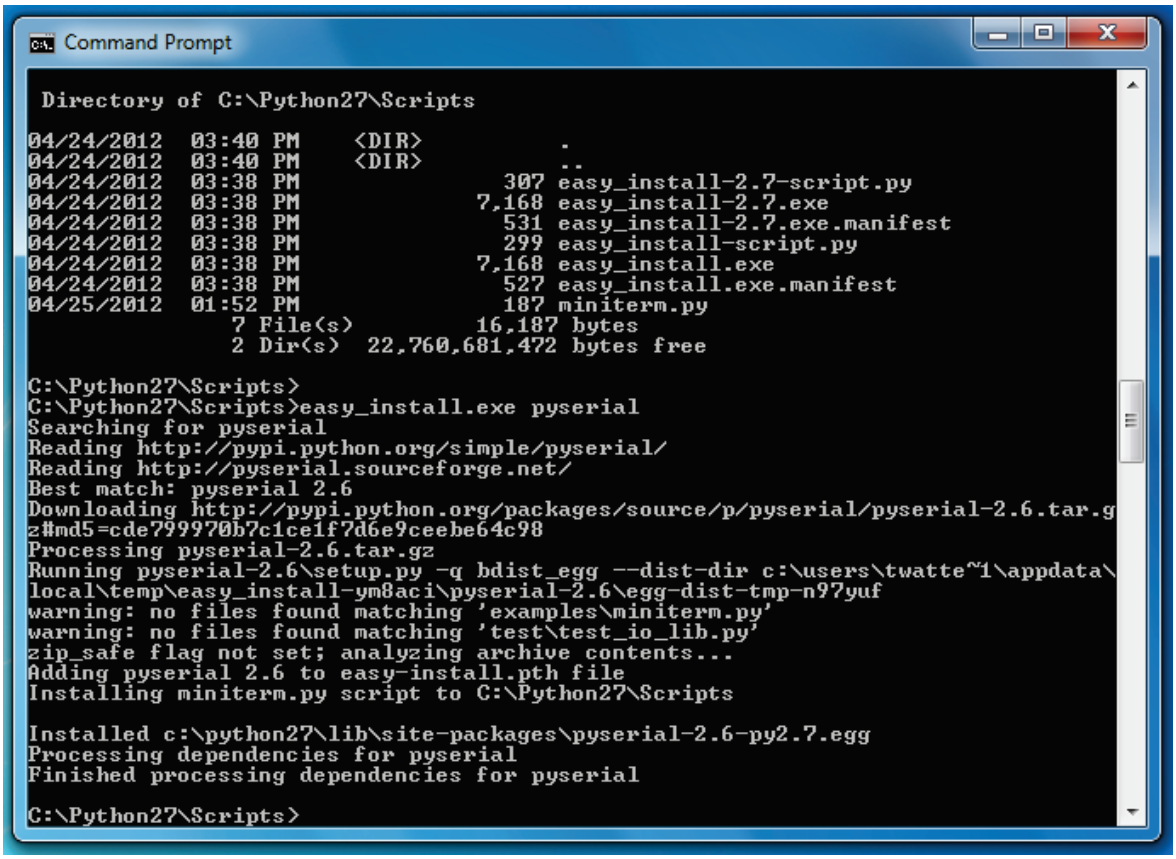
PySerial のインストール

64 ビットの Windows 7 では、Pyserial のインストール手順が異なります。以下の 2 つのステップを実行します。

1. http://pypi.python.org/pypi/setuptools から setuptools をインストールします。本書の執筆時点での最新バージョンは、setuptools 0.6c11 です。
    1. Windows 向けのインストールのセクションに移動し、手順に含まれるリンクから *ez_setup.py* をダウンロードします。
    2. スクリプト・アイコンをダブルクリックして、ダウンロードした *ez_setup.py* ファイルを実行します。Python フォルダ内に Scripts というフォルダが作成されます。



2. 上記ステップで作成された *easy_install.exe* スクリプトを使用して、PySerial をインストールします。*easy_install.exe* スクリプトは、Python フォルダ内（例：C:\Python27\Scripts）にあります（Python のインストール時に別の場所を指定した場合、このフォルダの場所は異なる可能性があります）。
    1. Windows Command Prompt を開きます。C:\Python27\Scripts フォルダに移動します（Command Prompt を開く方法は、http://www.sevenforums.com/tutorials/947-command-prompt.html を参照）。
    2. *easy_install.exe* を実行し、コマンド・ラインの引数として「pyserial」を渡します。
    例：*easy_install.exe pyserial*



PyWin32 のインストール

このステップは省略可能です。ソースから Serial Mux Configurator を実行しない場合、この作業を実行する必要はありません。Serial Mux Configurator は、実行可能ファイルから実行することができます。

PyWin32 は、Windows 固有のサポートを Python に追加します。SmartMesh SDK に含まれるアプリケーションのほとんどはプラットフォームに依存しませんが、Serial Mux Configurator などのアプリケーションには、プラットフォーム固有の要件があります。PyWin32 のインストーラは、各種の Python バージョンによって異なります。

1. PyWin32 インストーラをダウンロードします。Pywin32 ダウンロード・ページ（http://sourceforge.net/projects/pywin32/files/pywin32/）で、使用する Python バージョンと、32 ビットまたは 64 ビットのいずれのバージョンを実行するかに応じて、適切な Pywin32 インストーラを見つけます。
本書の執筆時点では、Python 2.7 向けの最新 64 ビット・ビルドは、pywin32 build 217 でした。
2. インストーラを実行します。

Linux（Ubuntu）へのインストール

Python 2.7 のインストール

Python 2.7 は、標準の Ubuntu ディストリビューションに含まれています。Python が含まれていない場合、Ubuntu パッケージ・マネージャからインストールできます。

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

PySerial のインストール

PySerial は、Ubuntu パッケージ・マネージャを使用してインストールできます。

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

Macintosh OS X へのインストール

Python 2.7 のインストール

Python 2.7 は、標準の OS ディストリビューションに含まれています。Python が含まれていない場合、macports などのパッケージ・マネージャを使用してインストールできます。

```
$ sudo port install python27
```

PySerial のインストール

PySerial は、こちらから入手できます。/documentation/pyserial.rst ファイル内の手順に従います。

インストールのテスト

1. */src/*ディレクトリに移動し、次に *bin/InstallTest* ディレクトリに移動します。
2. *InstallTest.py* をダブルクリックします。
3. インストールが完了している場合、Python コマンド・ウィンドウが開き、以下のテキストが表示されます。

```
Installation test script - Dust Networks SmartMeshSDK


Step 1. Python version
You are running Python 2.7.1 PASS

Step 2. PySerial installation
PASS

Press Enter to exit.
```

4. 全てのテストで「*PASS*」が表示されることを確認します。

3.2.6 ステップ 2d - Stargazer のインストール

Stargazer GUI

Stargazer GUI は、ネットワークの視覚化と相互作用を可能にするオプション・コンポーネントです。Stargazer をインストールする前に、シリアル・ドライバ（ステップ 2a）と SerialMux（ステップ 2b）をインストール済みであることを確認します。

前提条件

Stargazer をインストールするには、Windows XP または Windows 7 を実行しているコンピュータに SmartMesh IP マネージャを接続しており、SerialMux をインストールしている必要があります。これらのコンポーネントについて、詳しくは SmartMesh IP ツール・ガイドの「セットアップ」セクションを参照してください。

Stargazer のダウンロード

ダウンロード・ファイルは、Linear Design Tools サイトの「Dust Networks」部分にあります。Stargazer のダウンロード・リンクは、「Software Utilities」セクションに含まれています。このリンクをクリックすると、最新バージョンがダウンロードされます。

- ファイルを解凍して、インストーラ（setup.exe）とその他のインストール・ファイルを展開します。
- インストーラを実行します。.NET フレームワークがインストールされていない場合は、ダウンロードするように要求される場合があります。
- デスクトップに Stargazer を起動するショートカットが作成されます。

Stargazer の使用法について、詳しくは本書の「Stargazer GUI」セクションを参照してください。

Stargazer のインストール

Stargazer のディストリビューションは、*StargazerInstaller* zip ファイルとして提供されています。

SerialMux がインストールされており、マネージャに接続されたコンピュータに、Stargazer をインストールします。Microsoft .NET Framework 4.0 がインストールされていない場合は、インストーラによって自動的にインストールされます。インストーラには、使用許諾条件を表示したプロンプトが含まれています。インストールを続行するには条件に同意する必要があります。

1. Stargazer インストーラ（*setup.exe*）を起動します。
2. インストール・ウィザードに従って、Stargazer アプリケーションをインストールします。デスクトップに Stargazer アプリケーションのショートカットが追加されます。Stargazer を実行するには、.NET 4.0 Client のランタイムが必要です。このコンポーネントがシステムにインストールされていない場合、インストールの許可が求められます。Stargazer を正しく動作させるためには、.NET ランタイムのインストールを許可する必要があります。

使用している Windows のバージョンによって、ファイルがインストールされる場所は異なります。

- Windows XP の場合、実行可能ファイルは *C:\Program Files\Dust Networks\Stargazer* にインストールされ、構成ファイルは *C:\Documents and Settings\All Users\Application Data\Dust Networks\Stargazer\Default* にインストールされます。
- Windows 7 の場合、実行可能ファイルは *C:\Program Files\Dust Networks\Stargazer*（64 ビット・システムの場合は *C:\Program Files (x86)\Dust Networks\Stargazer*）にインストールされ、構成ファイルは *C:\ProgramData\Dust Networks\Stargazer\Default* にインストールされます。

3.3 トラブルシューティング

3.3.1 Linux での FTDI ドライバのインストール

ここでは、一般的な Linux ディストリビューションに対するトラブルシューティングのヒントを提供します。

> ⚠ SmartMesh SDK に含まれる全てのコンポーネントが、Linux でテストされているわけではありません。ここから先の作業は自己責任となりますのでご注意ください。

Ubuntu での FTDI ドライバのインストール

最近の Ubuntu リリース（12.04 以降）では、標準リリースに FTDI ドライバが含まれているため、デバイスの接続時に、カーネルによって FTDI ドライバがロードされます。

```
$ 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
```

上記の出力から判断すると、デバイスの API との通信に使用するシリアル・ポートは、*/dev/ttyUSB3* になります。

非特権ユーザー（root 以外）としてこのデバイスにアクセスするには、権限を修正する必要があります。以下のコマンドを使用して、*dev/ttyUSB2* と*/dev/ttyUSB3* の権限を変更します。

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

3.3.2 Macintosh OS X での FTDI ドライバのインストール

> ⚠ SmartMesh SDK に含まれる全てのコンポーネントが、OS X でテストされているわけではありません。ここから先の作業は自己責任となりますのでご注意ください。

OS X での FTDI ドライバのインストール

OS X 10.5、10.6、10.7 では、http://www.ftdichip.com/Drivers/VCP.htm からダウンロードできるディスク・イメージを使用して、FTDI ドライバをインストールします（OS X Installation Guide を参照）。

ドライバをインストールしたら、USB ケーブルを接続します。シリアル・ポートに接続するツールへの入力として、デバイス名が必要になります。

デバイス名を確認するには、ターミナル・アプリケーションに以下のコマンドを入力します。

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

/dev ディレクトリに、以下の形式でいくつかのエントリが出力されます。

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

デバイスがどの USB ポートに接続しているかによって、xxxxxxxx はデバイスのシリアル番号かロケーション文字列のいずれかになります。最後の文字（A、B、C、D のいずれか）がシリアル・ポートを示しています。C で終わるポートは CLI ポートであり、D で終わるポートは API ポートです。

screen プログラム（OS X に付属）をシリアル・ターミナルとして使用すると、CLI ポートに接続できます。

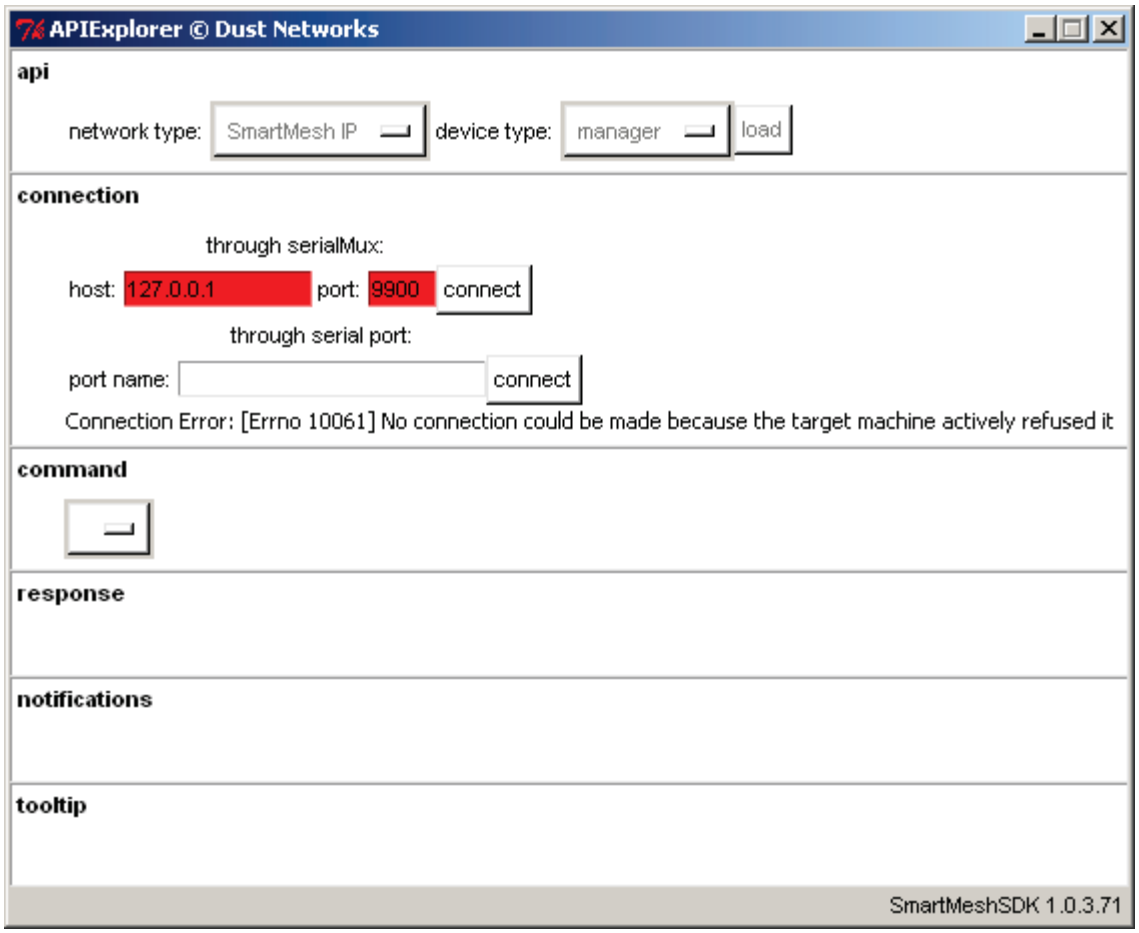
```
$ screen /dev/tty.usbserial-01234567C 9600

> help
...
```

3.3.3 マネージャ接続に関するトラブル

症状 - 接続拒否エラー

SerialMux を介してマネージャに接続しようとすると、接続拒否エラーが表示されます（以下の APIExplorer のスクリーンショットを参照）。



解決策

接続拒否エラーには、さまざまな原因が考えられます。

- 原因 1 - SerialMux が実行されていない
  - SerialMux が実行中であることを確認します。サービスの管理方法については、SmartMesh IP ツール・ガイドの「SerialMux の構成」セクションを参照してください。
- 原因 2 - SerialMux は実行されているが、誤った TCP ポートをリスニングしている（例：上の図で SerialMux が 9901 ポートをリスニングしていた）
  - SmartMesh SDK に含まれる MuxConfig ツールを使用して、正しい TCP ポートを確認します。
- 原因 3 - SerialMux が正しいポートで実行されているが、マネージャが接続されていない
  - Windows Device Manager を開き、マネージャが割り当てられた COM ポートがリストに含まれることを確認します。また、MuxConfig ツールで、SerialMux が正しい COM ポートを指していることを確認します。マネージャの電源がオンになっており、青色 LED（INDICATOR_0）が点灯していることを確認します。

症状 - 接続エラー

シリアル経由でマネージャに接続しようとすると、接続エラーが表示されます。

解決策

接続エラーには、いくつかの原因が考えられます。

- 原因 1 - マネージャの電源がオンになっていない
  - マネージャの電源がオンになっており、青色 LED（INDICATOR_0）が点灯していることを確認します。
- 原因 2 - 指定されたシリアル・ポートが存在しない
  - 以前はポートが存在していた場合
    - Windows Device Manager を開き、いったんマネージャの接続を解除してから再接続します。表示される COM ポートを書き留めて、使用しているツール（例：APIExplorer）で正しいポートを選択します。
  - マネージャを接続してもポートが表示されない場合、DC9006A ボードで 3 つの LED が点灯していることと、FTDI VCP ドライバをインストールしていることを確認します。

3.3.4 通知の未取得

SmartMesh IP マネージャでは、サブスクライブしない限り、デフォルトでは通知が取得されません。*サブスクライブ* API については、SmartMesh IP Manager API Guide を参照してください。API のテストには、SmartMesh SDK に含まれる APIExplorer を使用できます。現在定義されている全ての通知をサブスクライブする場合、0x76（118）ビットマップを使用します。

3.3.5 ネットワーク ID の変更

> ⚠ この作業が必要になるのは、その他の SmartMesh IP ネットワークと同じ無線空間でネットワークを運用している場合のみです。

ネットワーク識別子（netid）は、ネットワークの 16 ビット識別子です。デフォルトでは、1229 に設定されています。全てのデバイスで netid を変更するには、以下の手順に従います。

- USB 経由でデバイスをコンピュータに接続します。
- 上記設定を使用して、シリアル・ターミナルを使用しているデバイスの CLI ポートを開きます。
- デバイスのスイッチをオンにします。
- SmartMesh IP マネージャに接続している場合、以下のコマンドを実行します（ここでは、netid に 100 を設定）。

```
> 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
```

- SmartMesh IP モートに接続している場合、以下のコマンドを入力します。

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

SmartMesh IP mote, ver 1.1.0.37
```

- 全てのデバイスに対して上記の処理を繰り返します。

新しい netid を有効にするには、各デバイスをリセットする必要があります。

3.3.6 マスター/スレーブ

モードの動作

モートには、ジョインおよびコマンド終了の動作を制御する 2 つのモードがあります。

- **マスター・モード**では、モートが実行するアプリケーションがコマンドを終了し、ジョインを制御します。デフォルトでは、スタータ・キットに含まれる全てのモートが**マスター・モード**に設定されています。**マスター・モード**では API は無効化されています。
- **スレーブ・モード**では、モートにシリアル接続されたデバイスによってコマンドの終了とジョインの制御が実行されます。デフォルトでは、モート自体はネットワークにジョインしません。**スレーブ・モード**では、API が有効化されており、シリアル接続されたアプリケーション（APIExplorer など）がデバイスに接続します。*SetParameter*（SmartMesh IP のみ）を使用して `autojoin` が有効化されている場合、シリアル・アプリケーションが `join` コマンドを発行しなくても、スレーブであるモートがネットワークにジョインします。

モードは CLI の `set` コマンドで設定され、リセットするまで維持されます（不揮発性）。

LED

モート（DC9018）が**マスター・モード**になっている場合、モートは自動的に検索を開始するとともに、電源投入後すぐに STATUS_0 の LED が点滅し始めます。モートがジョインすると、STATUS_0 と STATUS_1 の LED が両方とも点灯します。スレーブ・モードになっていると LED は点灯しません。バッテリ切れと間違わないようにご注意ください。

> ⚠ DC9018 ボードの LED が点灯するのは、LED_EN ジャンパが短絡している場合のみです。マスター・モードの LED は、バージョン 1.1.2 以上の SmartMesh WirelessHART モートでサポートされています。

スレーブ・モードへの切り替え

スタータ・キット（DC9021A および DC9022A）に含まれるモートは、デフォルトで**マスター・モード**に設定されています。現在の設定を確認するには、USB ケーブル経由でモートをコンピュータに接続し、CLI コマンドの `get mode` を使用します。モートを**スレーブ・モード**に設定するには、CLI コマンドの `set mode` を使用します。

`get mode` コマンドを使用すると、現在のモードを表示できます。

```
> get mode
master
```

set mode コマンドを使用すると、**スレーブ・モード**に切り替えることができます。

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

⚠ モード変更を有効にするには、モートをリセットする必要があります。いったん設定したモードは、リセット後も維持されます。

マスター・モードへの切り替え

現在の設定を確認するには、USB ケーブル経由でモートをコンピュータに接続し、CLI コマンドの *get mode* を使用します。モートを**マスター・モード**に設定するには、CLI コマンドの *set mode* を使用します。

get mode コマンドを使用すると、現在のモードを表示できます。

```
> get mode
slave
```

set mode コマンドを使用すると、**マスター・モード**に切り替えることができます。

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

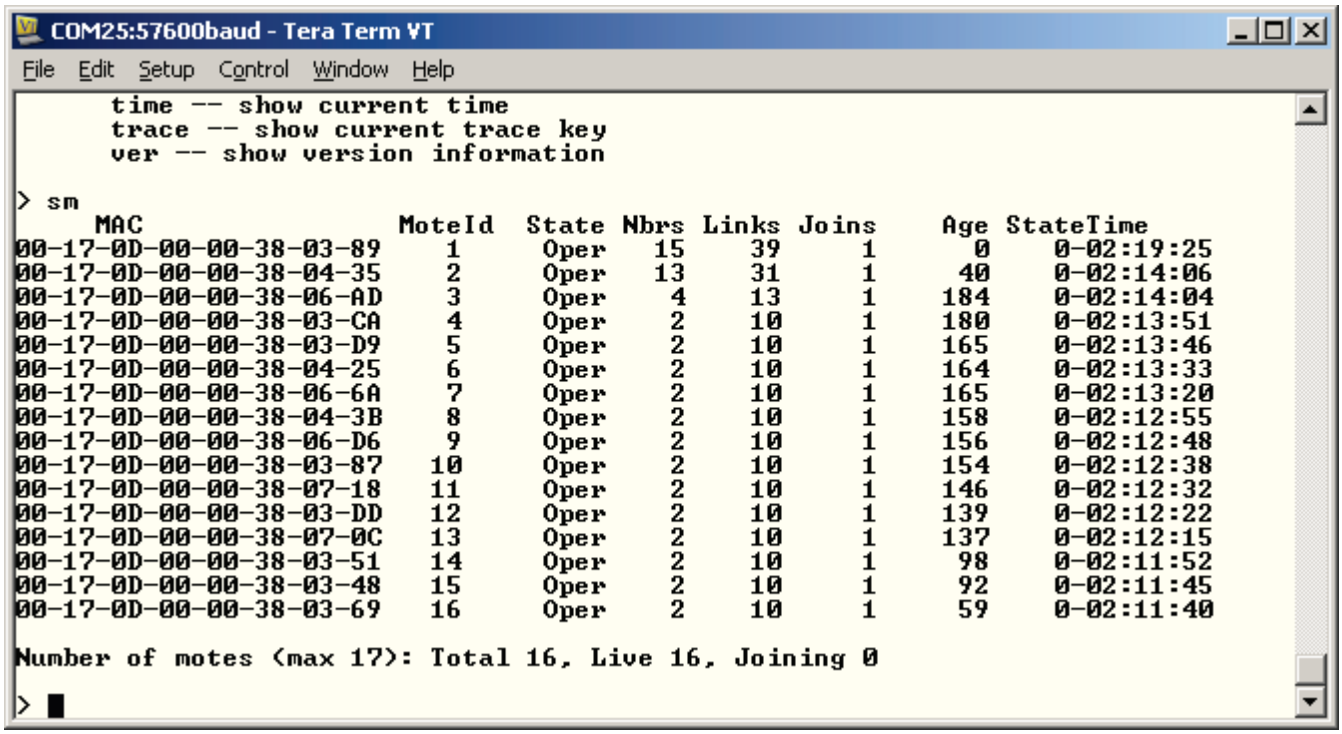
⚠ *set mode* コマンドを有効にするには、モートをリセットする必要があります。いったん設定したモードは、リセット後も維持されます。

4 シリアル・ターミナル・クライアント

ここでは、コマンド・ライン・インタフェース（CLI）を介して、デバイスとやり取りするために使用できるサードパーティのシリアル・ターミナル・クライアントを紹介します。モートまたはマネージャについての詳細は、それぞれの CLI ガイドを参照してください。

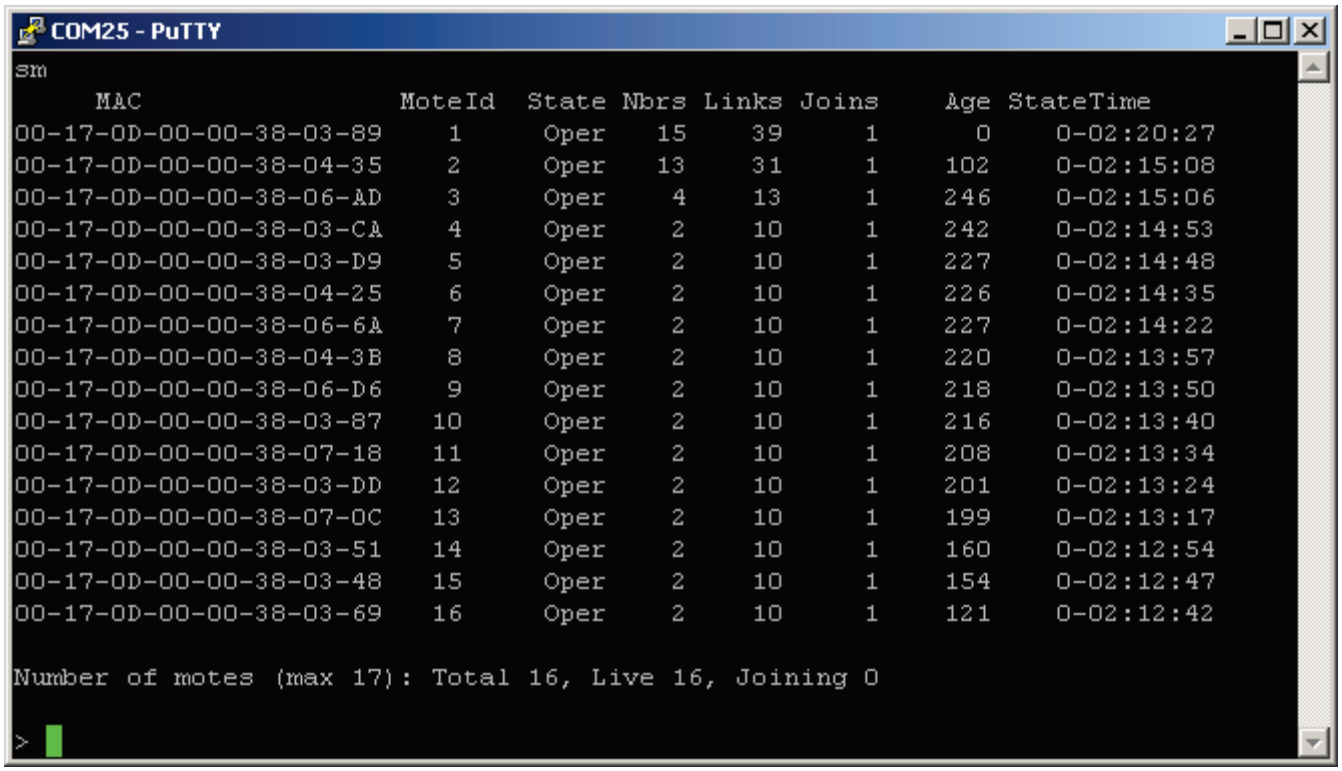
4.1 TeraTerm

- サポートされるプラットフォーム：Windows
- ダウンロード：http://ttssh2.sourceforge.jp/index.html.en



4.2 PuTTY

- サポートされるプラットフォーム：Windows
- ダウンロード：http://www.chiark.greenend.org.uk/~sgtatham/putty/



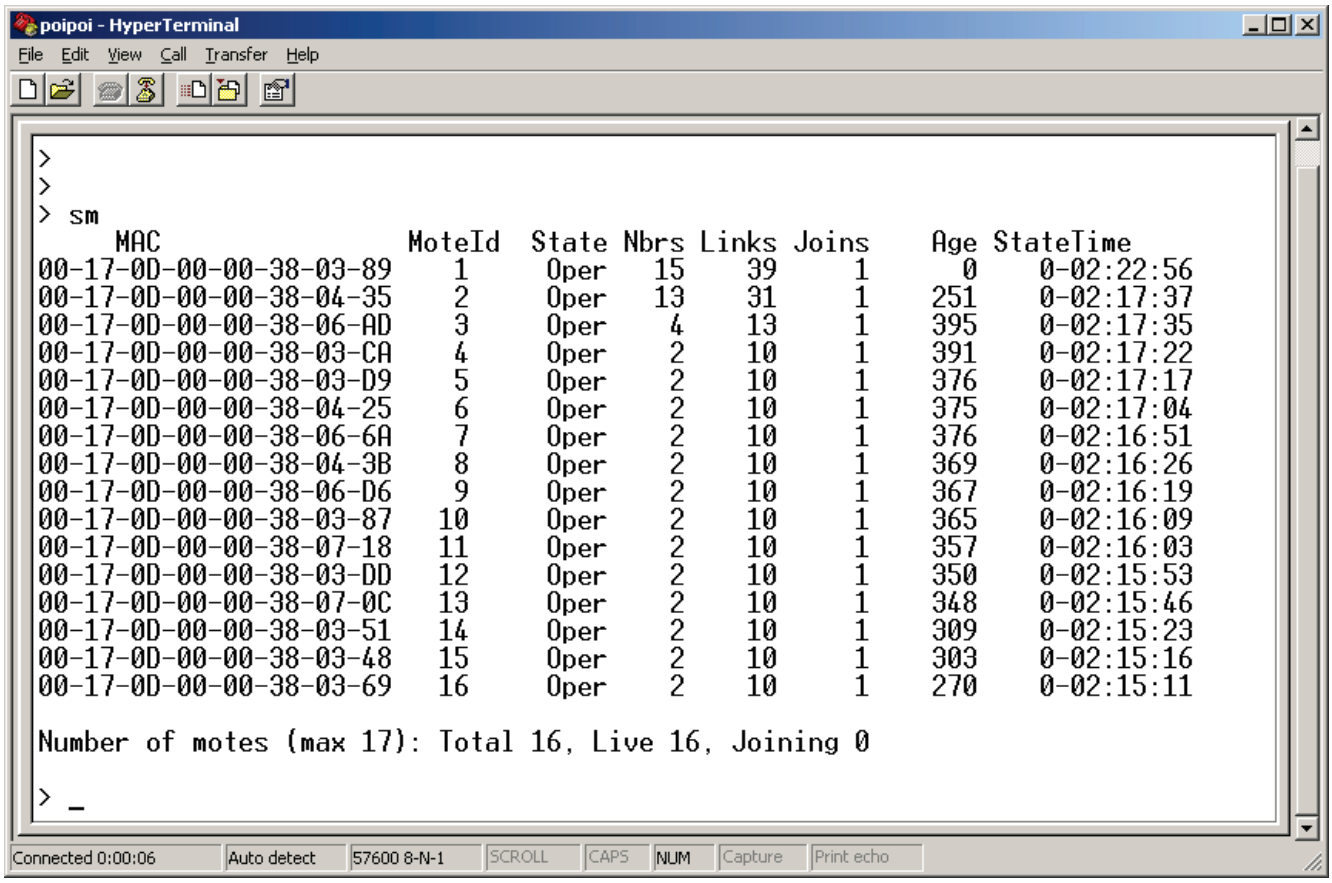
4.3 minicom

- サポートされるプラットフォーム：Linux、Unix
- ほとんどのディストリビューションにプリインストール済み。OS X では、fink や macports などのパッケージ・マネージャから使用可能。
- ダウンロード：http://alioth.debian.org/projects/minicom/

4.4 Microsoft Windows ハイパーターミナル

- サポートされるプラットフォーム：Windows XP
- Windows XP にプリインストール済み。

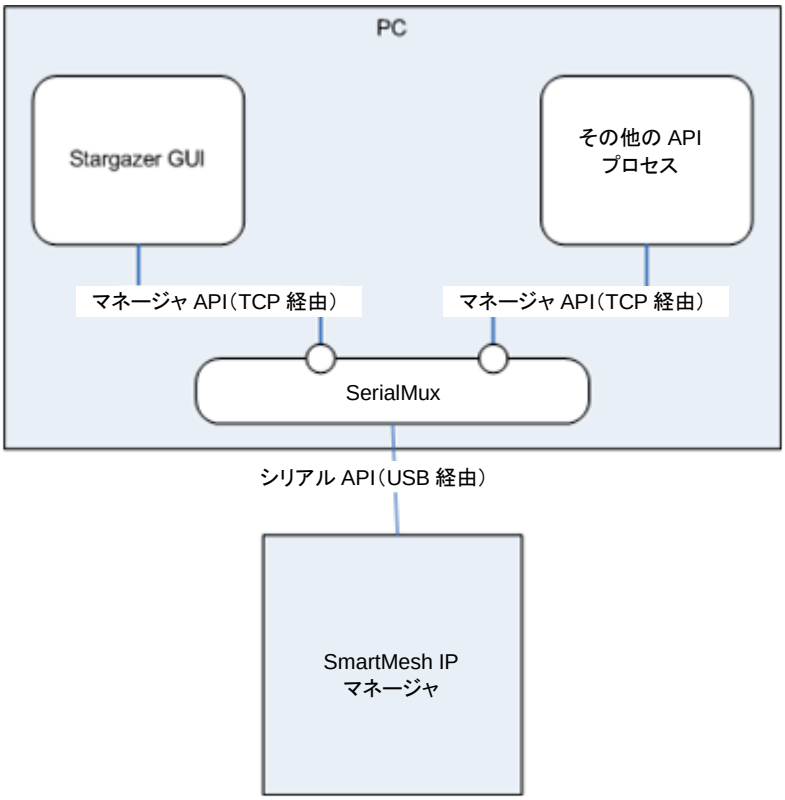
⚠ Windows Vista または Windows 7 にはインストールされていません。


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 シリアル API マルチプレクサ（SerialMux）

5.1 概要

ここでは、SmartMesh IP マネージャ向けのシリアル API マルチプレクサ（SerialMux）について説明します。SerialMux は単純なマルチプレクサであり、複数のプロセスが TCP 接続を介してマネージャのシリアル API と通信できるようにします。Windows では、SerialMux インストーラを使用して、バックグラウンド・サービスとして SerialMux をインストールすることができます。Stargazer GUI アプリケーションと SmartMesh SDK はどちらも、SerialMux を介してマネージャと通信します。



- インストールについては、本書の「セットアップ」セクションを参照してください。
- SerialMux の手動構成については、「SerialMux の構成」セクションを参照してください。
- SmartMesh SDK に含まれる *MuxConfig* ツールを使用すると、SerialMux の構成を編集できます。
- 「SerialMux のプロトコル」セクションでは、SerialMux を介したクライアントとマネージャの通信方法について説明しています。

5.2 SerialMux の構成

SerialMux は、シリアル・ポート（COM ポート）を使用してマネージャと通信し、特定の TCP ポートへのクライアント接続を受け入れます。シリアル・ポートと TCP ポートの構成パラメータは構成ファイルに格納されていますが、このファイルの場所は OS によって異なります。ここでは、構成パラメータを手動で編集する方法について説明します。ここで示す手順は参考までに記載しています。MuxConfig GUI ツールを使用すると、大幅に簡単に構成を追加および編集することができます。

手順は以下のとおりです。

1. SerialMux 構成ファイルの編集
2. SerialMux Windows サービスの再起動

5.2.1 ステップ 1：SerialMux 構成ファイルの編集

Windows XP では、以下の場所に構成ファイルがあります。

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

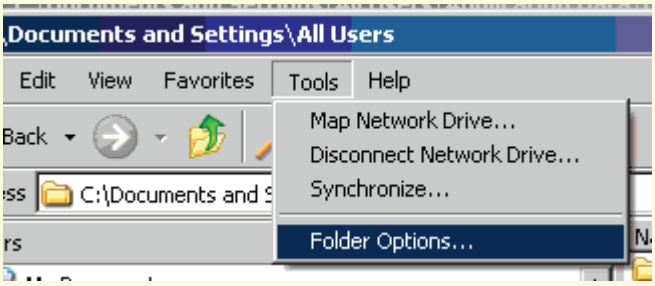
Windows 7 では、以下の場所に構成ファイルがあります。

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

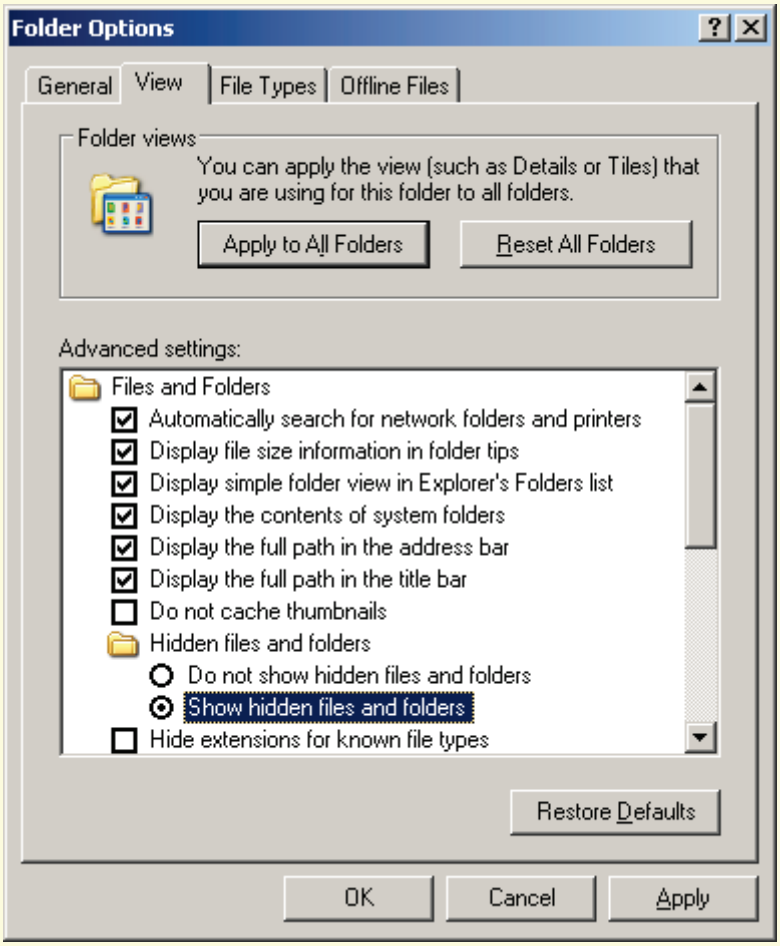
Windows では、*Application Data* フォルダがデフォルトで非表示になっています。

このフォルダを表示するには、以下の手順に従います。

1. Windows のブラウザ・ウィンドウで、**Tools > Folder Options** を選択します。



2. View タブで、Show hidden files and folders を選択します。



3. SerialMux 構成ファイルの編集が終わったら、上記手順で Do not show hidden files and folders を選択します。

テキスト・エディタで構成ファイルを開きます。以下の行を探します。

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

各要素の示す内容は以下のとおりです。

- *port* は、SerialMux がリスニングするシリアル（COM）ポートです（デフォルト値は COM1）。
- *listen* は、SerialMux が接続を受け入れる TCP ポートです（デフォルト値は 9900）。

設定を使用に適したシリアル・ポートへ変更し、他のサービスと競合しない TCP ポートを選択します。

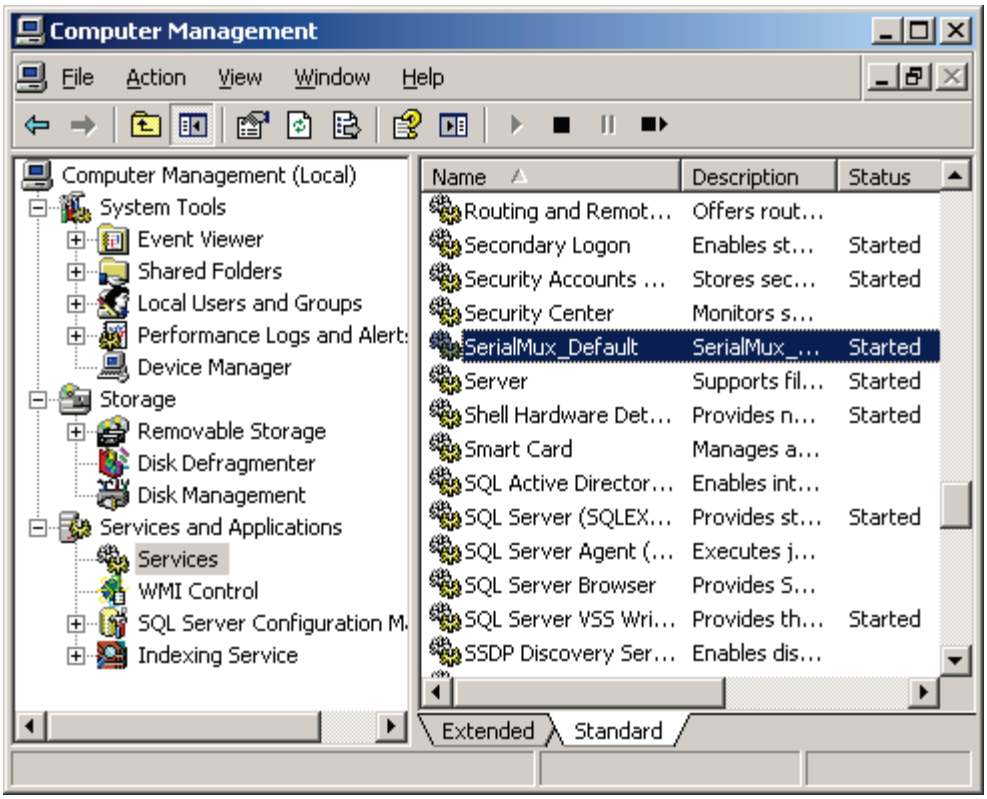
5.2.2 ステップ 2：SerialMux Windows サービスの再起動

SerialMux Windows サービスは、バックグラウンドで常時実行されています。SerialMux は起動時にのみ構成ファイルを読み取るため、構成ファイルの変更を有効にするには、サービスを再起動する必要があります。

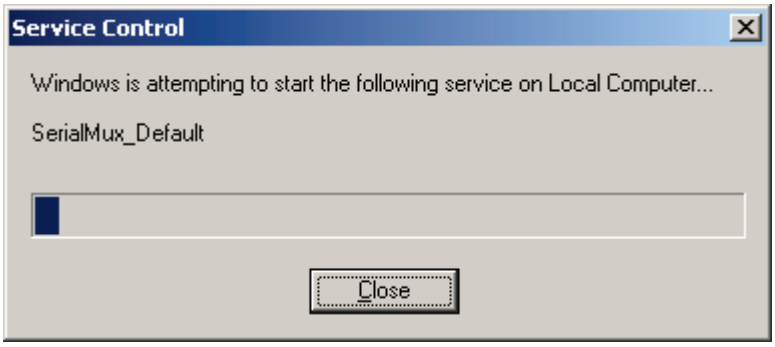
1. **Start** メニューで、**Settings > Control Panel** を選択します。
2. **Administrative Tools** を選択します。

3. **Computer Management** を選択します。

4. **Services and Applications > Services** に移動します。右側のリストで **SerialMux_Default** サービスを右クリックし、**restart** を選択します。



5. プログレス・バーが表示され、サービスが再起動されます。



5.2.3 SerialMux の詳細構成

SerialMux の構成パラメータは、構成ファイル内で設定します。
Windows XP では、以下の場所に構成ファイルがあります。

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

Windows 7 では、以下の場所に構成ファイルがあります。

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

インストーラはユーザーの入力に基づいて、単純なデフォルト構成ファイルを作成します。

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

構成ファイルのパラメータ

パラメータ名	概要
port	マネージャが接続されるシリアル・ポート。インストーラまたは Serial Mux Configurator で設定。
listen	クライアント接続用の TCP ポート。デフォルト・ポートは 9900。
authToken	Hello メッセージでクライアントが使用する必要のある認証トークン。デフォルト・トークンはバイト文字列「*0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37*」。
accept-anyhost	任意のコンピュータから SerialMux への接続を許可。デフォルトでは、ローカル・マシンからの接続のみ許可。
log-file	書き込み先のログ・ファイル名を指定。ログ・ファイルは、同じ構成を使用している他の SerialMux プロセスを検出するためのロック・ファイルとしても使用される。
log-level	出力するログ・メッセージの詳細レベルを指定。有効な値は、*trace*、*info*、*warning*、*error*。
log-num-backups	保存するバックアップ・ログ・ファイルの数。
log-max-size	ログ・ファイルを循環使用するログ・ファイル・サイズ（バイト）。

5.2.4 SerialMux から複数のマネージャへの接続

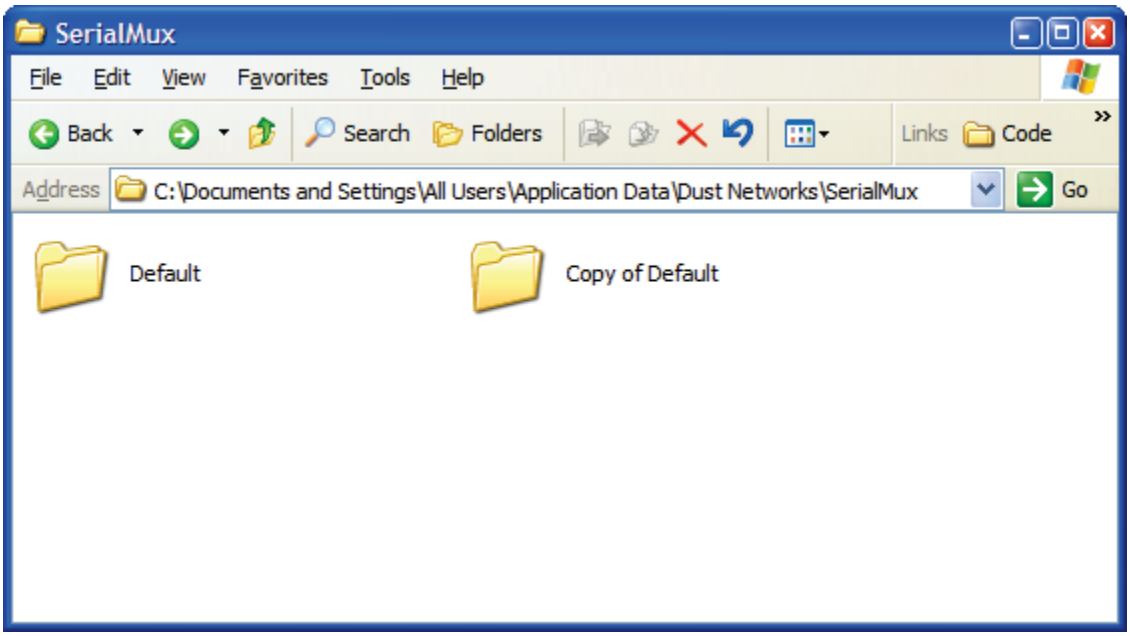
標準インストールの SerialMux でサポートされるのは、1 つのマネージャへの接続です。複数のマネージャに接続するには、SerialMux サービスのコピーを複数（マネージャごとに 1 つ）作成する必要があります。

ここで示す手順は参考までに記載しています。MuxConfig GUI ツールを使用すると、大幅に簡単に構成を追加および編集することができます。

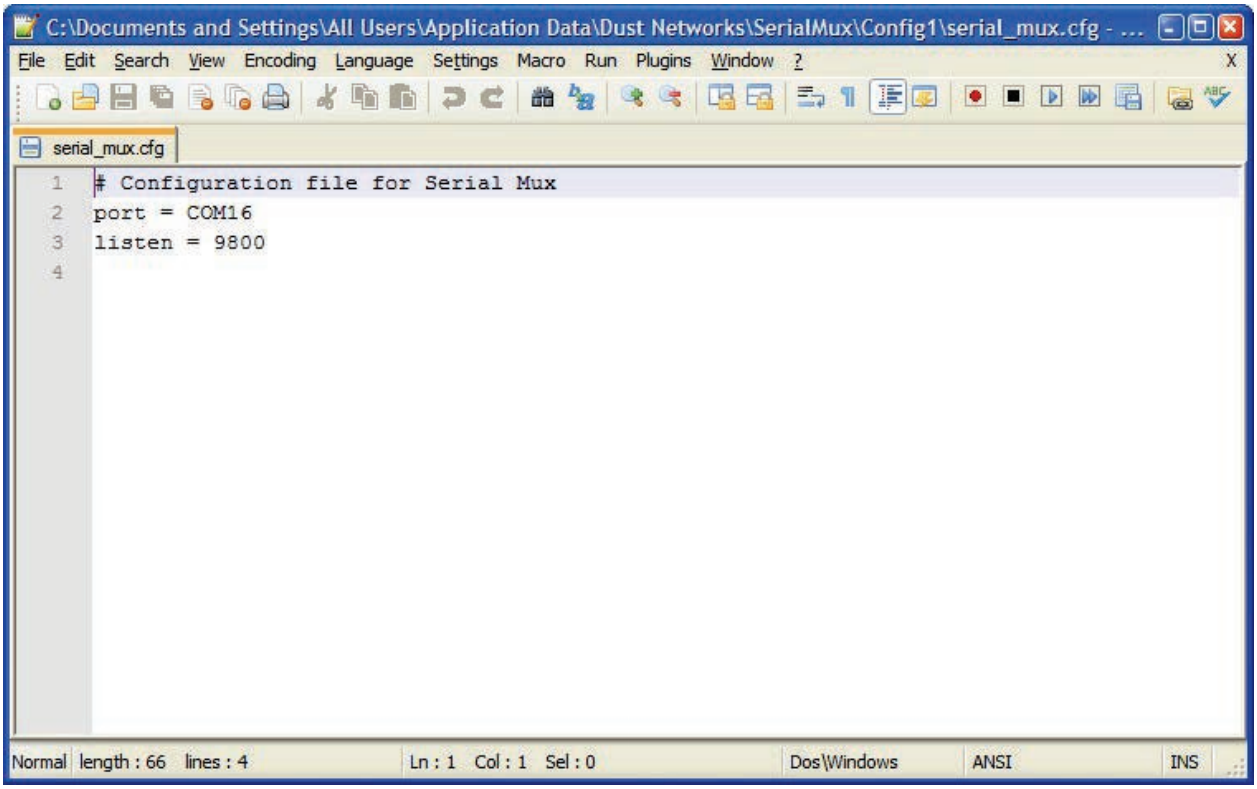
SerialMux 構成の作成

以下の手順に従うと、追加の SerialMux 構成を作成できます。

1. SerialMux の構成ディレクトリ「*Default*」をコピーして、名前を変更します（ここでは、「*Copy of Default*」を「*Config1*」に変更）。



2. コピーした構成ファイルを変更します。port には 2 番目のマネージャの API（4 番目）ポートを指定し、listen には他と競合しないポートを指定します。ここではポート 9800 を使用しています。



3. *sc* コマンドを使用して、新しい SerialMux サービスを作成します。一貫性を維持するため、サービスには構成ディレクトリに対応する名前を付けることをお勧めします（例：*SerialMux_Config1*）。
   Windows Command Prompt で、以下のコマンドを 1 行で入力します。ここでは、ページに合わせて 2 行に分けて表示しています。

```
> 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\""
```

   binPath パラメータには、サービスに対して実行するコマンド・ライン「*<path>/serial_mux.exe --daemon --directory <config path>*」が含まれています。パラメータ名の後ろの空白と引用符に注意してください。

警告：

⚠ インストーラは、構成ディレクトリ名が「Default」で、サービス名が「SerialMux_Default」であることを前提としています。
上記変更を行った後で、インストーラを使用して全ての SerialMux サービスを削除することはできません。
SerialMux をコンピュータから削除するには、SerialMux サービスを手動で削除する必要があります。

⚠ Stargazer では、SerialMux ポートを変更するための構成オプションが提供されていません。Stargazer が通信できるのは、デフォルト TCP ポートの SerialMux のみです。

インストールのテスト

この時点でまだコンピュータを再起動していない場合、2 番目の SerialMux サービスは実行されていません。サービスが正しく作成されたことを確認するには、以下の手順に従います。

1. **Start** メニューで、**Settings > Control Panel** を選択します。
2. **Administrative Tools** を選択します。
3. **Computer Management** を選択します。
4. **Services and Applications > Services** に移動します。右側のリストで **SerialMux_Config1** サービスを右クリックし、**start** を選択します。
5. プログレス・バーが表示され、サービスが起動されます。ここでエラーが表示される場合、コピーした構成ディレクトリへのパスが正しいかどうかを確認します。パスが正しくない（例：余分な空白がある）場合、サービスをいったん削除してから、正しいパラメータで再作成する必要があります。
   1. サービスを削除するには、Windows Command Prompt から以下のコマンドを実行します。

```
> sc delete SerialMux_Config1
```

2 つのサービスが実行中であることを確認したら、SmartMesh SDK に含まれる APIExplorer アプリケーションを使用して、両方のマネージャに接続できることを確認します。

1. APIExplorer のインスタンスを 2 つ起動します。
2. 両方のインスタンスをマネージャ API に対して構成します。
3. 一方をポート 9900 上の SerialMux に接続し、もう一方をポート 9800 上の SerialMux に接続するように設定します。
4. それぞれのマネージャに接続します。
5. 両方のマネージャに対してコマンドを実行することができます。ID に 1 を指定して *getMoteCfgByID* を実行すると、マネージャの MAC アドレスが表示されます。

5.3 SerialMux のプロトコル

シリアル API マルチプレクサは、SmartMesh IP マネージャのシリアル API とよく似た API を提供しますが、メッセージング・プロトコルとコマンド拡張にいくつかの違いがあります。SerialMux は、マネージャのシリアル・ポート経由で接続する代わりに、TCP 経由でクライアントに接続します。プロトコル・レイヤが異なるという点を除いて、クライアントはマネージャとの接続とほぼ同様の方法で SerialMux に接続できます。

5.3.1 基本動作

SerialMux は、起動されるとすぐにマネージャへの接続を試みます。接続に失敗しても、接続が確立されるまで定期的に再試行します。マネージャに接続すると、SerialMux は、(既知の構成可能ポートで)クライアントからの TCP 接続をリスニングします。クライアントは接続するとき、SerialMux に Hello メッセージを送信することになっています。この Hello メッセージには、クライアントの接続が認可されていることを示す秘密のトークンと、クライアントのプロトコル・バージョンが含まれています。秘密のトークンまたはプロトコル・バージョンが一致しない場合、SerialMux は、エラーと一緒に HelloResp を返して、接続を切断します。それ以外の場合、SerialMux はクライアント接続からコマンドを読み取って、マネージャに転送します。

マネージャのシリアル API が一度に処理できるのは、1 つの未処理リクエストのみであるため、クライアント側で、リクエストが SerialMux に送信される速度を制限する必要があります。SerialMux は、クライアントの TCP 接続から、一度に 1 つのリクエストのみを読み取ります。クライアントからのリクエストを一度に 1 つずつ処理する間、SerialMux は IP マネージャのシリアル API を読み取ります。レスポンスは適切なクライアントに転送されます。通知は、サブスクライブしている全てのクライアントにコピーされ、マネージャのシリアル API にアクノリッジが返されます。クライアントはリクエストを書き込み、同じ SerialMux への TCP 接続からレスポンスと通知の両方を読み取ります。マネージャとのシリアル API 接続がリセットされた場合、SerialMux はオープン中の API クライアントを終了し、マネージャに再接続します。SerialMux への再接続はクライアント側で実行する必要があります。

5.3.2 プロトコル

リクエストとレスポンスには、TCP ストリーム内のメッセージを識別するための短いヘッダが含まれています。メッセージ・データの前にコマンド・タイプが指定されます。コマンド・タイプと、関連付けられたリクエストおよびレスポンスのデータは、SmartMesh IP Manager API Guide で使用されているものと同じ構造になりますが、以下に示す Hello メッセージと Info メッセージが追加されます。

リクエスト

パラメータ	タイプ	概要
headerToken	INT8U[4]	4 バイトのメッセージ開始シーケンス
length	INT16U	残りのメッセージの長さ
reserved	INT16U	予約済みパラメータ。0 に設定。
commandType	INT8U	コマンド・タイプ（「Serial Mux のコマンド・タイプ」を参照）
data	INT8U[]	リクエスト・データ（シリアル API を参照）

レスポンス

パラメータ	タイプ	概要
headerToken	INT8U[4]	4 バイトのメッセージ開始シーケンス
length	INT16U	残りのメッセージの長さ
reserved	INT16U	予約済みパラメータ。0 に設定。
commandType	INT8U	コマンド・タイプ（「Serial Mux のコマンド・タイプ」を参照）
data	INT8U[]	レスポンス・データ（シリアル API を参照、レスポンス・コードを含む）

ヘッダのデータ・タイプ、バイト順序、伝送順序とコマンド構造については、SmartMesh IP Manager API Guide を参照してください。

5.3.3 接続

SerialMux は、構成可能な TCP ポートでクライアントからの TCP 接続をリスニングします。

ハンドシェイク

クライアントが SerialMux に接続するとき、SerialMux は、その他のコマンドが送信される前に Hello メッセージを受け取ることを想定しています。

Hello リクエスト

パラメータ	タイプ	概要
version	INT8U	クライアントのプロトコル・バージョン
authentication	INT8U[8]	認証鍵

Hello レスポンス

パラメータ	タイプ	概要
rc	INT8U	レスポンス・コード
version	INT8U	マネージャのプロトコル・バージョン

接続サンプル

SerialMux に Hello を送信するため、クライアントは SerialMux への TCP 接続を確立したのち、以下のバイト・ストリームを送信します。

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

認証鍵が一致すると、SerialMux は以下のレスポンスを返します。

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

getNetworkInfo コマンドを送信するため、クライアントは以下を送信します。

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

SerialMux は以下のレスポンスを返します。

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

5.3.4 Info コマンド

SerialMux の Info コマンドを使用すると、クライアントは SerialMux のバージョンを確認できます。

Info リクエスト

パラメータ	タイプ	概要

Info レスポンス

パラメータ	タイプ	概要
protocolVersion	INT8U	マネージャのシリアル API のプロトコル・バージョン
majorVersion	INT8U	SerialMux のメジャー・バージョン
minorVersion	INT8U	SerialMux のマイナー・バージョン
releaseVersion	INT8U	SerialMux のリリース・バージョン
buildVersion	INT16U	SerialMux のビルド番号

5.3.5 サブスクリプトと通知

SerialMux では、シリアル API によって提供されるサブスクライブ・コマンドと同じものを使用して、クライアントがマネージャからの通知をサブスクライブできます。クライアントは、受信する通知タイプをサブスクライブ・リクエスト内に指定します。同じタイプの通知を複数のクライアントがサブスクライブできます。SerialMux は、クライアントの TCP 接続上にある各クライアントに対して、通知をシリアライズして送信します、通知メッセージの間にコマンド・レスポンスが挿入される場合があります。SerialMux は通知のアクノリッジをマネージャに送り、クライアントごとに通知が異なる複雑な管理に対応します。

5.3.6 接続の解除

SerialMux はコマンド・レスポンスを待機する間、タイマーを維持します。タイマーがタイムアウトになると、SerialMux は以下の処理を実行します。

- マネージャの接続をリセットする
- 全ての接続済みクライアントを終了する
- Hello メッセージを送信してマネージャへの再接続を試みる

SerialMux からクライアント接続への書き込みができない場合、接続は終了されます。

5.3.7 SerialMux の定義

ここでは、API 構成で使用される定数と事前定義済みの値について説明します。

プロトコル定数

ヘッダ・トークンは 4 バイトのシーケンスです。

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

コマンド・タイプ

コマンド名	タイプ	概要
Hello	1	クライアントからの Hello メッセージ
Serial Mux Info	2	SerialMux の情報

その他全てのコマンド・タイプとペイロードは、SmartMesh IP マネージャのシリアル API によって定義されています。

レスポンス・コード

レスポンス・コード	値	概要
OK	0	エラーなしでコマンド実行を完了
Invalid Command	1	無効なコマンド・タイプ
Invalid Argument	2	無効な引数
Invalid Authentication	3	Hello メッセージに含まれる認証トークンが無効
Unsupported Version	4	Hello メッセージに含まれるバージョンがサポートされていない
Command Timeout	5	マネージャからのコマンド・レスポンスがタイムアウト

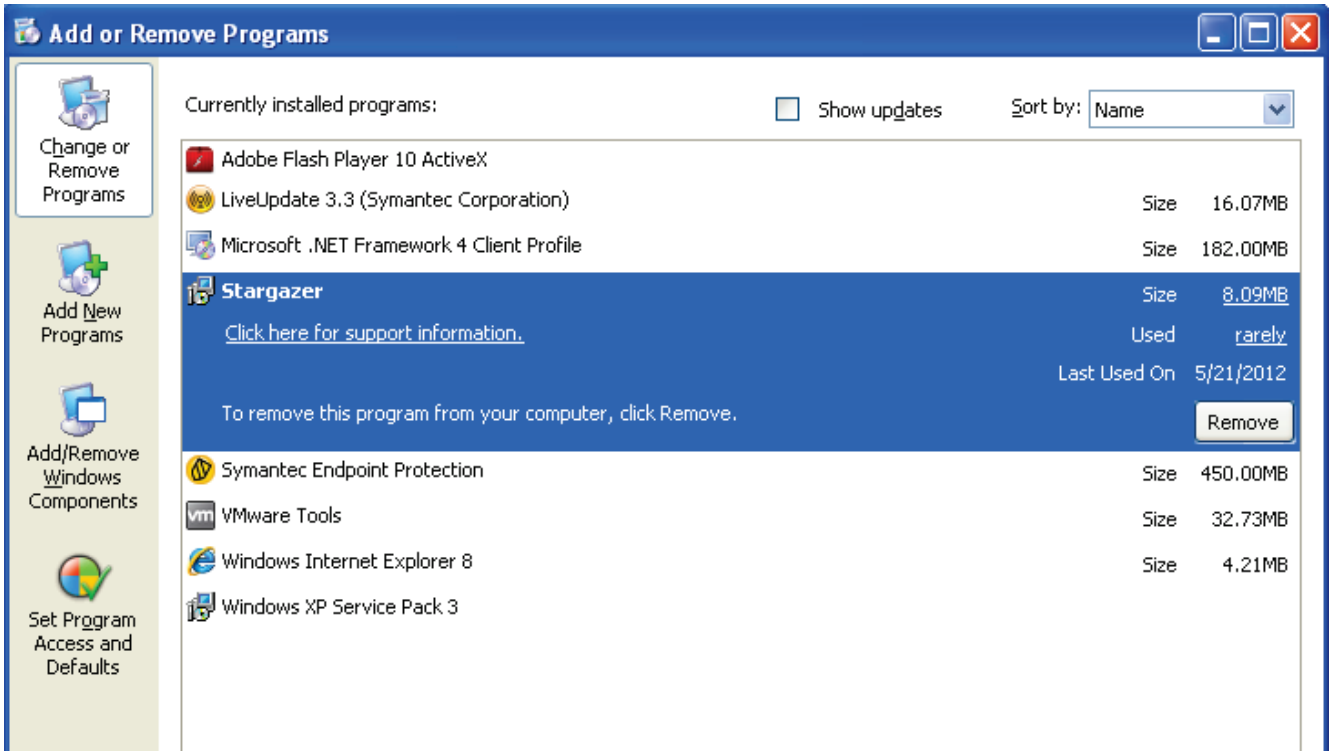
6 Stargazer GUI

6.1 Stargazer のアップグレード

Stargazer やその他のツールのインストーラは、その場でのアップグレードを常にサポートしているとは限りません。場合によっては、新しいバージョンにアップグレードするために、現在インストールしているバージョンを手動で削除する必要があります。これは特に、Stargazer 1.0.5.50 から 1.0.5.51 へのアップグレードに当てはまります。1.0.5.50 のコンポーネントは、別々のインストーラに分割されています。

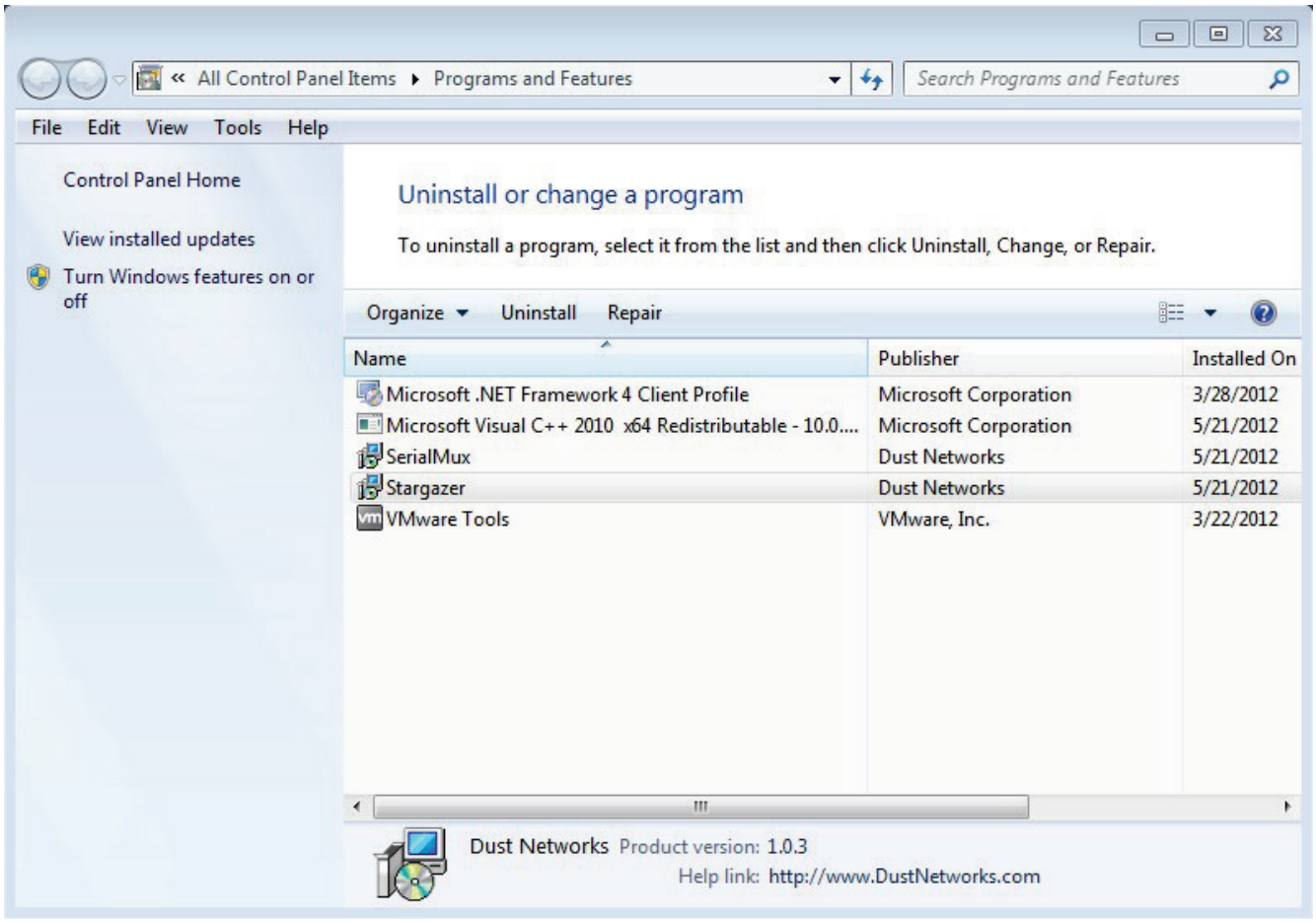
6.1.1 既存のアプリケーションの削除

Windows XP で、Control Panel から **Add / Remove Programs** を選択します。



リストから **Stargazer**（またはアップグレードするツール）を選択し、**Remove** ボタンをクリックします。

Windows 7 で、Control Panel から **Programs and Features** を選択します。



リストから **Stargazer**（またはアップグレードするツール）を選択し、右クリックして **Uninstall** をクリックします。インストール済みの依存コンポーネントがある場合（.NET Client Framework など）、インストール手順で指示されていない限り、これらを削除する必要はありません。

既存のインストールを削除したら、インストーラを使用した通常のインストール・プロセスに進みます。

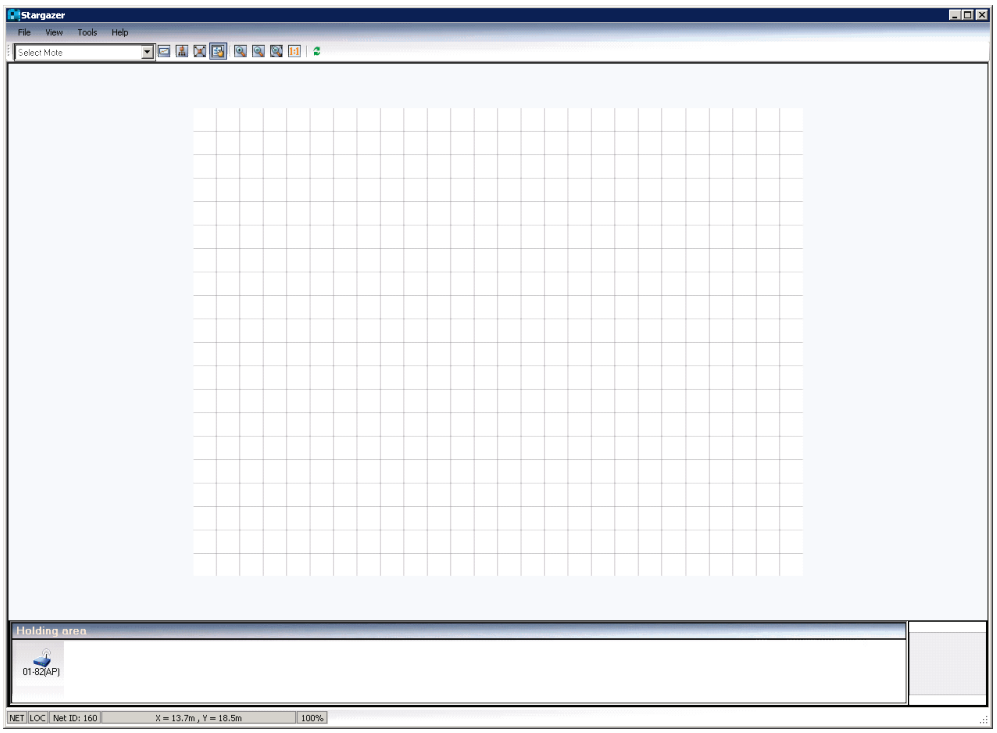
6.2 Stargazer の使用

6.2.1 概要

Stargazer を使用すると、SmartMesh IP マネージャに接続した Windows ベースのコンピュータからネットワークの構成と管理を実行できます。Stargazer はネットワーク・トポロジをグラフィカル表示します。簡単なポイント・アンド・クリック操作で、現在のデータを確認し、モートのステータスとパスを表示し、ネットワーク統計とアラートを監視することができます。また、任意で、サイト・マップの JPG または PNG イメージをインポートして、その上に重ねてネットワークを表示することができます。

Stargazer の起動

デスクトップ上にある Stargazer アイコン  をダブルクリックします。Stargazer を起動すると、アプリケーションは即座にマネージャに接続して、Stargazer ウィンドウ（最初は空白のグリッド）にネットワークをグラフィカル表示します。マネージャ  （「AP」と表示）は、ウィンドウ下部の **Holding Area** に表示されます。



モートの配置

次に、評価モジュール(モート)を配置します。Stargazer と連携させるには、SmartMesh IP モートを**マスター・モード**にする必要があります。APIExplorer や TempMonitor、またはその他の SmartMesh SDK サンプル・アプリケーションで使用するために、モートを**スレーブ・モード**に変更している場合は、**マスター・モード**に戻します。

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

⚠ スタータ・キット(DC9021A)に含まれるモートは、出荷時に**マスター・モード**に設定されています。モートの各モードと切り替え方法については、本書の「トラブルシューティング」セクションと SmartMesh IP ユーザー・ガイドを参照してください。

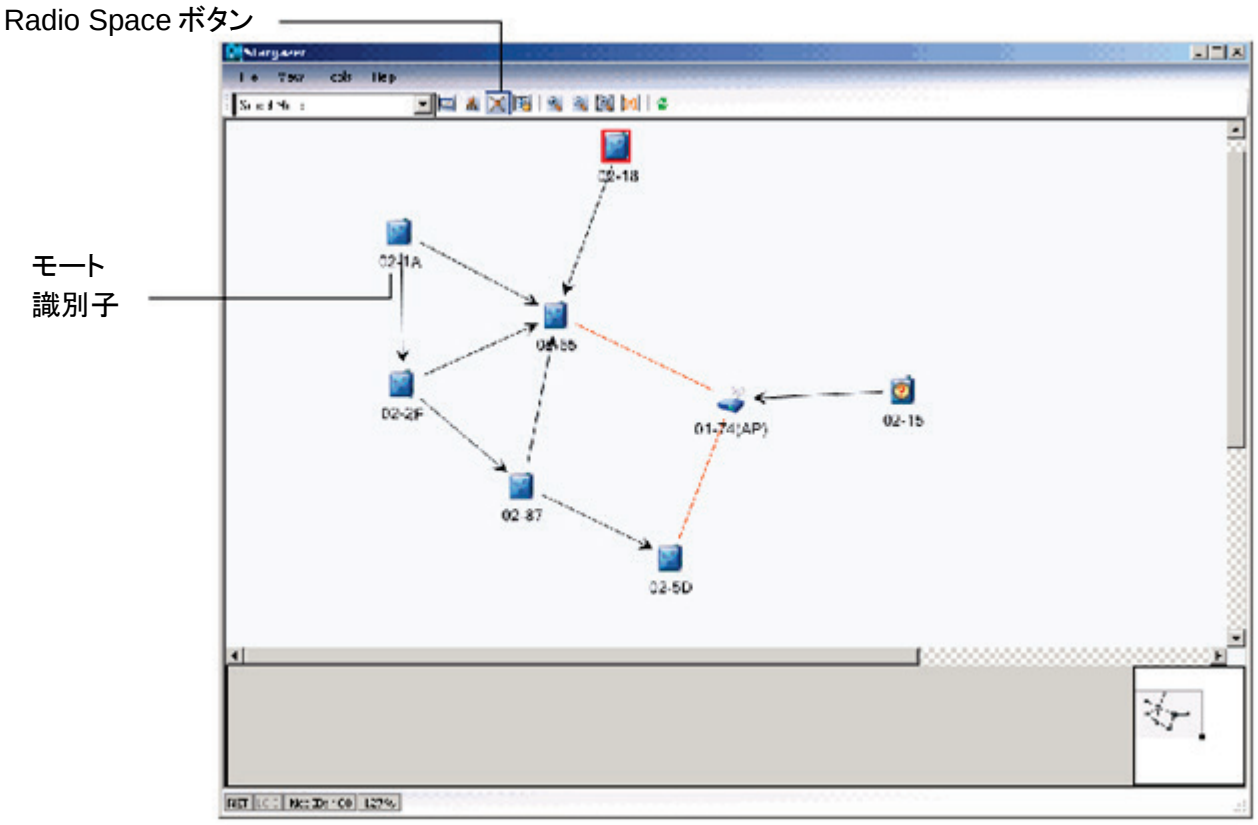
モートの電源をオンにすると、すぐに近くのモートを検索し始めて、ネットワークへのジョインを開始します。モートは数分以内に、信頼性の高いマルチチャンネル・メッシュ・ネットワークを形成します。

モートのインストールに関するガイドラインを以下に示します。

- マネージャに最も近いモートを最初にインストールします。
- モート間の最適な距離は、使用する RF 環境によって異なります。推奨される距離は、屋内の場合、10 メートル以内から 30 メートルまでです。屋外の場合は、50 メートル以内です。モジュール・エンクロージャは全天候型ではないことにご注意ください。
- モートは、地上 2 メートル以上に設置します。
- マネージャの範囲内に、少なくとも 3 つのモートが含まれるようにします。
- 各モートの範囲内に、少なくとも 3 つのモートが含まれるようにします。マネージャが範囲内にある場合、これも 1 つに数えることができます。

モートの配置手順

1. CR2032 バッテリをまだ装着していない場合は、プラス面を上にして取り付けます。**Power** スイッチを **On** の位置にスライドさせて、モートを起動します。
2. LED_EN ジャンパが装着されている場合、モートが作動しており、ネットワークを検索していることを示す **Status 0** の LED が点滅します。モートの起動に失敗した場合は、MOTE RESET ボタンを押してリセットします。それでも LED が点灯しない場合、バッテリを交換する必要があります。
3. 上記のガイドラインに従ってモートを配置します。
4. 残りのモートに対して、ステップ 1 から 3 を繰り返します。モートがネットワークに接続されるたびに、SmartMesh ウィンドウにそのアイコンが表示されます。
5. ウィンドウのツールバーにある **Radio Space** ボタン  をクリックすると、ネットワーク・トポロジが表示され、中央にマネージャが表示されます。デフォルトでは、モートは MAC アドレス(出荷時に各モートに割り当てられた一意のアドレス)の末尾 4 桁で識別されます。Preferences(File メニュー)を変更すると、モートの識別方法を変更できます。
6. 全てのモートを配置したら、次の「ネットワーク接続の確認」セクションに進みます。



LED_EN ジャンパを装着すると使用可能になるモートの**ステータス LED** は、ネットワーク接続に関する以下の情報を提供します。

Status 0	Status 1	モートの状態
点滅	オフ	モートがネットワークを検索中
オン	オフ	モートが検出したネットワークへのジョインを試行中
オン	オン	モートがネットワークにジョイン済みで稼働中

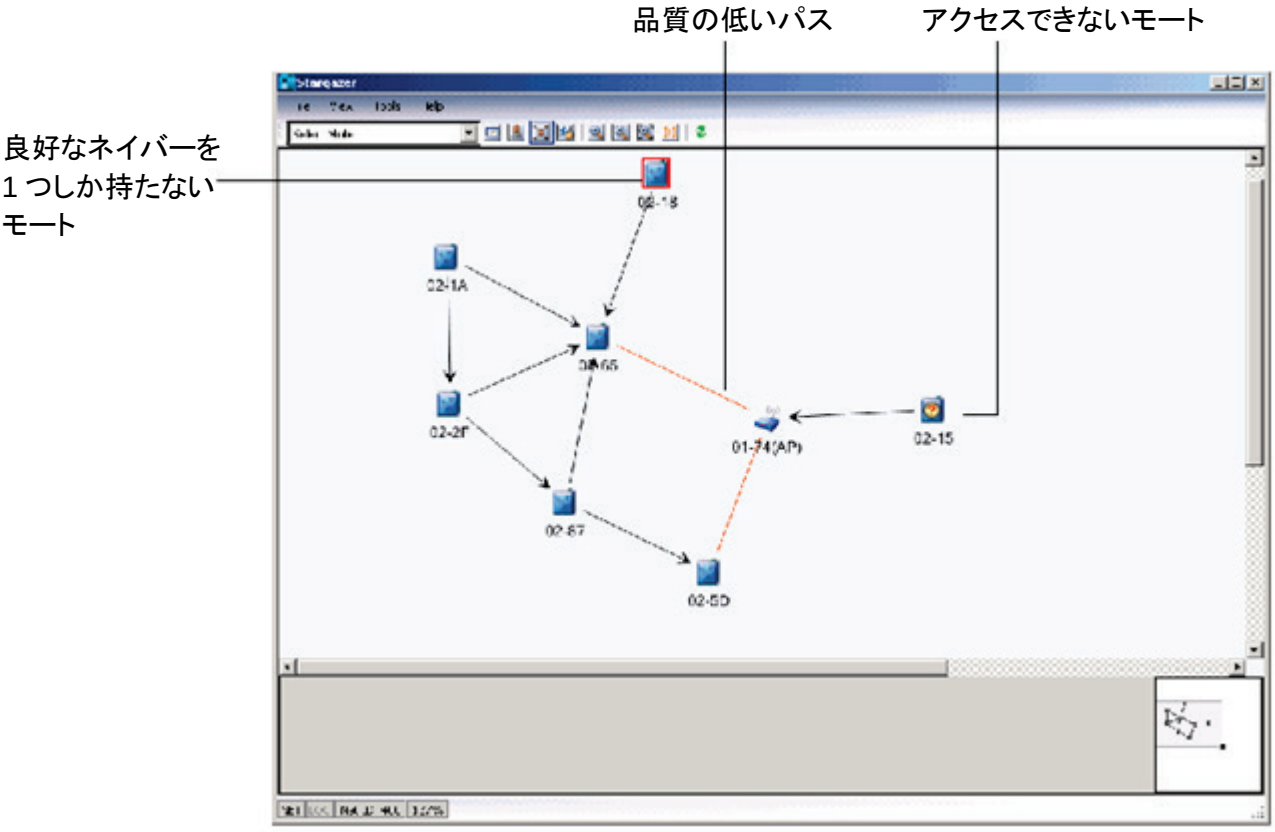
⚠ LED が有効になるのは、モートがマスター・モードに設定されているときだけです。

ネットワーク接続の確認

この時点で、全てのモートの配置が完了しており、モートはネットワークにジョインしています。それぞれのモートが少なくとも 1 つの親を持ち、データはマネージャの方向に流れています。SmartMesh IP マネージャは、絶えず先回りしてネットワーク・リンクを調整することで、自動的にネットワークを最適化して、高い信頼性を維持しながら、総合的なレイテンシと電力消費を改善します。おそらくは、何時間か経過するうちに、より最適な状態に向かって進化するネットワークの変化に気付くでしょう。次のステップはネットワーク接続の確認です。

ネットワーク接続の確認手順

1. ツールバーの *Refresh* ボタン  をクリックして、手動で表示を更新します。
2. 接続の問題を識別します。
    1. ネットワークにジョインしていないアクセス不可能なモートを探します。アクセス不可能なモートには、  アイコンが表示されます。
    2. 良好なネイバーを 2 つ持たないモートが複数あるかどうかを探します。良好なネイバーが指定された数（デフォルトは 2）に満たないモートは、赤色でハイライト表示されます。良好なネイバーとは、パス品質が 50%（デフォルト）を上回るネイバーです。パス品質は、使用されたパスに対する測定値と、使用されていないパスに対する RSSI ベースの概算値に基づいて計算されます。冗長ルーティングとネットワークの自己回復を可能にするために、各モートが 2 つ以上の親を持つ必要があります。親を 1 つしか持たないデバイスが 1 つだけ存在する必要があります。これは、ループを回避するためです。
    3. 無線信号強度（RSSI）が弱いパスを探します。信号強度が弱いパスはオレンジで表示されます。パスの RSSI 値を確認するには、パスにカーソルを合わせるか、またはパスをダブルクリックして詳細パス情報を表示します。

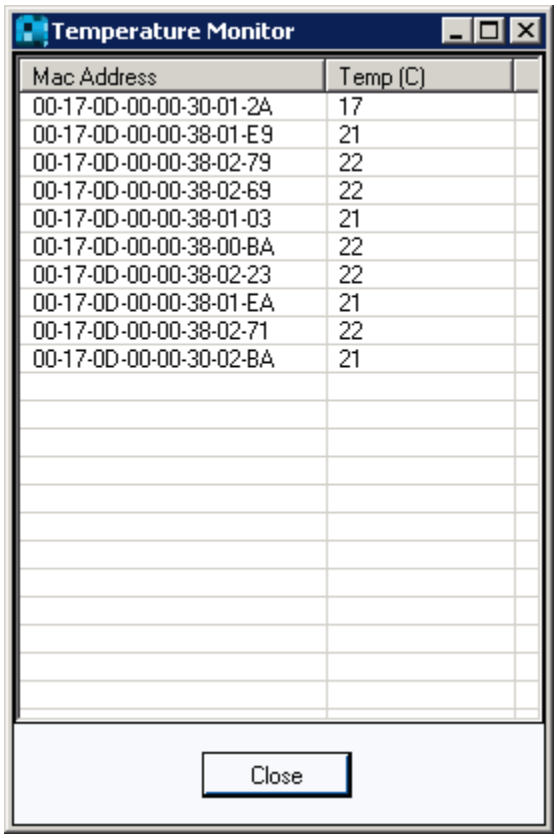


ネットワークが実行中になって 1 時間以上経過した後も接続の問題が続く場合は、以下のいずれかを実行して、手動介入します。

- 対象のモートを近接モートのより近くに移動する（下記手順を参照）。
- 対象のモートをマネージャのより近くに移動し、モートをリセットする（モートをリセットするには、MOTE RESET ボタンを押す）。
- 対象のモートを他のモートの見通し線内に移動し、モートをリセットする。
- 対象のモートを床上 2 フィート以上に移動し、モートをリセットする。
- 問題のモートと最も近い接続済みモートとの間に、1 つまたは複数のモートを追加する。
- 対象のモートとそのネイバーまたはマネージャとの間にある物体（特に重金属）を移動または除去するか、数を最小限に抑えて、モートをリセットする。

ネットワークが完全な冗長メッシュを形成したことを確認できたら、**Tools** メニューから **Temperature Monitor** を選択して、各モートに内蔵されたボード温度センサから取得したリアルタイム・データを表示します。各モートは、デフォルトで 30 秒ごとに読み取った温度を送信します。

⚠ 送信レートはリセットされるまで維持されるため、以前に 30 秒より早く（または遅く）送信するように設定している場合、ネットワークにジョインするたびに、このレートでの送信が継続されます。



以上で、SmartMesh IP ネットワークのセットアップが無事に完了し、マネージャへのライブ・データ送信が確認できました。

⚠ **Temperature Monitor** ウィンドウに表示されるのは、ネットワークにジョインしているモートの情報のみです。マネージャのリセットまたは電源投入以降に、ネットワークにジョインしていないモートがある場合、**Temperature Monitor** には表示されません。全てのモートがネットワークにジョインした後で、**Temperature Monitor** をいったん閉じてからもう一度開くと、全てのモートの情報が表示されます。

6.2.2 ネットワークの管理

ネットワークが実行中になったので、Stargazer アプリケーションを使用してリアルタイム・データを表示し、ネットワークの接続とパフォーマンスを評価することができます。Stargazer を使用すると、ネットワークをグラフィカル表示して、ネットワーク運用と個々のモートを完全にコントロールすることができます。

主なメニュー

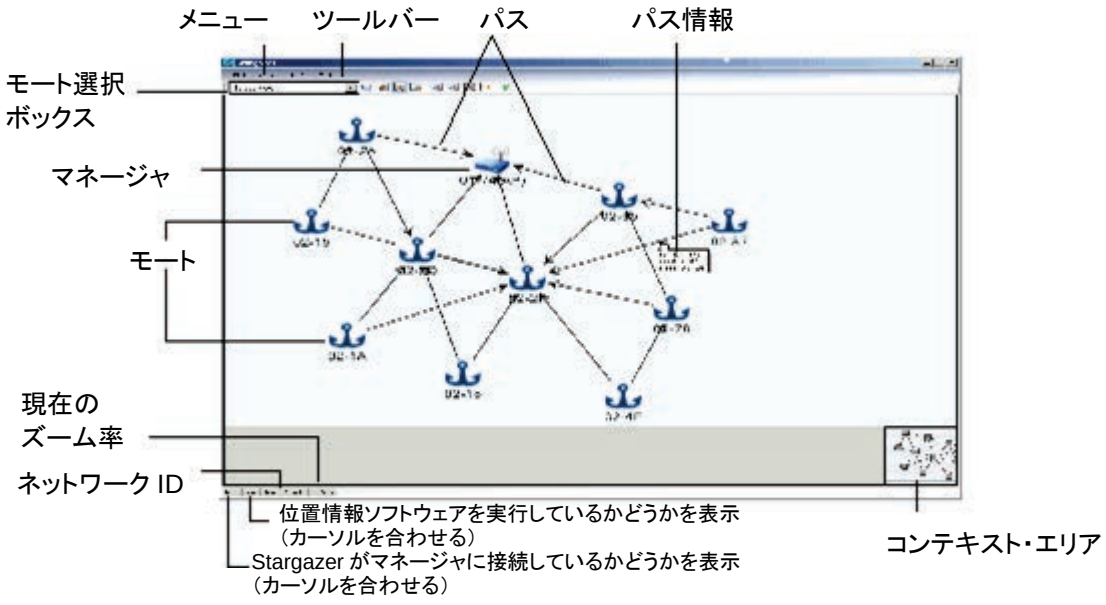
Stargazer の画面に含まれる主なメニューは以下のとおりです。

- *File* - ユーザー・プリファレンスの設定、情報の印刷、モート・ステートのスナップショットと統計情報の保存。
- *View* - 各種レイアウトでのネットワークの表示、ズーム・イン、ズーム・アウト。
- *Tools* - ネットワークとモートの設定、モートおよびパス情報の表示、ネットワーク・トラフィックの監視、センサの設定、その他のネットワーク管理操作（モートのリセット、ネットワーク ID の変更など）の実行。
- *Help* - Stargazer のバージョン情報とカスタマ・サービスの問い合わせ先の表示

上記以外に、モートやパスなどを選択して右クリックすると、ポップアップ・ショートカット・メニューが表示されます。メニュー・コマンドの一覧は、下記のコマンド・リファレンスに関するセクションを参照してください。

ネットワークの表示

Stargazer ウィンドウには、3 種類のビューでネットワーク・トポロジを表示できます。マップまたはグリッド上にネットワークを表示する Manual レイアウト、「ツリー」表示に似た Hierarchical レイアウト、マネージャを中心としたネットワーク・トポロジを表示する Radio Space レイアウトです。



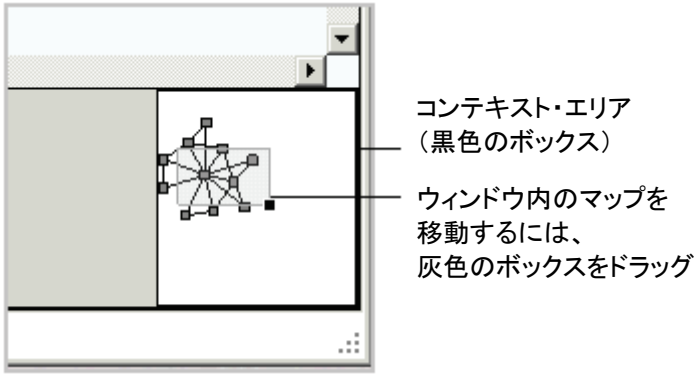
Stargazer ウィンドウ - Radio Space ビュー

以下のアイコンは全てのトポロジ・ビューに表示されます。

	**SmartMesh IP マネージャ** - マネージャはネットワークを制御し、ワイヤレス・メッシュ・ネットワークから受信したデータを送信します。
	**稼働中モート** - これは、ネットワークに接続されたモートです。Manual ビューの Holding Area に含まれる稼働中モートは、全てのビューで  として表示されます。稼働中モートが Manual ビューのメイン・ウィンドウにドラッグされると、全てのビューでアイコンが  に変わります。モートの識別子は、Preferences で設定できます。
	**ハイライト表示されたモート** - モートの接続状況（良好なネイバーの数）が Preferences に設定されたしきい値を下回る場合、モートは赤でハイライト表示されます。良好な接続には、デフォルトで 2 つのネイバーが必要です。
	**消失モート** - これは、以前のある時点でネットワーク内にあったが、現在はネットワークに接続されていないモートです。
	**パス** - この矢印はモート間の通信パスを表しており、常にモートの親の方向を指しています。Preferences に設定されたしきい値よりも品質が低い場合、パスは赤でハイライト表示されます。品質は、使用されたパスの安定性と、検出されたが使用されていないパスの RSSI（信号強度）によって決まります。

Stargazer ウィンドウ内でネットワークを移動するには、以下の操作方法があります。

- **クリック・アンド・ドラッグ** - ウィンドウ内の 1 か所をクリックしてドラッグすると、ネットワーク・イメージを移動できます。
- **コンテキスト・エリアの使用** - コンテキスト・エリア（「コンテキスト・エリア」セクションを参照）内の灰色の枠をドラッグすると、ネットワーク・イメージを移動できます。コンテキスト・エリアはウィンドウの右下にあり、ここにネットワーク全体のイメージが表示されています。灰色の枠は、現在表示されているネットワークを表しています。
- **スクロール・ホイールの使用** - ホイールを使用すると、ネットワーク・イメージのサイズを拡大または縮小できます。
- **アイコンのドラッグ** - Manual ビューでアイコンをドラッグすると、マップまたはグリッド上にモートあるいはマネージャを配置できます。



コンテキスト・エリア

以下のツールバー・ボタンを使用すると、ズーム・イン、ズーム・アウト、ウィンドウの更新を実行できます。

	**Zoom In** - マップ・サイズを拡大して中心にズーム・インします。マウスのスクロール・ホイールを使用してズームすることもできます。
	**Zoom Out** - ネットワークのサイズを縮小します。マウスのスクロール・ホイールを使用してズームすることもできます。
	**Fit in Window** — ネットワークのサイズをウィンドウに合わせて調整します。
	**Actual Size** — ネットワークを 100%に拡大して表示します。各アイコンは.ネイティブ解像度で表示されます。
	**Refresh** — ウィンドウ内のモート情報を更新します。

ネットワーク・ビューの変更

以下のツールバー・ボタンを使用すると、ネットワーク・ビューを切り替えることができます。

	**Table** — モートとそのステータスに関する情報を含む表を表示します。詳細については、本章の「モート情報の表示」セクションを参照してください。
	**Hierarchical** — マネージャを最上位に表示したネットワーク・トポロジを示します。このレイアウトでは、モートとマネージャの間にあるホップ数が簡単に分かります。
	**Radio Space** — マネージャを中央に表示したネットワーク・トポロジを示します。
	**Manual** — ネットワーク・トポロジをサイト・マップまたはグリッド上に表示します。このレイアウトでは、モートを地理的な位置に合わせて配置できます。

モートおよびパス情報の表示

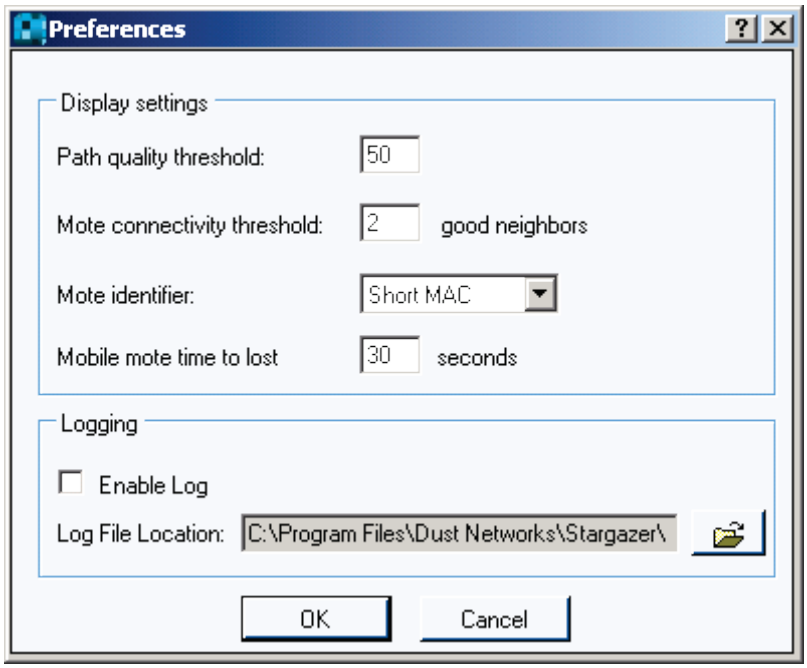
わずか数回のクリックで、選択したモートまたはパスに関する詳細なネットワーク情報を表示することができます。例えば、以下の操作を実行できます。

- カーソルをモートに合わせると、接続情報と位置座標が表示されます。
- モートをダブルクリックすると、構成の詳細情報が表示されます。
- カーソルをパスに合わせると、パスの品質と RSSI が表示されます。
- パスをダブルクリックすると、追加の詳細情報が表示されます。

Tools メニューで Motes または Paths をクリックすると、全てのモートまたはパスに関する詳細情報が表示されます。

Stargazer プリファレンスの変更

File メニューの Preferences コマンドを使用すると、表示プリファレンスを変更してロギングをオンにすることができます。ネットワークに問題がある場合、ロギングをオンにするよう求められる場合があります。このログはトラブルシューティングのために使用できます。



Preferences ウィンドウ

以下のプリファレンスを設定することができます。

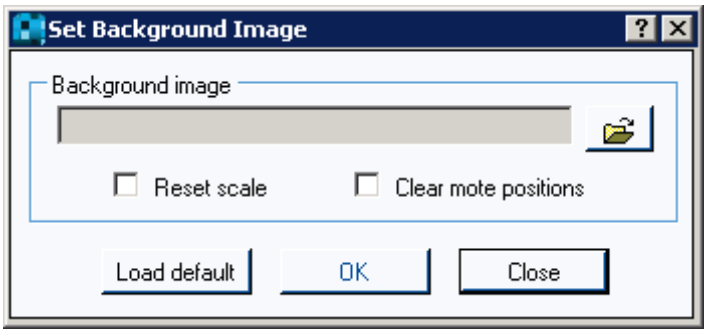
フィールド	概要
Path quality threshold	Stargazer でパスをハイライト表示するためのしきい値(0～100%)を設定します。パス品質のしきい値に満たないパスは、オレンジ色でハイライト表示されます。品質が 50%以上のパスは、良好なパスと見なされます。
Mote connectivity threshold	Stargazer でモートをハイライト表示するためのしきい値を設定します。モート接続のしきい値に満たないモートは、赤色でハイライト表示されます。 モートに必要とされる*良好なネイバー*の最小数を入力します。十分なパスの信号強度と品質を持つモートは、良好なネイバーと見なされます。
Mote identifier	Stargazer でのモート識別方法を指定します。以下のオプションから選択できます。 ● *Long MAC* — モートの 8 バイト EUI アドレスを表示します。MAC アドレスは、工場出荷時にモートに割り当てられた一意のアドレスです。 例:*00-17-0D-00-00-38-01-74* ● *Short MAC* — モートの MAC アドレスの末尾 2 バイトを表示します。 例:*01-74* ● *Mote ID* — モートのネットワーク割り当て ID を表示します。Mote ID は、モートがネットワークにジョインした順序をベースとしています。マネージャが常に 1 番であり、Mote ID は 2 から始まります。
Enable log	ネットワークの問題をトラブルシューティングするために、ログを有効化します。
Log file location	ログ・ファイルの場所を指定します。

背景イメージのインポート

Stargazer では、インポートしたサイト・マップ（PNG または JPG）を背景に表示して、モートの地理的な位置をマップ上に示すことができます。デフォルトで背景に表示されるのはグリッドです。

背景マップのインポート手順

1. ツールバーで **Manual** ボタン  をクリックして、Manual ビューに切り替えます。
2. **Tools** メニューで **Set Background Image** をクリックします。以下のダイアログボックスが表示されます。

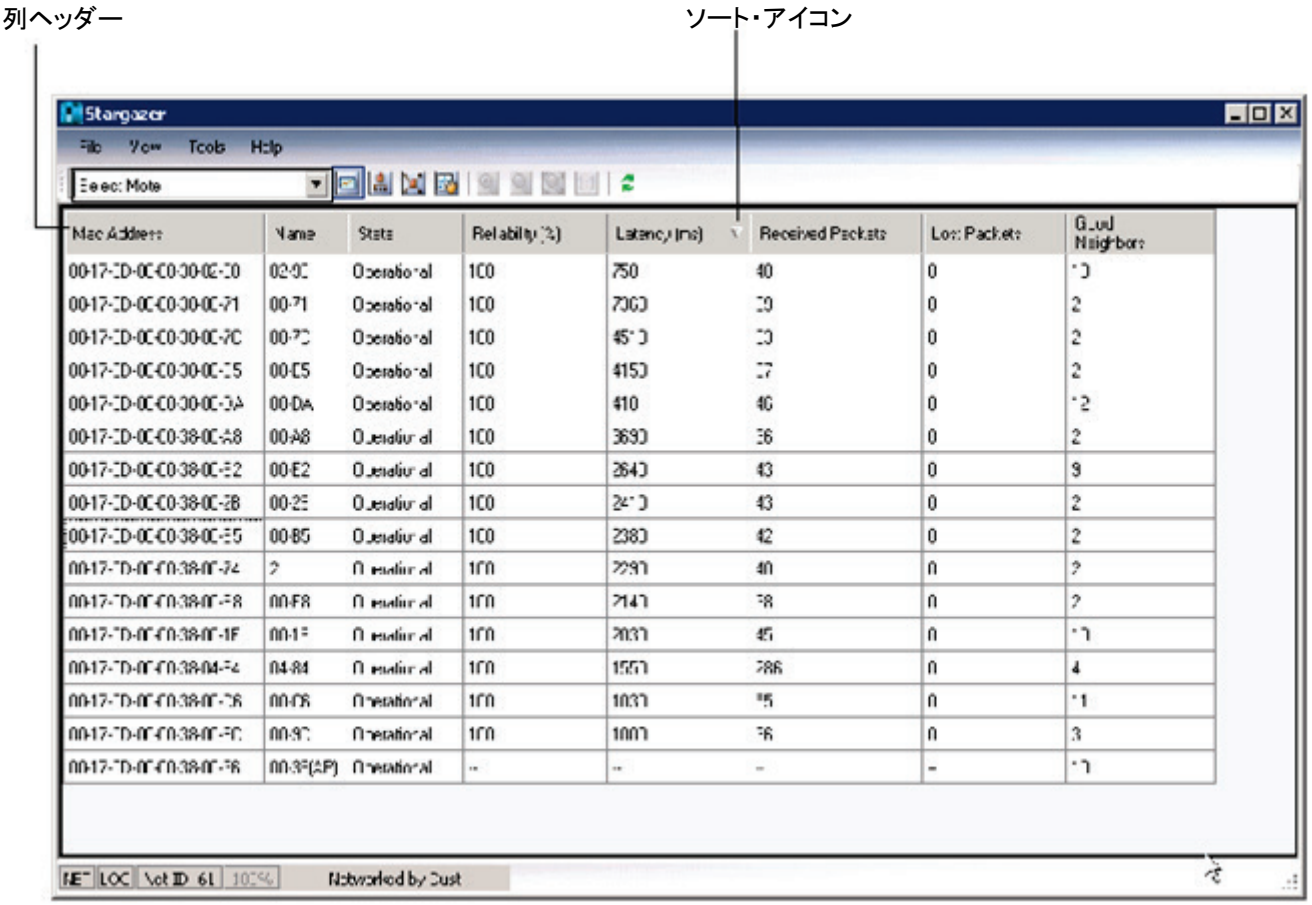


3. **Browse** ボタンをクリックし、サイト・マップ・イメージがある場所に移動して、**Open** をクリックします。
4. **OK** をクリックします。Stargazer で、Manual ビューにサイト・マップが表示されます。

モート情報の表示

ツールバーで Table アイコン  をクリックするか、Tools メニューから Motes をクリックすると、リスト形式でモートが表示され、列でソートすることができます。モート・リストには、現在のモートのステータスと統計情報が表示されます。モートの統計情報には、モートによって収集され、15 分おきにマネージャに送信するヘルス・レポートに含まれるパケット統計と、マネージャによって算出されたデータ・レイテンシ統計が含まれます。モートの働きが十分でない場合は、レイテンシが期待値よりも高くなったり、パケット損失が表示されたりするなどの兆候が統計情報に表れます。

ウィンドウ内の情報を更新するには、Refresh ボタンを使用します。列のカテゴリでリストをソートするには、列のヘッダをクリックします。シングルクリックすると昇順にソートされ、ダブルクリックすると降順にソートされます。列ヘッダのソート・アイコンに、ソート順が表示されます。



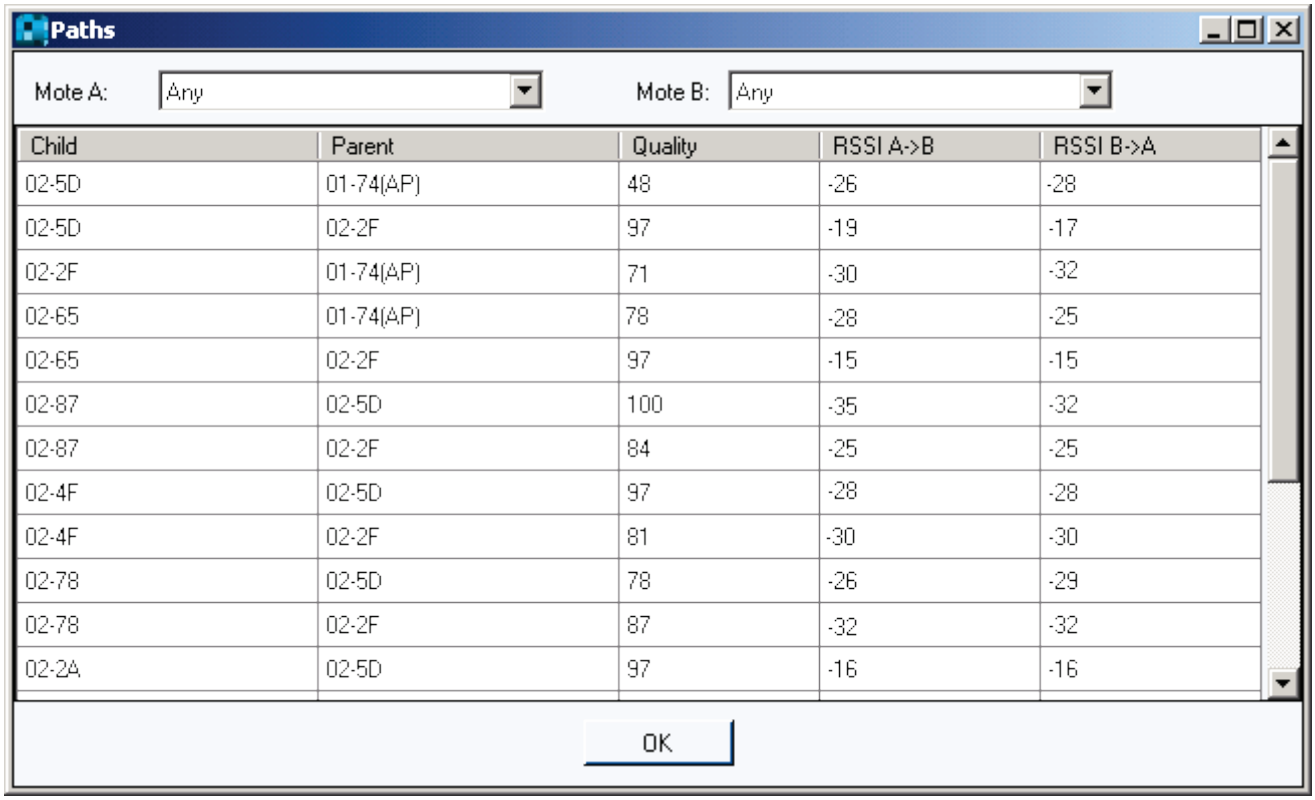
Mote Table ビュー

モート表に表示される情報は、以下のとおりです。

フィールド	概要
MAC Address	工場出荷時にモートに割り当てられた一意のアドレス。
Name	Preferences で設定されたモートの識別子。
State	ネットワークに含まれるモートの現在のステータス。 ● *Negotiating*: モートがネットワークへのジョインを試行している ● *Operational*: モートがネットワークに接続されており、データを送信している ● *Lost*: モートは現在ネットワークにジョインしていない
Reliability	損失することなくマネージャに送信されたパケットの割合
Latency	データ・パケットが発信元から宛先に届くまでにかかる平均時間（ミリ秒）
Received Packets	マネージャがモートから受信したパケット数
Lost Packets	受信が見込まれたが、実際には受信しなかったパケット数
Good Neighbors	プリファレンスで設定されたしきい値を上回るパス品質を持つ近接モートの数

パス情報の表示

Tools メニューで Paths をクリックすると、ネットワーク内の全てのパスの品質および信号強度が表示されます。パスをダブルクリックすると、同じ情報が選択したパスのみに対して表示されます。



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

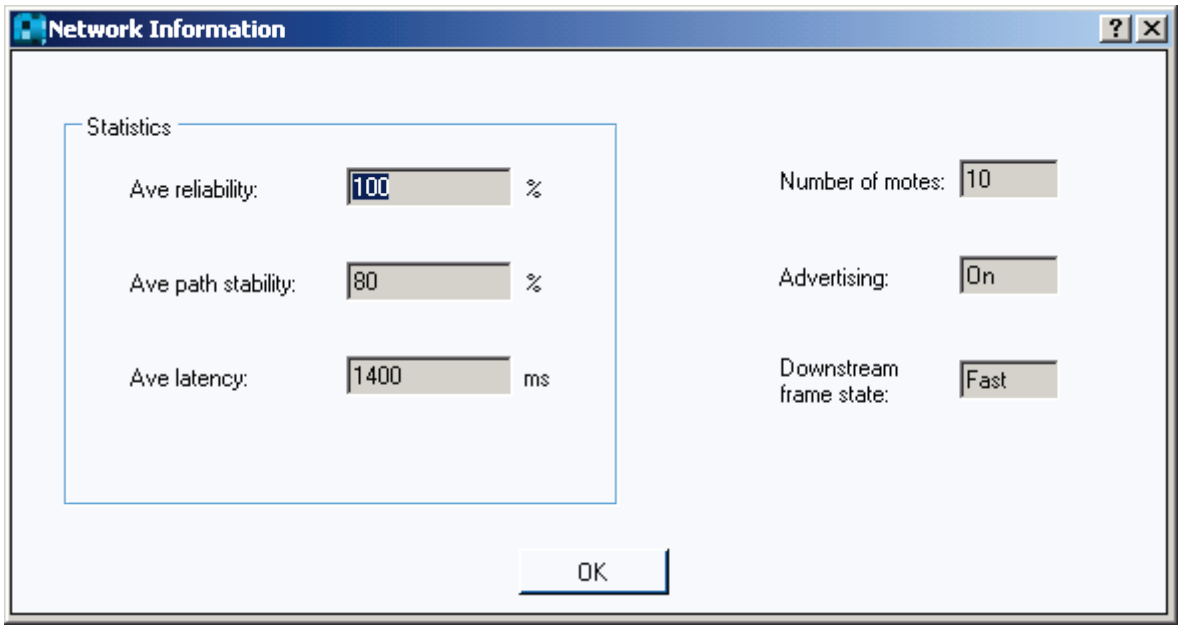
Paths ウィンドウ

Paths ウィンドウに表示される情報は、以下のとおりです。

フィールド	概要
Child	このパスへのアップストリーム・データを送信しているモート。
Parent	このパスへのアップストリーム・データを受信しているモート。
Quality	0～100 の値（%）で示されるパス品質。50%以上のパス品質は良好と見なされます。30%を下回るパスがある場合、可能であれば最適化で削除されます。
RSSI A->B	子モートから親モートへの送信における無線信号強度（dBm）。
RSSI B->A	親モートから子モートへの送信における無線信号強度（dBm）。

ネットワーク統計の表示

SmartMesh IP マネージャは、15 分ごとに各モートから受信するヘルス・レポートに基づいて、データの信頼性、パスの安定性、データ・レイテンシに関する統計情報を生成します。Network Information ウィンドウは、ネットワーク存続期間を通じた統計サマリーと現在のネットワーク情報を提供します。



Network Information ウィンドウ

Network Information ウィンドウに表示される情報は、以下のとおりです。

フィールド	概要
Average reliability	予想されたデータ・パケットのうち実際に受信された割合。信頼性が 100%である場合、予想された全てのデータ・パケットが実際に受信されたことを意味します。ここに表示される値は、ネットワークの平均値です。
Average stability	パスの安定性は、モート間の送信が成功した比率を示します。これは、送信モートがアクノリッジを受け取ったデータ・パケットの割合です。送信モートがアクノリッジを受信しなかった場合、別のパスで再送信される可能性があります。メッシュ・ルーティング特有の利点により、パスの安定性が非常に低い場合も、データの信頼性を 100%にすることができます。
Average latency	データ・パケットが発信元のモートからマネージャに届くまでにかかる平均時間（ミリ秒）。データ・レイテンシはネットワーク内の場所によって変動しますが、ここ表示される値は平均データ・レイテンシです。
Number of motes	ネットワークに含まれるモートの数。
Advertising	Advertising が On になっている場合、ネットワークの存在を知らせるアドバタイズメント・パケットが、モートおよびマネージャによって頻繁に送信されます。ネットワークにジョインしようとするモートは、ジョインのためにアドバタイズメント・パケットをリスニングします。Advertising が Off になっている場合、アドバタイズメント・パケットは送信されません。
Downstream frame state	ネットワークの現在の動作速度を Fast または Normal で示します。形成過程にあるネットワークは、モートのジョインを促すために高速で稼働します。いったんネットワークが形成されると、電力を節約するために低い（通常の）速度で稼働します。

ネットワーク統計の表示手順

- **Tools** メニューで **Network** をクリックしてから **Information** をクリックします。

ネットワーク統計の消去

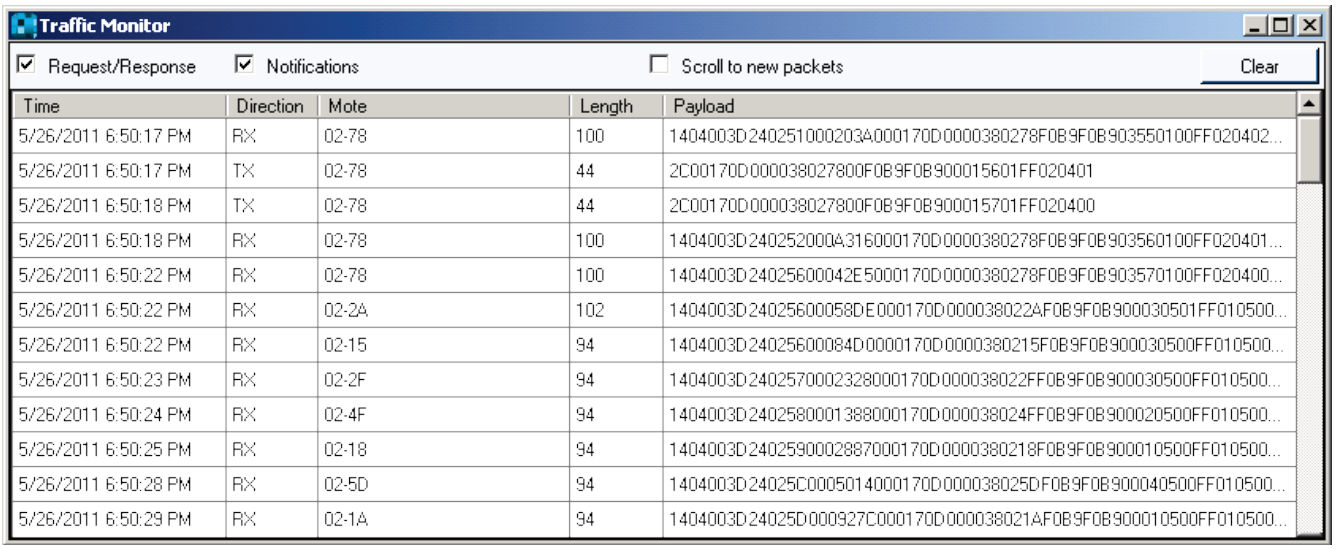
新しいネットワーク統計を収集する場合、データの信頼性、パスの安定性、データ・レイテンシに対する現在の平均統計情報をマネージャから削除することができます。マネージャは統計の収集を継続して、新しい統計セットを参照用に提供します。

全ての統計情報の消去手順

- **Tools** メニューで **Network** をクリックしてから **Clear Statistics** をクリックします。

ネットワーク・トラフィックの監視

Tools メニューで Traffic Monitor をクリックすると、Stargazer とマネージャの間のネットワーク通信を監視することができます。Traffic Monitor ウィンドウには、Stargazer とマネージャ間で交換されたリクエスト/レスポンス・パケットと、モートから受信したイベントおよびデータ通知が表示されます。



Traffic Monitor ウィンドウ

Traffic Monitor ウィンドウに表示される情報は、以下のとおりです。

フィールド	概要
Request/Response チェック・ボックス	このチェック・ボックスが選択されている場合、Stargazer とマネージャ間で交換されたリクエスト/レスポンス・パケットが表示されます。
Notifications チェック・ボックス	このチェック・ボックスが選択されている場合、Stargazer がモートから受信したレスポンス・パケットとデータ・パケットが表示されます。
Scroll to new packets	このチェック・ボックスが選択されている場合、常に最新のパケットまでスクロールします。
Clear	Traffic Monitor の表示を消去します。

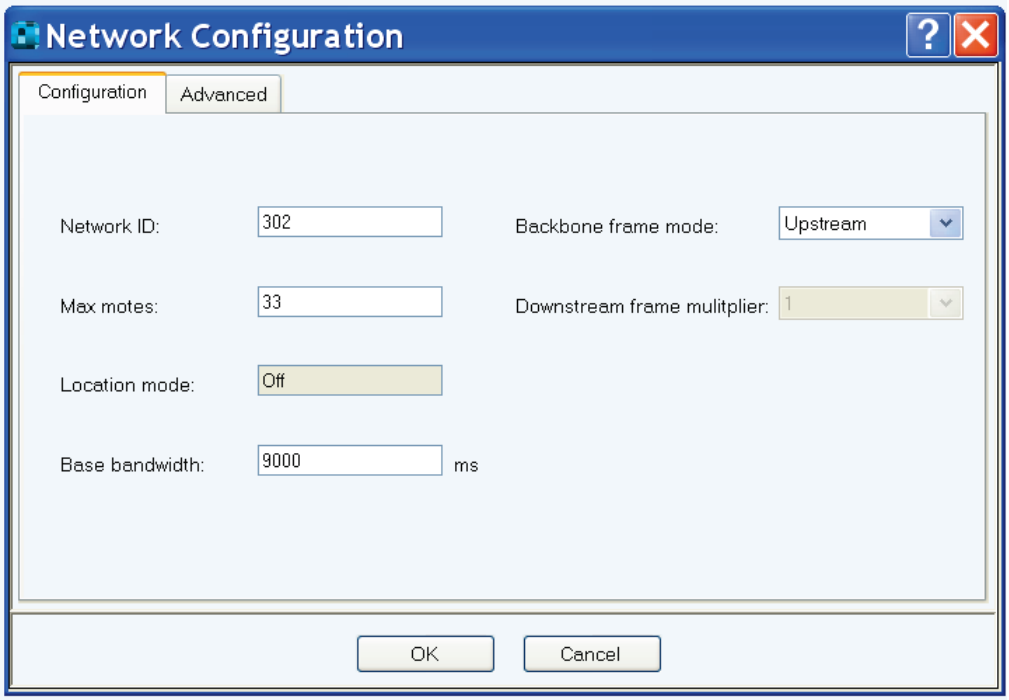
フィールド	概要
Time	Stargazer がパケットを送信した時刻、または Stargazer がパケットを受信した時刻。
Direction	パケットが送信された方向。TX は、パケットが Stargazer によって送信されたことを示し、RX は、Stargazer がモートからパケットを受信したことを示します。
Mote	Stargazer にパケットを送信したモート、または Stargazer からパケットを受信したモート。識別子は MAC アドレスまたは Mote ID です（Preferences での設定によって異なる）。
Length	ヘッダ情報を含むパケットの長さ。
Payload	パケットの内容（16 進数）。

低レイテンシ・モード

SmartMesh IP は低レイテンシ・モードを提供しており、通信回線で電源供給されているモートのバックボーンが存在する場合、アップストリーム通信（モートからマネージャ）または双方向通信の速度を向上します。レイテンシの短縮は、Stargazer などの GUI ベースのインタフェースよりも、制御システム（照明制御やビルオートメーションなど）の方が明白になりますが、Stargazer を使用して低レイテンシ・モードを設定することもできます。

ネットワークの低レイテンシ設定手順

1. **Tools** メニューで **Network** をクリックしてから **Configuration** をクリックします。以下のウィンドウが表示されます。



2. **Backbone frame mode** フィールドで、以下のいずれかのオプションを選択します。
    1. **Upstream** — モートからマネージャに送信されるイベント通知の速度を向上します。
    2. **Bidirectional** — モートとマネージャの間のリクエスト/レスポンス通信の速度を向上します。
3. **OK** をクリックします。
4. **Tools** メニューで **System** をクリックしてから **Reset System** をクリックします。マネージャのソフトウェア・プロセスとワイヤレス通信が再起動されます。ネットワークが再形成されると、新しい構成が有効になります。

モート・アプリケーションとの通信

Communicate with Application ウィンドウを使用すると、Stargazer を使用して、モートの組み込みアプリケーションと通信することができます。評価キットに付属しているデフォルトのモートには、Cortex M3 マイクロプロセッサ上で稼働し、Eterna の組み込み機能を使用するアプリケーションが 4 つ組み込まれています。各アプリケーションは以下のとおりです。

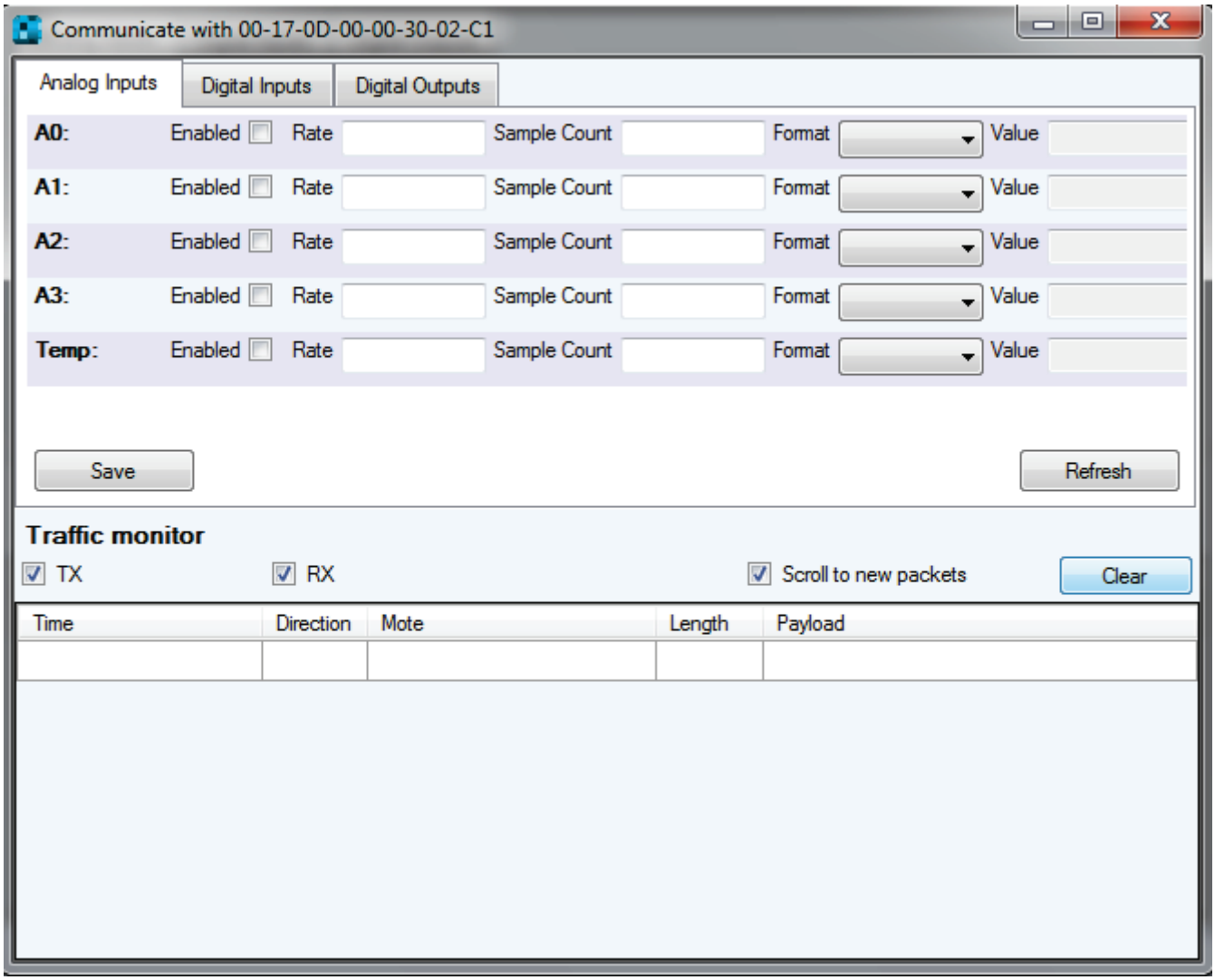
- 温度（Eterna の組み込み温度センサを使用）
- アナログ入力（4 チャネル）
- デジタル入力（4 チャネル）
- デジタル出力（3 チャネル）

データの取得と送信は、アプリケーションとチャネルごとに個別に設定できます。

アナログ入力の範囲は 0～1.8V です。この範囲外の信号は、デバイスの損傷を引き起こす場合があります。

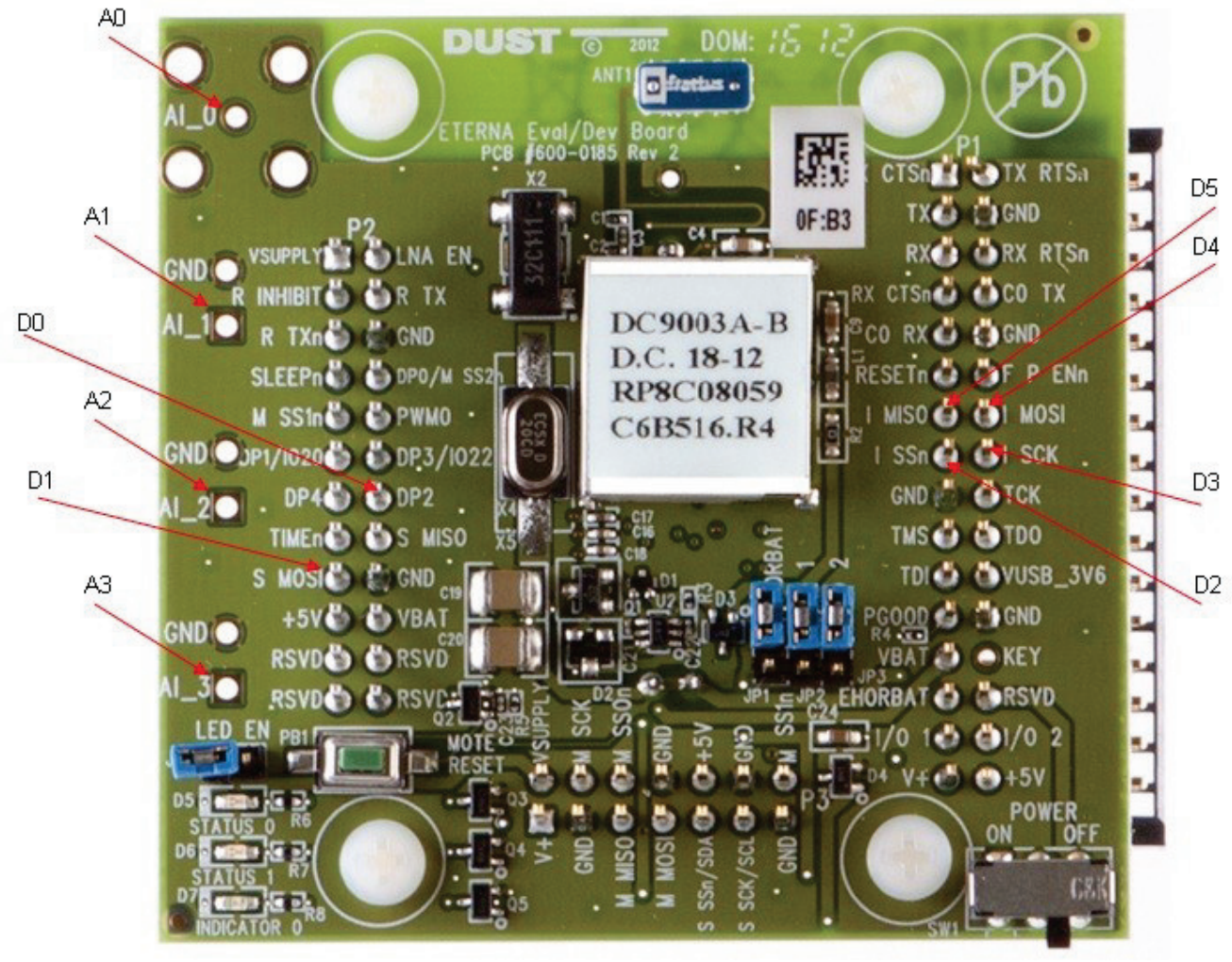
消費電力を最小限に抑えるため、**マスター・モード**での動作時には、使用しないデジタル入力（D0～D3）を解除することをお勧めします。

Communicate with Application ウィンドウを表示するには、モートを右クリックして **Communicate with Application** を選択します。センサ入力および出力（チャネル）のリストと構成パラメータを表示したウィンドウが開きます。明示的に更新しない限り、モートの現在のステータスは表示されません。ウィンドウを更新して現在のステータスを表示するには、**Refresh** ボタンをクリックします。モート上で稼働する全てのアプリケーションのステータスを確認するには、この操作を全てのタブで繰り返す必要があります。



⚠ 上記のアプリケーションが稼働するには、モートをマスター・モードに設定している必要があります。

Eternal 評価ボードの I/O の場所については、以下の図を参照してください。



アナログ入力アプリケーション

Analog Inputs タブでは、温度監視とアナログ I/O 監視という 2 つのアプリケーションにアクセスできます。このウィンドウに表示されるフィールドは、以下のとおりです。

フィールド	概要
Refresh ボタン	モート上のセンサ・アプリケーションに、現在のセンサ構成情報とデータ値を問い合わせます。データの取得中は、ウィンドウ内のタブ領域が灰色で表示されます。
Temp:	モートに組み込まれた温度センサ。温度センサはデフォルトで有効になっており、30 秒ごとに温度を報告するように設定されています。測定値は、**Tools** メニューからアクセスできる **Temperature Monitor** ウィンドウに表示されます。
A0～A3	アナログ入力チャネル
Enabled	このチェック・ボックスが選択されている場合、このチャネルに関するデータがセンサから送信されます。
Rate	データの取得ペースをミリ秒で指定します。指定できる範囲は、1,000～300,000 ミリ秒です。 ● 例えば、Rate に 30,000 ミリ秒と指定した場合、センサは 30 秒ごとにデータを取得します。

フィールド	概要
Sample Count	このフィールドに 1 を設定すると、パケットあたり 1 つのデータ・サンプルが送信されます。このフィールドに 3 を設定すると、パケットあたり 3 つのサンプルが提供されます。指定された数のサンプルがパケットに入らない場合（パケット内に格納できるのは 27 個のサンプルまで）、格納できない余分のサンプルは破棄されます。モートは、指定されたサンプル数に達するとパケットを送信します。
Format	パケット内のデータ形式を定義します。 ● *All* を指定すると、全てのデータ・サンプルがパケットに含まれます（パケットに格納できないサンプルは削除されます）。 ● *Stats* を指定すると、全てのサンプルの最大値、最小値、平均値が提供されます。
Value	温度監視アプリケーションの場合、Value には測定された温度（摂氏、0.01 度単位）が表示されます。 アナログ入力アプリケーションの場合、Value には、Refresh ボタンをクリックしたときに入力ピンから読み取られた現在の電圧が表示されます。 ● *電圧の範囲は 0～1.800mV であり、0～1800 までの数値で表されます。例えば、636 という値は 636mV に相当します。* 注:ピンに何も接続されていない場合、値は変動し、有効な範囲内の任意の値を取ります。
Save ボタン	構成の変更を保存します。モートの構成中は、ウィンドウ内のタブ領域が灰色で表示されます。
Traffic Monitor	Stargazer によって送受信されたトラフィックを表示します。詳細については、「ネットワーク・トラフィックの監視」セクションを参照してください。
Payload	パケット・ペイロード。このペイロードにはパケット・ヘッダは含まれていませんが、メインの Traffic Monitor に表示されるペイロードにはヘッダが含まれています。詳細については、本書の「オンチップ・アプリケーション・プロトコル」セクションに示すサンプル・ペイロードを参照してください。

アナログ・センサ（A2 または Temperature）の構成手順

Stargazer で、モートを右クリックして **Communicate with Application** を選択します。

Communicate with Application ダイアログボックスが表示されます。デフォルトで、**Analog Inputs** タブが選択されています。

1. **Refresh** をクリックして、モート上のセンサ・アプリケーションから現在の構成情報を取得します。

Stargazer はモート上のセンサ・アプリケーションに一連のコマンドを送って、全てのアナログ・センサに設定されている現在の構成を要求します。

1. アナログ入力の **Enabled** チェック・ボックスを選択します。
2. 上記の表を参考にして、**Rate**、**Sample Count**、**Format** を指定します。
3. **Save** をクリックします。

Stargazer が、モート上のセンサ・アプリケーションに構成コマンドを送信します。コマンドを送信してからレスポンスを受け取るまでの間、**Analog Inputs** タブは灰色で表示されます。設定が完了すると、ウィンドウ下部の **Traffic Monitor** 領域に、センサから受け取ったデータ通知が表示され始めます。

- 注:3 秒ごとに 1 パケットを超える速度でモートが送信するように設定している場合、Save ボタンを押した後、このデータ速度に対応するために必要な追加リンクをマネージャが割り当てるまでに、数分間の遅延が生じる場合があります。コマンドは順序どおりに処理する必要があるため、この間、モートは追加の構成コマンドを受信できない可能性があります。構成コマンドが有効になっていない(**Traffic Monitor** に変更が反映されていない)場合、もう一度 **Save** ボタンを押します。

任意のタイミングで **Refresh** ボタンを押して、**Value** フィールドに最新のセンサ測定値を表示できます。**Value** フィールドは自動的に更新されないため、手動で **Refresh** ボタンを押す必要があります。

デジタル入力アプリケーション

Digital Inputs タブでは、最大 4 つのデジタル入力を監視するアプリケーションにアクセスできます。このウィンドウに表示されるフィールドは、以下のとおりです。

フィールド	概要
Refresh ボタン	モート上のセンサ・アプリケーションに、現在のセンサ構成情報とデータ値を問い合わせます。データの取得中は、ウィンドウ内のタブ領域が灰色で表示されます。
D0～D3	デジタル入力チャネル
Enabled	このチェック・ボックスが選択されている場合、このチャネルに関するデータがセンサから送信されます。
Rate	データの取得ペースをミリ秒で指定します。指定できる範囲は、1,000～300,000 ミリ秒です。 ● 例えば、Rate に 30,000 ミリ秒と指定した場合、センサは 30 秒ごとにデータを取得します。
Sample Count	このフィールドに 1 を設定すると、パケットあたり 1 つのデータ・サンプルが送信されます。このフィールドに 3 を設定すると、パケットあたり 3 つのサンプルが提供されます。指定された数のサンプルがパケットに入らない場合(パケット内に格納できるのは 54 個のサンプルまで)、格納できない余分のサンプルは破棄されます。モートは、指定されたサンプル数に達するとパケットを送信します。
Format	パケット内のデータ形式を定義します。 ● *All* を指定すると、全てのデータ・サンプルがパケットに含まれます(パケットに格納できないサンプルは削除されます)。 ● *On change* を指定すると、入力が 0 から 1、または 1 から 0 に変更されたときにサンプルを提供します。 ● *On rising* を指定すると、立ち上がりエッジ(0 から 1 への変化)が確認されるたびにサンプルを提供します。 ● *On falling* を指定すると、立ち下がりエッジ(1 から 0 への変化)が確認されるたびにサンプルを提供します。
Value	1 または 0 のどちらかの値になります。 最小 I/O レベルについては、LTC5800-IPM データシートを参照してください。 注:ピンに何も接続されていない場合、値は変動し、有効な範囲内の任意の値を取ります。

フィールド	概要
Save ボタン	構成の変更を保存します。モートの構成中は、ウィンドウ内のタブ領域が灰色で表示されます。
Traffic Monitor	Stargazer によって送受信されたトラフィックを表示します。詳細については、「ネットワーク・トラフィックの監視」セクションを参照してください。
Payload	パケット・ペイロード。このペイロードにはパケット・ヘッダは含まれていませんが、メインの Traffic Monitor に表示されるペイロードにはヘッダが含まれています。詳細については、本書の「オンチップ・アプリケーション・プロトコル」セクションに示すサンプル・ペイロードを参照してください。

デジタル入力(D0～D3)の構成手順

Stargazer で、モートを右クリックして **Communicate with Application** を選択します。

Communicate with Application ダイアログボックスが表示されます。**Digital Inputs** タブを選択します。

1. **Refresh** をクリックして、モート上のセンサ・アプリケーションから現在の構成情報を取得します。

Stargazer はモート上のセンサ・アプリケーションに一連のコマンドを送って、全てのアナログ・センサに設定されている現在の構成を要求します。

1. デジタル入力の **Enabled** チェック・ボックスを選択します。
2. 上記の表を参考にして、**Rate**、**Sample Count**、**Format** を指定します。
3. **Save** をクリックします。

Stargazer が、モート上のセンサ・アプリケーションに構成コマンドを送信します。コマンドを送信してからレスポンスを受け取るまでの間、**Digital Inputs** タブは灰色で表示されます。設定が完了すると、ウィンドウ下部の **Traffic Monitor** 領域に、センサから受け取ったデータ通知が表示され始めます。

- 注:3 秒ごとに 1 パケットを超える速度でモートが送信するように設定している場合、Save ボタンを押した後、このデータ速度に対応するために必要な追加リンクをマネージャが割り当てるまでに、数分間の遅延が生じる場合があります。コマンドは順序どおりに処理する必要があるため、この間、モートは追加の構成コマンドを受信できない可能性があります。構成コマンドが有効になっていない(**Traffic Monitor** に変更が反映されていない)場合、もう一度 **Save** ボタンを押します。

任意のタイミングで **Refresh** ボタンを押して、**Value** フィールドに最新のセンサ測定値を表示できます。**Value** フィールドは自動的に更新されないため、手動で **Refresh** ボタンを押す必要があります。

デジタル出力アプリケーション

Digital Outputs タブからは、DC9018 ボード上の 2 つの出力ピン(D4 および D5)と INDICATOR_0(青色)LED にアクセスできます。このウィンドウに表示されるフィールドは、以下のとおりです。

フィールド	概要
Refresh ボタン	モート上のセンサ・アプリケーションに、現在のセンサ構成情報とデータ値を問い合わせます。データの取得中は、ウィンドウ内のタブ領域が灰色で表示されます。

フィールド	概要
D4 および D5 インジケータ	デジタル出力チャネル。インジケータの接続先は INDICATOR_0 LED です。
Value	以下の 3 つのステータスうちのいずれかになります。 ● *No change* - 値は変わっていません。既存の出力がすでに有効化されている場合、その他の出力を構成するために使用できます。 ● 0 - 低出力（gnd）に設定します。 ● 1 - 高出力（Vsupply)に設定します。
Save ボタン	構成の変更を保存します。モートの構成中は、ウィンドウ内のタブ領域が灰色で表示されます。
Traffic Monitor	Stargazer によって送受信されたトラフィックを表示します。詳細については、「ネットワーク・トラフィックの監視」セクションを参照してください。
Payload	パケット・ペイロード。このペイロードにはパケット・ヘッダは含まれていませんが、メインの Traffic Monitor に表示されるペイロードにはヘッダが含まれています。詳細については、本書の「オンチップ・アプリケーション・プロトコル」セクションに示すサンプル・ペイロードを参照してください。

デジタル出力（D4、D5、インジケータ）の構成手順

Stargazer で、モートを右クリックして **Communicate with Application** を選択します。

Communicate with Application ダイアログボックスが表示されます。**Digital Outputs** タブを選択します。

1. **Refresh** をクリックして、モート上のセンサ・アプリケーションから現在の構成情報を取得します。

Stargazer はモート上のセンサ・アプリケーションにコマンドを送って、全てのアナログ・センサに設定されている現在の構成を要求します。

1. 上記の表を参考にして、デジタル出力の **Value** を指定します。
2. **Save** をクリックします。

Stargazer が、モート上のセンサ・アプリケーションに構成コマンドを送信します。レスポンスを受け取るまで、**Digital Outputs** タブは灰色で表示されます。いつでも **Refresh** ボタンを押して、現在の設定を確認できます。

7 SmartMesh IP SDK

> **参考**
>
> このセクションの目的は、SmartMesh SDK の**概要を分かりやすく**説明することです。
>
> 全ての機能、パラメータ、変数に関する**技術的な詳細**については、インストール・ディレクトリの*/doc/*フォルダにある Doxygen ベースのドキュメントを参照してください。

7.1 SmartMesh SDK について

SmartMesh SDK は、SmartMesh IP または SmartMesh WirelessHART ネットワークを、お使いのアプリケーションに簡単に統合するための Python パッケージです。このパッケージには、接続先デバイスのアプリケーション・プログラミング・インタフェース（API）が実装されています。SmartMesh SDK に含まれた一連のサンプル・アプリケーションを利用することで、プログラマは速やかに API を理解し、大規模システムの一部として使用できるようになります。

7.2 SmartMesh SDK の特長

- **全ての API 定義**：SmartMesh IP マネージャ、SmartMesh IP モート、SmartMesh WirelessHART マネージャ、SmartMesh WirelessHART モートの全 API をサポートしています。コマンド定義をコピー/貼り付けする必要はありません。
- **低レベル・コネクタ**：シリアル、XML-RPC、SerialMux を含む任意の送信メディアを介して、任意の Dust Networks デバイスにアプリケーションを接続できます。低レベル・モジュールを開発する必要はありません。
- **dustUI ビジュアル・ライブラリ**：一般的なルック・アンド・フィールを含み、簡単にカスタマイズできる標準 GUI 要素のセットです。
- **完全なソース・コード**：SmartMesh SDK の内部処理を詳しく調べることができます。
- **サンプル・アプリケーション**：GUI ベースとスクリプト・ベースのサンプル・アプリケーションを、ソース・コードとバイナリ形式で提供しています。
- **包括的なドキュメント**：概要レベルのハンズオン導入ガイドと、Doxygen ベースのソース・コード・ドキュメントを提供しています。
- **移植性**：標準の Python インストール以外に必要なものはありません。Python をサポートする全てのプラットフォーム（Microsoft Windows、Linux、MacOS を含む）で実行できます。
- **拡張性**：より大規模なアプリケーションへの統合を想定して設計されています。

7.3 ドキュメントの構成

ここからは、以下の構成でガイドを提供します。記載した順序でお読みいただくことをお勧めします。

- 「フォルダの内容」では、上記でインストールした *SmartMeshSDK-full-X.X.X.X.zip* ファイルの内容を簡単に説明します。
- SmartMesh SDK の内部を詳しく説明する前に、「サンプル・アプリケーション」では、SmartMesh SDK に含まれるサンプル・アプリケーション（バイナリ形式とソース・コード形式）の概要を説明します。
- 「アーキテクチャ」では、SmartMesh SDK ソース・コードの概要と主なパッケージおよびオブジェクトについて説明します。

- 「dustUI ライブラリ」では、グラフィカル・ユーザー・インタフェース・オブジェクトのライブラリについて説明します。
- 「SmartMeshSDK ライブラリ」では、API 定義と API コネクタ・オブジェクトを含む SmartMesh SDK のコア・コンポーネントについて説明します。

7.4 フォルダの内容

SmartMeshSDK-full-X.X.X.X.zip ファイルを解凍すると、以下のディレクトリ構造が作成されます。

- *SmartMeshSDK-X.X.X.X/*内に作成されるサブ・ディレクトリ
  - */src/*には、SmartMesh SDK とサンプル・アプリケーションのソース・コードが含まれます。Python はインタープリタ型の言語であるため、ソース・ファイルから直接アプリケーションを実行できます。
  - */win/*には、py2exe ユーティリティを使用して生成された、コンパイル済みのサンプル・アプリケーションが含まれます。これを使用すると、Python をインストールすることなく、Windows コンピュータでアプリケーションを実行できます。
  - */doc/*には、Doxygen を使用して生成された、HTML ベースの SmartMesh SDK ドキュメントが含まれます。
  - */api/*には、C ヘッダ・ファイルとサンプル・コードが含まれます。

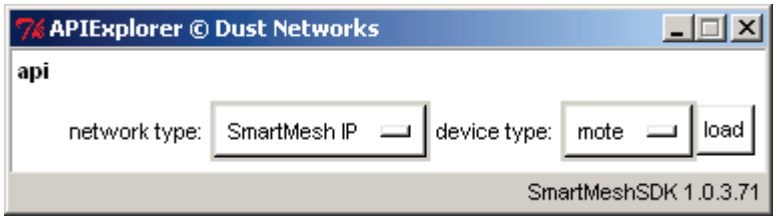
7.5 サンプル・アプリケーション

7.5.1 アプリケーションの実行

サンプル・アプリケーションを実行する方法は 2 種類あります。

コンパイル済みの Windows 実行可能ファイルを使用する方法

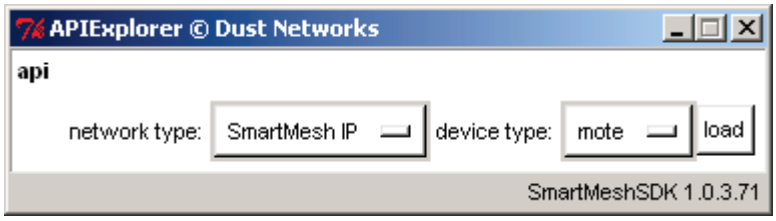
*/win/*ディレクトリには、py2exe ユーティリティを使用して、Windows 実行可能ファイルとしてコンパイルされたサンプル・アプリケーションが含まれています。
例えば、*/win/APIExplorer.exe* をダブルクリックすると、APIExplorer アプリケーションが起動されます。



アプリケーションの動作を**変更しない**場合は、この方法でアプリケーションを実行することをお勧めします。

ソース・ファイルを使用する方法

Python はインタープリタ型のプログラミング言語であるため、アプリケーション・スクリプトをダブルクリックして起動することもできます。
例えば、*src/bin/APIExplorer/APIExplorer.py* をダブルクリックすると、APIExplorer アプリケーションが起動されます。



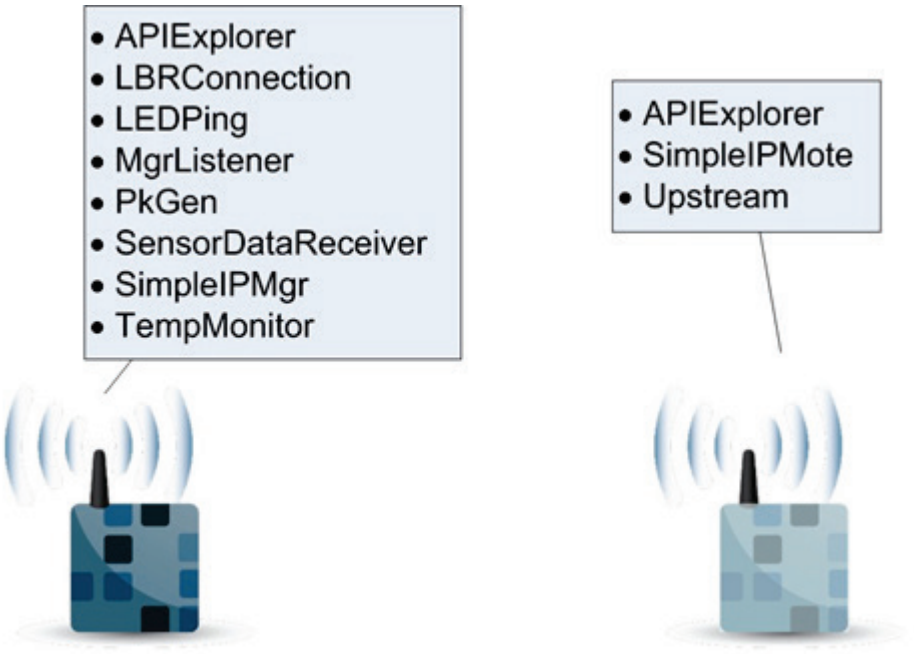
ソース・コードを編集してアプリケーションの動作を変更している場合、この方法でアプリケーションを実行することをお勧めします。

7.5.2 色分け表示

一部のアプリケーションでは、フィールドにデータを入力できます。それぞれの色分けが意味する内容は、以下のとおりです。

色	意味
緑	入力された値は有効で使用可能。
黄色	入力された値が有効でない。入力した文字列の種類とその長さを確認してください（例：整数の入力時に全ての文字が数字になっているかどうか）。
オレンジ	入力された値はアプリケーションに受け入れられたが、警告が発生した（例：接続しようとしたデバイスに電源が入っていない）。
赤	入力された値はアプリケーションに受け入れられたが、エラーが発生した（例：存在しないシリアル・ポートを開こうとした）。

7.5.3 概要



SmartMesh IP
マネージャ

SmartMesh IP
モート

- APIExplorer：APIExplorer は、全ての SmartMesh デバイスのアプリケーション・プログラミング・インタフェース（API）とのやり取りを可能にするグラフィカル・ユーザー・インタフェースです。
- InstallTest：InstallTest は、シンプルなコンソールアプリケーションで（グラフィカルでない）、SmartMesh SDK のサンプルをソースから実行するために必要な Python コンポーネントが正しくインストールされているかどうかをテストします。
- LBRConnection：LBRConnection アプリケーションを使用すると、稼働中の LBR に接続して、LBR に関する情報と通過するパケット数を表示できます。
- LEDPing：LEDPing アプリケーションは、デフォルト・ファームウェアを使用して、**マスター・モード**で稼働中の SmartMesh IP モートの LED を継続して切り替え続けます。
- MgrListener：MgrListener アプリケーションは SmartMesh IP マネージャに接続して、全てのタイプの通知をサブスクライブし、タイプごとの数を表示します。
- MuxConfig：Serial Mux Configurator は、SerialMux の構成を管理するための GUI エディタを提供しています。
- OTAP Communicator：OTAP Communicator は、Over-the-Air Programming（OTAP）を介してモートにファームウェアをロードするためのツールです。
- PkGen：PkGen は SmartMesh IP マネージャまたは SmartMesh WirelessHART マネージャに接続して、**マスター・モード**で稼働しているモート上のパケット生成アプリケーションにコマンドを送信できるようにします。
- SensorDataReceiver：SensorDataReceiver アプリケーションは、SmartMesh IP マネージャに接続し、SmartMesh IP モートに接続した *Upstream* アプリケーションから送信されたデータを表示します。
- Simple：SimpleIPMgr と SimpleIPMote は、プログラムを使用して各デバイスの API とやり取りする方法を示すコンソール・アプリケーション（グラフィカルでない）です。これらはサンプル・コードとしての役割を果たします。
- TempMonitor：TempMonitor は SmartMesh IP マネージャまたは SmartMesh WirelessHART マネージャに接続して、**マスター・モード**で稼働しているモート上の温度取得アプリケーションにコマンドを送信できるようにします。
- Upstream：Upstream アプリケーションは、SmartMesh IP モートのステート・マシンを、ジョイン、サービス要求、ソケット・バインディングへと遷移させます。ユーザーは、ダミーの「センサ」データをアプリケーションに入力して、SmartMesh IP マネージャまたはインターネット・ホストに送信できます。

1.0.4 で導入されたサンプル・アプリケーション

- DC2126A：DC2126A ボードからデータを受け取って解析するサンプル・アプリケーションです。
- Xively：センサ・データを Xively に送信し、変更をサブスクライブします。

7.5.4 APIExplorer

概要

APIExplorer は、全ての SmartMesh デバイスのアプリケーション・プログラミング・インタフェース（API）とのやり取りを可能にするグラフィカル・ユーザー・インタフェースです。

接続できるコンポーネントは以下のとおりです。

- SmartMesh IP マネージャ
- SmartMesh WirelessHART マネージャ
- **スレーブ**・モードで実行中の SmartMesh IP モート
- **スレーブ**・モードで実行中の SmartMesh WirelessHART モート

モートの各モードについては、SmartMesh IP ユーザー・ガイドと SmartMesh WirelessHART User's Guide を参照してください。

実行方法

APIExplorer アプリケーションの実行方法は以下のとおりです。

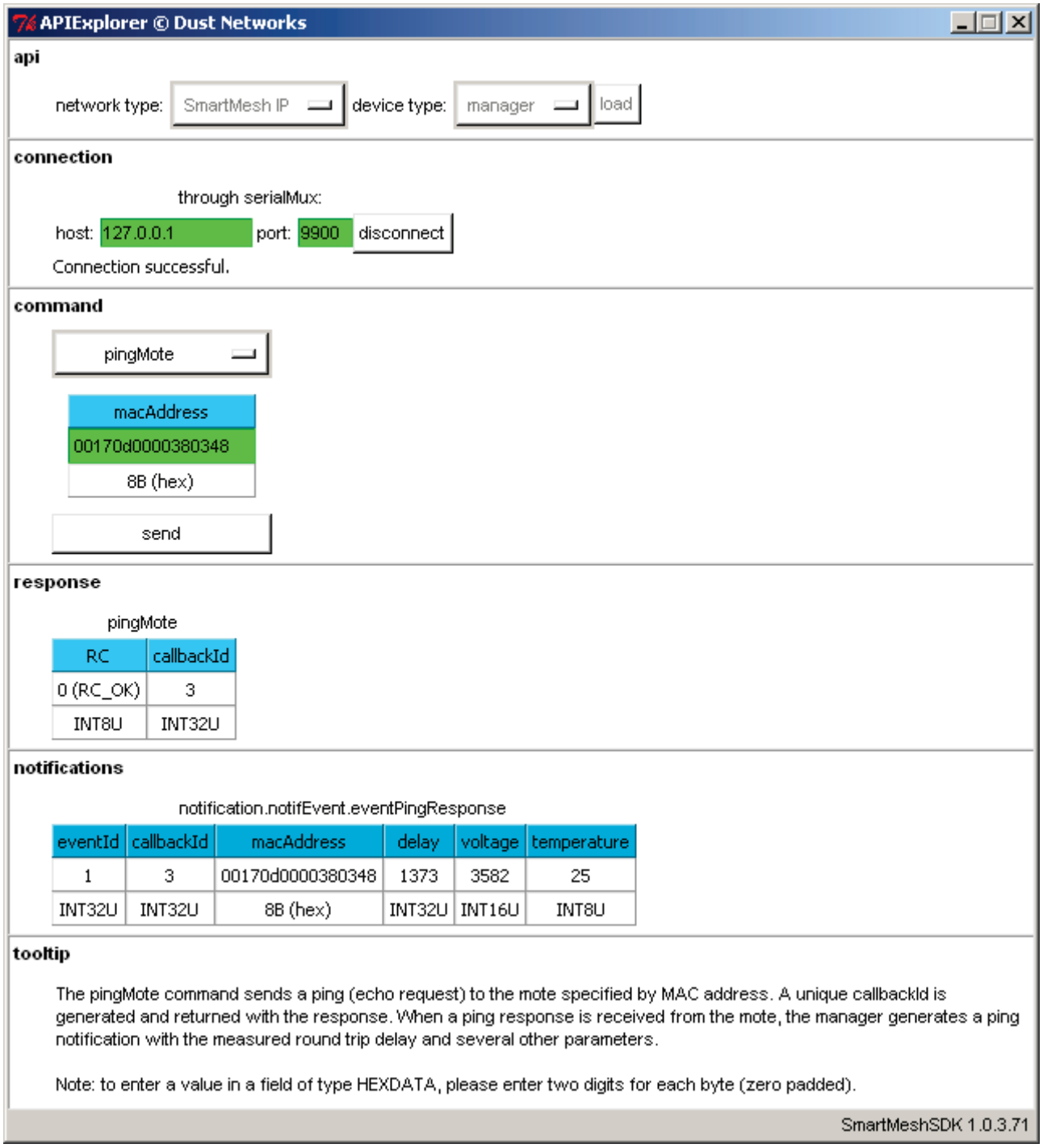
- Windows 実行可能ファイル（*/win/ APIExplorer.exe*）をダブルクリックする
- ソース・ファイル（*/src/ bin/APIExplorer/APIExplorer.py*）をダブルクリックする（Windows 以外の OS では追加の手順が必要になる場合あり）

説明

APIExplorer は GUI ベースのアプリケーションであり、以下のデバイスの API インタフェースと相互作用します。

- SmartMesh IP マネージャ
  - シリアル経由で接続
  - SerialMux 経由で接続
- SmartMesh IP モート
  - シリアル経由で接続 - API を有効にするには、モートを**スレーブ**・モードで実行する必要があります
- SmartMesh WirelessHART マネージャ
  - XML-RPC 経由で接続
- SmartMesh WirelessHART モート
  - シリアル経由で接続 - API を有効にするには、モートを**スレーブ**・モードで実行する必要があります

以下に、SmartMesh IP マネージャに接続した後の APIExplorer ウィンドウの例を示します。接続方法と使用可能なコマンドが異なるため、フレームの内容はデバイスによって異なります。



インタフェースは、以下の 6 つのフレームで構成されています。

- **api** フレームでは、接続するネットワークおよびデバイスの種類を選択します。*load* ボタンを押すと、対応する API 定義が設定され、その他のフレームに表示されます。
- **connection** フレームには、デバイスへの接続方法を指定します。**api** フレームで選択したネットワークおよびデバイスの種類によって、複数の接続オプションが表示される場合があります。
- **command** フレームでは、デバイスに送信するコマンドを選択します。デバイスの API に含まれる全てのコマンドを使用できます。全てのコマンド・パラメータを入力する必要があります。*set* フォームを使用する前に、*get* フォームを使用してコマンド引数をよく理解しておくことをお勧めします。**send** ボタンを押すと、connection フレームで選択したコネクタを使用して、コマンドがデバイスに送信されます。
- 送信したコマンドに対するレスポンスは、**response** フレームに表示されます。
- デバイスによって送信された通知は、**notifications** フレームに表示されます。

> ⚠ SmartMesh IP マネージャなどのデバイスでは、通知を受け取る前に（*subscribe* コマンドを使用して）サブスクライブしておく必要があります。

- コマンドを選択すると、その説明が **tooltip** フレームに表示されます。

APIExplorer を使用したチュートリアル

APIExplorer は、以下のチュートリアルで使用されています。

- SmartMesh IP 向け
  - マネージャの対話操作
  - モートの対話操作
  - HDLC フレームのロギング
  - アップストリーム通信
  - ダウンストリーム通信

- SmartMesh WirelessHART 向け
  - マネージャの対話操作
  - モートの対話操作

7.5.5 DC2126A

概要

DC2126A ボードからデータを受け取って解析するサンプル・アプリケーションです。接続できるコンポーネントは以下のとおりです。

- SmartMesh IP マネージャ

> ⚠ このアプリケーションを使用できるのは、SmartMesh IP ネットワーク内で DC2126A ボードが稼働している場合のみです。

実行方法

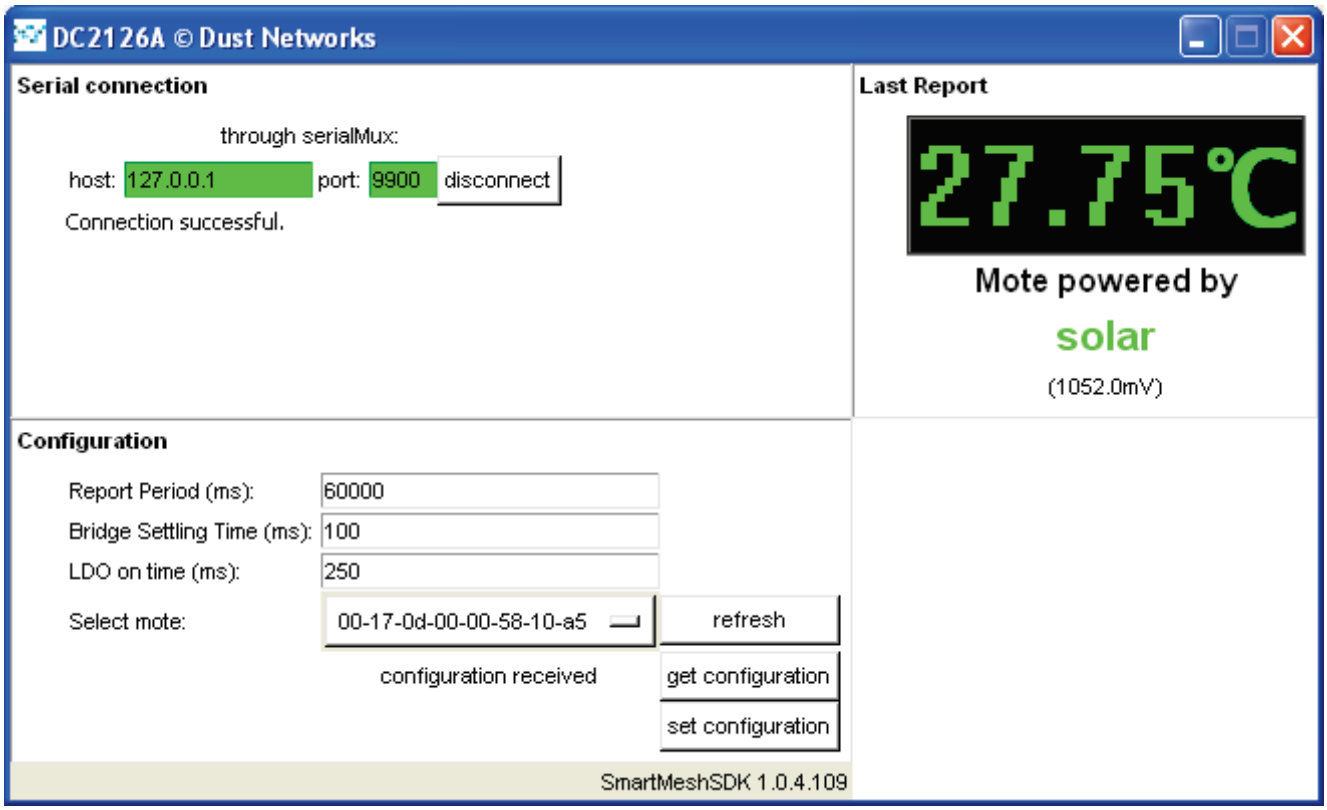
DC2126A アプリケーションの実行方法は以下のとおりです。

- Windows 実行可能ファイル(*/win/ DC2126A.exe*)をダブルクリックする
- ソース・ファイル(*/src/ bin/DC2126A/DC2126A.py*)をダブルクリックする(Windows 以外の OS では追加の手順が必要になる場合あり)

説明

DC2126A は、GUI ベースのサンプル・アプリケーションです。DC2126A は、SerialMux またはシリアル接続経由で SmartMesh IP マネージャに接続します。既存の SmartMesh IP ネットワーク内に稼働中の DC2126A ボードがある場合、このアプリケーションは、各ボードが収集したデータを自動的に解析して表示します。

- DC2126A ボードが実行中になっており、SmartMesh IP ネットワークにジョインしていることを確認します。
- DC2126A サンプル・アプリケーションを起動して、SmartMesh IP マネージャに接続します。
  接続には SerialMux またはシリアル・ポートを使用できます。
- DC2126A サンプル・アプリケーションは SmartMesh IP マネージャに接続されると、ネットワーク内のモートを取得して、**Select mote** ドロップダウン・メニューに表示します。
- DC2126A がデータを送信するたびに、DC2126A サンプル・アプリケーションの右上にこのデータが表示されます。
- SmartMesh IP ネットワーク内の複数の DC2126A ボードによって生成されたデータをボード別に表示するには、**Select mote** ドロップダウン・メニューを使用して、対象ボードの MAC アドレスを選択します。
- **get configuration** ボタンをクリックすると、**Select mote** ドロップダウン・メニューで選択されたモートにパケットが送信され、現在の構成を要求します。モートから現在の構成が返されると、Configuration フォームに表示されます。
- 特定のモートの構成を変更する場合、Report Period、Bridge Settling time、LDO on time を入力して、**Select mote** ドロップダウン・メニューでモートの MAC アドレスを選び、**set configuration** をクリックします。
- **Select mote:**ドロップダウン・メニューで選択した MAC アドレスを更新するには、**refresh** ボタンをクリックします。



7.5.6 HrListener

概要

HrListener アプリケーションは、SmartMesh IP マネージャに接続してヘルス・レポート通知をサブスクライブし、受け取った HR の数をタイプ別に表形式で表示します。

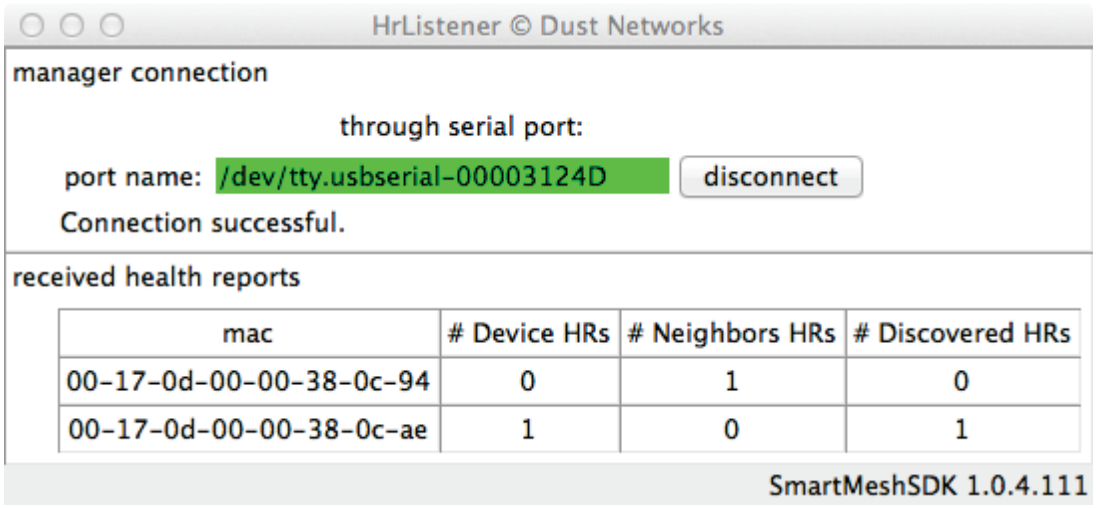
実行方法

HrListener アプリケーションの実行方法は以下のとおりです。

- Windows 実行可能ファイル(*/win/HrListener.exe*)をダブルクリックする
- ソース・ファイル(*/src/bin/HrListener/HrListener.py*)をダブルクリックする

説明

HrListener は GUI ベースのアプリケーションで、SmartMesh IP マネージャに接続してヘルス・レポート通知をサブスクライブし、受け取った HR の数をタイプ別に表形式で表示します。HR のタイプを判定するだけで、コンテンツの解析は行いません。



HrListener は解析したヘルス・レポートを、*receivedHRs.log* としてアプリケーション（exe または py）と同じディレクトリ内に保存します。以下にそのサンプルを示します。

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

7.5.7 InstallTest

概要

InstallTest は、シンプルなコンソール・アプリケーションで（グラフィカルでない）、SmartMesh SDK のサンプルをソースから実行するために必要な Python コンポーネントが正しくインストールされているかどうかをテストします。

実行方法

InstallTest アプリケーションの実行方法は以下のとおりです。

- Windows 実行可能ファイル(*/win/InstallTest.exe*)をダブルクリックする
- ソース・ファイル(*/src/bin/InstallTest/InstallTest.py*)をダブルクリックする

> ⚠ ソース・バージョンを実行するには Python が正しくインストールされている必要があるため、このアプリケーションを使用して Python のインストールをテストすることはできません。ただし、pyserial のインストールを検証することはできます。

説明

InstallTest は、小さいコマンド・ウィンドウ・アプリケーションであり、SmartMesh SDK をソースから実行するために必要な Python コンポーネントが正しくインストールされているかどうかをテストします。起動するとコマンド・ウィンドウが開き、以下のテキストが表示されます。

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

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

7.5.8 LBRConnection

概要

Low-power Border Router（LBR）は、SmartMesh IP マネージャとインターネットの間に配置されます。インターネットの IPv6 パケット形式を SmartMesh IP ネットワークの 6LoWPAN パケット形式に変換して、インターネット上のコンピュータ（インターネット・ホスト）と SmartMesh IP モートが通信できるようにします。

Linux コンピュータで稼働するサンプルの LBR 実装は、linear-tech.co.jp で提供されています。
LBRConnection アプリケーションを使用すると、稼働中の LBR に接続して、LBR に関する情報と通過するパケット数を表示できます。

実行方法

LBRConnection アプリケーションの実行方法は以下のとおりです。

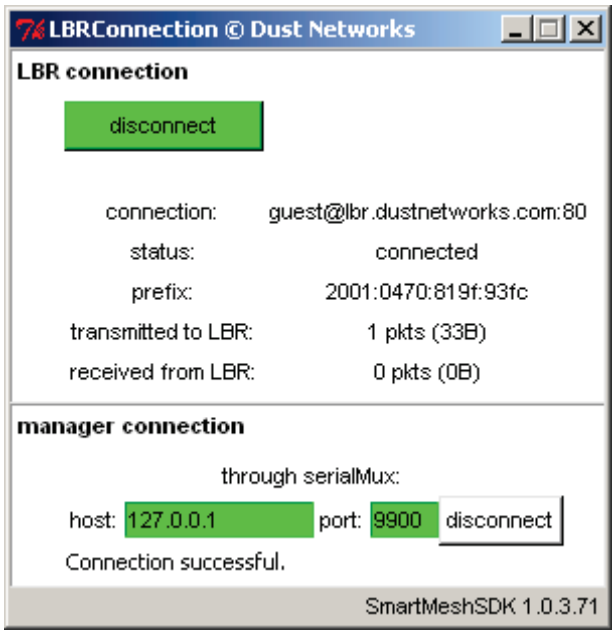
- Windows 実行可能ファイル（*/win/LBRConnection.exe*）をダブルクリックする
- ソース・ファイル（*/src/bin/LBRConnection/LBRConnection.py*）をダブルクリックする

説明

GUI の構成

LBRConnection アプリケーションは以下の 2 つのフレームで構成されています。

- LBR connection フレームで、LBR に接続します。
- manager connection フレームで、SmartMesh IP マネージャに接続します。



このアプリケーションは、内部で SmartMesh IP マネージャの IP 通知をサブスクライブします。通知を受け取ると、接続先の LBR にパケットを転送します。同様に、LBR からデータを受け取ると、SmartMesh IP マネージャに転送します。

.lbrauth ファイルの使用

LBRConnection アプリケーションから LBR に接続するには、以下の情報が必要です。

- LBR の IP アドレス
- LBR がリスニングしている TCP ポート番号
- ネットワークのユーザー名
- 認証情報

必要な認証情報はセキュリティ・レベルによって異なります。詳細については、LBR User Guide を参照してください。認証情報をフィールドに入力させる代わりに、LBRConnection アプリケーションは「LBR 認証ファイル」(*.lbrauth)を解析します。インストールした SmartMesh SDK には、サンプルの LBR 認証ファイルが含まれています。

表示される情報

LBR connection フレームに表示される情報は以下のとおりです。

フィールド	概要
connection	LBR に接続すると、ユーザー名と LBR のアドレスが、*username@address* という形式で表示されます。
status	現在のステータスが、*connected* または *disconnected* のいずれかで表示されます。
prefix	LBR に接続すると、LBR からネットワークに割り当てられた IPv6 プリフィックスが表示されます。
transmitted to LBR	LBR に送信されたパケット数とバイト数が表示されます。
received from LBR	LBR から受信したパケット数とバイト数が表示されます。

チュートリアル

このアプリケーションは、以下のチュートリアルで使用されています。

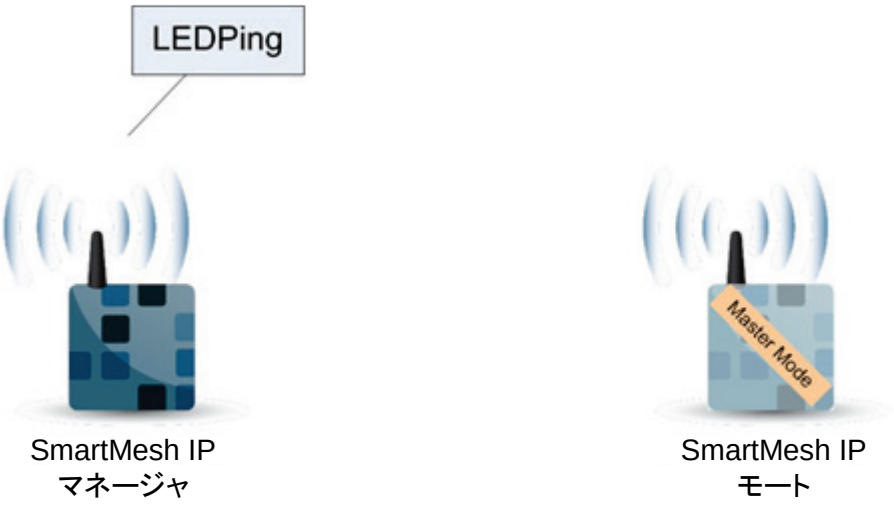
- SmartMesh IP 向け
  - インターネット統合

7.5.9 LEDPing

概要

LEDPing アプリケーションは、デフォルト・ファームウェアを使用して、**マスター・モード**で稼働中の SmartMesh IP モートの LED を継続して切り替え続けます。

セットアップ



実行方法

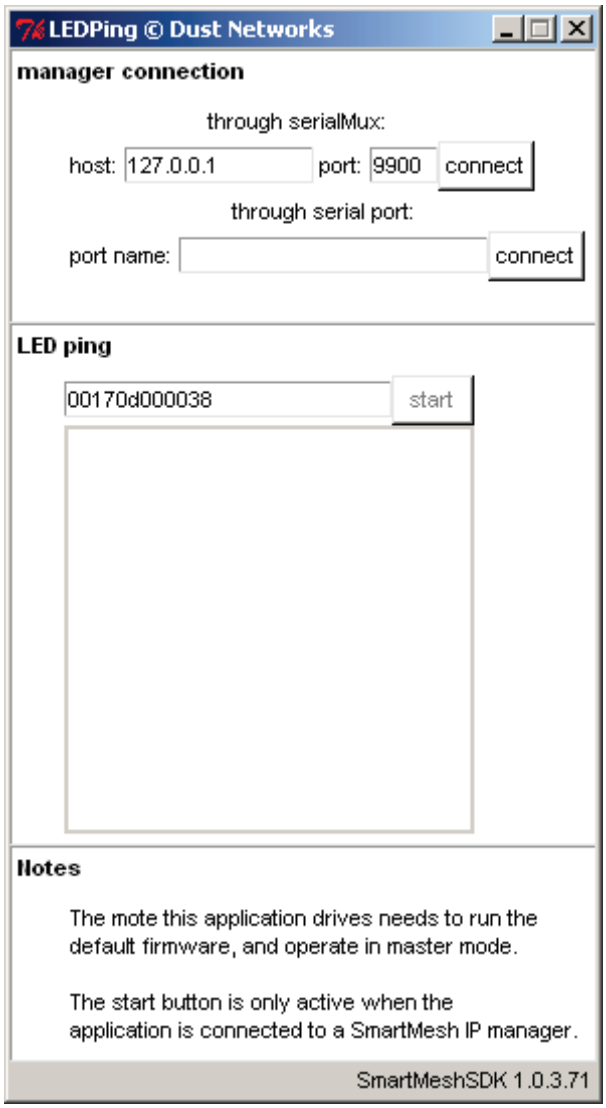
LBRConnection アプリケーションの実行方法は以下のとおりです。

- Windows 実行可能ファイル(*/win/LEDPing.exe*)をダブルクリックする
- ソース・ファイル(*/src/bin/LEDPing/LEDPing.py*)をダブルクリックする

GUI の構成

LEDPing アプリケーションは以下の 2 つのフレームで構成されています。

- **manager connection** フレームで、SmartMesh IP マネージャに接続します。
- **LED ping** フレームで SmartMesh IP モートを選択すると、その LED ステートを監視できます。
- **Notes** フレームには、役に立つヒントが表示されます。



説明

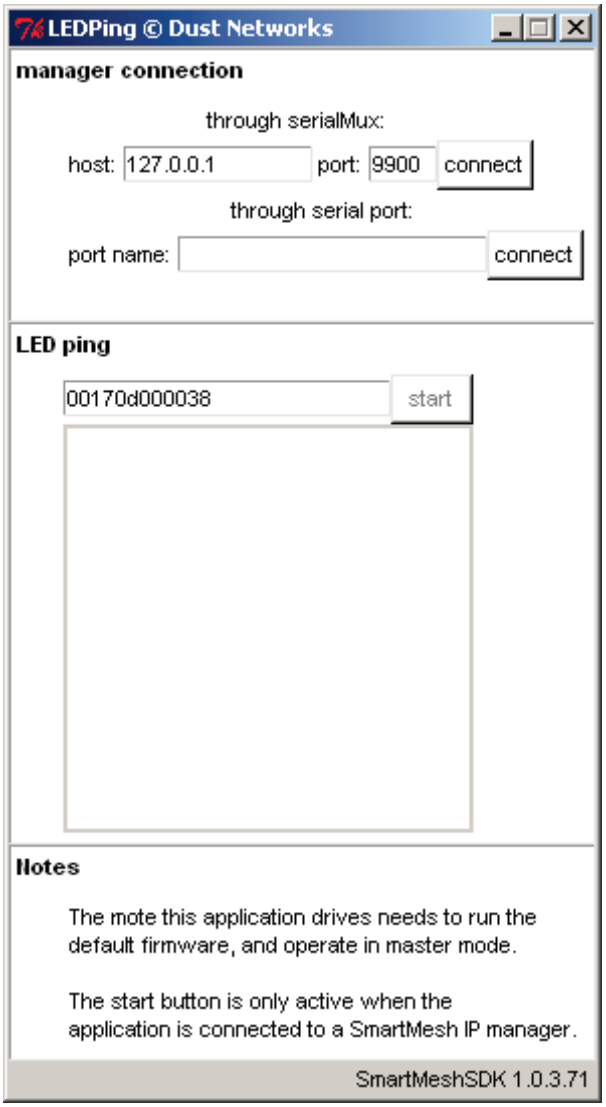
SmartMesh IP モートのデフォルト・ファームウェアには、オンチップ・アプリケーション・プロトコル（OAP）が実装されています。LEDPing アプリケーションは SmartMesh IP マネージャに接続して、ネットワーク内にある全ての SmartMesh IP モートに OAP パケットを送信することで、*INDICATOR* LED を切り替え続けます。**start** ボタンが押された後の動作は以下のようになります。

- LEDPing アプリケーションは選択された SmartMesh IP モートに対して、LED をオンにする OAP パケットを送信する。
- SmartMesh IP モートはパケットを受け取ると、LED をセットして、アプリケーション・レベルのアクノリッジを送信する。
- LEDPing はアプリケーション・レベルのアクノリッジを受け取ると、LED をオフにする新しい OAP パケットを送信する。
- **stop** ボタンが押されるまで、上記処理が繰り返される。

このアプリケーションを使用すると、ネットワーク内のラウンドトリップ時間を確認できます。デフォルト構成では、2～3 秒ごとに LED が点滅します。ネットワーク構成を変更（バックボーン・モードの切り替えなど）すると、点滅速度を大幅に向上できる場合があります。

アプリケーションの使用

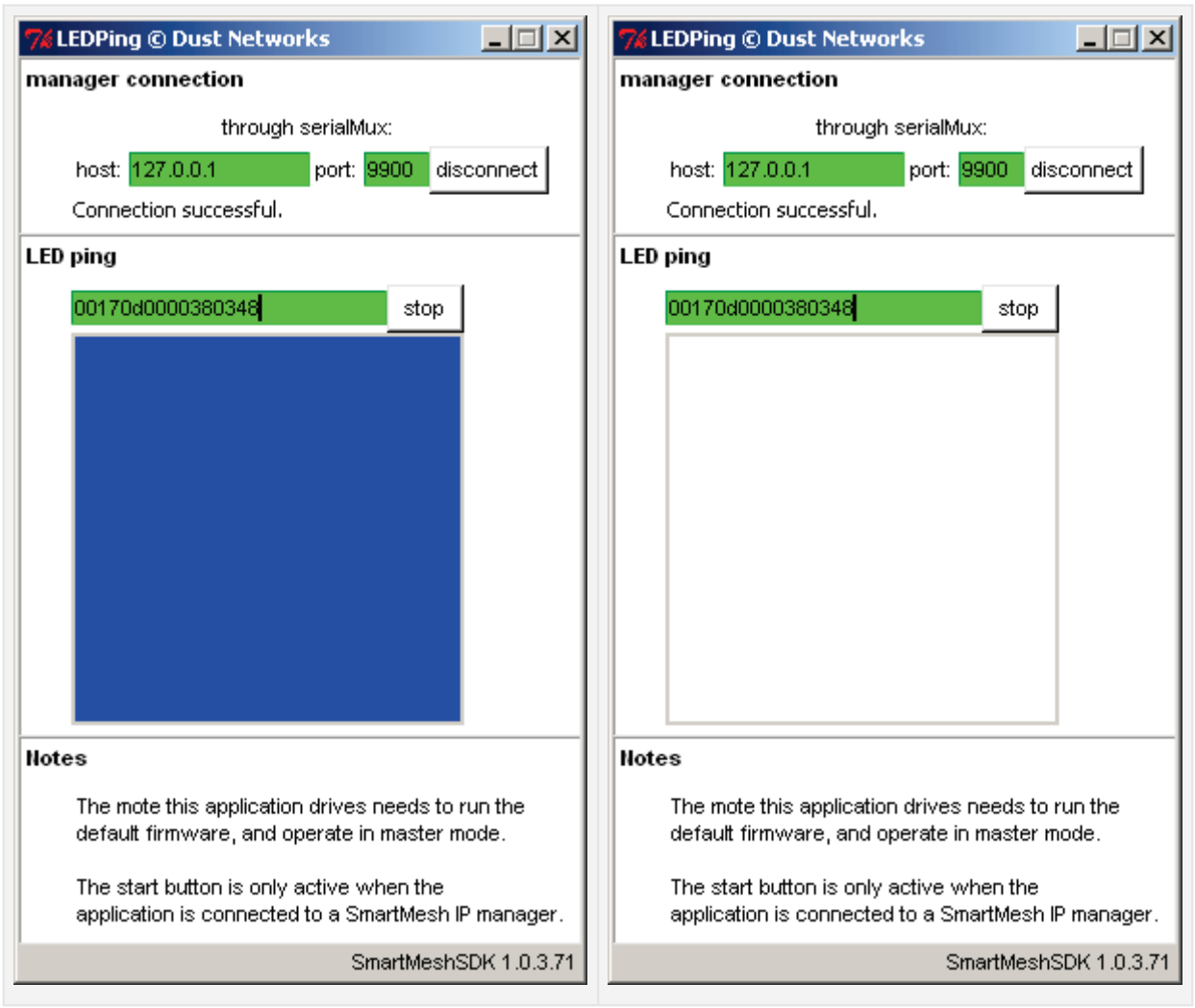
- LEDPing アプリケーションを起動します。



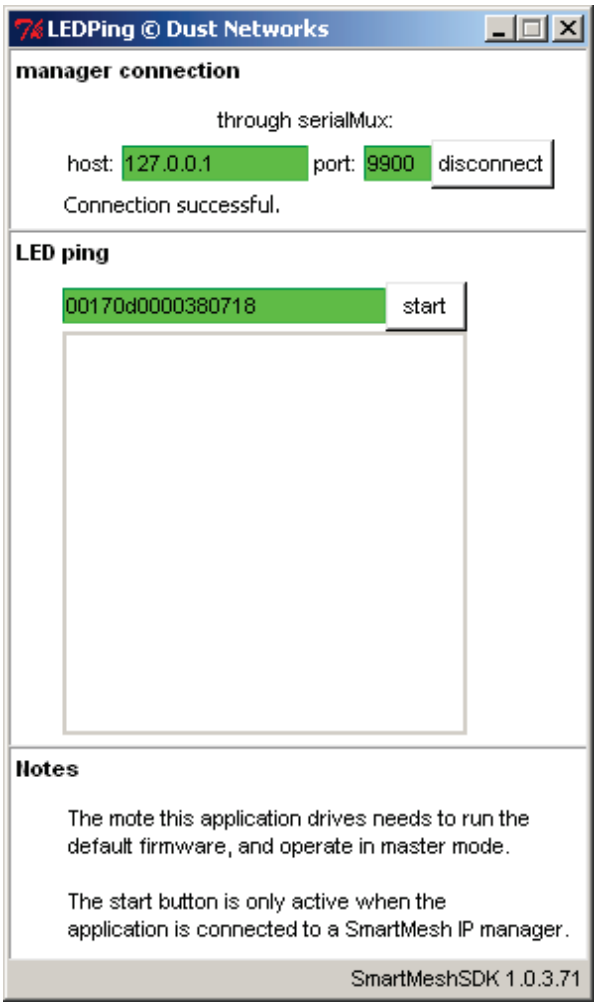
- SerialMux 経由か、またはシリアル・ポートから直接、SmartMesh IP マネージャにアプリケーションを接続します。接続に成功すると、**manager connection** フレームのフィールドが緑色に変わります。

- ネットワーク内で現在稼働している任意のモートの MAC アドレスを入力します。**start** ボタンを押します。

- LEDPing アプリケーションが、SmartMesh IP モート上の LED と同じタイミングで青色に点滅します。



- 点滅を中断するには、**stop** ボタンをクリックします。



7.5.10 MgrListener

概要

MgrListener アプリケーションは SmartMesh IP マネージャに接続して、全てのタイプの通知をサブスクライブし、受け取ったタイプ別のカウンタを表示します。

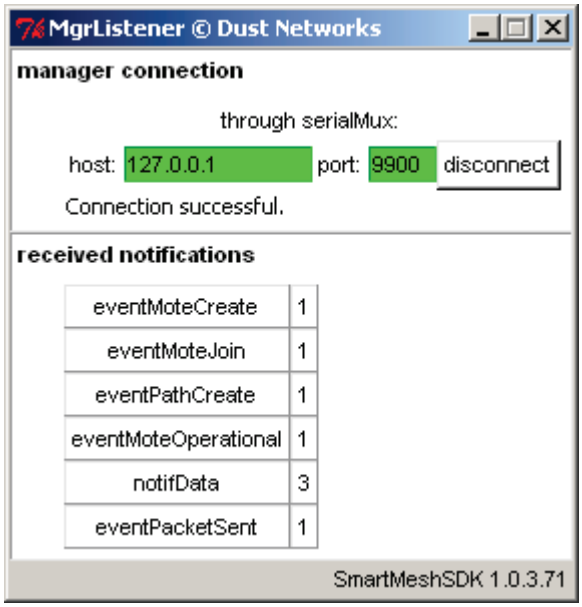
実行方法

MgrListener アプリケーションの実行方法は以下のとおりです。

- Windows 実行可能ファイル(*/win/MgrListener.exe*)をダブルクリックする
- ソース・ファイル(*/src/bin/MgrListener/MgrListener.py*)をダブルクリックする

説明

MgrListener は GUI ベースのアプリケーションで、SmartMesh IP マネージャに接続して全ての通知をサブスクライブし、受け取ったそれぞれの通知の数を表示します。



7.5.11 MuxConfig

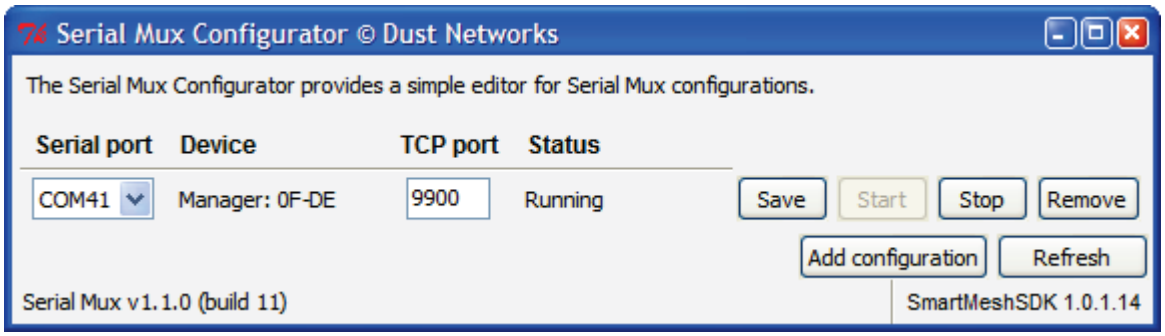
概要

Serial Mux Configurator は、SerialMux の構成を管理するための GUI エディタを提供します。

それぞれの構成が、マネージャに接続される 1 つの SerialMux プロセス(Windows のバックグラウンド・サービスとして実行)に相当します。コンピュータに接続するマネージャが複数ある場合、SerialMux 構成も複数セットアップする必要があります。

Configurator が提供する機能は以下のとおりです。

- SerialMux 構成の追加、更新、削除（例：別のシリアル・ポートへのマネージャの接続）
- SerialMux プロセスの開始と停止
- 接続先デバイスの基本デバイス情報の表示

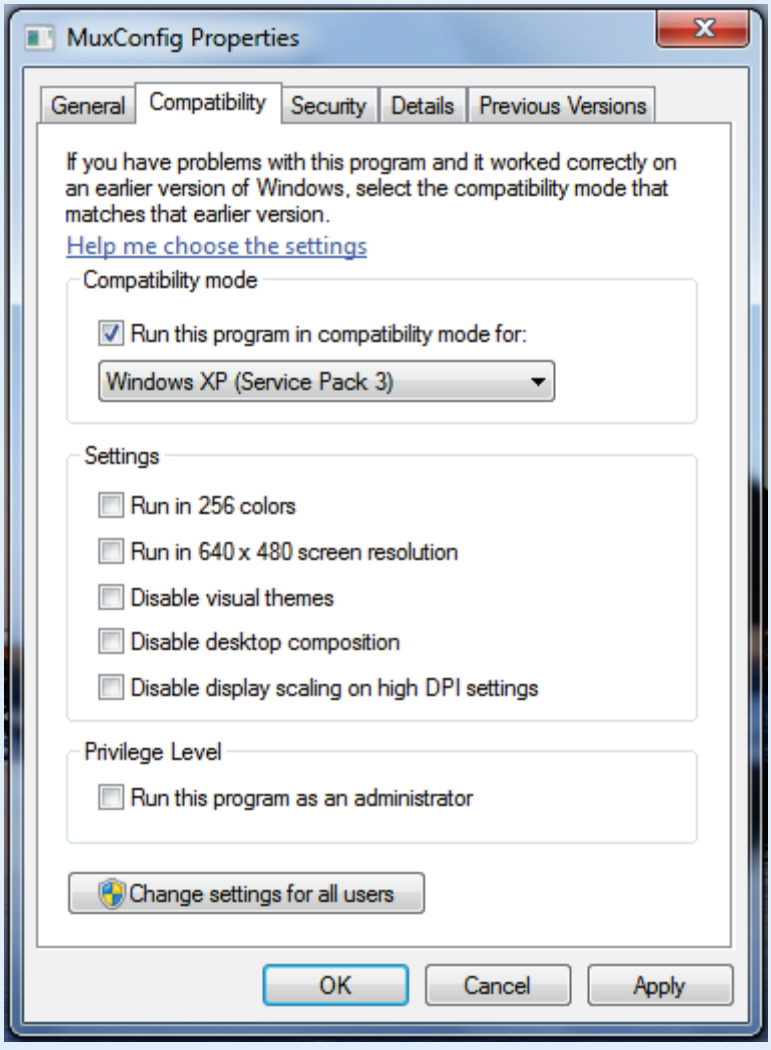


Configurator の実行方法

Configurator は現在 SmartMesh SDK に付属していますが、実行するには、Serial Mux をインストールする必要があります。Configurator を実行するには、コンパイル済みの Windows 実行可能ファイル（*MuxConfig.exe*）をダブルクリックします。

Windows 7 では、XP 互換モードで Configurator を実行する必要があります。一部のバージョンの Windows 7 は XP 互換モードをサポートしていません。これらのバージョンを使用する場合、SmartMeshSDK v1.0.2.53 以降を使用し、管理者として Configurator を実行します。

1. *MuxConfig.exe* を見つけて右クリックし、**Properties** を選択します。
2. Compatibility タブで **Run this program in compatibility mode for:**をチェックし、**Windows XP (Service Pack 3)**を選択します。互換モードが使用できない場合は、**Run this program as an administrator** をチェックします。**OK** をクリックします。



Configurator の使用

Configurator を起動すると、現在の SerialMux 構成リストがウィンドウに表示されます。
シリアル・ポート・リスト（Serial Port コンボボックス）には、USB シリアル・ポートのリストが表示されます。
シリアル・ポート・リストには、非 Dust デバイスに関連付けられたポートが含まれる場合があります。

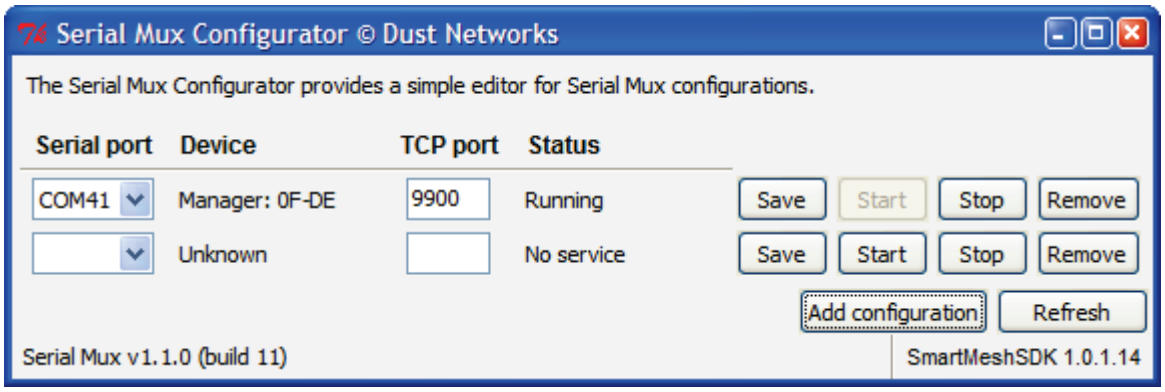
コンフィグレーションの編集

構成を編集するには、Serial port フィールドと TCP port フィールド（またはそのいずれか）を変更します。*Save* をクリックして構成を更新します。SerialMux サービスがあらかじめ実行されている場合、Configurator はこのサービスを停止してから再起動します。

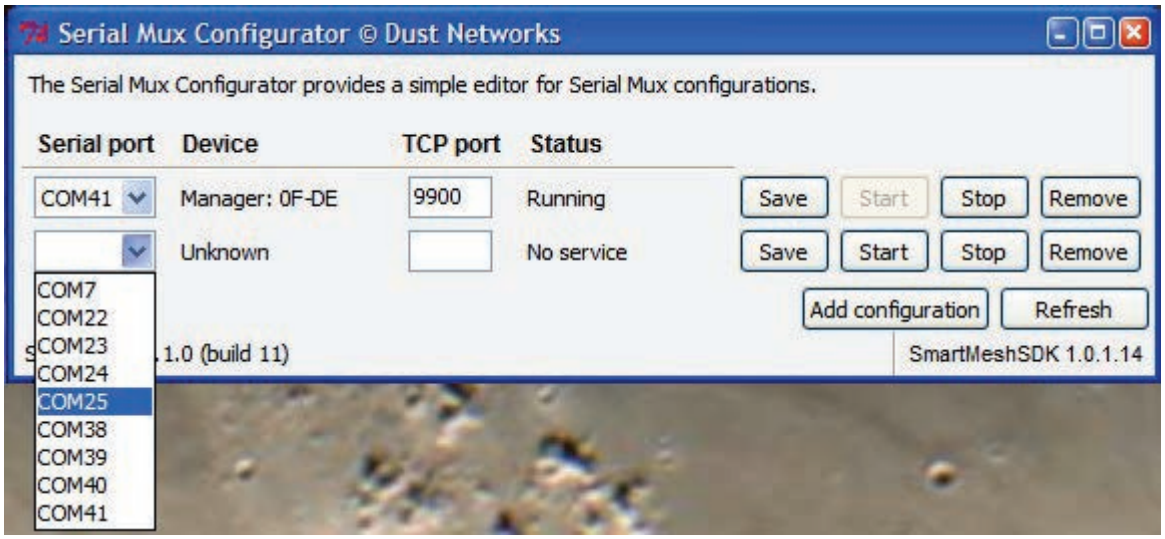
TCP port には、その他の TCP ポートとは異なる番号を指定する必要があります。

新しい構成の作成

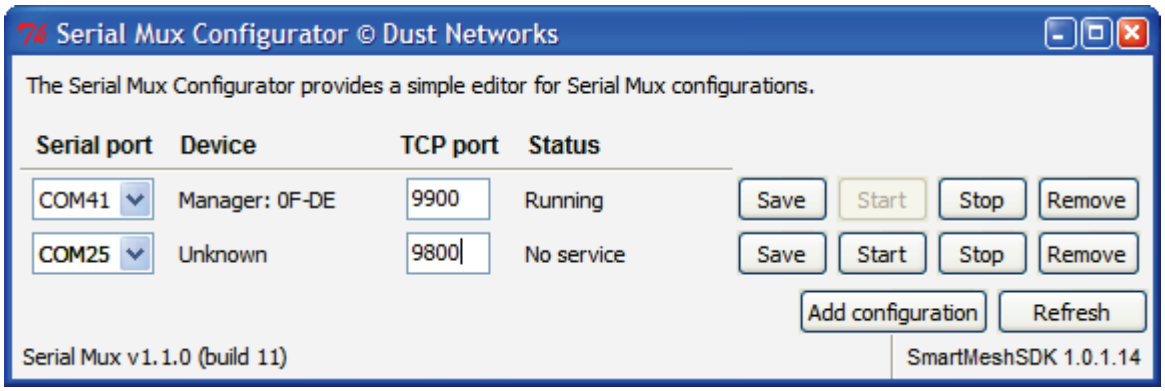
Add configuration ボタンをクリックすると、空の構成が作成されます。



Serial port フィールドと TCP port フィールドを編集して、必要な構成を指定します。TCP port には、その他の SerialMux サービスと競合しない番号を指定する必要があります（コンピュータ上のその他のサービスが使用する TCP ポートと競合する可能性もあります）。

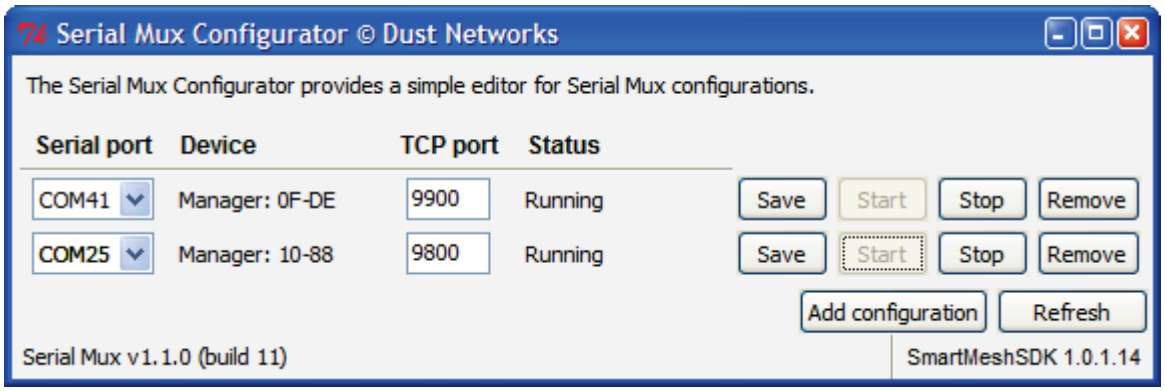


Save をクリックして構成を更新します。



Start をクリックして、SerialMux プロセスを開始します。

SerialMux がマネージャに接続されると、Device フィールドが Unknown から Manager に変わります。APIExplorer などのインタラクティブなツールを使用して、SerialMux の TCP ポート上のマネージャに接続します。



上記の例では、APIExplore をポート 9900（SerialMux のデフォルト)に接続すると、MAC アドレスの末尾に *0F-DE* を持つマネージャに接続され、ポート 9800 に接続すると、MAC アドレスの末尾に *10-88* を持つマネージャに接続されます。

構成の削除

Remove ボタンをクリックすると、特定の SerialMux 構成を削除できます。関連付けられた SerialMux プロセスが実行中である場合、このプロセスを停止してから SerialMux 構成が削除されます。

7.5.12 OTAP Communicator

概要

OTAP Communicator は、Over-the-Air Programming（OTAP）を介してモートにファームウェアをロードするためのツールです。

OTAP プロセスはワイヤレス・ネットワーク経由でソフトウェア更新やファイルを送信するので、各モートに物理的にアクセスすることなくソフトウェア・アップグレードを配信することができます。OTAP プロセスはファイル・データを小さいチャンクで送信し、完了したかどうかをモートに問い合わせて、モートが受信していない場合はもう一度チャンクを送信します。OTAP プロセスを使用して、ネットワーク経由で完全なモート・イメージをロードすると、一般に長い時間（数時間単位）がかかります。

OTAP Communicator は、Windows のコマンド・ライン実行可能ファイルまたは Python ソース・コードとして、SmartMesh SDK に含まれています。
OTAP Communicator を実行するには、SmartMesh IP マネージャに Serial Mux が接続されている必要があります。

コマンド・ライン引数

OTAP Communicator は、Windows のコマンド・ライン・プロンプトから実行できます。このプログラムの入力引数は、ロードするファイルのパスであり、任意で宛先モートのリストを渡すこともできます。ファイルの拡張子が *.otap* または *.otap2*（SmartMesh IP モートで使用できるファイル形式は *.otap2*）でない限り、ファイルは通常のファイルとしてモートのファイル・システムにロードされます。

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

オプション	
-v	デバッグ用の詳細出力を有効化します。
-p	SerialMux の TCP ポートを指定します。
-m	宛先モートの MAC アドレスを指定します。このオプションは複数回指定できます。-m オプションを指定しない場合、ファイル（またはイメージ）は全てのモートに配信されます。

例

OTAP Communicator を実行すると、役に立つメッセージがいくつか表示されます。

```
$ OTAPCommunicator.exe -m 00-17-0d-00-00-38-01-8d mote_ip_1_1_0_36_oski.otap2
Welcome to the OTAP communicator console
Loading mote_ip_1_1_0_36_oski.otap2
Starting OTAP for mote_ip_1_1_0_36_oski.otap2

Starting handshake with 1 motes
Handshake completed, 1 motes accepted
Starting data transmission of 2798 blocks, 1 motes left
...<long wait>...
Starting status query to 1 motes
Data complete. No motes left on incomplete list
Starting commit for 'mote_ip_1_1_0_36_oski.otap2' to 1 motes
00-17-0D-00-00-38-01-8D committed mote_ip_1_1_0_36_oski.otap2 [FCS=0x6ca0]

Successful OTAP to:

00-17-0D-00-00-38-01-8D
```

上記の出力が示す内容は以下のとおりです。

- OTAP Communicator は、*mote_ip_1_1_0_36_oski.otap2* ファイルをロードしました。
- OTAP Communicator は、1 つのモートに対して 1 つの OTAP セッションを開始しました（コマンド・ラインで 1 つのモートが指定されていたため）。

- 1 つのモートが OTAP ハンドシェイクを受け入れました。OTAP Communicator はこのモートに対して、ファイル送信を開始します。ファイル・データは正しく送信されました。
- コミットに成功するたびに、MAC アドレスと FCS が表示されます。
- 成功した配信と失敗した配信が一覧で表示されます。

詳細な出力は、*otap_communicator.log* ログ・ファイルとして、OTAP Communicator が実行されたディレクトリ内に保存されます。

7.5.13 PkGen

概要

PkGen は SmartMesh IP マネージャまたは SmartMesh WirelessHART マネージャに接続して、**マスター・モード**で稼働しているモート上のパケット生成アプリケーションにコマンドを送信できるようにします。

モートの各モードについては、SmartMesh IP ユーザー・ガイドと SmartMesh WirelessHART User's Guide を参照してください。

PkGen の実行

PkGen アプリケーションの実行方法は以下のとおりです。

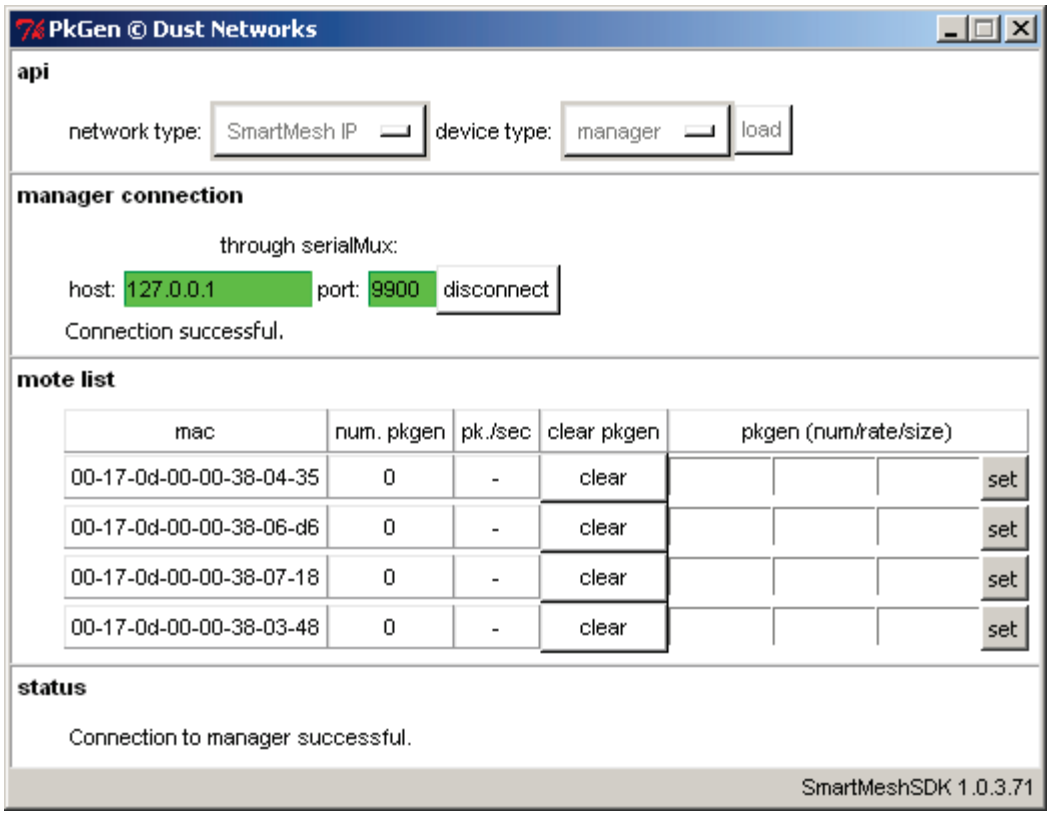
- Windows 実行可能ファイル(*/win/ PkGen.exe*)をダブルクリックする
- ソース・ファイル(*src/ bin/PkGen/PkGen.py*)をダブルクリックする(Windows 以外の OS では追加の手順が必要になる場合あり)

説明

PkGen アプリケーションは以下の 4 つのフレームで構成されています。

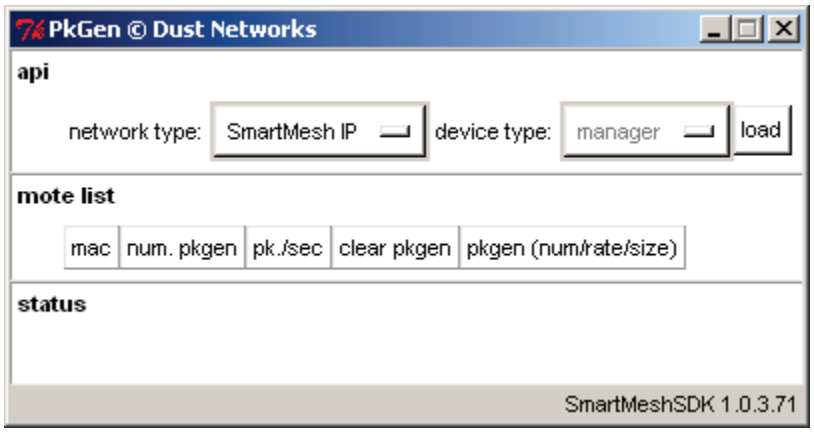
- **api** フレームでは、接続先として SmartMesh IP または SmartMesh WirelessHART のいずれかのネットワークを指定します。ここではマネージャに接続しているため、device type セレクタは無効化されています。
- **manager connection** フレームでは、マネージャへの接続方法を指定します。
- マネージャに接続すると、**mote list** に、現在ネットワーク内にあるモートのリストが表示されます。
- **status** フレームには、一般的なステータス情報が表示されます。

以下に、ユーザーが SmartMesh IP マネージャに接続した後の PkGen アプリケーション・ウィンドウの例を示します。SmartMesh WirelessHART マネージャに接続した場合も、API タブと connection タブ以外は同じように表示されます。

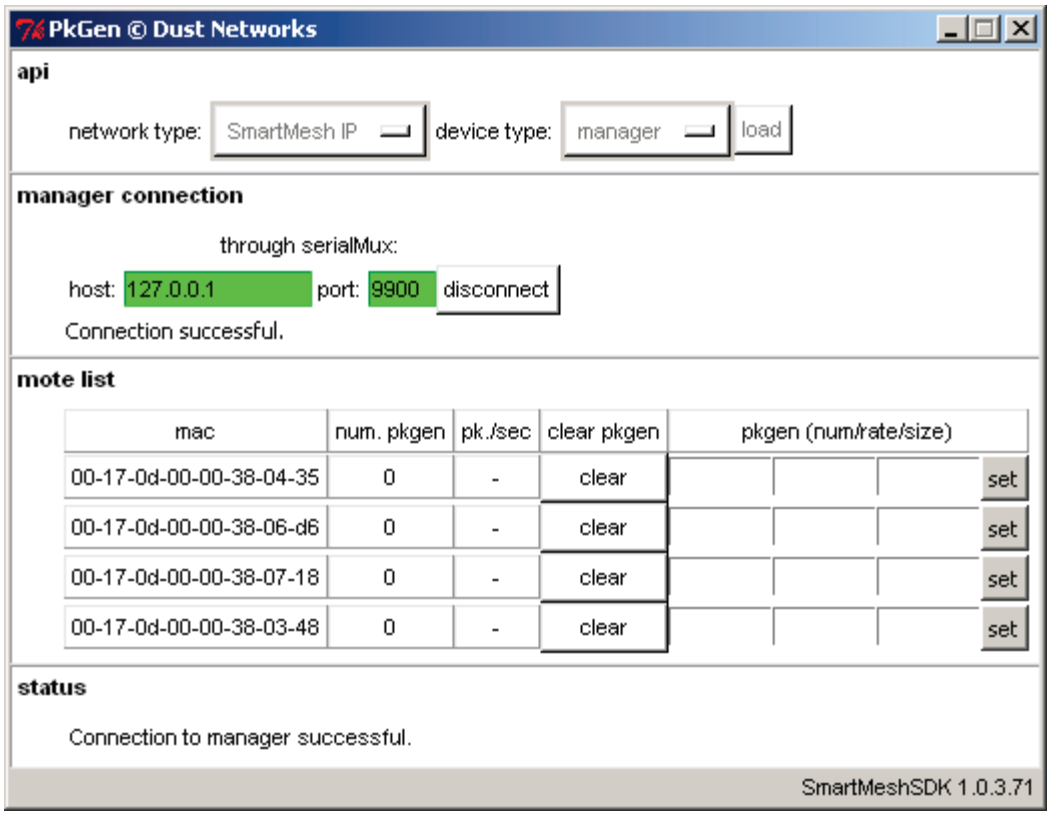


手順

- PkGen アプリケーションを起動すると、以下のウィンドウが表示されます。



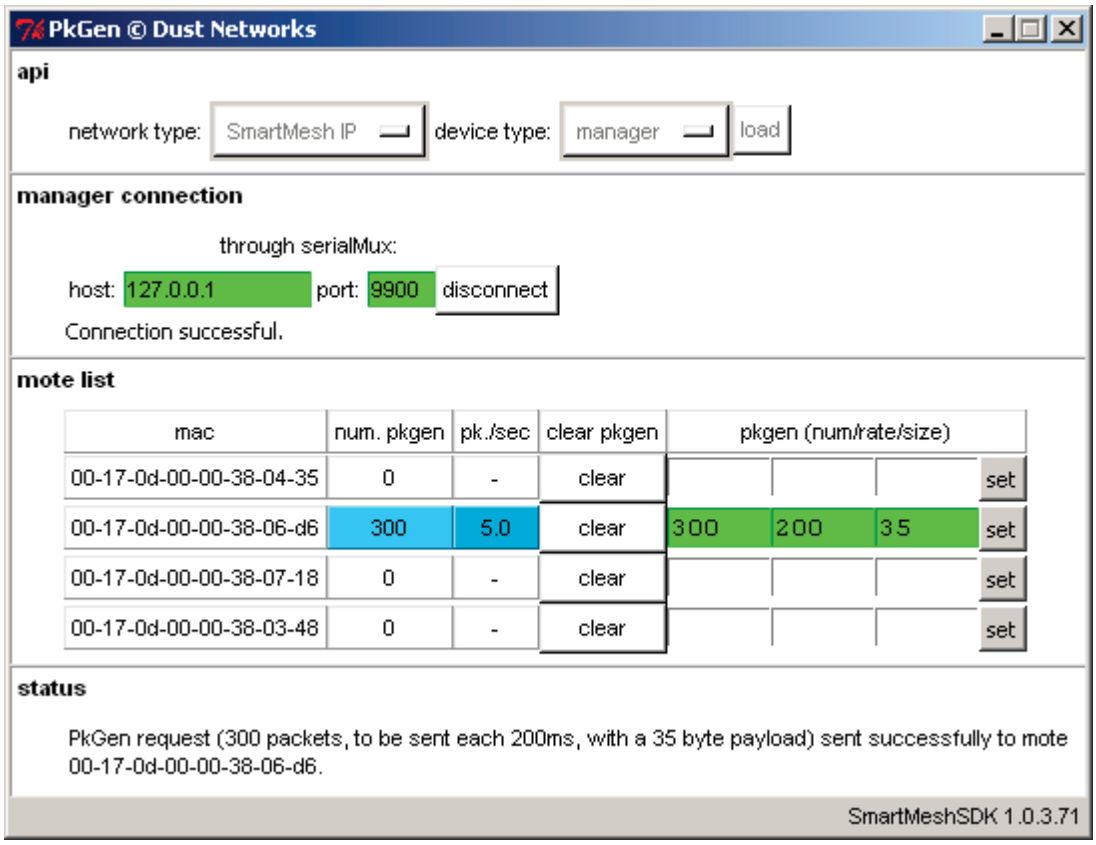
- 接続先のネットワーク・タイプを選択します（ここでは、SmartMesh IP を選択）。
- マネージャへの接続方法を選択します（ここでは SerialMux を使用）。マネージャに接続すると、ネットワーク内で現在**稼働中**ステートになっているモートのリストが取得されます。



⚠ 稼働中モードではないモートやアクセス・ポイントは、アプリケーションに表示されません。

⚠ **稼働中**モードのモートは、**マスター**・モードであるか**スレーブ**・モードであるかに関係なく全て表示されます。ただし、相互作用の対象になるのは、マスター・モードで稼働しているモートのみです。

- 目的のモートに対して、モート上で実行中の PkGen アプリケーションに送信する構成を入力します。
  - *num* は、モートで生成するパケット数です。
  - *rate* は、連続する 2 つのパケット間の間隔(ミリ秒)です。
  - *size* は、パケット・ペイロードのサイズ(バイト)です。
- 下記の構成では、モート *00-17-0d-00-00-38-06-d6* が、それぞれが 35 バイトのペイロードを含む 300 個のパケットを、200 ミリ秒の間隔で送信します。特定のネットワーク・サイズとトポロジに対して、このペースは全てのデバイスでサポートされるとは限らないためご注意ください。



- 表示されるカウンタは以下のとおりです。
  - *num* - モートによって生成され、受け取ったパケットの数。対象となるパケットは、モートの PkGen アプリケーションによって生成されたパケットのみです。生成されるデータ・タイプが異なる場合、ここでのカウントの対象にはなりません。
  - *pk./sec* - アプリケーションが 1 秒あたりに受け取ったパケット数のレート。
- **clear** ボタンを押すと、カウンタが消去されます。

7.5.14 SensorDataReceiver

概要

SensorDataReceiver アプリケーションは、SmartMesh IP マネージャに接続して、SmartMesh IP モートに接続した *Upstream* アプリケーションから送信されたデータを表示します。

実行方法

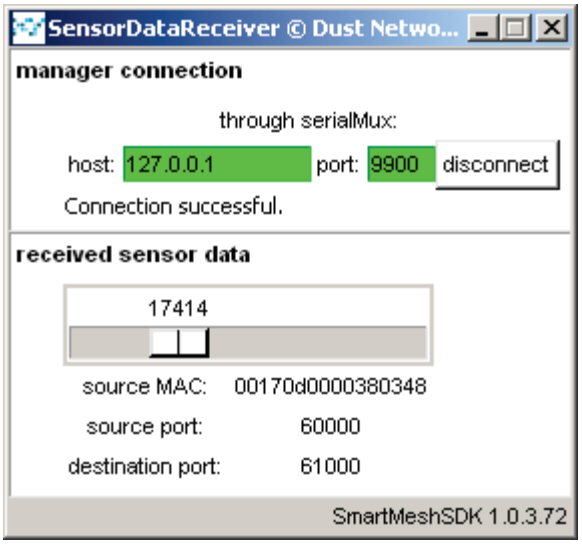
LBRConnection アプリケーションの実行方法は以下のとおりです。

- Windows 実行可能ファイル(*/win/SensorDataReceiver.exe*)をダブルクリックする
- ソース・ファイル(*/src/bin/SensorDataReceiver/SensorDataReceiver.py*)をダブルクリックする

説明

SensorDataReceiver アプリケーションは以下の 2 つのフレームで構成されています。

- **manager connection** フレームで、SmartMesh IP マネージャに接続します。
- **received sensor data** フレームには、SmartMesh IP モートに接続した Upstream アプリケーションから受信したデータが表示されます。



このアプリケーションは、内部で SmartMesh IP マネージャのデータ通知をサブスクライブします。通知を受け取ると、スライダに最初の 2 バイトを設定し、パケットに関する情報を表示します。

フィールド	概要
source MAC	データの送信元となるモートの MAC アドレス。
source port	送信元の UDP ポート番号(データが送信された SmartMesh IP モート上のソケットに関連付けられたポート番号)。
destination port	データを受け取った SmartMesh IP マネージャ上の UDP ポート番号。

フィールド	概要
timestamp generated	SmartMesh IP モートがデータを生成した時点のネットワーク・タイムスタンプ（秒）。
timestamp received	SmartMesh IP マネージャがデータを受信した時点のコンピュータ・タイムスタンプ（秒）。

このアプリケーションにデータが表示されるのは、モートに接続された *Upstream* アプリケーションを同時に使用している場合のみです。

7.5.15 SimpleIPMgr および SimpleIPMote

概要

SimpleIPMgr と SimpleIPMote は、プログラムを使用して各デバイスの API とやり取りする方法を示すコンソール・アプリケーション（グラフィカルでない）です。これらはサンプル・コードとしての役割を果たします。

実行方法

このサンプル・アプリケーションは、さまざまなスクリプトを集めたものです。

- *SimpleIPMgr* は、SmartMesh IP マネージャの API を実行します。
- *SimpleIPMote* は、SmartMesh IP モートの API を実行します。

スクリプトの実行方法は 2 種類あります。

- */win/*内の該当する Windows 実行可能ファイルをダブルクリックする
- */src/bin/Simple/*にあるソース・ファイルをダブルクリックする

説明

スクリプトを使用して、SmartMesh SDK の機能を詳細に確認することができます。以下に、スクリプトの出力例を示します。

```
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.
```

7.5.16 TempMonitor

概要

TempMonitor は SmartMesh IP マネージャまたは SmartMesh WirelessHART マネージャに接続して、**マスター・モード**で稼働しているモート上の温度測定アプリケーションにコマンドを送信できるようにします。

モートの各モードについては、SmartMesh IP ユーザー・ガイドと SmartMesh WirelessHART User's Guide を参照してください。

TempMonitor の実行

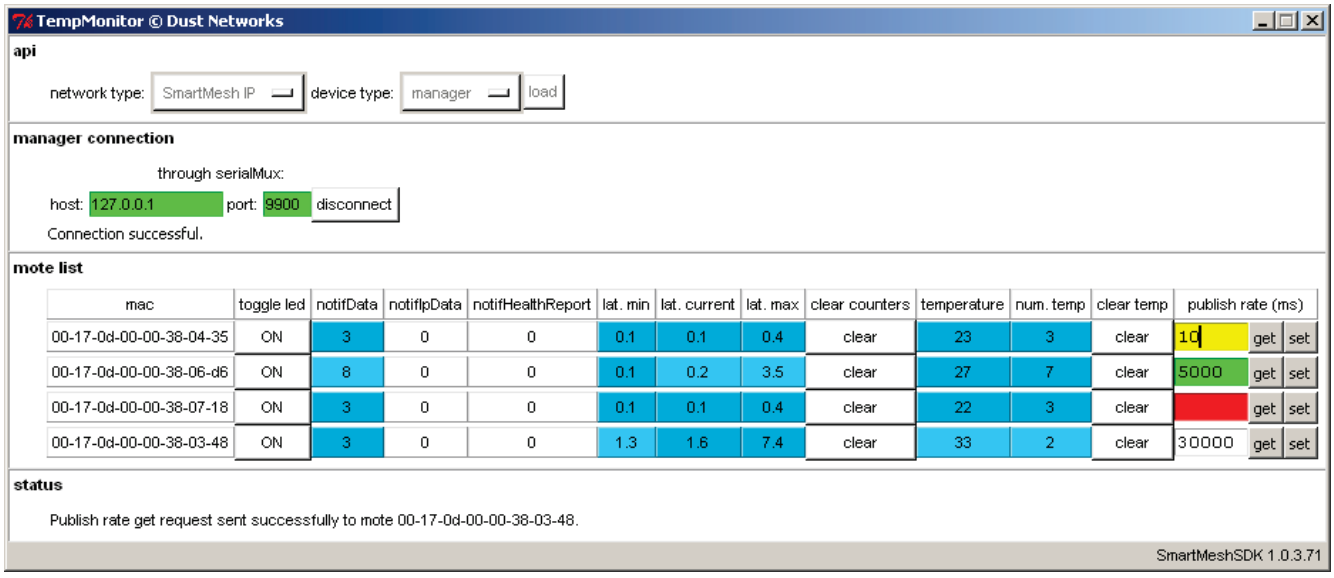
TempMonitor はマネージャに接続して、**マスター・モード**で稼働中のモートと相互作用します。

TempMonitor アプリケーションの実行方法は以下のとおりです。

- Windows 実行可能ファイル（*/win/TempMonitor.exe*）をダブルクリックする
- ソース・ファイル（src/bin/TempMonitor/TempMonitor.py）をダブルクリックする

説明

以下に、ユーザーが SmartMesh IP マネージャに接続した後の TempMonitor アプリケーション・ウィンドウの例を示します。SmartMesh WirelessHART マネージャに接続した場合も、API タブと connection タブ以外は同じように表示されます。



説明

TempMonitor は GUI ベースのアプリケーションであり、マネージャに接続してネットワーク内のモートをリスト表示して、**マスター・モード**で稼働中の一部のモートと相互作用できるようにします。主な機能は以下のとおりです。

- モートの INDICATOR_0 LED の切り替え（サポートされるのは SmartMesh IP モートのみです。

SmartMesh WirelessHART モートの toggle led フィールドで ON ボタンをクリックすると、ボタン・テキストが「N.A.」に変わります）。

- 各モートから受信したデータ・パケット数（*notifData*）の監視
- 各モートから受信したヘルス・レポート数（*notifHr*）の監視
- パケットのレイテンシ情報の監視
- 温度の取得と、任意のモートでの温度送信レートの設定

このアプリケーションの用途は以下のとおりです。

- モートの LED 切り替えによる、不定期のダウンストリーム・トラフィックの生成（SmartMesh IP モートのみのサポート）
- 温度送信レートの設定による、連続的なアップストリームの生成

TempMonitor は内部で、Dust が開発したアプリケーション・プロトコルの OAP を使用する、モート上の小さいアプリケーションと相互作用します。このモート常駐アプリケーションは、モートが**マスター・モード**で稼働しているときのみ有効です。

7.5.17 Upstream

概要

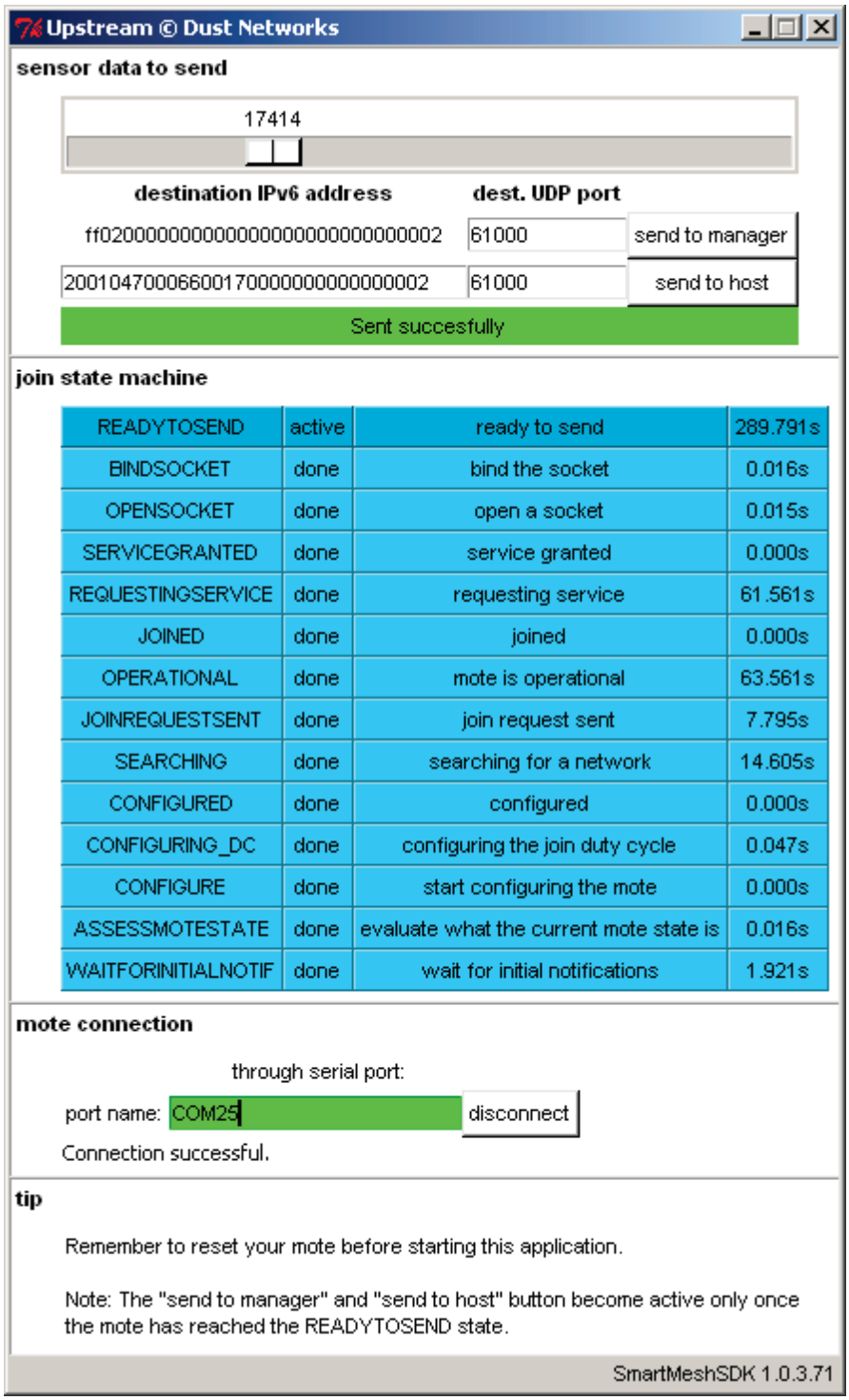
Upstream アプリケーションは、SmartMesh IP モートのステート・マシンを、ジョイン、サービス要求、ソケット・バインディングへと遷移させます。ユーザーは、ダミーの「センサ」データをアプリケーションに入力して、SmartMesh IP マネージャまたは
インターネット・ホストに送信できます。

実行方法

Upstream アプリケーションの実行方法は以下のとおりです。

- Windows 実行可能ファイル（*/win/Upstream.exe*）をダブルクリックする
- ソース・ファイル（*/src/bin/Upstream/Upstream.py*）をダブルクリックする

説明


Upstream © Dust Networks
sensor data to send
17414
destination IPv6 address
dest. UDP port
ff020000000000000000000000000002
61000
send to manager
20010470006600170000000000000002
61000
send to host
Sent succesfully
join state machine
READYTOSEND active ready to send 289.791s
BINDSOCKET done bind the socket 0.016s
OPENSOCKET done open a socket 0.015s
SERVICEGRANTED done service granted 0.000s
REQUESTINGSERVICE done requesting service 61.561s
JOINED done joined 0.000s
OPERATIONAL done mote is operational 63.561s
JOINREQUESTSENT done join request sent 7.795s
SEARCHING done searching for a network 14.605s
CONFIGURED done configured 0.000s
CONFIGURING_DC done configuring the join duty cycle 0.047s
CONFIGURE done start configuring the mote 0.000s
ASSESSMOTESTATE done evaluate what the current mote state is 0.016s
WAITFORINITIALNOTIF done wait for initial notifications 1.921s
mote connection
through serial port:
port name: COM25
disconnect
Connection successful.
tip
Remember to reset your mote before starting this application.
Note: The "send to manager" and "send to host" button become active only once the mote has reached the READYTOSEND state.
SmartMeshSDK 1.0.3.71

Upstream アプリケーションは以下の 3 つのフレームで構成されています。

- **mote connection** フレームで、アプリケーションを SmartMesh IP モートに接続します。
- **join state machine** フレームには、モートのジョインプロセスにおける最新ステートが表示されます（各ステートについては下記を参照）。右端の列には、各ステートの経過時間が表示されます。
- **sensor data to send** フレームでは、ユーザーが 16 ビットの値（ダミーのセンサ・データ）を選んで、マネージャまたはインターネット・ホストにデータを送信できます。

ジョインステート

⚠ SmartMesh IP モートが以下の全ステートを遷移できるようにするには、Upstream アプリケーションに接続する前にモートをリセットする必要があります。リセットするには、以下のいずれかの方法を使用します。

- APIExplorer アプリケーションを使用して *reset* コマンドを送信する
- CLI コマンド *reset* を使用する
- SmartMesh IP モートの電源をいったん切ってから再度オンにする

ステート名	概要
WAITFORINITIALNOTIF	SmartMesh IP モートがリセットされた直後の *BOOT* 通知などの、予想される通知を待機します。通知を受け取ったかどうかに関係なく、このステートは規定のタイムアウト後にも残ります。
ASSESSMOTESTATE	現在のモート・ステートを評価します。 ● モートのステータスを取得して、稼働中である場合は *READYTOSEND* ステートに進みます。 ● このモートで実行中のサービスを取得して、実行中のサービスがある場合は *READYTOSEND* ステートに進みます。 ● 上記以外は *CONFIGURE* ステップに進みます。
CONFIGURE	移行中のステート。
CONFIGURING_DC	ジョインのデューティ・サイクルを 100%に設定します。
CONFIGURED	移行中のステート。このステートは、SmartMesh IP モートが構成済みで、ネットワークへのジョイン準備が整っていることを示します。
SEARCHING	*join* コマンドを発行し、ネットワークからアドバタイズメントを受け取るまで待機します（*JOINSTART* イベント通知）。
JOINREQUESTSENT	SmartMesh IP モートが稼働中になるまで待機します（*OPERATIONAL* イベント通知）
OPERATIONAL	ベース帯域幅が機能するまで待機します（*SVC_CHG* イベント通知）。
JOINED	移行中のステート。このステートは、SmartMesh IP モートネットワークにジョイン済みであることを示します。
REQUESTINGSERVICE	マネージャに対してサービスを要求します。サービスのインストールが通知されるまで待機します（*SVC_CHG* イベント通知）。

ステート名	概要
SERVICEGRANTED	移行中のステート。このステートは、SmartMesh IP モートがサービスを取得しており、データ送信を開始できることを示します。
OPENSOCKET	UDP ソケットを開きます。
BINDSOCKET	ソケットを UDP ポート 60000 にバインドします。
READYTOSEND	ジョインプロセスの最終ステート。

センサ・データの送信

sensor data to send フレームは、モートが *READYTOSEND* ステートになっているときのみ有効になります。モートがこのステートになったらセンサ値を設定して、以下のいずれかを実行できます。

- **sendto manager** ボタンを押して、マネージャに送信する
- IPv6 アドレスを指定してから **send to host** ボタンを押して、インターネット上のホストに送信する

マネージャに送信する場合、マネージャの既知の IPv6 アドレス(*ff02::2*、
省略しない場合は *ff020000000000000000000000000002*)にデータが送信されます。

このアプリケーションは、マネージャに接続された SensorDataReceiver アプリケーションと組み合わせて使用されることを想定しています。

データをインターネットに送信するためには、LBRConnection アプリケーションを使用して、マネージャを Low-power Border Router に接続する必要があります。

7.5.18 Xively

概要

センサ・データを Xively に送信し、変更をサブスクライブします。

Xively（https://xively.com/）は、センサ・データを送信できるクラウドベースのサービスです。Xively で提供される多数のライブラリを使用すると、Xively に保存されたデータの視覚化と相互作用を実行するアプリケーションを簡単に開発できます。

SmartMesh SDK に含まれる Xively サンプル・アプリケーションは、以下の処理を実行します。

- **マスター・モード**でデフォルト・ファームウェアを実行する SmartMesh IP モートによって生成された温度データを送信します。これにより、https://xively.com/、または独自開発した Xively 対応アプリケーションで、データを時系列に表示できます。
- **マスター・モード**でデフォルト・ファームウェアを実行する SmartMesh IP モートの LED に対応するデータストリームをサブスクライブします。これにより、https://xively.com/、または独自開発した Xively 対応アプリケーションから、評価モート上の LED を作動させることができます。

実行方法

Xively アプリケーションの実行方法は以下のとおりです。

- Windows 実行可能ファイル（/win/Xively.exe）をダブルクリックする
- ソース・ファイル（*/src/bin/Xively/Xively.py*）をダブルクリックする（Windows 以外の OS では追加の手順が必要になる場合あり）

説明

Xively サンプル・アプリケーションは SmartMesh IP マネージャに接続して、温度データをサブスクライブします。Xively サービスに情報を送信できるようにするには、Xively Master API Key を入力する必要があります。

はじめに、http://www.xively.com で Xively の開発者アカウントを作成します。

アカウントを作成したら、http://www.xively.com にログインして Master API Key を作成します。

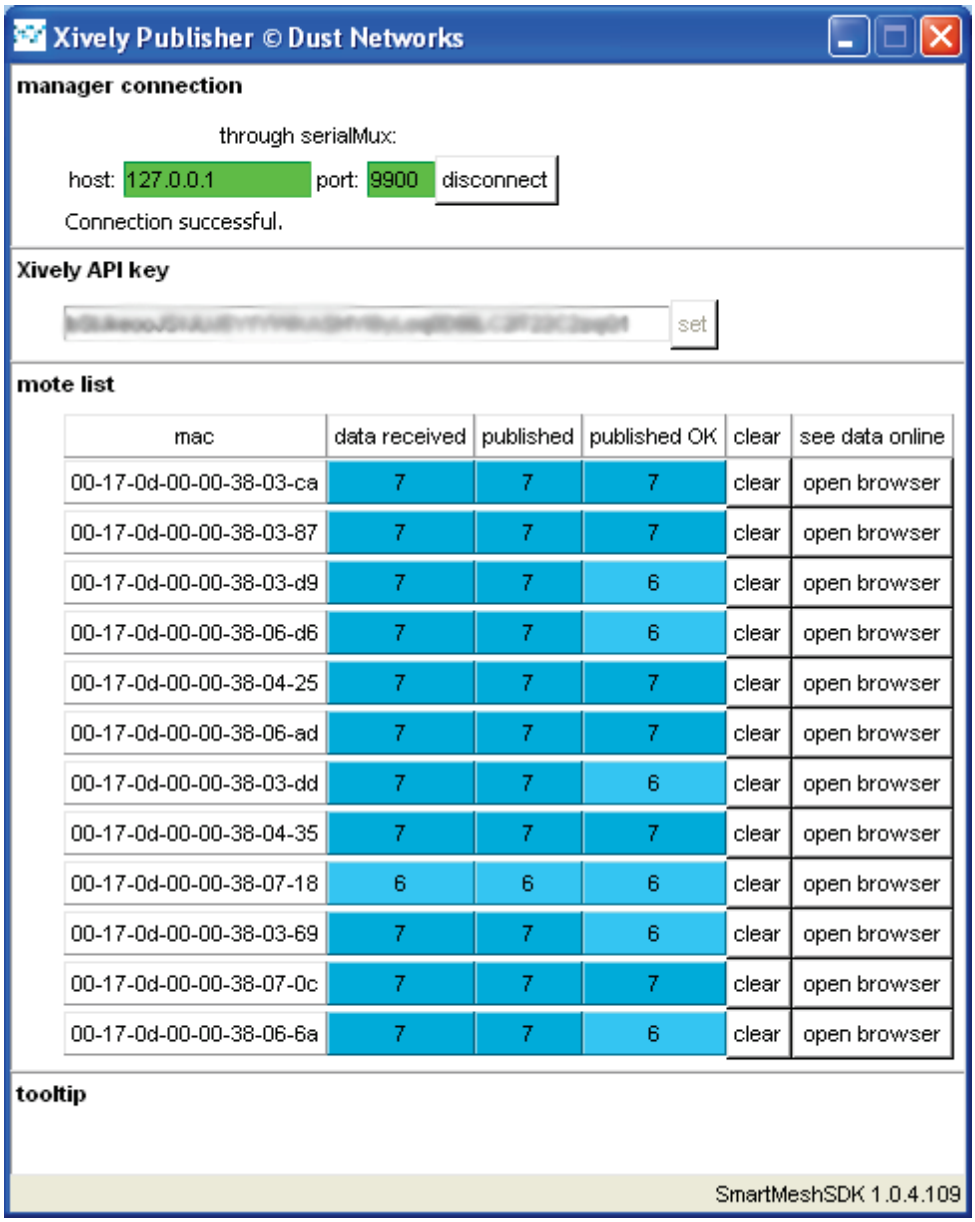
- Web Tools ドロップダウン・メニューから Settings を選択します。
- Settings セクションで Master Keys を選択します。
- Add Master Key をクリックします。
- キー・タイトルを入力して全てのチェック・ボックス（*READ*、*CREATE*、*UPDATE*、*DELETE*）と Access Private フィールドを選択します。

Master Key をコピーします。

⚠ 使用する SmartMesh IP ネットワークで、1 つ以上の SmartMesh IP モートがマスター・モードで
デフォルト・ファームウェアを実行していることを確認します。

Xively にデータを送信する手順は以下のとおりです。

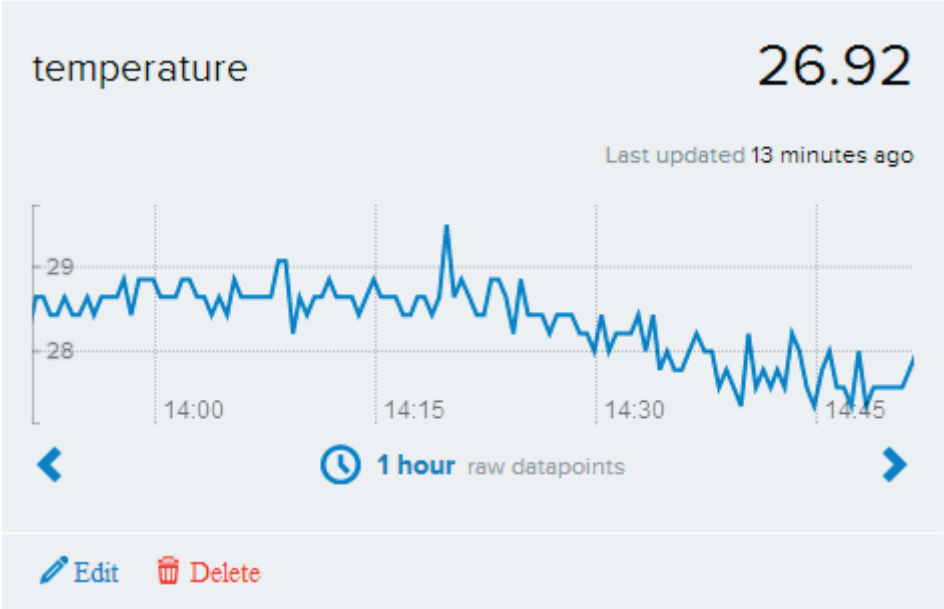
- Xively サンプル・アプリケーションを起動します。
- SerialMux またはシリアル・ポート経由で、SmartMesh IP マネージャに Xively サンプル・アプリケーションを接続します。
- ネットワーク内のモートが自動的にリストに表示されます。
- 上記手順で作成した Xively Master API Key を **Xively API key box** に入力し、**set** をクリックします。
- モートが温度を送信するたびに、以下の処理が実行されます。
  - **data received** カウンタが増加します。
  - **published** カウンタが増加し、Xively サンプル・アプリケーションが Xively にデータを送信しようとしていることが示されます。
  - **published OK** カウンタが増加し、送信に成功したことが示されます。



データの表示手順は以下のとおりです。

- http://www.xively.com にログインして、**Web Tools→Manage** に移動します。
- 「SmartMesh IP Starter Kit」という名前の新製品が作成されました。

- この製品をクリックすると、SmartMesh IP ネットワーク内のデバイスがリストに表示されます。
- それぞれのデバイスに、以下の 2 つのデータストリームが関連付けられています。
- 「temperature」データストリームには、送信された温度が時系列に格納されています。



- 「led」データストリームは、モートの LED を作動させるために使用されます。

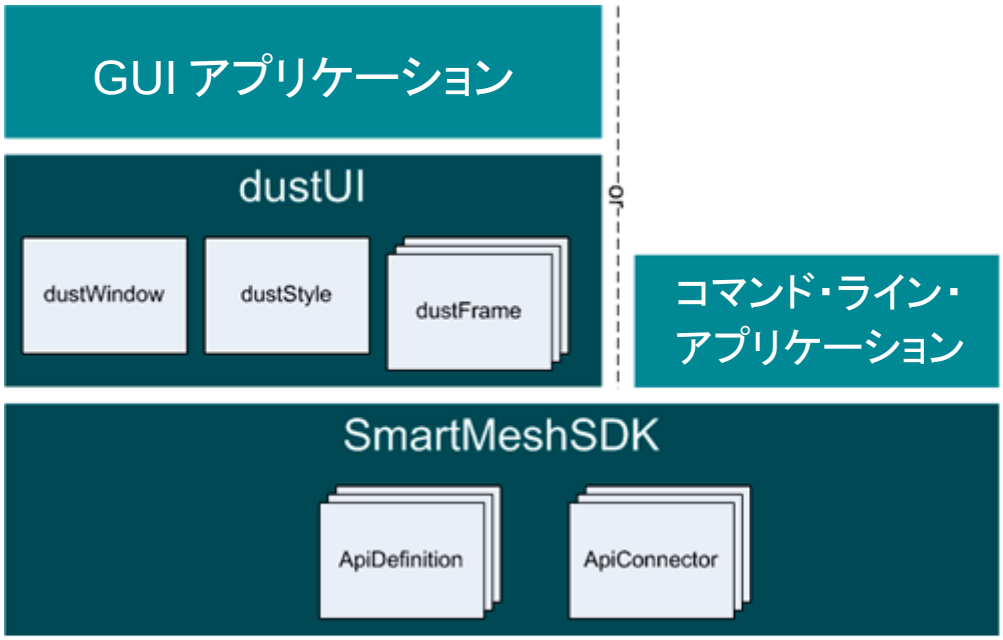
モートの LED を作動させる手順は以下のとおりです。

> ⚠ モートに **LED_EN** ジャンパが装着されていることを確認します。

- http://www.xively.com から、モートのデバイス・ページで LED データストリームをクリックします。
- ボード上の LED スイッチがオンになります。
- 上記手順を繰り返して **0** を入力すると、LED スイッチがオフになります。

7.6 アーキテクチャ

7.6.1 概要

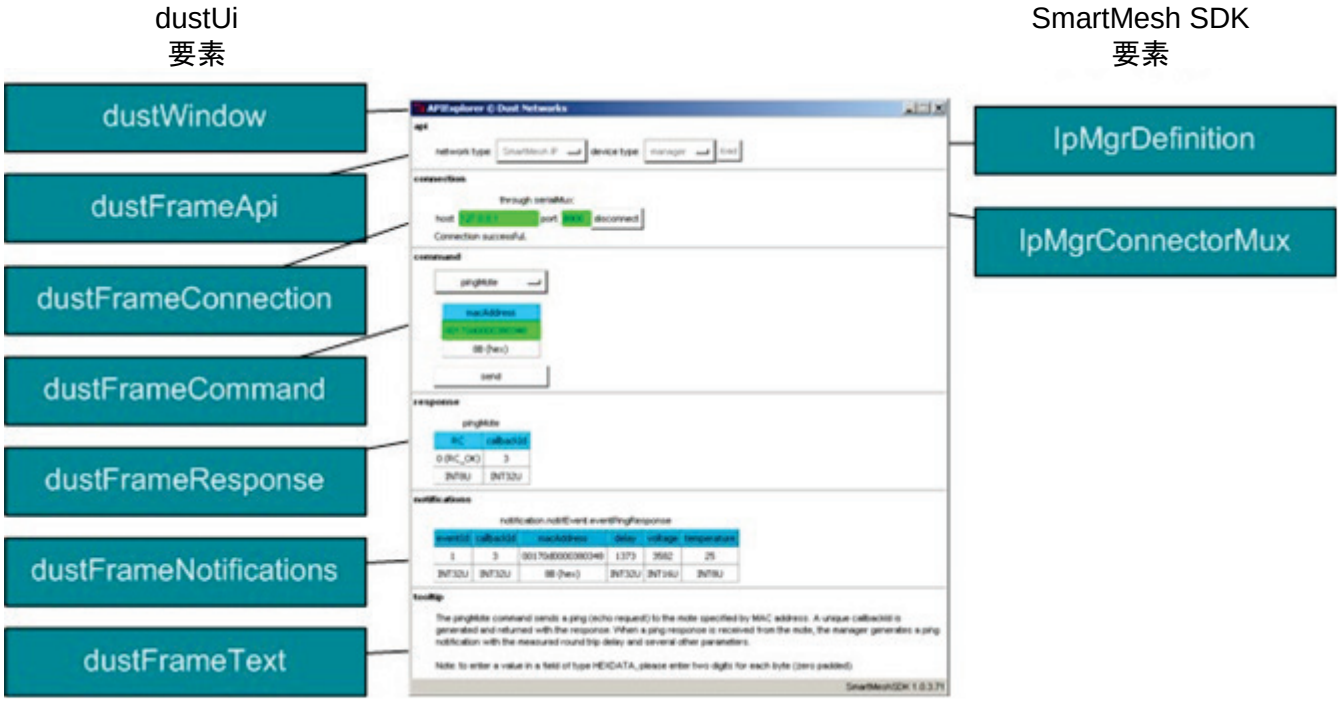


SmartMesh SDK は以下の 3 つのレイヤで構成されています。

- **SmartMeshSDK** レイヤには、一連の ApiDefinition と ApiConnector が含まれています。
  - **ApiDefinition** には、特定の API のコマンド、レスポンス、通知と、これらを操作する関数を定義されています。
  - **ApiConnector** は、何らかの送信メカニズムを介して物理デバイスに接続するために使用されます。
- **dustUI** レイヤは、GUI アプリケーションの構築に使用できるビジュアル要素のライブラリで、以下の要素で構成されています。
  - **dustWindow** は、アプリケーションのメイン・ウィンドウを表します。
  - **dustStyle** は共通のスタイルシートであり、dustUI ライブラリを介して使用されます。
  - 一連の **dustFrame** は、GUI 要素です。
- 上記を使用して、以下の 2 種類のアプリケーションを構築できます。
  - SmartMeshSDK レイヤ上に直接配置される**コマンド・ライン・アプリケーション**
  - 複数の dustFrame 要素から 1 つのウィンドウを構成し、各要素を相互接続する **GUI アプリケーション**

7.6.2 APIExplorer アプリケーションの構成例

下図に、APIExplorer アプリケーションの内部構成を示します。



メイン・アプリケーションが各種のフレームを管理します。APIExplorer アプリケーションの実行手順は以下のとおりです。

1. ユーザーが **api** フレームで API 定義を選択します。この定義（ここでは *IpMgrDefinition*）がメイン・アプリケーションに返され、別のフレームに渡されます。
2. ユーザーが **connection** フレームで API 接続を選択します。この接続（ここでは *IpMgrConnectorMux*）がメイン・アプリケーションに返され、別のフレームに渡されます。
3. **command** フレームがロードされた API 定義を調べて、コマンドを含むドロップダウン・メニューを動的に構築します。
4. **command** フレームは、デバイスへのコマンドの送受信用に作成されたコネクタを使用します。
5. メイン・アプリケーションがレスポンスを受け取り、表示用に **response** フレームに送ります。
6. メイン・アプリケーションが通知を受け取り、グローバル変数に格納します。**notifications** フレームは定期的にグローバル変数をポーリングして、その内容を表示します。
7. 有用な情報がある場合、メイン・アプリケーションは **tooltip** フレームにこれを表示します。

> ⚠ dustFrame 要素が互いを呼び出すことはありません。フレーム間での情報の受け渡しはメイン・アプリケーションの役割です。

7.7 dustUI ライブラリ

7.7.1 概要

dustUI ライブラリは一連の GUI 要素で構成されており、これらを組み合わせて 1 つのアプリケーションが形成されます。このライブラリは、Python のデファクト・スタンダード GUI パッケージである Tkinter をベースとしています。
dustUI ライブラリ内の各モジュールは、SmartMesh SDK インストール・ディレクトリの */src/ SmartMeshSDK/dustUI/*フォルダ内にあります。

7.7.2 モジュールの概要

以下で説明するモジュールは、アプリケーションに組み込むことを想定して設計されていますが、**スタンドアロン・アプリケーションとして実行**することもできます。ソース・ファイルをダブルクリックすると、ダミー・アプリケーションが表示され、GUI 要素がどのように表示されるかを確認できます。

dustStyle.py

このモジュールは、全ての dustUI GUI 要素のルックアンドフィールを設定します。このファイルを編集するだけで、アプリケーションの外観を簡単にカスタマイズできます。

dustWindow.py

このモジュールは各アプリケーションのウィンドウであり、以下の要素を含んでいます。

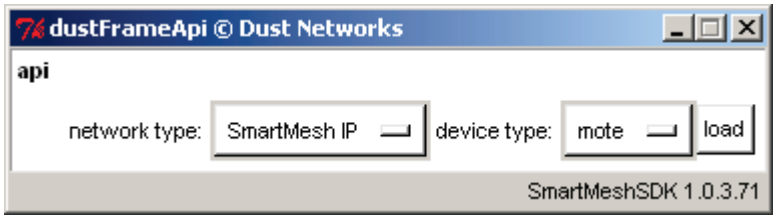
- Dust Networks のロゴと著作権記号

- アプリケーション名（最上部に表示）
- SmartMesh SDK のバージョン（最下部に表示）

dustFrame.py

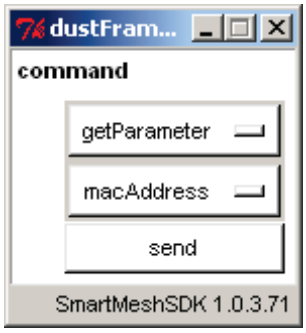
これは全ての dustFrame の親クラスであり、継承を目的として作成されています。

dustFrameApi.py



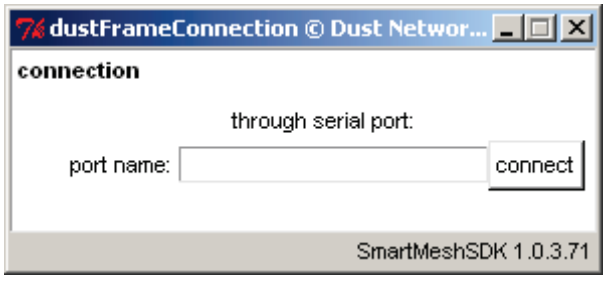
ロードする API 定義を指定します。

dustFrameCommand.py



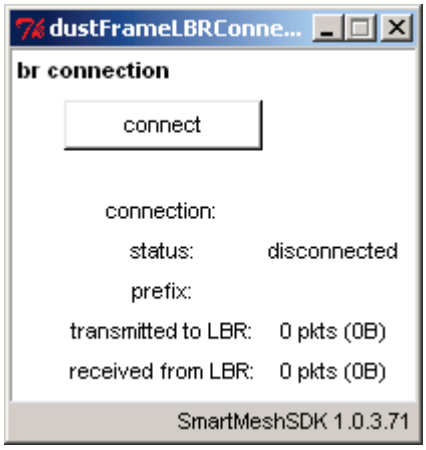
使用可能なコマンドを表示して送信します。

dustFrameConnection.py



コネクタ・タイプを選択し、デバイスに接続します。

dustFrameLbrConnection.py



LBR に接続します。

dustFrameFields.py

親クラスです。

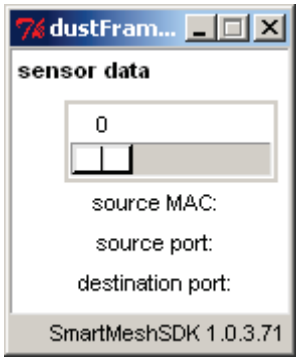
dustFrameResponse.py

レスポンスに含まれるフィールドを表示します。

dustFrameNotifications.py

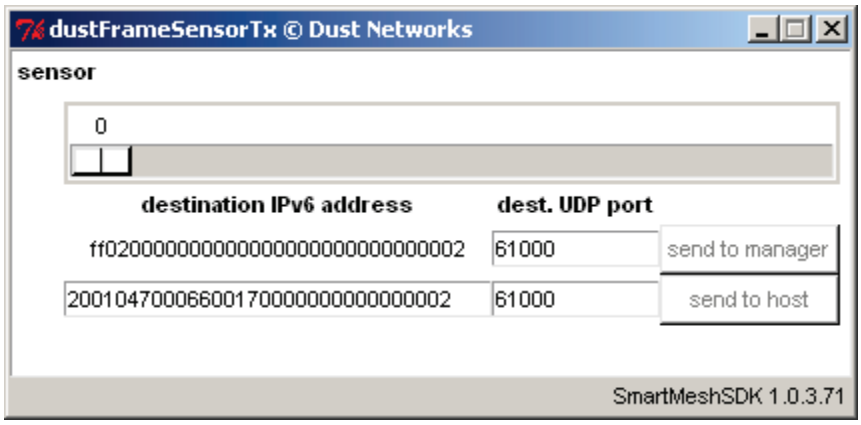
通知に含まれるフィールドを表示します。

dustFrameSensorData.py



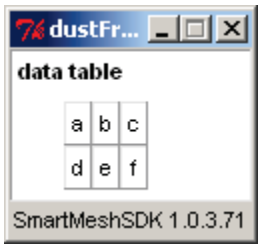
センサ・データを表示します。

dustFrameSensorTx.py



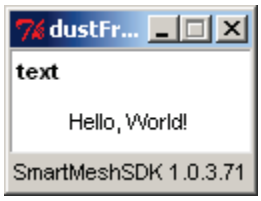
センサ・データを送信します。

dustFrameTable.py



2 次元表として、リストのリストを表示します。

dustFrameTooltip.py



任意のテキストを表示します。

7.8 SmartMeshSDK ライブラリ

7.8.1 概要

SmartMeshSDK ライブラリに含まれる一連のモジュールは、全ての Dust Networks デバイスに接続され、関連する全ての アプリケーション・プログラミング・インタフェース（API）を実装します。

SmartMeshSDK は、SmartMesh SDK インストール・ディレクトリの*/src/SmartMeshSDK/*フォルダ内にあります。

SmartMeshSDK ライブラリは、以下の 2 種類のモジュールで構成されています。

- 特定のデバイスの API を定義する API 定義
- 物理的なデバイス接続を可能にする API コネクタ

7.8.2 モジュールの概要

API 定義

全ての API 定義は、*/src/SmartMeshSDK/ApiDefinition/*ディレクトリ内にあります。

全ての API 定義の親クラスとなるのが *ApiDefinition.py* であり、このクラスから以下のモジュールが継承されます。

モジュール名	API 定義の対象デバイス
IpMgrDefinition.py	SmartMesh IP マネージャ
IpMoteDefinition.py	SmartMesh IP モート
HartMgrDefinition.py	SmartMesh WirelessHART マネージャ
HartMoteDefinition.py	SmartMesh WirelessHART モート

API コネクタ

全ての API コネクタの親クラスとなるのが *ApiConnector.py* であり、このクラスから以下のモジュールが継承されます。

モジュール名	接続先デバイス	接続プロトコル
IpMgrConnectorMux	SmartMesh IP マネージャ	SerialMux
IpMgrConnectorSerial	SmartMesh IP マネージャ	シリアル
IpMoteConnector	SmartMesh IP モート	シリアル
HartMgrConnector	SmartMesh WirelessHART マネージャ	XML-RPC
HartMoteConnector	SmartMesh WirelessHART モート	シリアル

8 ネットワークの対話操作

8.1 概要

このセクションにはさまざまなチュートリアルが含まれており、それぞれが、SmartMesh IP ネットワークが持つ特定の側面に焦点を合わせています。チュートリアルを実行するために、ワイヤレス・センサ・ネットワークや Dust Networks 製品に関する予備知識は不要です。

基本チュートリアル:

- 「初期のネットワーク_bookmark86」では、デフォルト設定でネットワークを起動し、Stargazer GUI アプリケーションを使用してネットワーク形態を監視します。
- 「マネージャの対話操作」では、SmartMesh IP マネージャにログインし、APIExplorer アプリケーションを使用して、SmartMesh IP マネージャと接続済みの SmartMesh IP モートに関する情報を取得します。
- 「モートの対話操作」では、APIExplorer アプリケーションを使用して SmartMesh IP モートを操作し、外部のマイクロコントローラの動作を再現します。モートをネットワークにジョインさせて、SmartMesh IP マネージャにデータを送信します。

上級者向けトピック:

- 「スクリプトによる API の実行」では、グラフィカル・ユーザー・インタフェースではなくスクリプトから SmartMesh SDK を使用する方法を紹介します。
- 「HDLC フレームのロギング」では、SmartMesh SDK のロギング機能を使用して、SmartMesh IP モートのシリアル接続経由で送信されるデータ・ストリームを確認します。
- 「アップストリーム通信」では、Upstream アプリケーションを実行します。このアプリケーションは SmartMesh IP モートのステート・マシンをジョイン、サービス要求、ソケット・バインディングへと遷移させ、ユーザーが SmartMesh IP マネージャにデータを送信できるようにします。
- 「ダウンストリーム通信」では、APIExplorer アプリケーションを使用して SmartMesh IP マネージャから SmartMesh IP モートにデータを送信します。
- 「インターネット統合」では、SmartMesh IP マネージャを Low-power Border Router に接続し、SmartMesh IP モートがインターネット上のコンピュータと直接データを交換できるようにする方法について説明します。

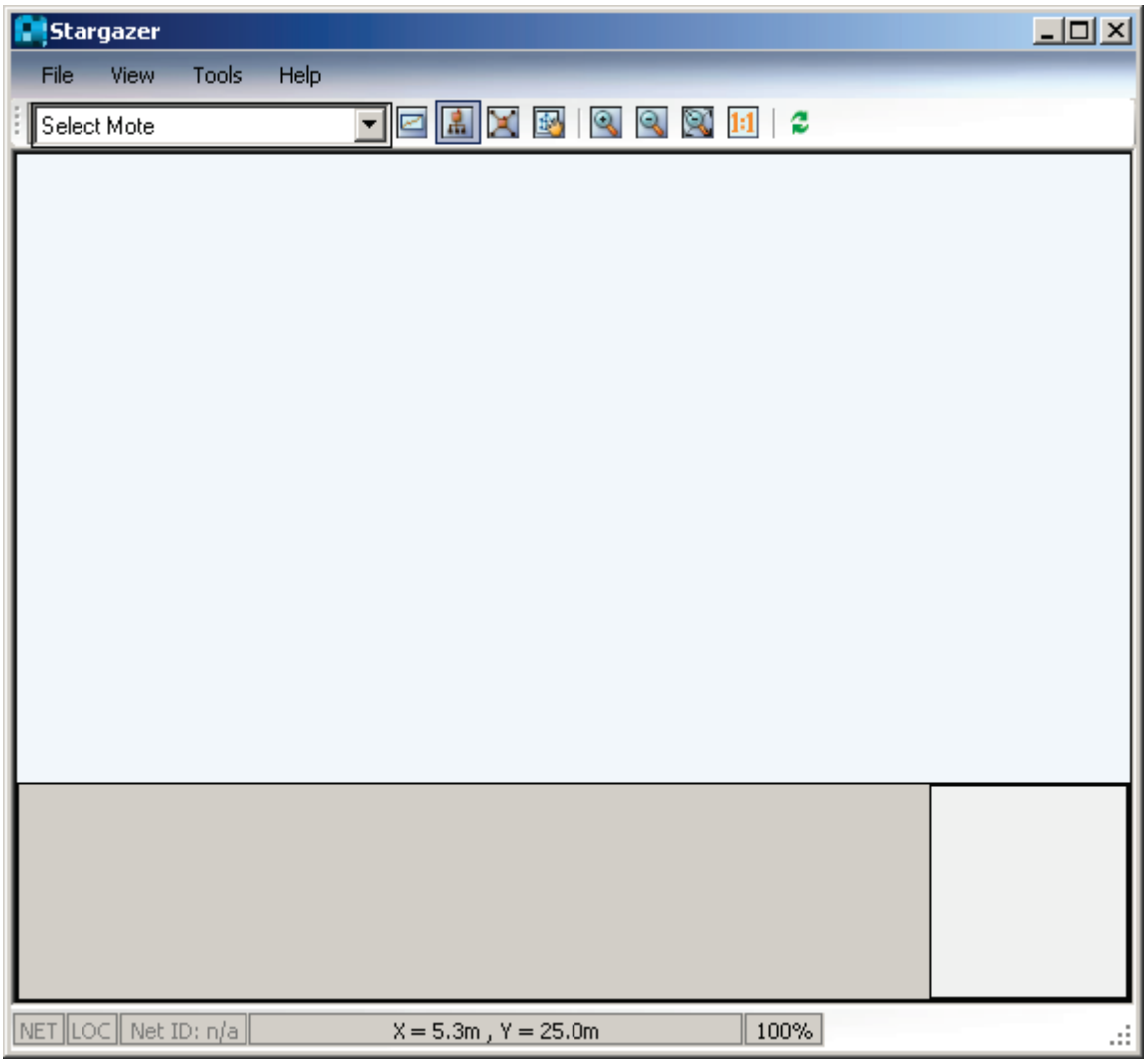
8.2 初期のネットワーク

8.2.1 概要

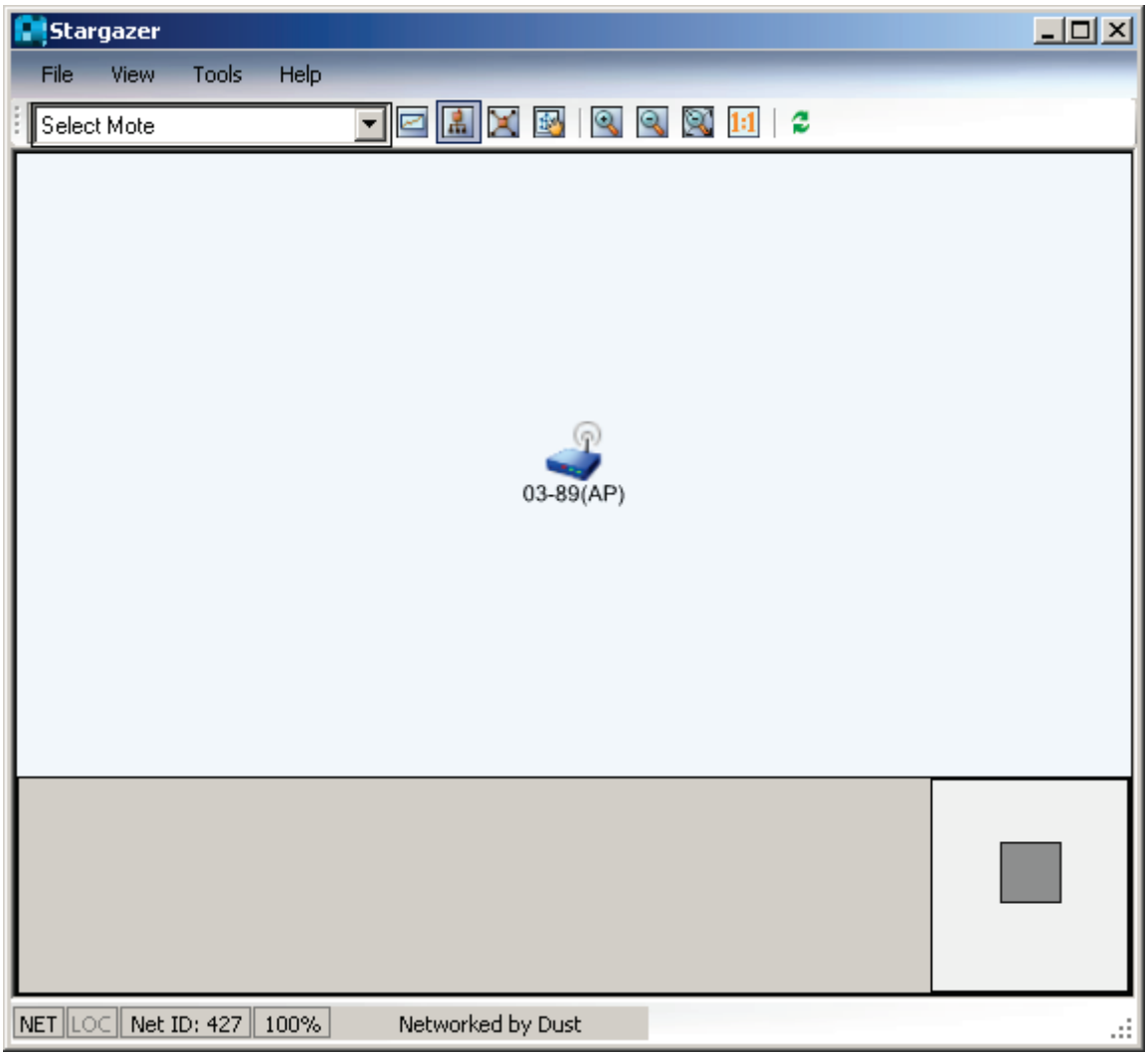
このチュートリアルでは、Stargazer GUI と SmartMesh IP マネージャを使用して、メッシュ・ネットワークの形態を監視します。本書の前半に記載したインストール手順が完了していることが前提となります。

ネットワークの構築

1. 全てのモートおよびマネージャの電源を切ります。
2. SmartMesh IP マネージャをコンピュータに接続します。
3. Stargazer アプリケーションを起動します。以下のウィンドウが表示されます。

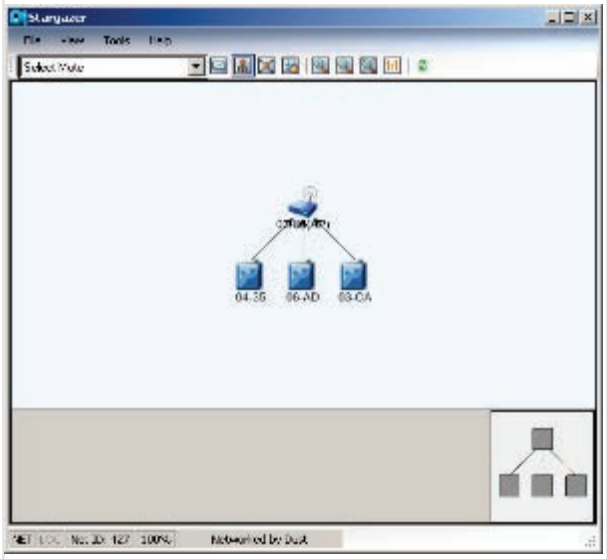


4. SmartMesh IP マネージャの電源を入れると、Stargazer ウィンドウに表示されます。

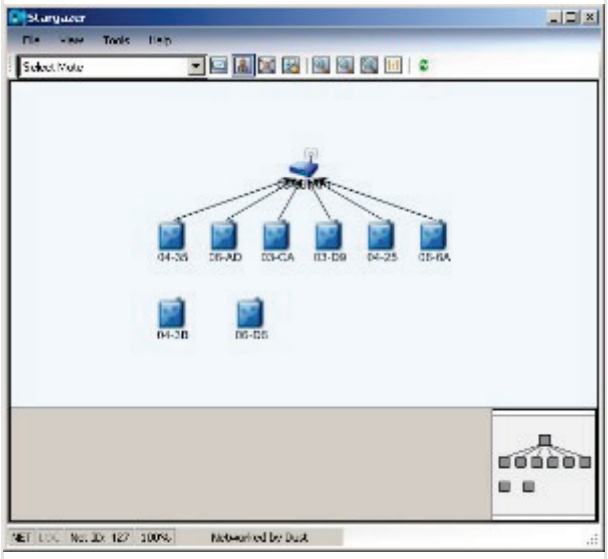


5. SmartMesh IP 各モートの電源を入れます。どのスイッチを先に入れても問題ありません。SmartMesh IP モートそれぞれがジョインすると、ネットワークが自動的に構成されます。ネットワークの構築状況を監視します。

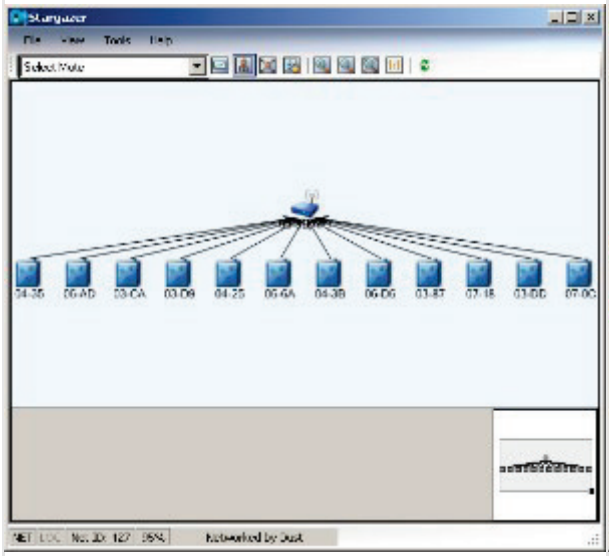
モート数:3



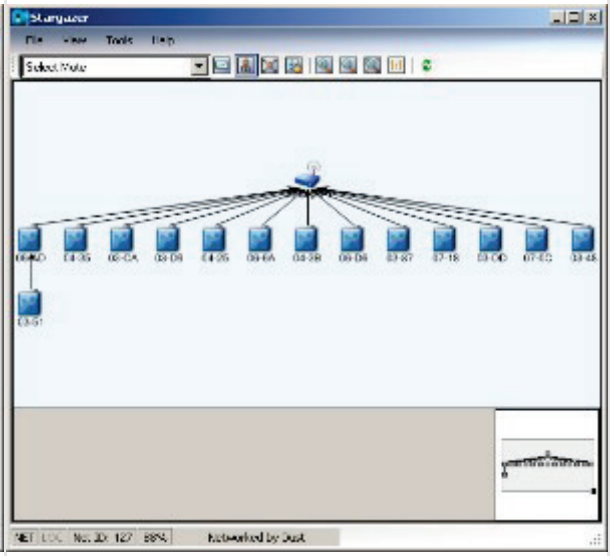
モート数:8



モート数:12



モート数:14

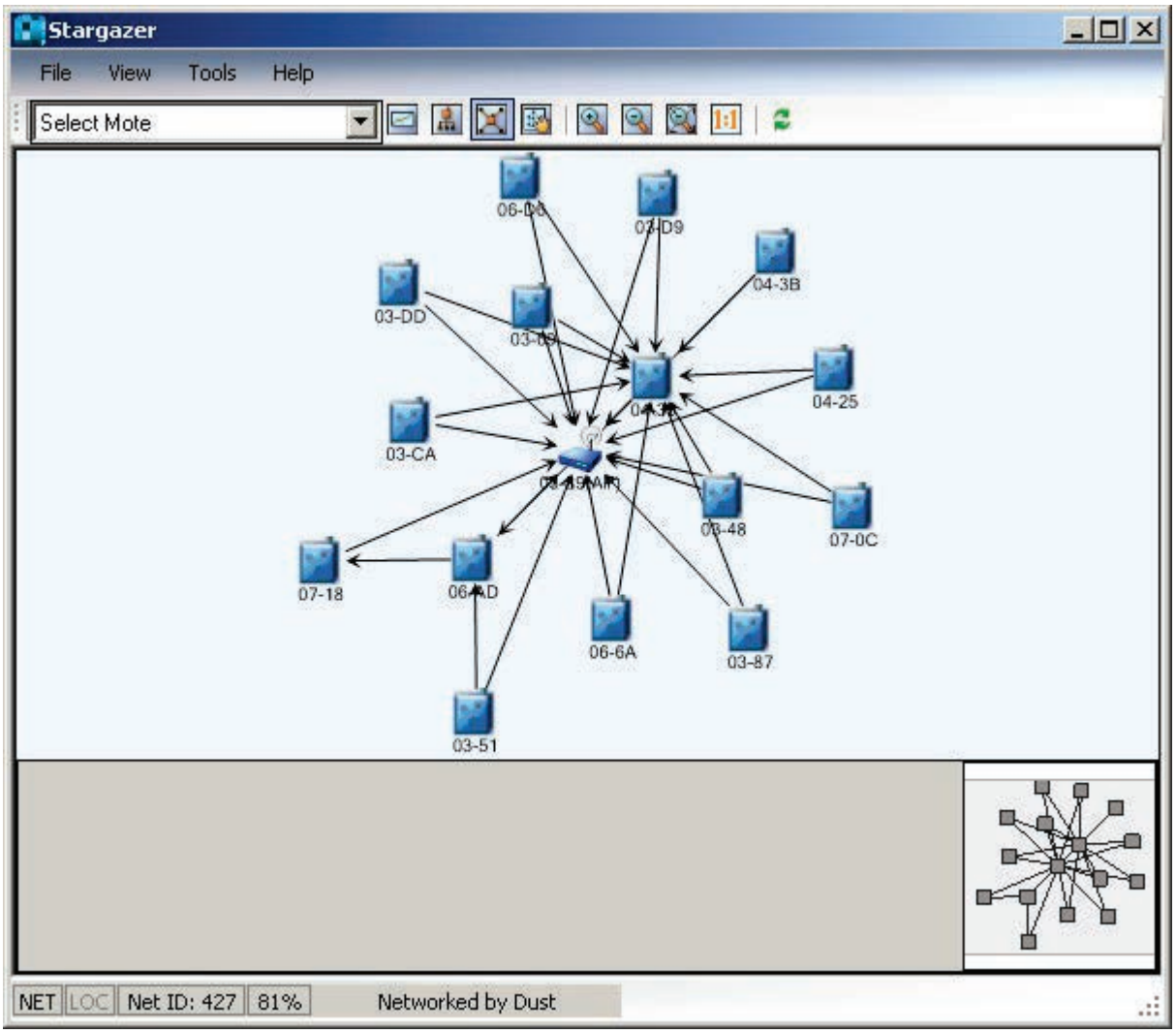


APIExplorer や TempMonitor、またはその他の SmartMesh SDK サンプル・アプリケーションで使用するために、モートをスレーブ・モードに変更している場合は、**マスター・モード**に戻します。

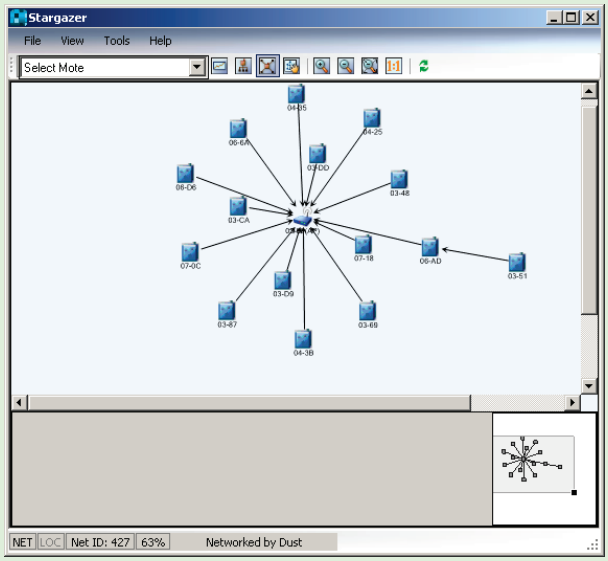
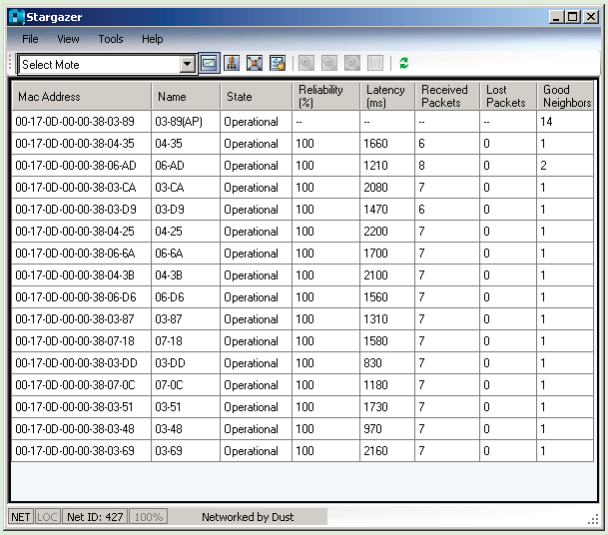
```
> set mode master
> reset
```

⚠ スタータ・キット（DC9021A）に含まれるモートは、出荷時に**マスター・モード**に設定されています。モートの各モードとその切り替え方法については、SmartMesh IP ユーザー・ガイドを参照してください。スタータ・キットにはモートが 5 つ含まれていますが、マネージャには 32 まで追加できます。マネージャに外部 RAM が装着されている場合、最大 100 まで追加できます。

6. 全てのノードがジョインしたら、SmartMesh IP マネージャによる最適化ルールの適用によって、トポロジが冗長メッシュに変わる様子を観察します。



表示領域の上にあるアイコンを使用すると、ビューを切り替えることができます。

Stargazer Radio Space ビュー	Stargazer Tabular ビュー
	

Stargazer の GUI について、詳しくは「Stargazer の使用」を参照してください。

以上で、SmartMesh IP ネットワークのパフォーマンスを評価する準備が整いました。推奨事項については、アプリケーション・ノート「How to Evaluate Network and Device Performance」を参照してください。

8.2.2 一般的な問題

Stargazer にモートがまったく表示されない

- SmartMesh IP マネージャの電源は入っていますか。
  Stargazer は SmartMesh IP マネージャと通信することで、ネットワーク情報を取得します。
  SmartMesh IP マネージャの電源が入っていない場合、Stargazer は情報を表示できません。

- 接続しているデバイスは SmartMesh IP マネージャですか。
  接続先を確認するには、デバイスの CLI ポートに接続し（転送レート 9600bps、データ・ビット 8、パリティなし、ストップ・ビット 1、フロー制御なしに設定）、以下のコマンドを入力します。

```
> 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
>
```

  上記のような出力が表示されない場合、接続先は SmartMesh IP マネージャではないため、Stargazer には情報は表示されません。
- SerialMux がリスニングしているポートは、SmartMesh IP マネージャの API ポートですか。
  そうでない場合、SerialMux はマネージャと通信できません。
  「SerialMux の構成」セクションの手順に従って構成を確認し、必要に応じて変更します。

Stargazer に表示されないモートがある

- SmartMesh IP モートの電源は入っていますか。
- SmartMesh IP モートは**マスター・モード**になっていますか。
  モートの各モードについては、本書の「トラブルシューティング」セクションを参照するか、または SmartMesh IP ユーザー・ガイドを参照してください。
- SmartMesh IP モートに設定されたネットワーク ID は、SmartMesh IP マネージャと同じですか。
  ネットワーク ID の確認および変更手順については、本書の「トラブルシューティング」セクションを参照してください。

8.3 マネージャの対話操作

8.3.1 概要

このステップでは、SmartMesh IP マネージャ（DC2274A あるいは DC9020A / DC9020B + DC9006）とやり取りします。CLI を使用する場合はターミナル・クライアントから、API を使用する場合は APIExplorer アプリケーションから実行します。

CLI を使用した対話操作

SmartMesh IP マネージャには以下の 2 つのシリアル・ポートがあります。

- シリアル・ターミナル・ソフトウェアと直接やり取りするためのコマンド・ライン・インタフェース（CLI）ポート
- SmartMesh SDK を使用してやり取りするためのアプリケーション・プログラミング・インタフェース（API）ポート

1. SmartMesh IP マネージャの CLI ポート（4 つあるうちの 3 番目）にシリアル・ターミナル・クライアントを接続します（転送レート 9600bps、データ・ビット 8、パリティなし、ストップ・ビット 1、フロー制御なしに設定）。
2. *help* と入力すると、使用可能なコマンドのリストが表示されます。

```
> help
help <command>
Commands:
   mlog
   login
   logout
```

3. マネージャにログインすると、使用できるコマンドが増えます。

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

4. *sm* コマンド（"show motes"の短縮形）を使用すると、接続済みのモートのリストが表示されます。この時点で接続されているのは、内部のアクセス・ポイント（MoteId 1）のみです。

```
> 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. show mote コマンドを使用すると、特定のモートに関する情報が表示されます。

```
> 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. *show stat* コマンドを使用すると、ネットワークに関する統計情報が表示されます。

```
> 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
>
```

ネットワークが形成されたら、各モートとネットワーク全体に関する情報が表示されます。

7. 上記操作を終了したら、*logout* コマンドを使用してログアウトします。

```
> logout
```

CLI およびマネージャの対話操作については、以下の資料を参照してください。

- SmartMesh IP ユーザー・ガイド
- SmartMesh IP Manager CLI Guide

APIExplorer を使用した API による対話操作

SmartMesh SDK について

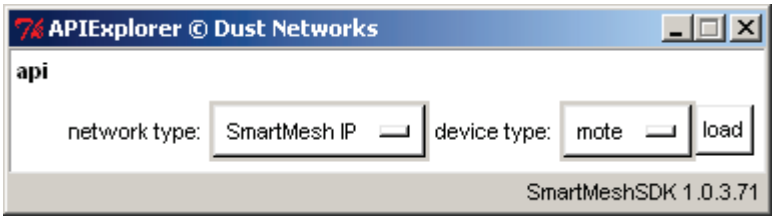
SmartMesh SDK は、SmartMesh IP または SmartMesh WirelessHART ネットワークのアプリケーション統合を簡素化する Python パッケージです。このパッケージには、接続先デバイスのアプリケーション・プログラミング・インタフェース（API）が実装されています。SmartMesh SDK に含まれた一連のサンプル・アプリケーションを利用することで、プログラマは速やかに API を理解し、大規模システムの一部として使用できるようになります。

「セットアップ」セクションでインストールした SerialMux は、Windows サービスです。SerialMux はバックグラウンドで実行され、SmartMesh IP マネージャの API ポートをリスニングします。複数のアプリケーションが TCP ソケットを介して SerialMux に接続することで、SmartMesh IP モートの API ポートを共有します。

このセクションでは、SerialMux 経由で SmartMesh IP マネージャに接続し、SmartMesh SDK を使用して API による対話操作を実行します。

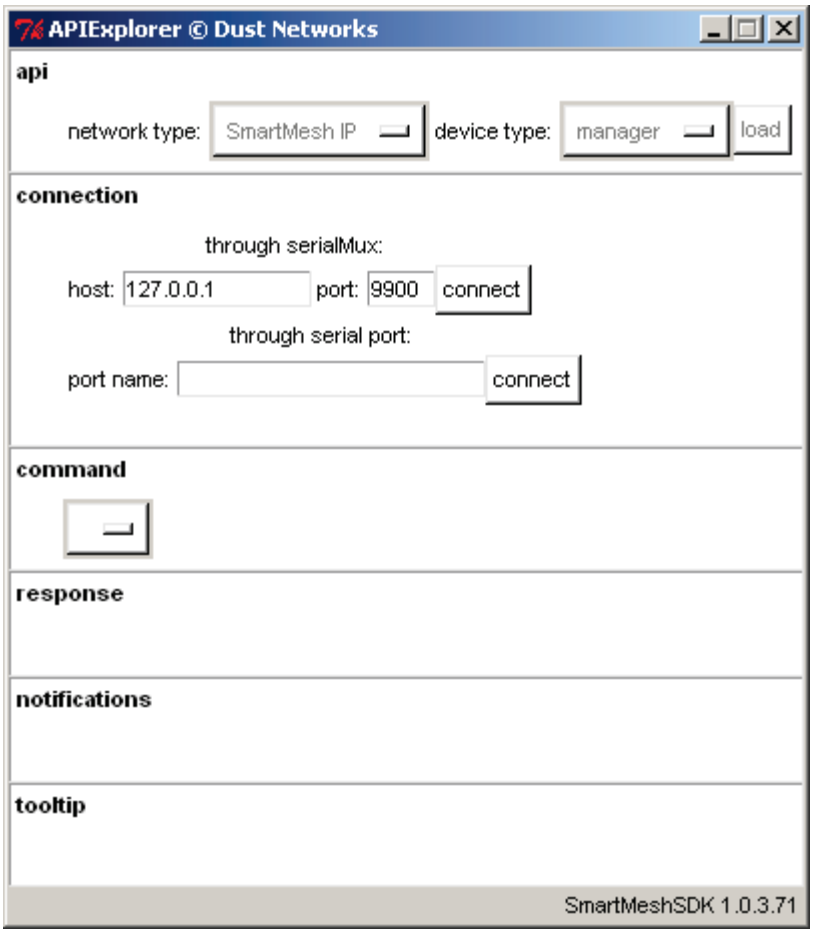
マネージャへの接続

1. SmartMesh IP マネージャがコンピュータに接続されていることを確認します。
2. *SmartMeshSDK* ディレクトリにある *win/APIExplorer.exe* プログラムをダブルクリックします。APIExplorer のウィンドウが開きます。



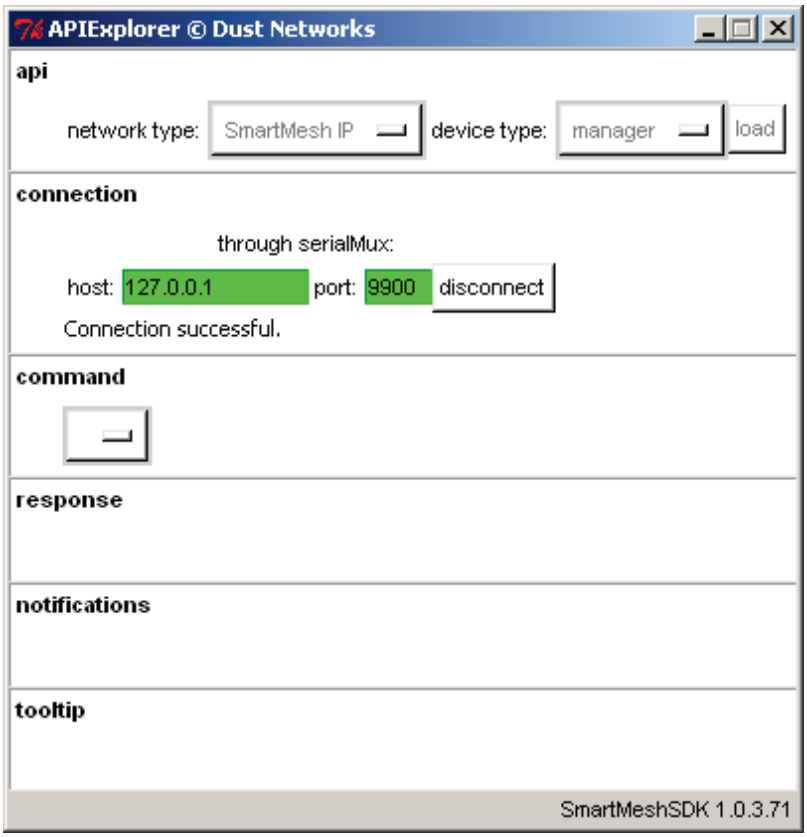
3. SmartMesh IP マネージャに接続するための情報を、以下のとおりに選択します。
   - *network type*:*SmartMeshIP*
   - *device type*:*manager*

4. **load** ボタンをクリックします。



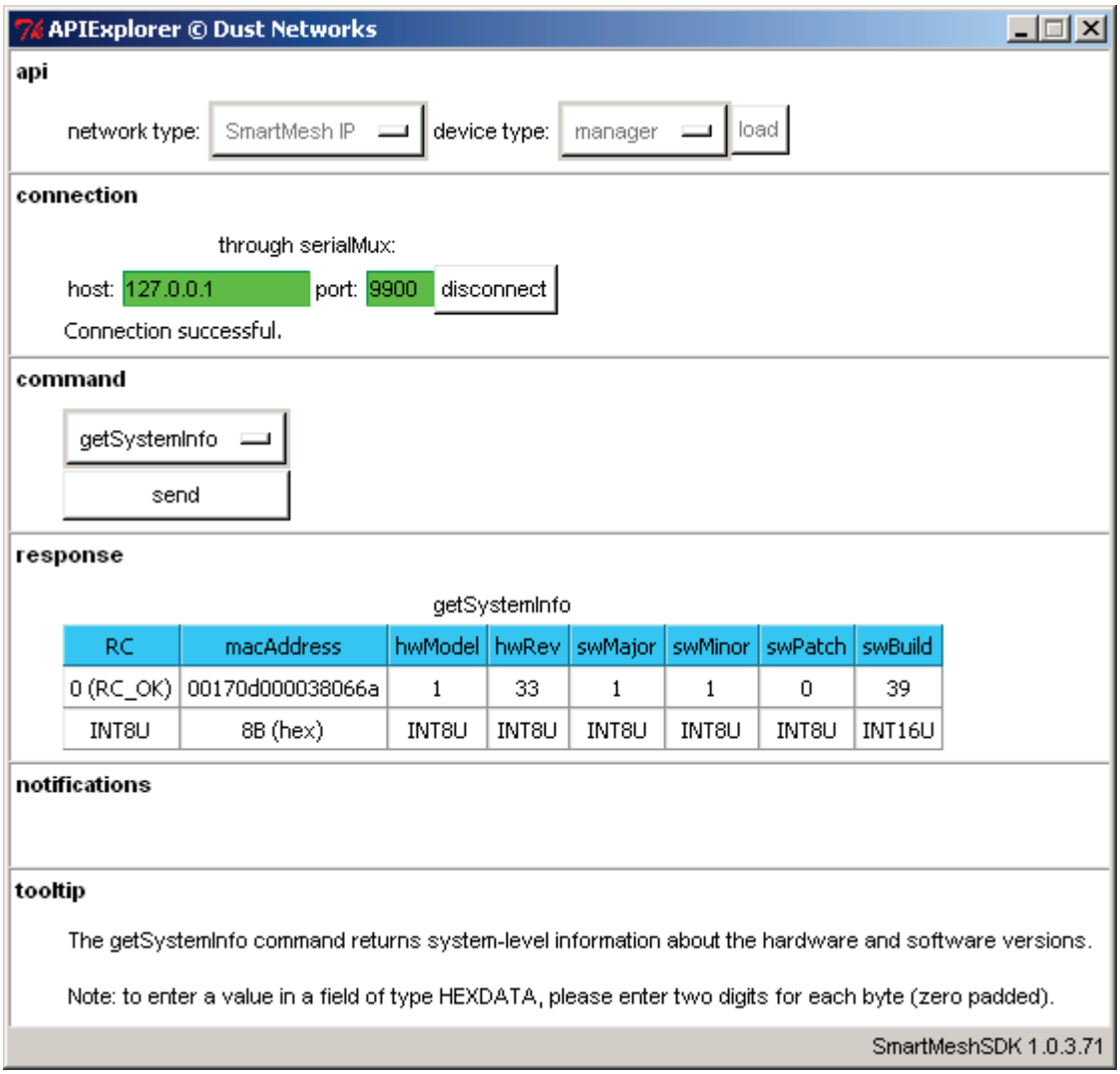
5. **connection** フレームには SmartMesh IP マネージャへの接続オプションが 2 つ表示されています。ここでは、SerialMux を選択します。
   connection フレームの SerialMux セクションで、以下を入力します。
   - *host*:*127.0.0.1*
   - *port*:9900

6. connect ボタンをクリックします。接続が成功するとフレームのフィールドが緑色になります。



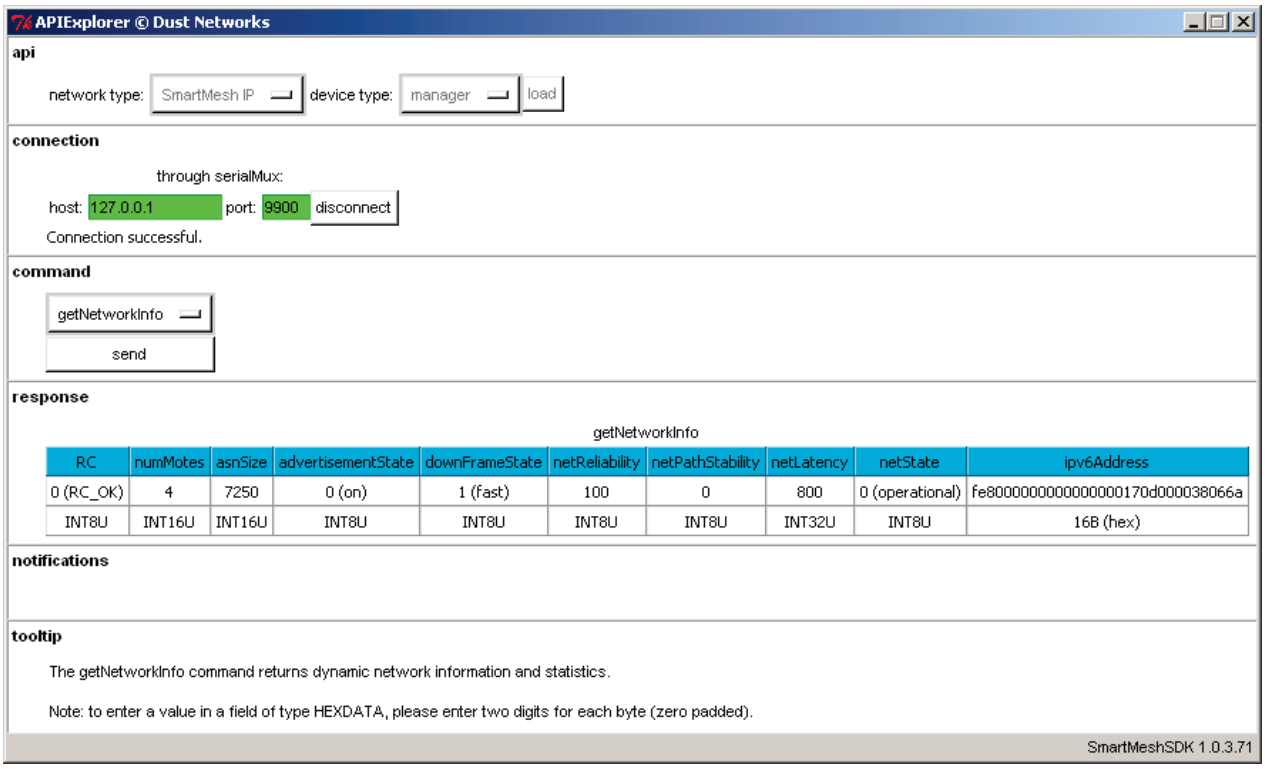
マネージャおよびネットワーク情報の取得

1. **command** フレーム内のドロップダウン・メニューには、SmartMesh IP Manager API Guide で定義されたコマンドが全て含まれています。*getSystemInfo* を選んで **send** をクリックします。各フィールドの名前、値、形式を含むレスポンスが、**response** フレームに表示されます。



2. 上のイメージには以下の情報が表示されています。
   - SmartMesh IP マネージャの MAC アドレス：*00:17:0d:00:00:38:06:6a*
   - SmartMesh IP マネージャのハードウェアおよびソフトウェアに関する情報

1. 同様に、*getNetworkInfo* コマンドを使用すると、マネージャに接続したメッシュ・ネットワークに関する情報が表示されます。



2. 上のイメージには以下の情報が表示されています。
   - マネージャに接続されたモートの数（**numMotes** フィールド）
   - スロットの長さ：7.25ms（**asnSize** フィールド）

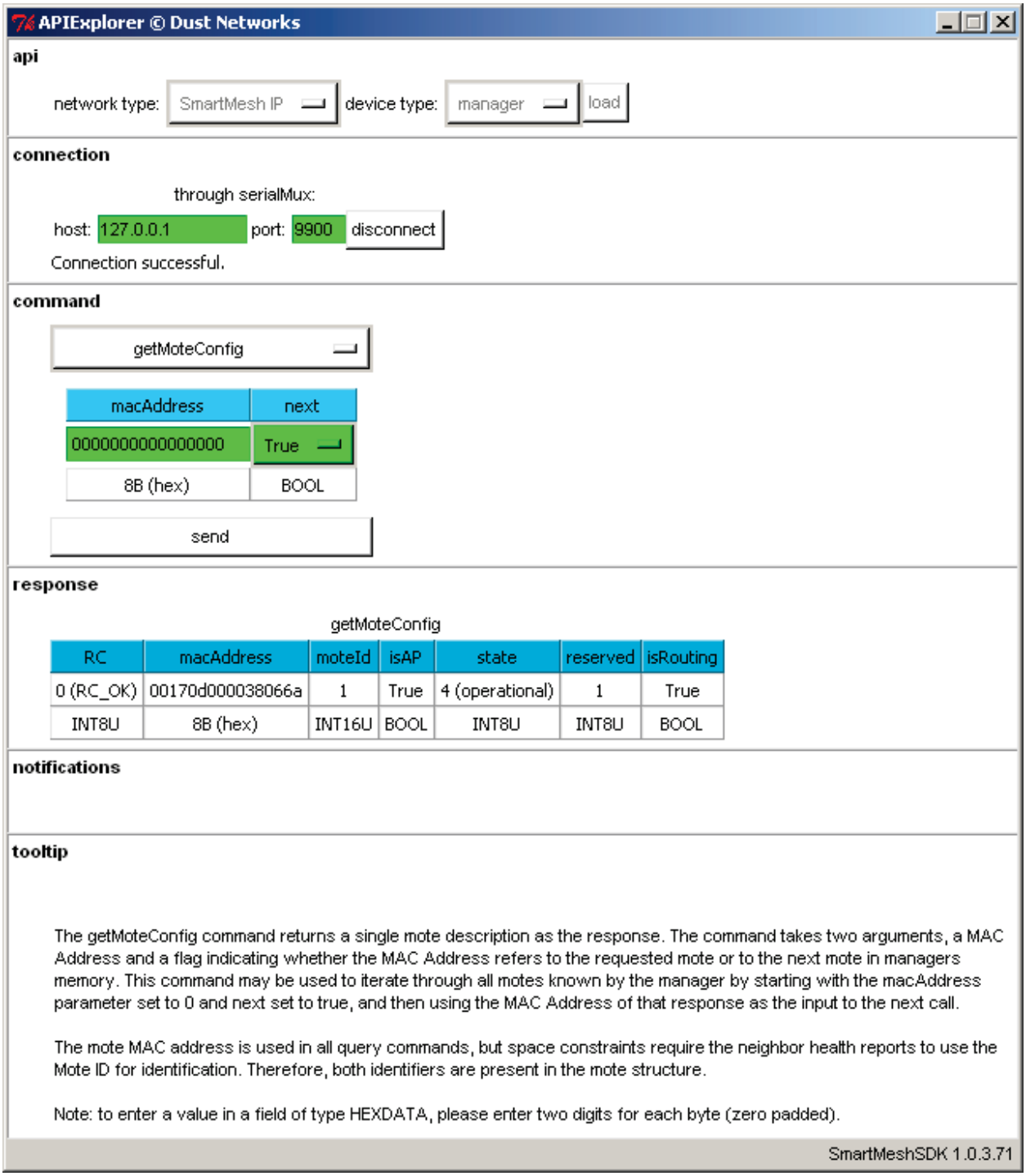
コマンドおよびフィールドの一覧については、SmartMesh IP Manager API Guide を参照してください。

モート情報の取得

1. *getMoteConfig* コマンドを選択して、以下のとおりに入力します。

   - **MAC** フィールド：*0000000000000000*
   - **next** フィールド：*True*

1. **send** ボタンをクリックすると、ネットワーク内の最初のデバイスに関する情報がマネージャから返されます。これは SmartMesh IP マネージャ自体のモートで、*アクセス・ポイント*(AP)と呼ばれます(**isAP** フィールドが *1* になっています)。



2. **macAddress** フィールドの値を書き留めておきます。このフィールドの値は AP の MAC アドレスであり、ここでは *00170d000038066a* になっています。

表内の任意のフィールドを**右クリック**すると、**コピー**を実行できます。

編集可能フィールドを**右クリック**すると、**貼り付け**を実行できます。

通知のサブスクライブ

ここまでのセクションでは、SmartMesh IP マネージャにコマンドを送ることで、即座にレスポンスが得られました。これ以外に、ユーザーが問い合わせを実行しなくても、イベントが発生したタイミングで SmartMesh IP マネージャから通知を送信することができます。通知にはさまざまな種類があり、SmartMesh IP Manager API Guide で詳しく説明されています。

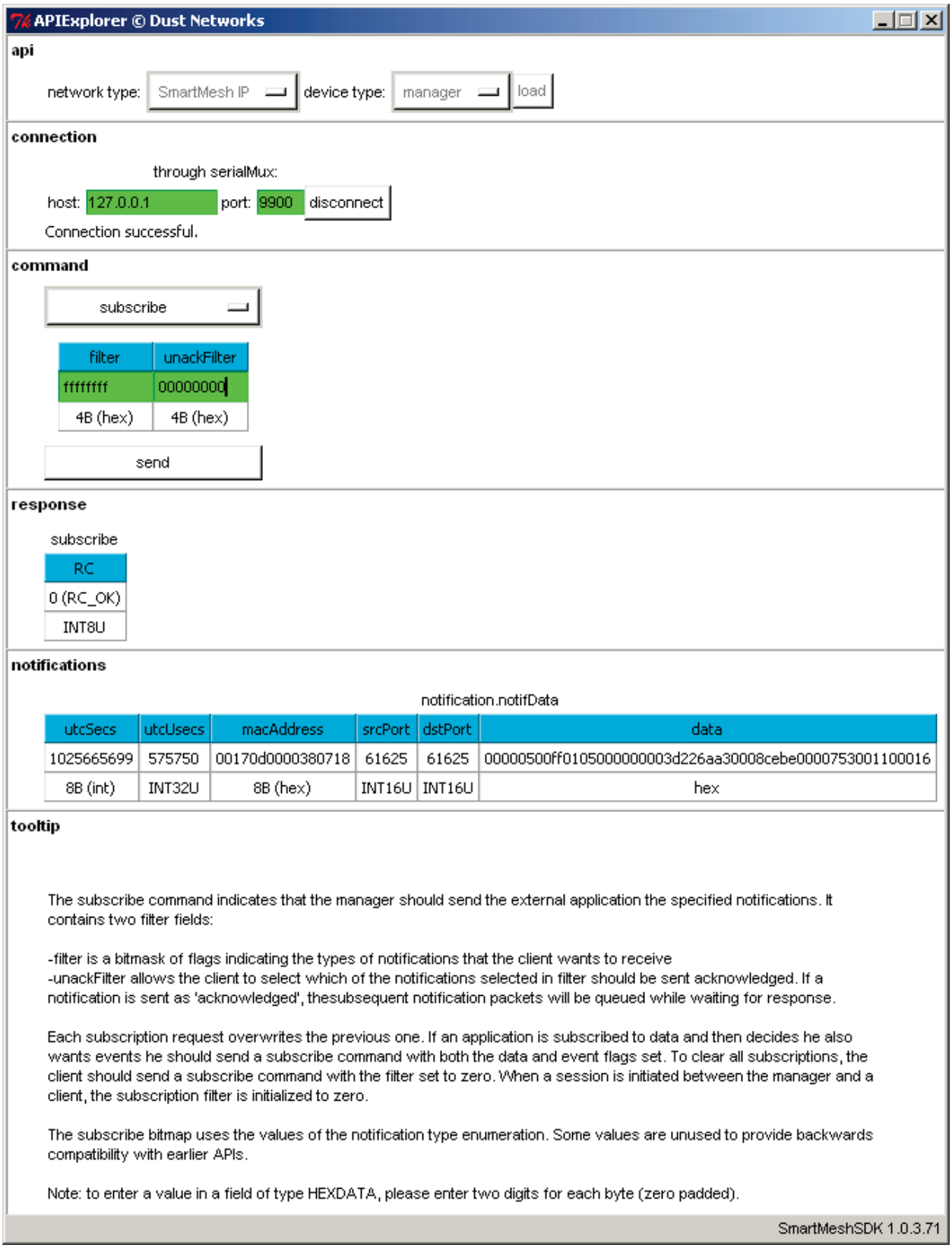
マネージャはデフォルトでは通知を*送信しません*。*subscribe* コマンドを使用して、どの通知タイプを受け取るかを指定する必要があります。

1. *subscribe* コマンドを選択して、以下のとおりに入力します。

    - **filter** フィールド: `ffffffff`
    - **unackFilter** フィールド: `00000000`

1. **send** をクリックした後は、使用可能な全ての通知が送信されます。

2. ネットワークが稼働している場合、**notifications** フレームに通知が表示されます。



参考

API およびマネージャの対話操作については、以下の資料を参照してください。

- SmartMesh IP ユーザー・ガイド
- SmartMesh IP Manager API Guide

8.3.2 一般的な問題

CLI を使用しても出力が表示されない

- デバイスの電源は入っていますか。
- シリアル・ターミナルがデバイスの CLI ポートに接続されていますか。
- シリアル・ターミナルの設定は正しいですか。

以下の表に、全てのデバイスに使用できるシリアル設定を示します。

デバイス	シリアル・ポート番号	用途	ボーレート	データ・ビット	パリティ	ストップ・ビット
SmartMesh IP マネージャ	3 番目*	CLI	9600	8	N	1
	4 番目*	API	115200**	8**	N**	1**
SmartMesh IP モート	3 番目*	CLI	9600	8	N	1
	4 番目*	API	115200**	8**	N**	1**

* FTDI ドライバによって作成されたシリアル・ポート
** デフォルト値

SerialMux 経由でマネージャに接続したが、出力が表示されない

APIExplorer から SerialMux への接続に成功しても（connection フレームのフィールドが緑色）、SerialMux から SmartMesh IP マネージャへの接続に失敗する場合があります。この場合、SerialMux がリスニングしているシリアル・ポートが、SmartMesh IP マネージャの API ポートではない可能性があります。

SerialMux の構成を変更するには、「SerialMux の構成」に記載された手順を参照してください。

8.4 モートの対話操作

8.4.1 概要

このステップでは、SmartMesh IP モート（DC9018A-B/DC9018A-B/DC9018B-B+ DC9006）とやり取りします。CLI を使用する場合はターミナル・クライアントから、API を使用する場合は APIExplorer アプリケーションから実行します。

CLI を使用したモートの対話操作

SmartMesh IP モートには以下の 2 つのシリアル・ポートがあります。

- シリアル・ターミナルと直接やり取りするための CLI ポート
- SmartMesh SDK を使用してやり取りするための API ポート

1. SmartMesh IP モートの CLI ポートにシリアル・ターミナル・クライアントを接続します（9600 ボー、データ・ビット 8、パリティなし、ストップ・ビット 1、フロー制御なしに設定）。
2. *help* と入力すると、使用可能なコマンドのリストが表示されます。

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

3. *minfo* コマンドを使用すると、SmartMesh IP モートのネットワーク関連情報が表示されます。

```
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. デフォルト（マスター・モード）では、モートは起動時に自動的にネットワークにジョインします。*reset* コマンドを使用して、モートを強制的にリセットします。数分経つと（SmartMesh IP マネージャが実行中の場合）、CLI にジョインの進捗が表示されます。SmartMesh IP モートがジョインしている間に *minfo* コマンドを使用すると、ステートの変化を確認できます。

```
> 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
```

APIExplorer を使用した API による対話操作

SmartMesh SDK

SmartMesh SDK は、センサ/アクチュエータ・デバイスへの SmartMesh IP モートの統合を簡素化する Python パッケージです。このパッケージには、OEM マイクロプロセッサがシリアル UART インタフェース経由でモートに対して実行するアプリケーション・プログラミング・インタフェース（API）コールが実装されています。SmartMesh SDK に含まれた一連のサンプル・アプリケーションを利用することで、プログラマは速やかに API を理解し、大規模システムの一部として使用できるようになります。

このセクションでは、Windows のシリアル COM ポート経由で SmartMesh IP モートに接続し、SmartMesh SDK を使用して API による対話操作を実行します。

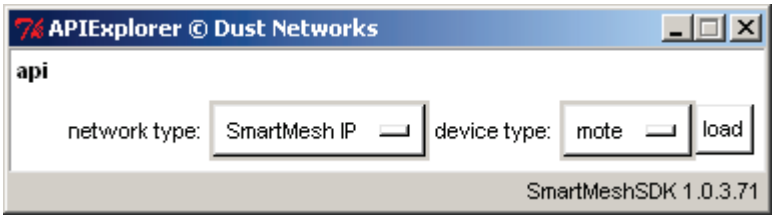
デバイスへの接続

1. SmartMesh IP モートがコンピュータに接続されていることを確認します。
   SmartMesh IP モートが**スレーブ・モード**で実行されていることを確認するため、モートの CLI で以下のコマンドを実行します。

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

> ⚠ スタータ・キット（DC9021A）に含まれるモートは、出荷時に**マスター・モード**に設定されています。
> モートの各モードと切り替え方法については、本書の「トラブルシューティング」セクションと SmartMesh IP ユーザー・ガイドを参照してください。

2. *SmartMeshSDK* ディレクトリにある *win/APIExplorer.exe* アプリケーションをダブルクリックします。
   APIExplorer のウィンドウが開きます。

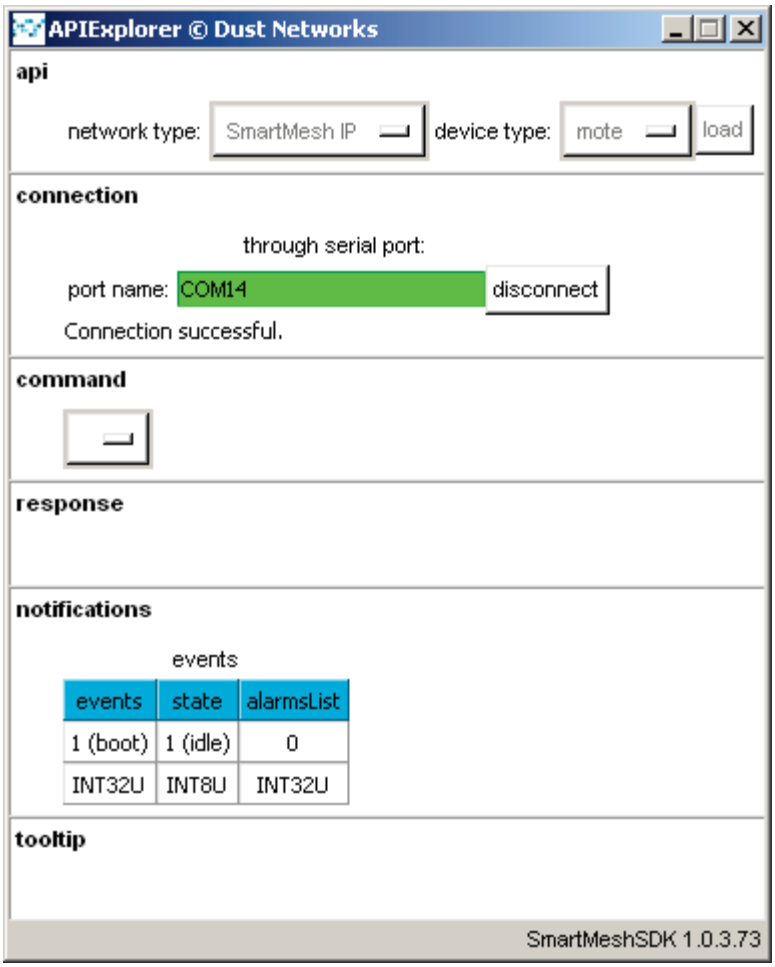


3. SmartMesh IP モートに接続するための情報を、以下のとおりに選択します。
   - *network type*:*SmartMeshIP*
   - *device type*:*mote*

4. **load** ボタンをクリックします。

5. **connection** フレームに以下を入力します。
   - *`port name`*: SmartMesh IP モートの API ポート番号

6. **connect** をクリックします。接続に成功すると、フィールドが緑色に変わります。



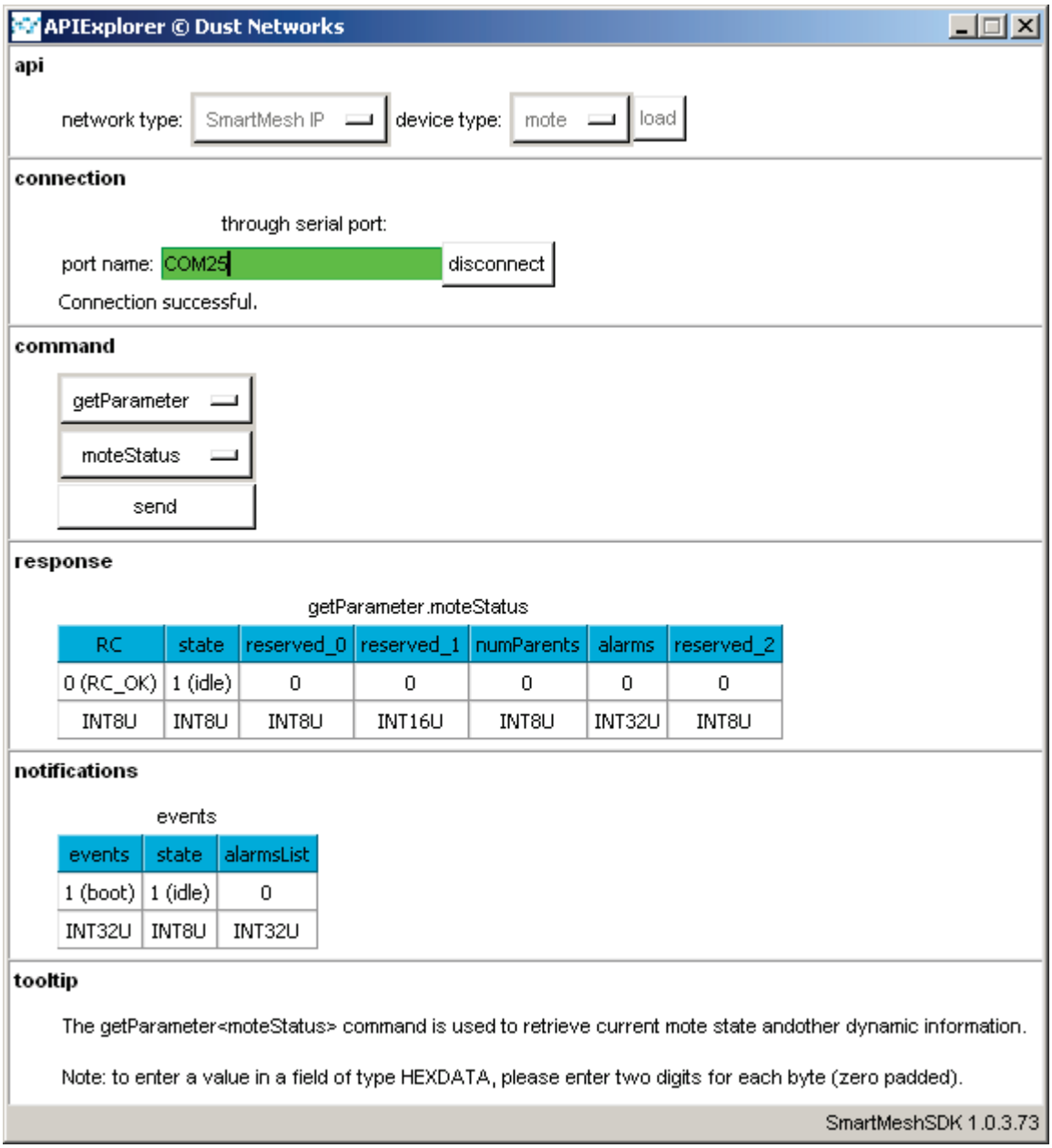
モートからの通知の取得

SmartMesh IP マネージャとは異なり、SmartMesh IP モートから通知を取得する場合、サブスクライブは必要ありません。

SmartMesh IP モートは起動されると、「ブート・イベント」通知を定期的に送信し、API シリアル・ポートをリスニングしている外部デバイス（APIExplorer アプリケーションなど）によるアクノリッジを受け取るまで続けます。上のスナップショットに表示されているのが、モート起動イベントです。

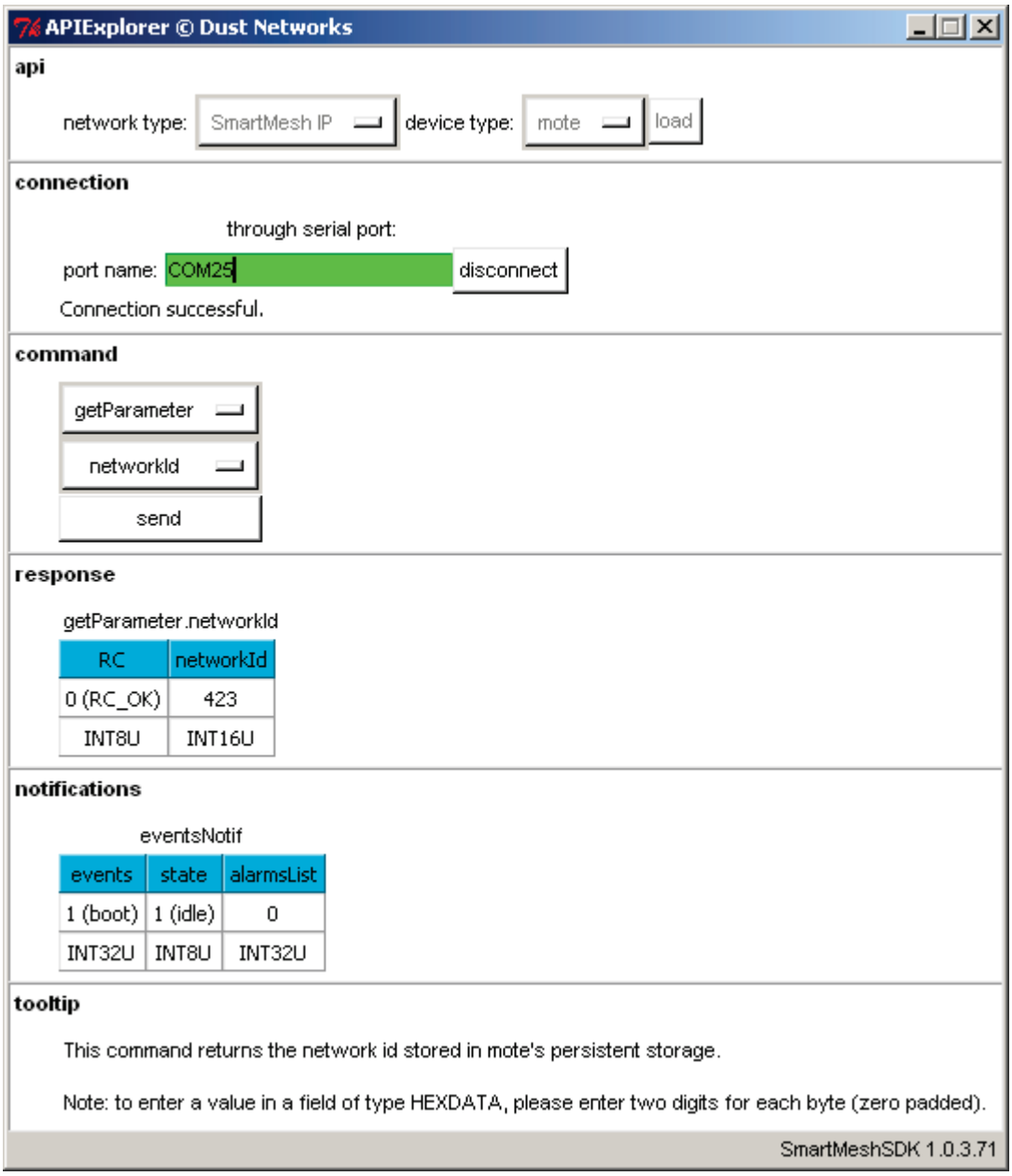
モート情報の取得

1. **command** フレーム内のドロップダウン・メニューには、SmartMesh IP Mote API Guide で定義されたコマンドが全て含まれています。*getParameter* と *moteStatus* を選択して、**send** をクリックします。各フィールドの名前、値、形式を含むレスポンスが、**response** フレームに表示されます。



2. 上のイメージでは、モートが *idle* ステートになっていることが分かります。つまり、モートはネットワークにジョインしようとしていません。

3. *getParameter.networkId* コマンドを使用して、SmartMesh IP モートに正しいネットワーク ID が設定されていることを確認します。

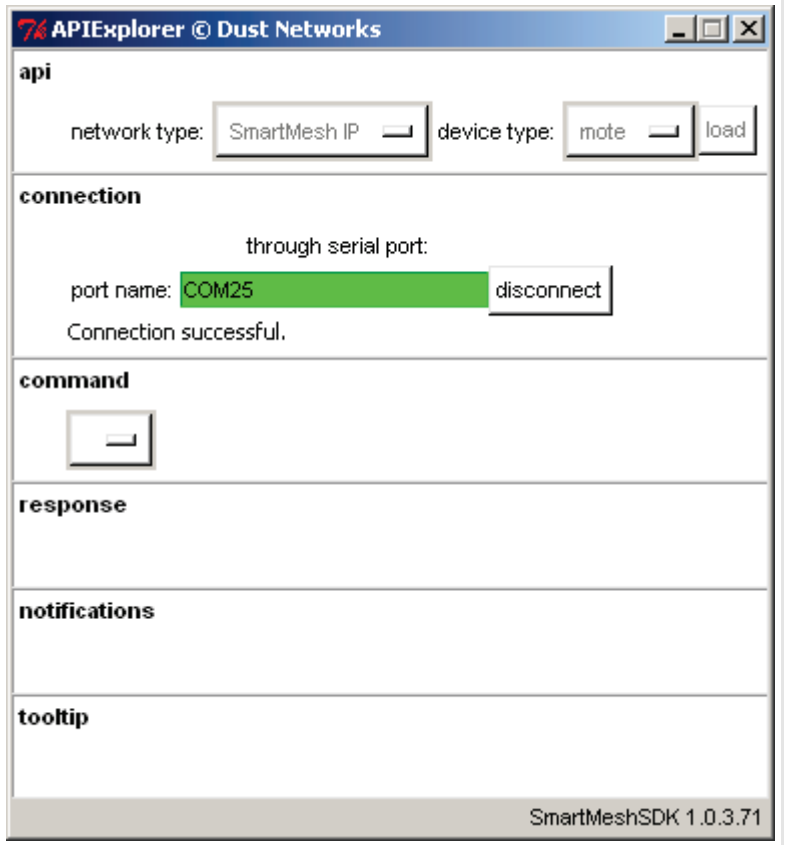


モートのネットワークへのジョイン

1. 最初の APIExplorer アプリケーションを起動して、SmartMesh IP マネージャに接続します。
2. また、2 番目の APIExplorer アプリケーションを起動して、SmartMesh IP モートに接続します（まだ起動していない場合のみ）。

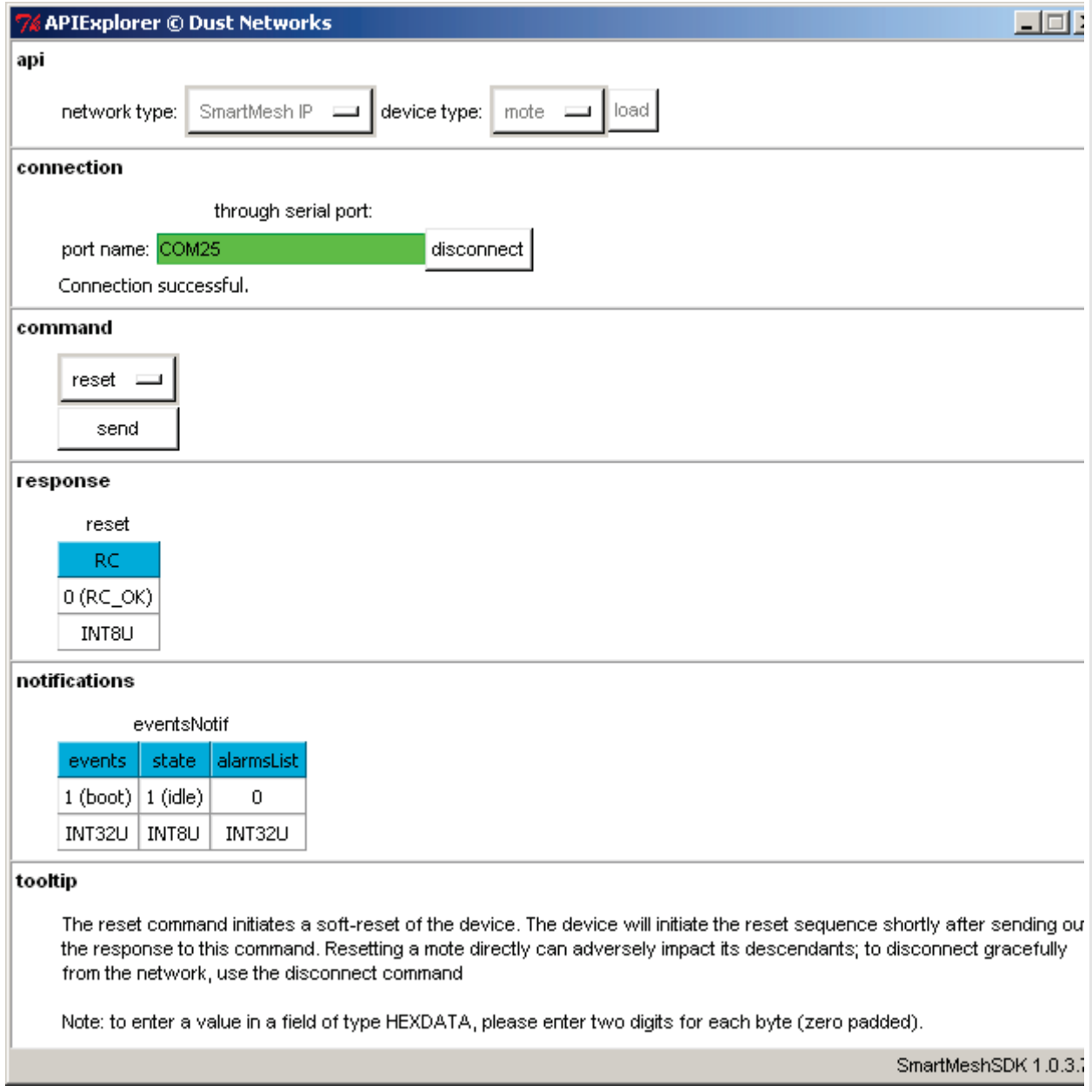
3. 分かりやすくするため、2 つの APIExplorer ウィンドウを並べて表示することをお勧めします。

SmartMesh IP モート



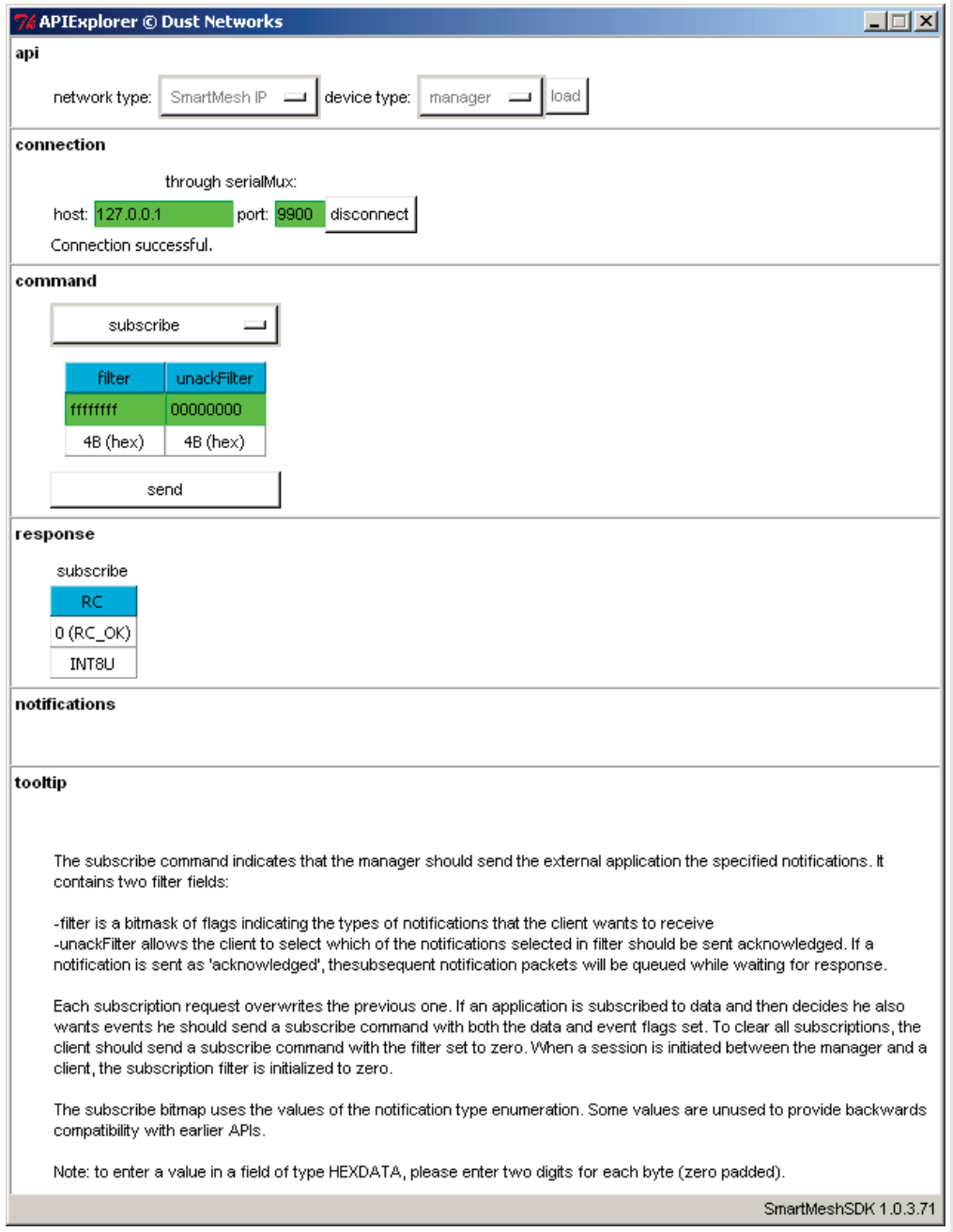
4. *reset* コマンドを実行して、SmartMesh IP モートをリセットします。数秒経つと、SmartMesh IP モート側に *boot* イベント通知が表示されます。

SmartMesh IP モート



5. SmartMesh IP マネージャ側では、全ての通知を *サブスクライブ*します。ネットワークが稼働中になっている場合、不定期に通知を受け取ります。

SmartMesh IP マネージャ

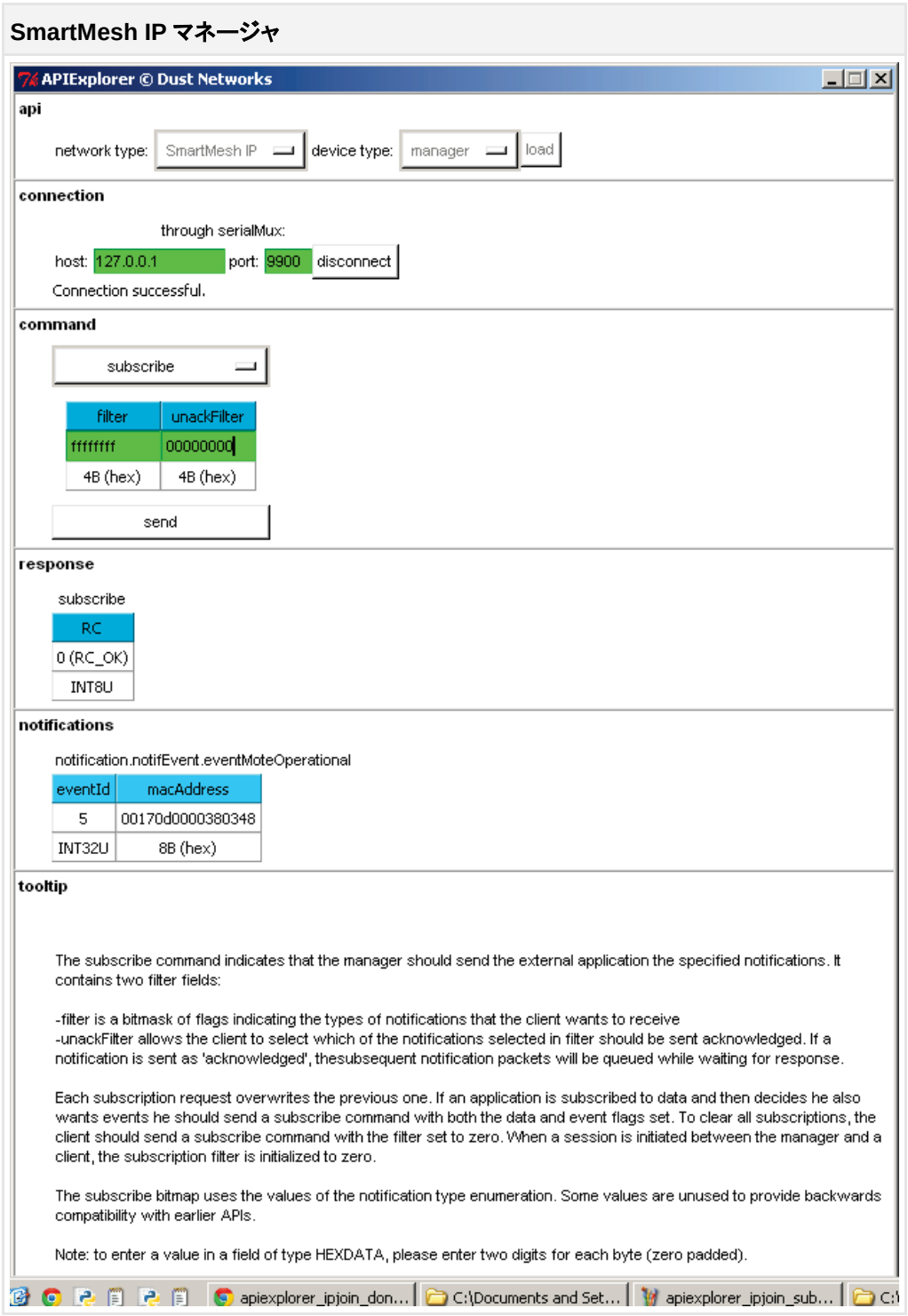


6. SmartMesh IP モート側で、*join* コマンドを実行します。これによりモートがネットワークにジョインし、SmartMesh IP モート側と SmartMesh IP マネージャ側の両方に以下の通知が送信されます。

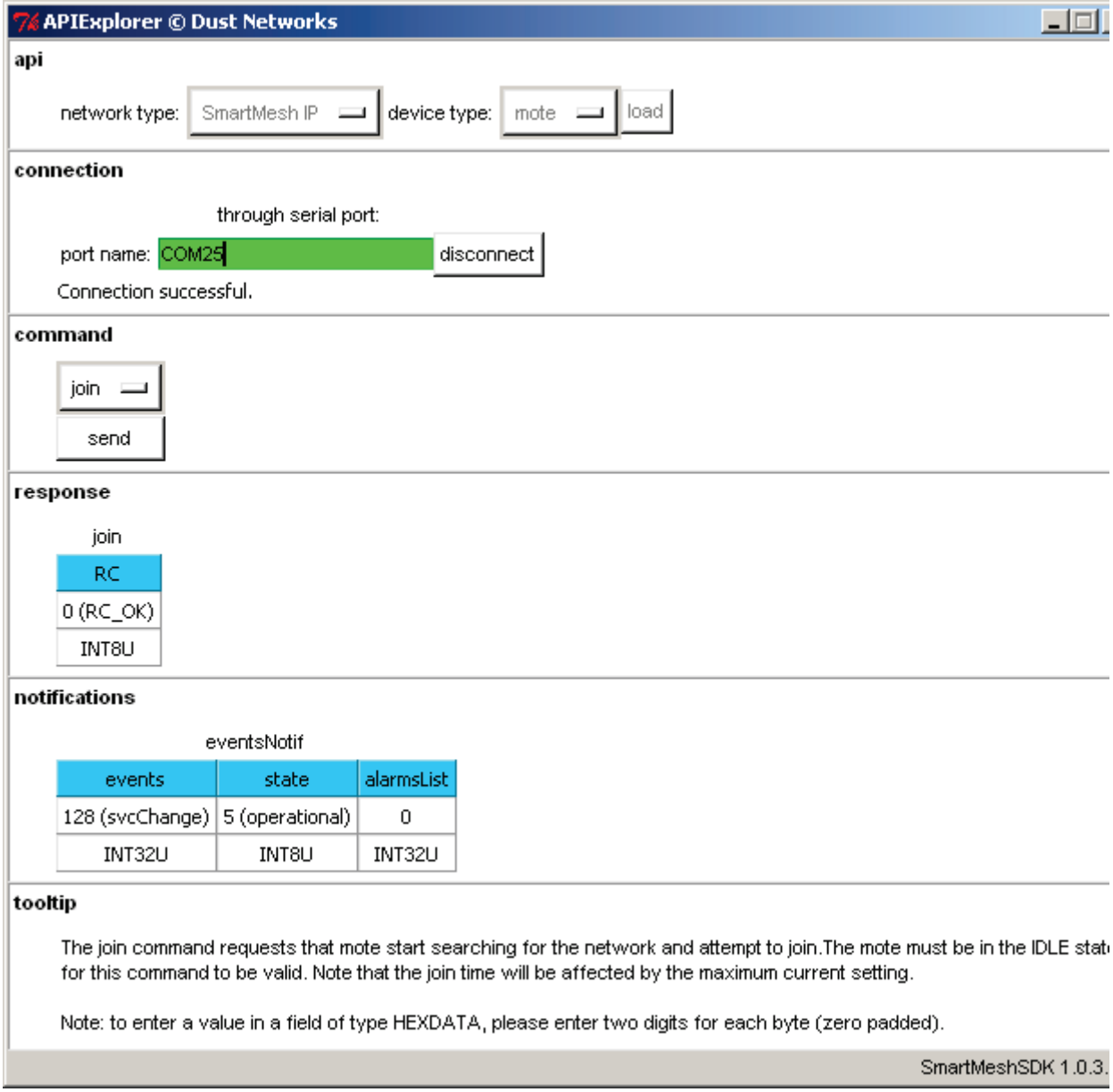
SmartMesh IP マネージャ側		SmartMesh IP モート側	
通知	**説明**	**通知**	**説明**
		joinStart	アドバタイズメントを受け取って、SmartMesh IP マネージャにジョインリクエストを送信した
eventsMoteJoin	ジョインリクエストを受け取った		
eventPathCreate	新しい SmartMesh IP モート用のパスを作成した		
		operational	SmartMesh IP モートがネットワークにジョインした
eventMoteOper	パスが正しく実装され、SmartMesh IP モートの稼働が確認された		
		scvChange	SmartMesh IP モートはベース帯域幅を受け取った

7. SmartMesh IP モートがネットワークにジョインすると、各ウィンドウは以下のように表示されます。

SmartMesh IP マネージャ



SmartMesh IP モート

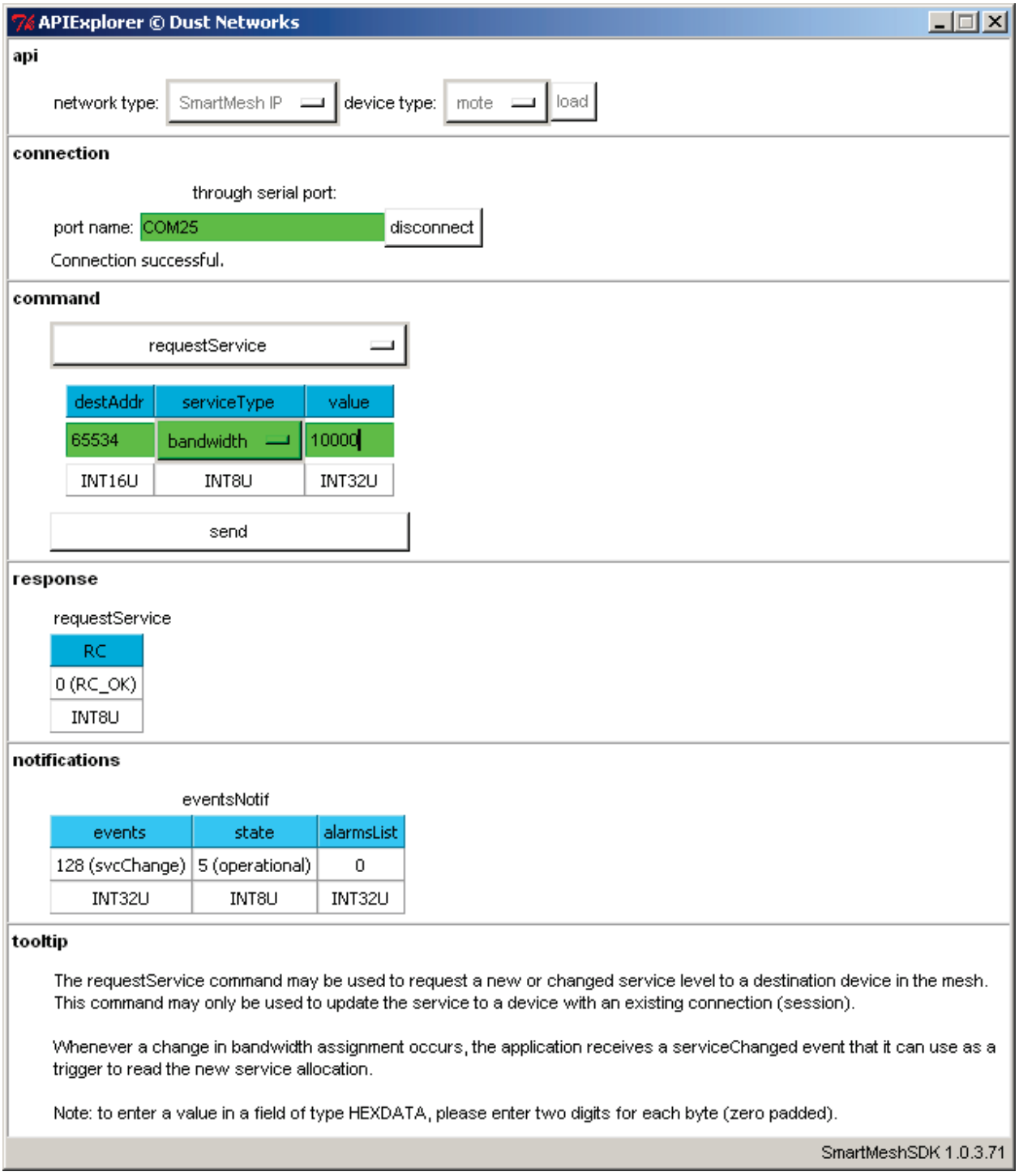


モートによるサービスの要求

このセクションでは、SmartMesh IP モートから SmartMesh IP マネージャに対して帯域幅を要求します。

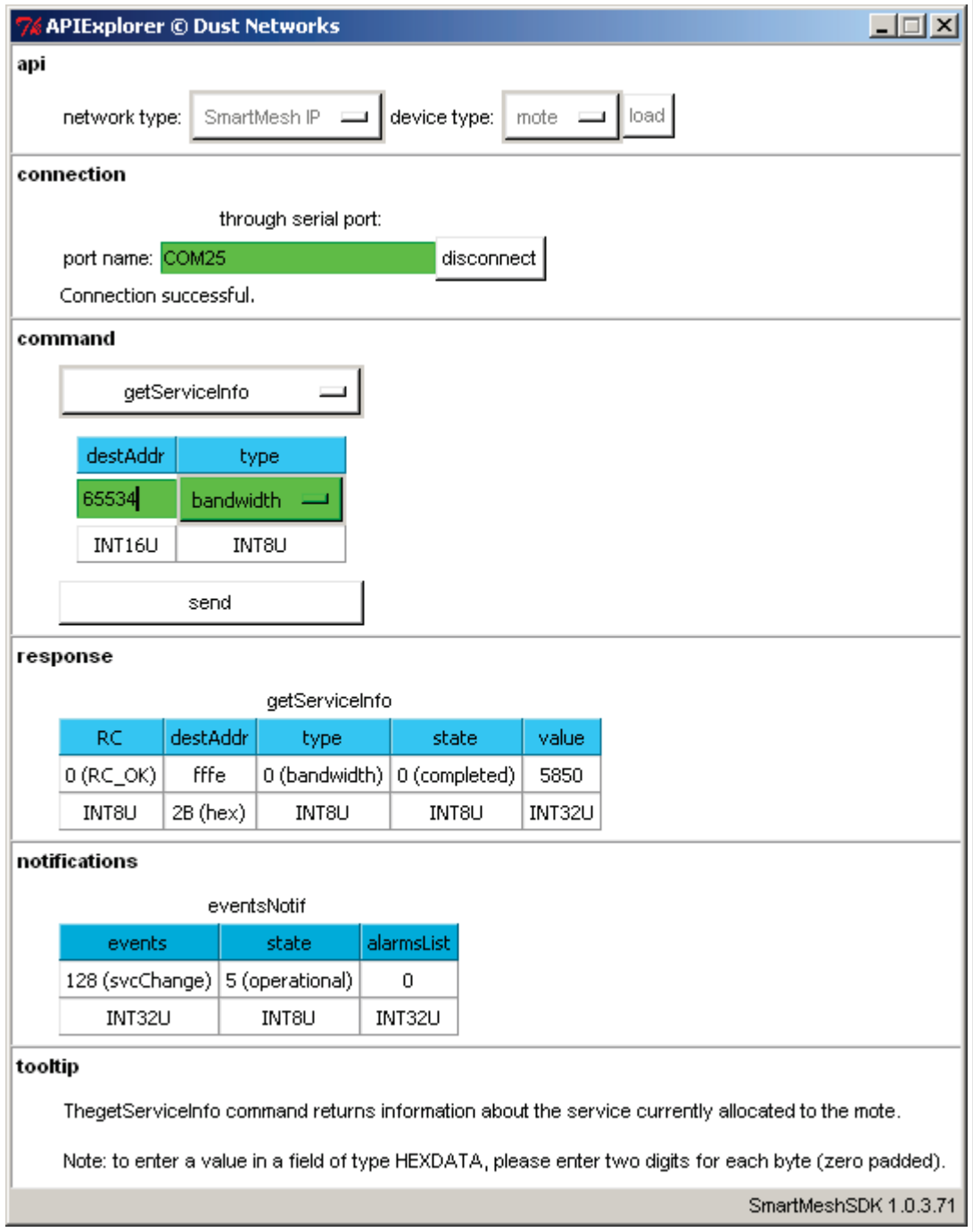
1. SmartMesh IP モート側で、*requestService* コマンドを実行して帯域幅を要求し、10 秒ごとに 1 つのパケットを SmartMesh IP マネージャに送信します。
   - **destAddr**:*65534*(SmartMesh IP マネージャの既知のアドレス)
   - **serviceType**:*bandwidth*
   - **value**:*10000*(送信間隔、ミリ秒)

2. **send** を押すと SmartMesh IP マネージャがリクエストを受け取り、要求された帯域幅を実装して、帯域幅の実装を要求元の SmartMesh IP モートに伝えます。その結果、SmartMesh IP モート側に *svcChange* イベント通知が表示されます。



割り当てられた帯域幅を確認するには、SmartMesh IP マネージャ側で getMoteInfo コマンドを実行し、SmartMesh IP モート側では getServiceInfo コマンドを実行します。

SmartMesh IP モート

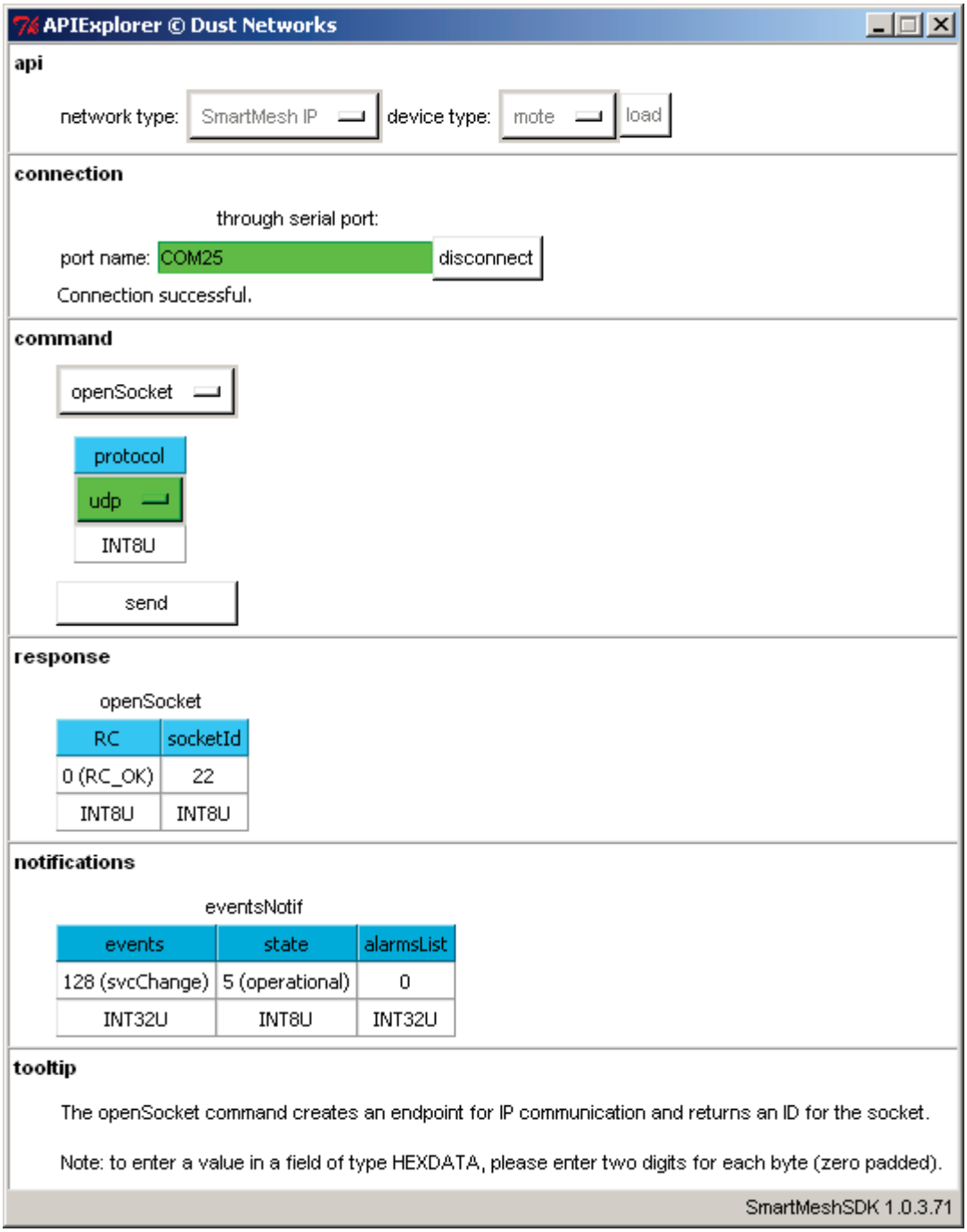


⚠ 割り当てられた帯域幅は送信間隔として示され、数字が小さいほど帯域幅は大きくなります。SmartMesh IP マネージャは帯域幅の割り当て時に余裕を持たせるため、実際に割り当てられる間隔は要求したものよりも小さくなります。

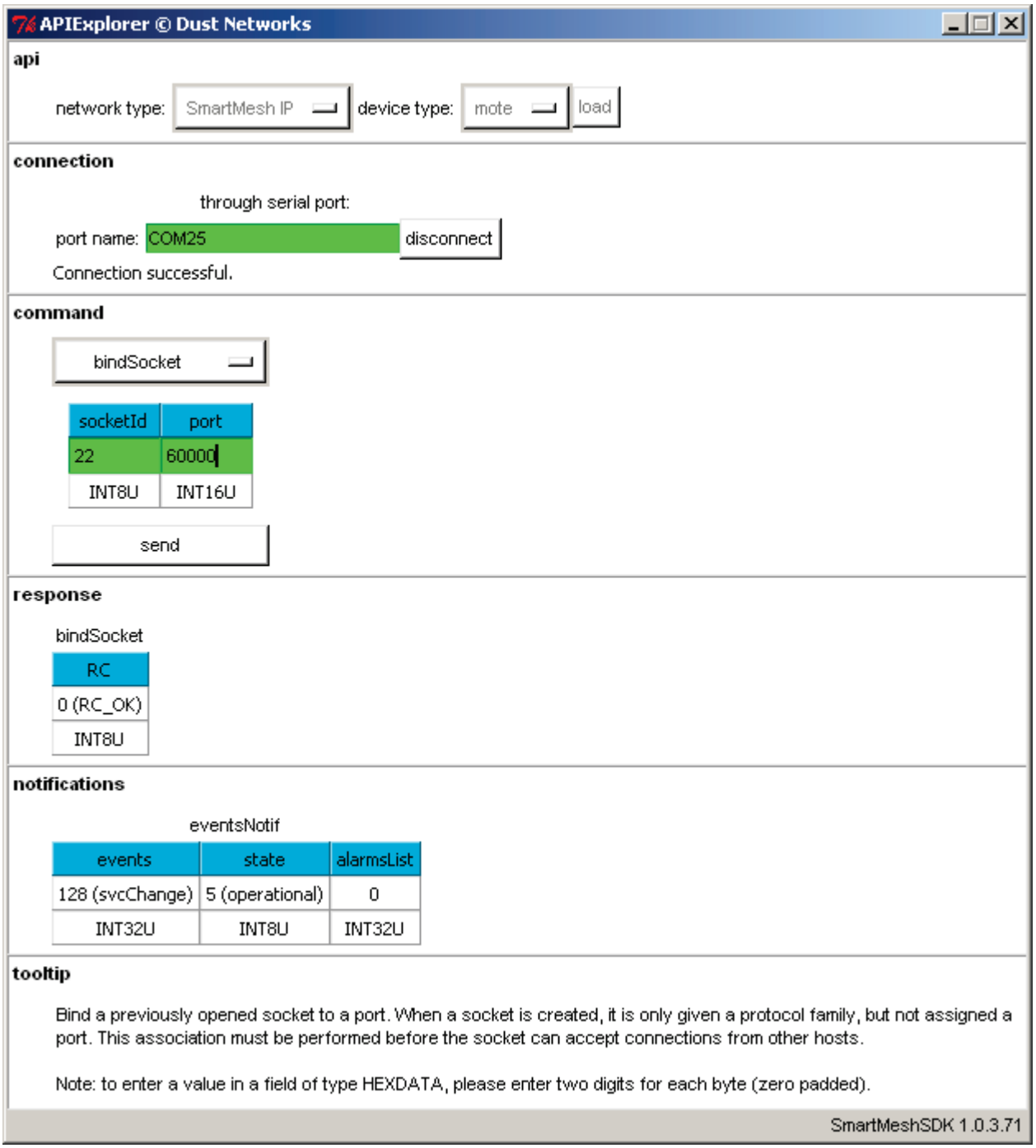
モートの UDP ソケットの準備

このセクションでは、SmartMesh IP モートに新しい UDP ソケットを開き、UDP ポート番号にバインドします。

1. SmartMesh IP モート側で *openSocket* コマンドを使用して、新しいソケットを作成します。socketId フィールドに表示される値を書き留めておきます。



2. **socketId** の値をハンドラとして使用して、新しく作成したソケットを任意の UDP ポート番号（ここでは *60000*）にバインドします。

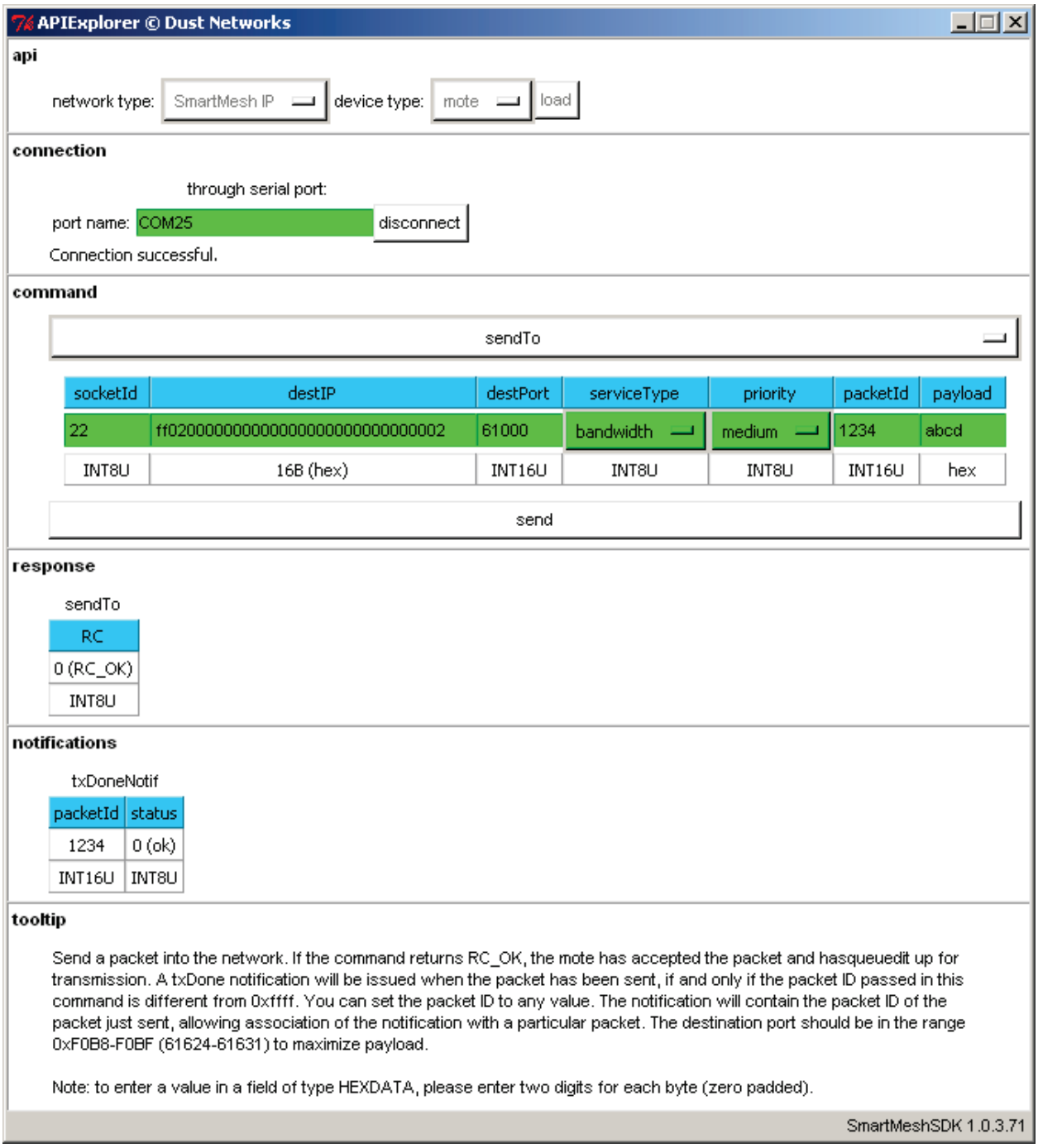


⚠ ソケットを開いてバインドしない限り、データを送信することはできません。これは、送信するデータの UDP ソース・ポートを識別するためにソケットが使用されるためです。

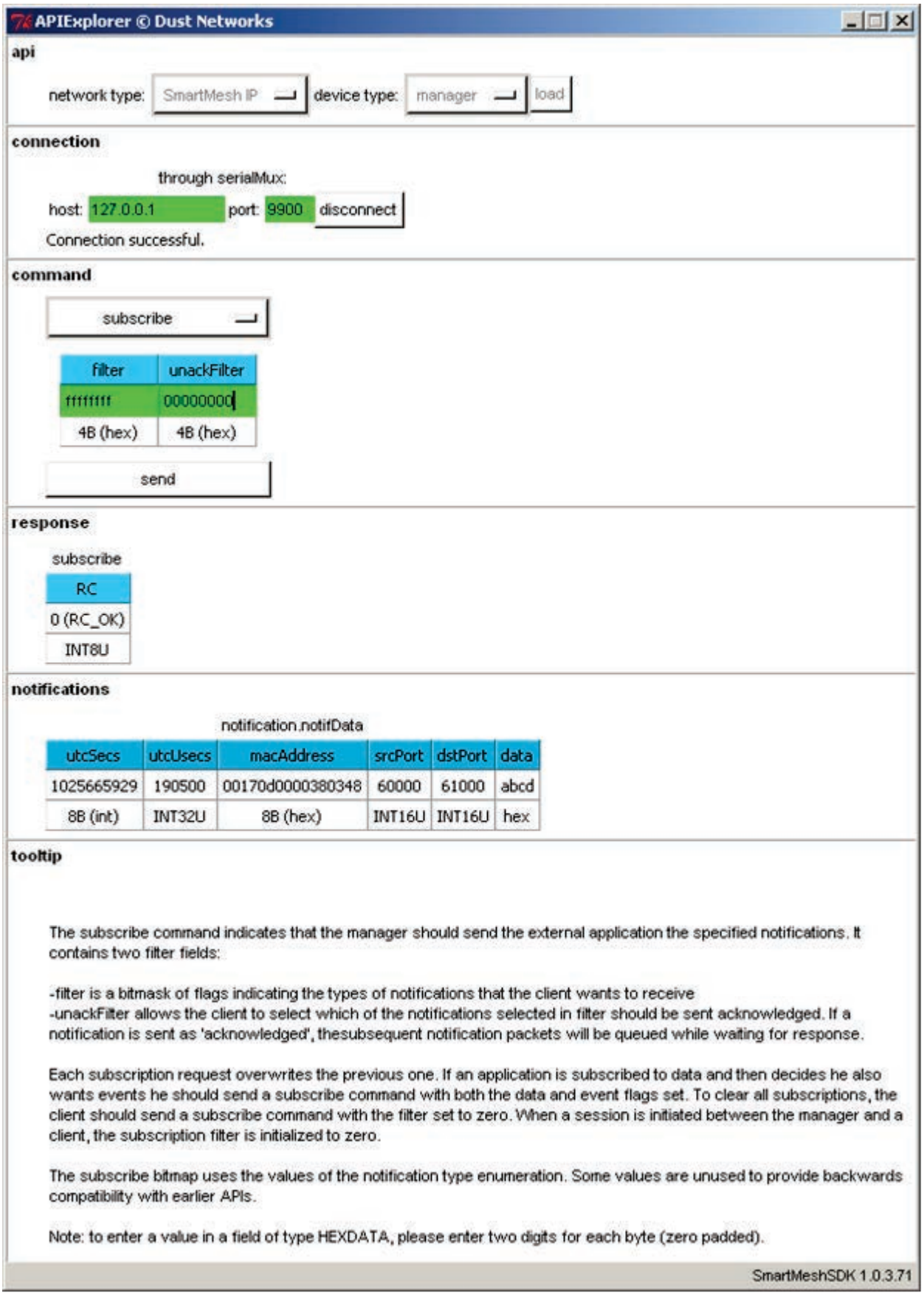
モートからマネージャへのデータ送信

このセクションでは、SmartMesh IP モートから SmartMesh IP マネージャに対してデータを送信します。

1. SmartMesh IP モート側で *sendTo* コマンドを使用して、マネージャにパケットを送信します。
   SmartMesh IP ネットワークでは、マネージャの IPv6 は *ff02::2* であり、これを 16 進数にすると *ff020000000000000000000000000002* になります。
2. パケットの転送が受け入れられると、SmartMesh IP モート側で *txDone* 通知が生成されます。この通知内に指定された **packetId** は、*sendTo* コマンド内で指定された **packetId** と一致します。



3. SmartMesh IP マネージャ側でデータ・パケットを受け取ると、*notifData* 通知が生成されます。


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. SmartMesh IP マネージャ側にあるパケットが、SmartMesh IP モートから送信されたパケット（UDP ポート *60000* から UDP ポート *61000* に送信され、ペイロード *abcd* を含む）と一致することを確認します。送信側と受信側の UDP ポート番号を一致させる必要はありません。

モートに対する ping の実行

前のセクションでは、モートからマネージャにデータを送信しました。ここでは、マネージャからモートにコマンドを送信します。モートに対して ping を送信すると、送信されたワイヤレス・リクエストがモートに着くまでに複数のホップを経由します。モートはコマンドを受け取るとすぐに ping レスポンスを生成します。このレスポンスは、別のホップを経由してマネージャに戻る場合があります。
これはモートに送信できるコマンドのうち最も簡単なコマンドであり、モートが正しく動作しているかどうかを確認したり、ラウンドトリップ時間を測定したりするために使用されます。

CLI の使用

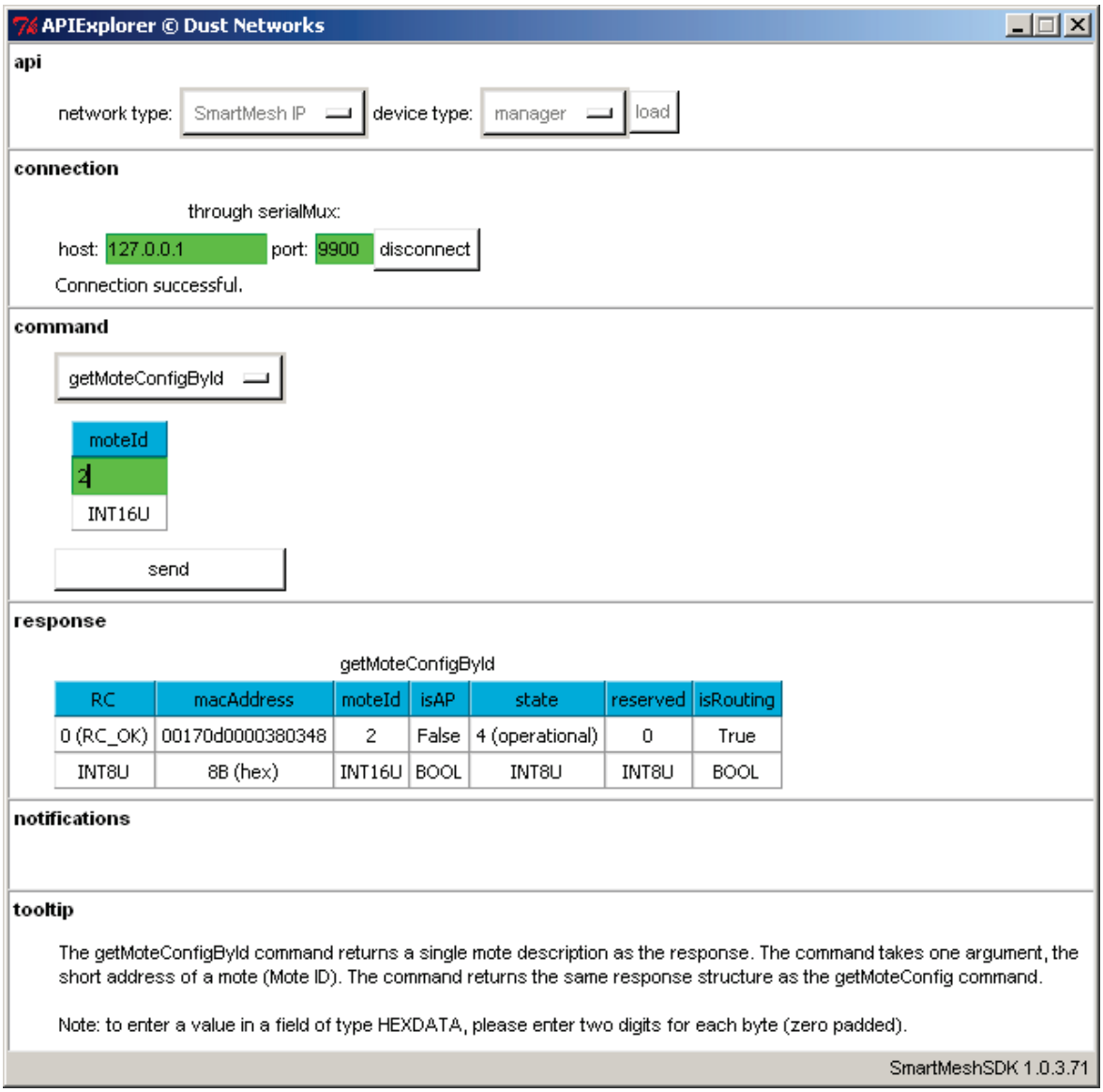
「マネージャの対話操作」の説明に従って、マネージャの CLI インタフェースにログインします。モート 2 に ping コマンドを送信します。モートの ID を確認するには、前述した *sm* コマンドを使用します。ID=2 を持つモートからレスポンスが返され、ラウンドトリップ遅延時間、温度、電源電圧が表示されます。

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

API の使用

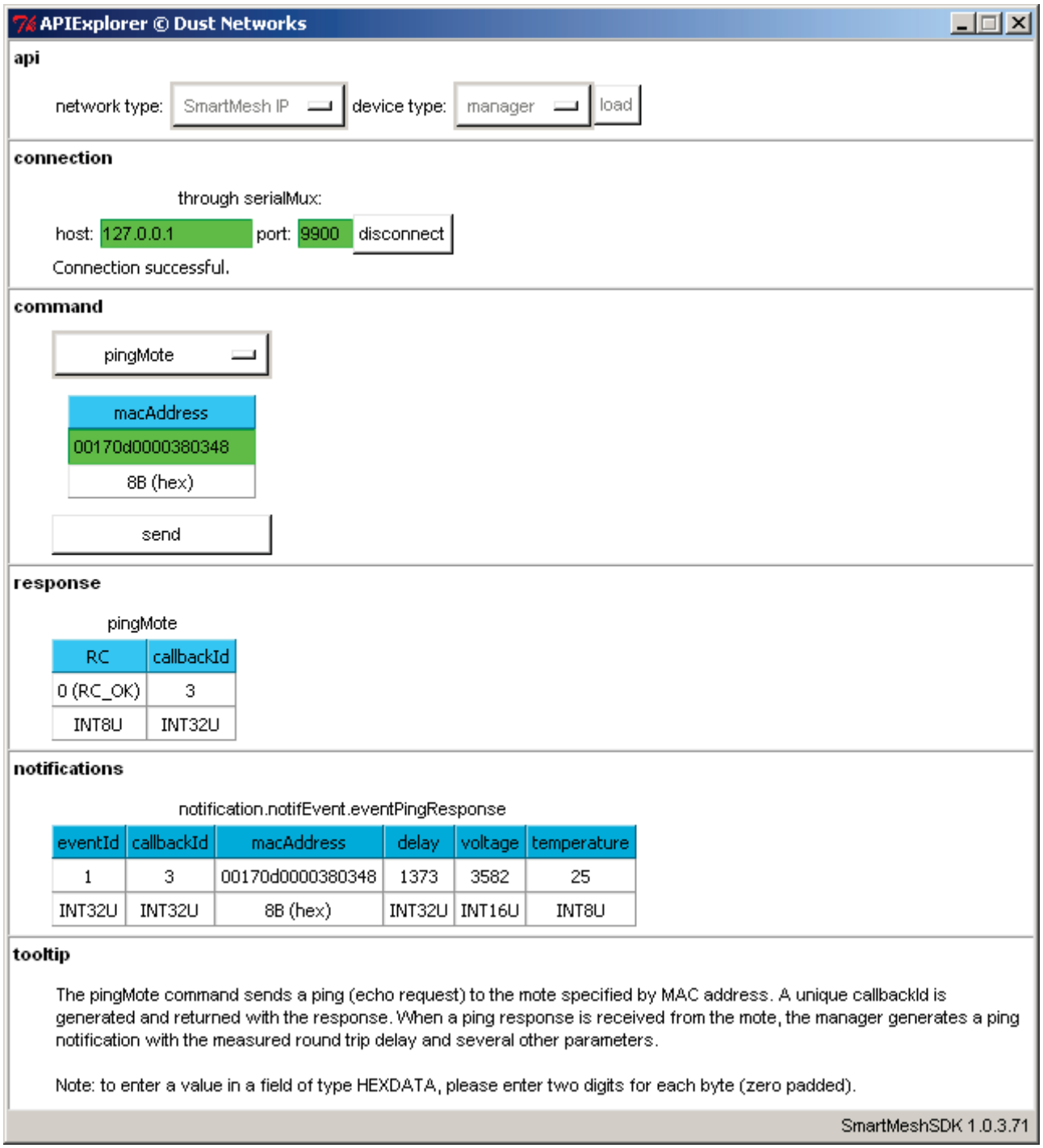
⚠ CLI では多くの場合、moteID が使用されますが、API ではモートの参照に MAC アドレスが使用されます。moteID と MAC アドレスの変換を実行するには、*getMoteConfigByID* コマンドを使用します。

1. *getMoteConfigByID* コマンドを使用して **moteId** フィールドに 2 を入力し、**send** ボタンをクリックします。
   - マネージャから、ネットワーク内のモート 2 に関する情報が返されます。



2 *pingMote* コマンドを選択し、上記で確認したモートの MAC アドレスを **macAddress** フィールドに入力します。

3 コマンドが認識されたことを示すレスポンスが返され、その数秒後に、ping を送ったモートからレスポンスが返されたという通知が返されます。ラウンドトリップ時間のほかに、レスポンスにはモートの電源電圧と温度の情報が含まれます。



⚠ マネージャ側で *eventPingResponse* 通知を受け取るには、この通知タイプをサブスクライブしている必要があります。

ℹ マネージャの *ping* コマンドは、Unix/Linux または DOS の ping コマンドと機能的に似ていますが、同じではありません。Unix/Linux または DOS の ping コマンドを実行すると、デバイスに ICMP echo コマンドが送信されます。

API およびモートの対話操作については、以下の資料を参照してください。

- SmartMesh IP ユーザー・ガイド
- SmartMesh IP Mote API Guide

8.4.2 一般的な問題

デバイスにコマンドを送信するとアプリケーションがハングする

SmartMesh IP モートまたは SmartMesh WirelessHART モートに接続しており、モートが**スレーブ・モード**で実行されていない場合、この問題が発生します。

モートの各モードの説明とその切り替え方法については、本書の「トラブルシューティング」セクションを参照してください。

アプリケーションがモートに接続されない

- モートの電源は入っていますか。
- アプリケーションがデバイスの API ポートに接続されていますか。
- このポートにあらかじめ接続しているアプリケーションがほかにありませんか。

以下の表に、全てのデバイスに使用できるシリアル設定を示します。

デバイス	シリアル・ポート番号	用途	ボーレート	データ・ビット	パリティ	ストップ・ビット
SmartMesh IP マネージャ	3 番目*	CLI	9600	8	N	1
	4 番目*	API	115200**	8**	N**	1**
SmartMesh IP モート	3 番目*	CLI	9600	8	N	1
	4 番目*	API	115200**	8**	N**	1**

* FTDI ドライバによって作成されたシリアル・ポート
** デフォルト値

8.5 上級者向けトピック

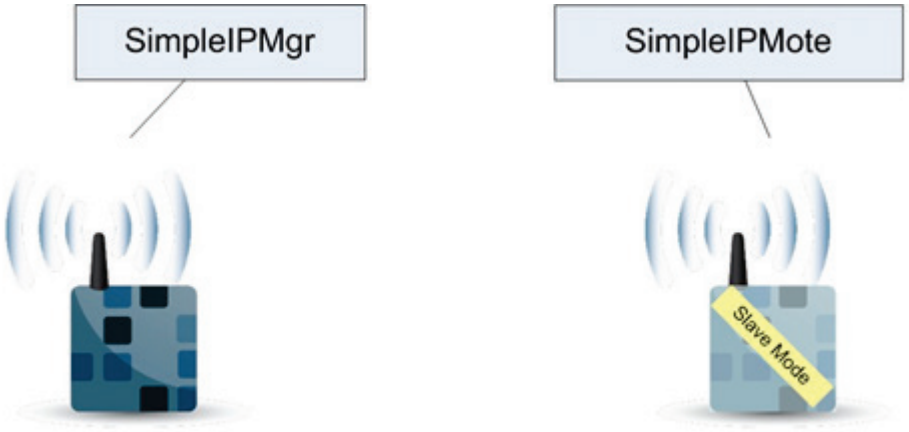
8.5.1 スクリプトによる API の実行

概要

ここでは、グラフィカル・アプリケーションではなく対話型スクリプトを使用して、SmartMesh IP モートの API を実行します。

この方法を使用すると、独自のアプリケーションのベースとして SmartMesh SDK を使用できます。ここで使用するサンプル・アプリケーションを出発点として使用するために、ここではコンパイル済みの実行可能ファイルではなく、Python ソース・コードから直接アプリケーションを実行します。

セットアップの概要



SmartMesh IP
マネージャ

SmartMesh IP
モート

手順

ここからは、SmartMesh SDK に含まれる以下のスクリプトを使用します。

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

> **参考**
>
> このセクションでは、スクリプトから SmartMesh SDK を使用する方法をごく簡単に説明します。
>
> 詳しい説明については、「SmartMesh IP SDK」セクションを参照してください。

SmartMesh IP マネージャの対話操作

> SmartMesh IP マネージャがコンピュータに接続されており、電源が入っていることを確認します。

SimpleIpMgr.py の実行

1. Windows ブラウザで、SmartMeshSDK ディレクトリを開きます。
2. */bin/Simple* に移動します。
3. *SimpleIpMgr.py* をダブルクリックします。コマンド・ウィンドウが表示されます。
4. 次の行が表示されたら

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

   y と入力して、Enter キーを押します。

5. 次の行が表示されたら

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

   SerialMux と同じマシンでこのスクリプトを実行している場合は、Enter キーを押します。

6. 次の行が表示されたら

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

   SerialMux がデフォルトの TCP ポート設定で実行されている場合は、Enter キーを押します。

7. 次の行がスクリプトの最終行です。

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

   Enter キーを押してウィンドウを閉じます。

以下のトレースには、スクリプトの全出力が含まれています。

```
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.
```

このスクリプトによって実行される 3 つのステップは、以下のとおりです。

1. SmartMesh IP マネージャに接続する。
2. SmartMesh IP マネージャが接続されているネットワークのステータスを取得する。
3. SmartMesh IP マネージャへの接続を解除する。

SimpleIpMgr.py の理解

SimpleIpMgr.py スクリプトをテキスト・エディタで開いて、内容を確認します（スクリプトをダブルクリックしないでください。ダブルクリックした場合、デフォルトではファイルが開く代わりにスクリプトが実行されます）。

ファイル全体にわたってコメントが含まれているため、実行時の出力結果と照らし合わせて確認することができます。

以下に、理解に役立つヒントを示します。

- *IpMgrConnector* オブジェクト（およびそのインスタンス *connector*）は、SerialMux を介して SmartMesh IP マネージャに物理的に接続するエンティティです。
- *raw_input()* はスクリプトを一時停止する Python 関数であり、ユーザーが文字列を入力して Enter キーを押すまで待機します。

参考

詳しい説明については、「SmartMesh IP SDK」セクションを参照してください。

SmartMesh IP モートの対話操作

SmartMesh IP モートがコンピュータに接続されて、電源が入っており、**スレーブ・モード**で実行されていることを確認します。

SimpleIpMote.py の実行

1. Windows ブラウザで、SmartMeshSDK ディレクトリを開きます。
2. */bin/Simple* に移動します。
3. *SimpleIpMote.py* をダブルクリックします。コマンド・ウィンドウが表示されます。
4. 次の行が表示されたら

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

y と入力して、Enter キーを押します。

5. 次の行が表示されたら

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

SmartMesh IP モートの API ポートのシリアル・ポート番号を入力して、Enter キーを押します。

6. 次の行がスクリプトの最終行です。

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

Enter キーを押してウィンドウを閉じます。

以下のコード・ブロックには、スクリプトの全出力が含まれています。

```
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.
```

このスクリプトによって実行される 4 つのステップは、以下のとおりです。

1. SmartMesh IP モートの API コマンド・リストをロードし、SmartMesh SDK の機能を使用してコマンド情報を取得する。
2. SmartMesh IP モートに接続する。
3. SmartMesh IP モートのステータスを取得する。
4. SmartMesh IP モートへの接続を解除する。

SimpleIpMote.py の理解

SimpleIpMote.py スクリプトをテキスト・エディタで開いて、内容を確認します（スクリプトをダブルクリックしないでください。ダブルクリックした場合、デフォルトではファイルが開く代わりにスクリプトが実行されます）。

ファイル全体にわたってコメントが含まれているため、実行時の出力結果と照らし合わせて確認することができます。

以下に、理解に役立つヒントを示します。

- *IpMoteDefinition* オブジェクト（およびそのインスタンス *apidef*）は、SmartMesh IP モートの API 定義（SmartMesh IP モートに送信できるコマンドのリスト）を表しています。
- *IpMoteConnector* オブジェクト（およびそのインスタンス *connector*）は、SmartMesh IP モートの API ポートに物理的に接続するエンティティです。

> **参考**
>
> 詳しい説明については、「SmartMesh IP SDK」セクションを参照してください。

一般的な問題

アプリケーションがモートに接続されない

- モートの電源は入っていますか。
- アプリケーションがデバイスの API ポートに接続されていますか。
- このポートにあらかじめ接続しているアプリケーションがほかにありませんか。

以下の表に、全てのデバイスに使用できるシリアル設定を示します。

デバイス	シリアル・ポート番号	用途	ボーレート	データ・ビット	パリティ	ストップ・ビット
SmartMesh IP マネージャ	3 番目*	CLI	9600	8	N	1
	4 番目*	API	115200**	8**	N**	1**
SmartMesh IP モート	3 番目*	CLI	9600	8	N	1
	4 番目*	API	115200**	8**	N**	1**

* FTDI ドライバによって作成されたシリアル・ポート
** デフォルト値

SerialMux 経由でマネージャに接続したが、出力が表示されない

APIExplorer から SerialMux への接続に成功しても（connection フレームのフィールドが緑色）、SerialMux から SmartMesh IP マネージャへの接続に失敗する場合があります。この場合、SerialMux がリスニングしているシリアル・ポートが、SmartMesh IP マネージャの API ポートではない可能性があります。

SerialMux の構成を変更するには、「SerialMux の構成」に記載された手順を参照してください。

8.5.2 HDLC フレームのロギング

概要

このステップでは、APIExplorer のロギング機能を使用して、コンピュータと SmartMesh IP モートの間で交換されるデータを表示します。

セットアップの概要

SmartMesh IP
マネージャ

SmartMesh IP
モート

手順

参考

このセクションでは、SmartMesh SDK のロギング機能を使用する方法をごく簡単に説明します。詳しい説明については、「SmartMesh IP SDK」セクションを参照してください。

1. Windows のブラウザ・ウィンドウを開き、SmartMeshSDK ディレクトリに移動します。
2. *win/*に移動します。
3. *APIExplorer.log* ファイルが存在する場合は削除します。
4. *APIExplorer.exe* をダブルクリックして、APIExplorer アプリケーションを起動します。
5. SmartMesh IP モートに接続します。
6. *reset* コマンドを送信して、*boot* イベント通知を受け取るまで待機します。
7. SmartMesh IP モートへの接続を解除し、APIExplorer アプリケーションを終了します。
8. 新しく作成された *APIExplorer.log* ファイルをテキスト・エディタで開きます。Windows コマンド・ラインで別のディレクトリから APIExplorer を実行している場合、*APIExplorer.log* ファイルはそのディレクトリ内に保存されます。

APIExplorer.log には、APIExplorer アプリケーション内の全てのモジュールの動作が含まれています。以下に、その出力例を示します。

```
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
```

以下に、ファイルの理解に役立つヒントを示します。

- 全てのエントリは以下の形式でロギングされます。

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

- *Hdlc* モジュールから出力されるエントリは、シリアル・ポート経由で送信されたデータそのものを示します。
- *SerialConnector* モジュールから出力されるエントリは、これらのデータがシリアライズ/デシリアライズされた方法を示します。

一般的な問題

APIExplorer.log が非常に長い

APIExplorer から出力されるログは、アプリケーションが起動されるたびに *APIExplorer.log* ファイルの最後に追加されます。空のロギング・セッションを再開するには、APIExplorer アプリケーションを起動する前に *APIExplorer.log* ファイルを削除します。

8.5.3 アップストリーム通信

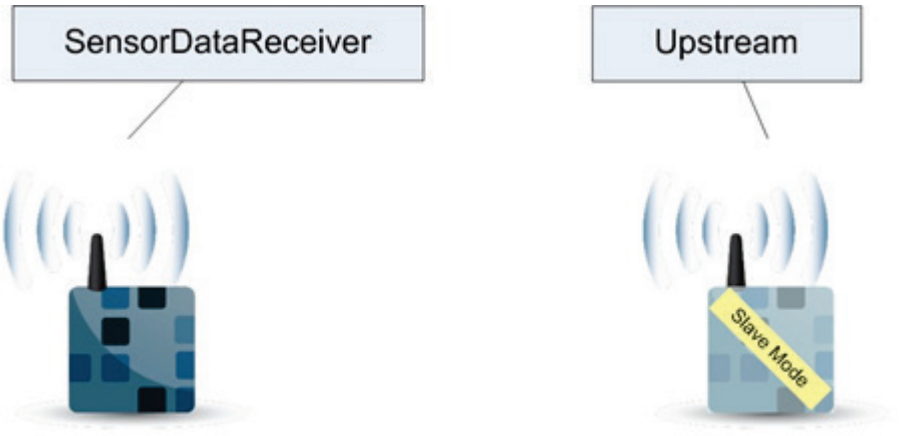
定義

アップストリーム通信は、SmartMesh IP モートから SmartMesh IP マネージャに向かう通信です。

概要

このステップでは、UpStream アプリケーションを使用して、SmartMesh IP モートのステート・マシンを起動からデータ送信まで遷移させます。

セットアップの概要



SmartMesh IP
マネージャ

SmartMesh IP
モート

手順

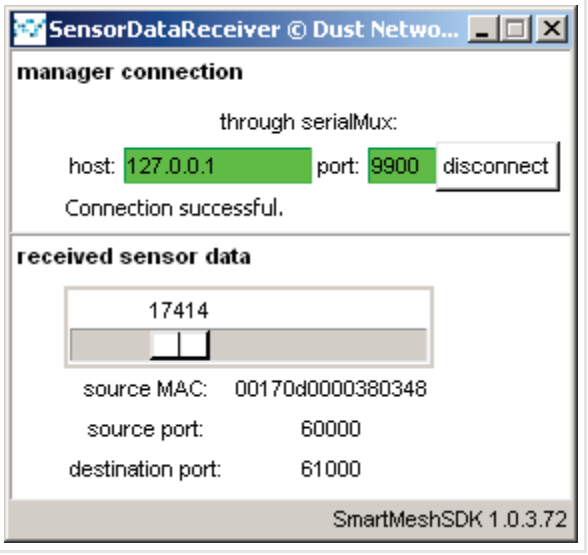
UpStream アプリケーションは、APIExplorer を使用した基本的ウォークスルーの手順をプログラムから実行したものと考えることができます。

1. SmartMesh IP モートをセットアップします。
    1. SmartMesh IP モートに正しいネットワーク ID が設定されており、**スレーブ・モード**で実行されていることを確認します。
    2. 電源を入れ直すか、または APIExplorer で *reset* コマンドを実行して、SmartMesh IP モートをリセットします。
    3. SmartMeshSDK ディレクトリに移動してから、*/win/* に移動します。
    4. *Upstream.exe* プログラムをダブルクリックします。アプリケーションが開きます。
    5. SmartMesh IP モートの API ポート番号を入力して、**connect** をクリックします。
    6. Upstream アプリケーションによって、SmartMesh IP モートのステート・マシンが以下のとおりに遷移します。
        - 初期の *boot* イベント通知を待機します。

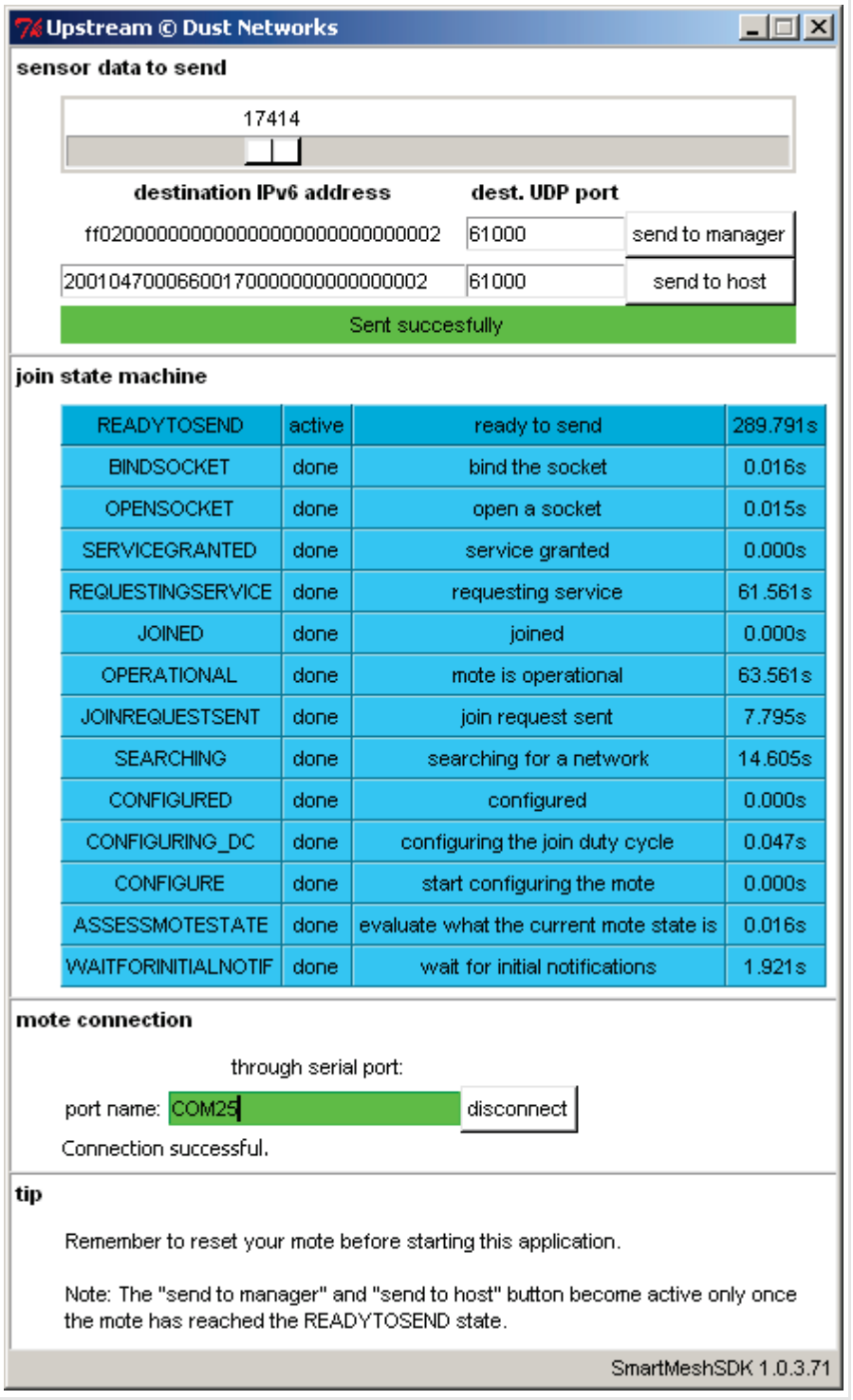
- ジョインのデューティ・サイクルを設定します。
- *join* コマンドを発行します。SmartMesh IP モートがネットワークの検索を開始します。
- ネットワークにジョインすると、SmartMesh IP モートはサービスを要求します。
- サービスが提供されると、モートのステートが *READYTOSEND* に変わり、sensor data to send フレームが使用可能になります。

2. SmartMesh IP マネージャをセットアップします。
    1. SmartMeshSDK ディレクトリに移動してから、*/win/*に移動します。
    2. *SensorDataReceiver.exe* プログラムをダブルクリックします。アプリケーションが開きます。
    3. manager connection フレームでオプションを選択し、SmartMesh IP マネージャに接続します。
3. SmartMesh IP モートから SmartMesh IP マネージャにデータを送信します。
    1. UpStream アプリケーションで SmartMesh IP モートに接続し、スライダの値を変更します。これはセンサによって収集される値を再現したものです。
    2. send to manager ボタンをクリックします。以下の UDP パケットが SmartMesh IP モートから SmartMesh IP マネージャに送信されます。
        - UDP source port:*61000*
        - UDP destination port:*61000*（または UpStream アプリケーションで設定されている値）
        - パケットには、UpStream アプリケーション上のスライダの値を表す 2 バイトのデータが含まれます。
    3. SmartMesh IP マネージャに接続された SensorDataReceiver アプリケーションがパケットを受け取ると、received sensor data フレームにデータが表示されます。

SmartMesh IP マネージャ側



SmartMesh IP マネージャ側



send to host ボタンを使用すると、任意の IP アドレスにデータを送信することができます。この機能には LBR 経由での転送が必要であり、詳しくは別のセクションで説明します。

⚠ UpStream アプリケーションに汎用性を持たせるため、モートのネットワーク ID は設定されていません。*UpStream.py* ソース・ファイルに該当するコードがコメントとして含まれており、必要に応じて使用できます。

一般的な問題

アプリケーションがモートに接続されない

- モートの電源は入っていますか。
- アプリケーションがデバイスの API ポートに接続されていますか。
- このポートにあらかじめ接続しているアプリケーションがほかにありませんか。

以下の表に、全てのデバイスに使用できるシリアル設定を示します。

デバイス	シリアル・ポート番号	用途	ボーレート	データ・ビット	パリティ	ストップ・ビット
SmartMesh IP マネージャ	3 番目*	CLI	9600	8	N	1
	4 番目*	API	115200**	8**	N**	1**
SmartMesh IP モート	3 番目*	CLI	9600	8	N	1
	4 番目*	API	115200**	8**	N**	1**

* FTDI ドライバによって作成されたシリアル・ポート
** デフォルト値

モートがネットワークにジョインしない

- ジョインのデューティ・サイクルとネットワークの状態によっては、SmartMesh IP モートがジョインするまでに何分かかかる場合があります。
- SmartMesh IP モートに設定されたネットワーク ID が、SmartMesh IP マネージャと同じであることを確認します。

8.5.4 ダウンストリーム通信

定義

ダウンストリーム通信は、SmartMesh IP マネージャから SmartMesh IP モートに向かう通信です。

概要

このステップでは、APIExplorer を使用して、SmartMesh IP マネージャから接続済みの SmartMesh IP モートにデータを送信します。

セットアップの概要

SmartMesh IP
マネージャ

SmartMesh IP
モート

手順

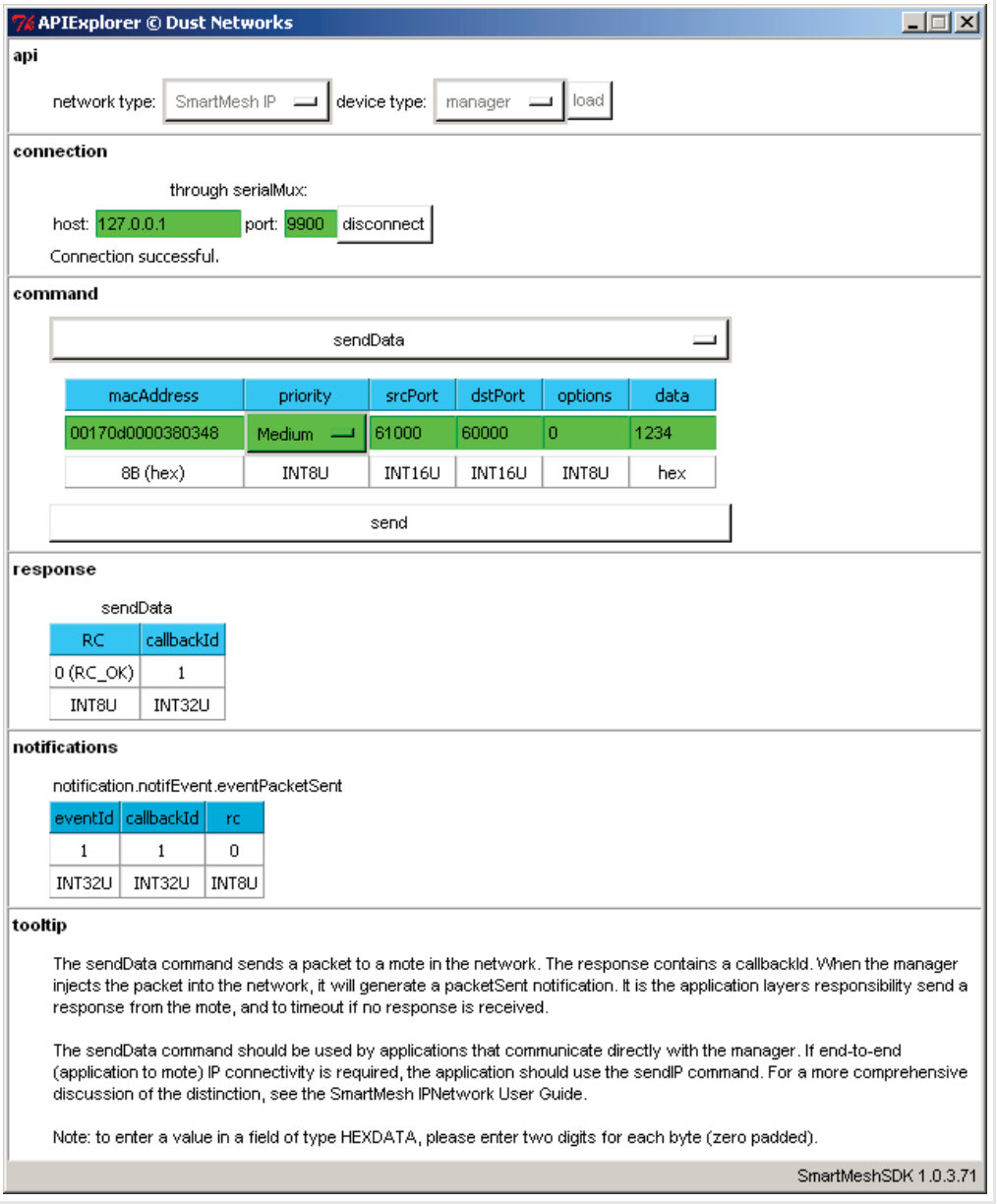
1. 基本的ウォークスルーの手順に従って、SmartMesh IP モートを SmartMesh IP マネージャに接続し、UDP ポート *60000* 上にソケットを開きます。
2. SmartMesh IP モートに接続された APIExplorer を使用して *getParameter.macAddress* を実行し、MAC アドレスを読み取ります（ここでは、*00170d0000380348*）。
3. また、2 番目の APIExplorer アプリケーションを起動して、SmartMesh IP マネージャに接続します（まだ起動していない場合のみ）。
4. 2 番目の APIExplorer アプリケーションで、*sendData* コマンドを実行し、以下のとおりにデータを入力します。
   - **macAddress**：SmartMesh IP モートから返された MAC アドレス
   - **priority**：*Medium*（ネットワーク内にその他のトラフィックがない場合、この優先順位によってパケットの伝達が変わることはありません）
   - **srcPort**：*61000*、または任意の 16 ビット・ポート番号
   - **dstPort**：*60000*（SmartMesh IP モート上に開かれたソケットの UDP ポート番号）
   - **options**：*0*（なし）

- **data**:*1234*、または任意の 16 進ペイロード

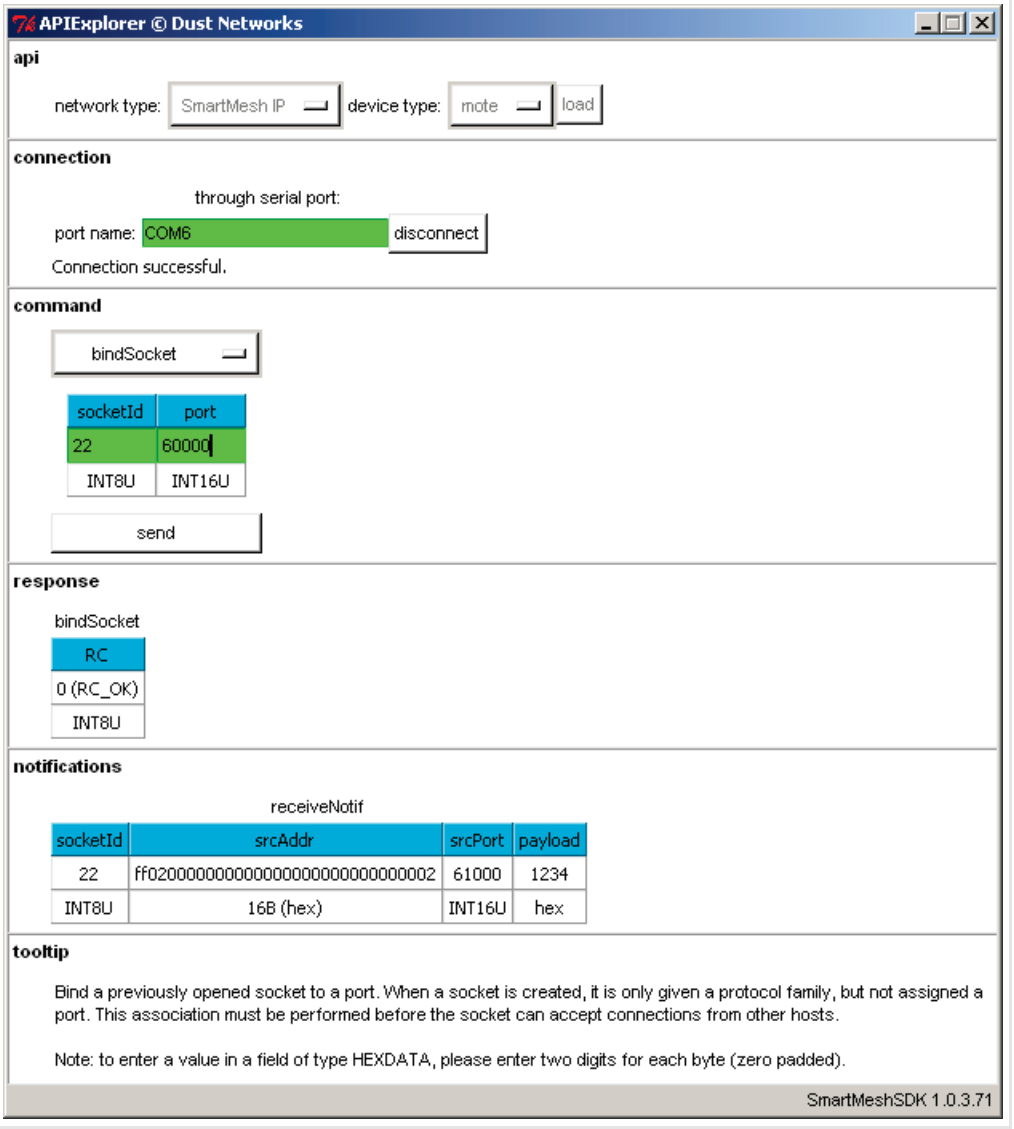
5. **send** をクリックします。パケットが SmartMesh IP マネージャから SmartMesh IP モートに送信されます。

パケットを受け取ると、SmartMesh IP モート側で *receive* 通知が起動され、パケットの送信元として、IPv6 アドレス *ff02::2*（SmartMesh IP マネージャの既知の IPv6 アドレス）と UDP ポート *61000* が表示されます。

SmartMesh IP マネージャ側



SmartMesh IP モート側



一般的な問題

デバイスにコマンドを送信するとアプリケーションがハングする

SmartMesh IP モートまたは SmartMesh WirelessHART モートに接続しており、モートが**スレーブ・モード**で実行されていない場合、この問題が発生します。

モートの各モードの説明とその切り替え方法については、本書の「トラブルシューティング」セクションを参照してください。

アプリケーションがモートに接続されない

- モートの電源は入っていますか。
- アプリケーションがデバイスの API ポートに接続されていますか。
- このポートにあらかじめ接続しているアプリケーションがほかにありませんか。

以下の表に、全てのデバイスに使用できるシリアル設定を示します。

デバイス	シリアル・ポート番号	用途	ボーレート	データ・ビット	パリティ	ストップ・ビット
SmartMesh IP マネージャ	3 番目*	CLI	9600	8	N	1
	4 番目*	API	115200**	8**	N**	1**
SmartMesh IP モート	3 番目*	CLI	9600	8	N	1
	4 番目*	API	115200**	8**	N**	1**

* FTDI ドライバによって作成されたシリアル・ポート
** デフォルト値

SerialMux 経由でマネージャに接続したが、出力が表示されない

APIExplorer から SerialMux への接続に成功しても（connection フレームのフィールドが緑色）、SerialMux から SmartMesh IP マネージャへの接続に失敗する場合があります。この場合、SerialMux がリスニングしているシリアル・ポートが、SmartMesh IP マネージャの API ポートではない可能性があります。

SerialMux の構成を変更するには、「SerialMux の構成」に記載された手順を参照してください。

モートがネットワークにジョインしない

- ジョインのデューティ・サイクルとネットワークの状態によっては、SmartMesh IP モートがジョインするまでに何分かかかる場合があります。
- SmartMesh IP モートに設定されたネットワーク ID が、SmartMesh IP マネージャと同じであることを確認します。

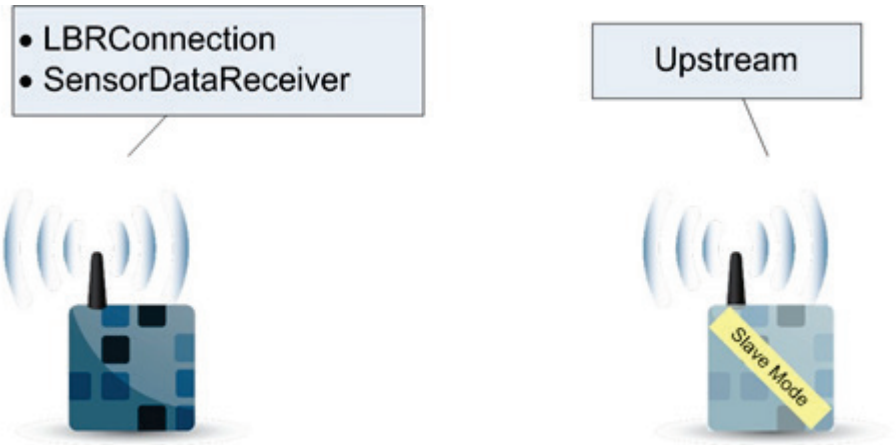
8.5.5　インターネット統合

概要

このステップでは、UpStream アプリケーションを使用して、SmartMesh IP モートから Web ページ http://motedata.dustnetworks.com/にデータを送信します。

SmartMesh IP マネージャに接続されたコンピュータがインターネットに接続されている必要があります。

セットアップの概要



手順

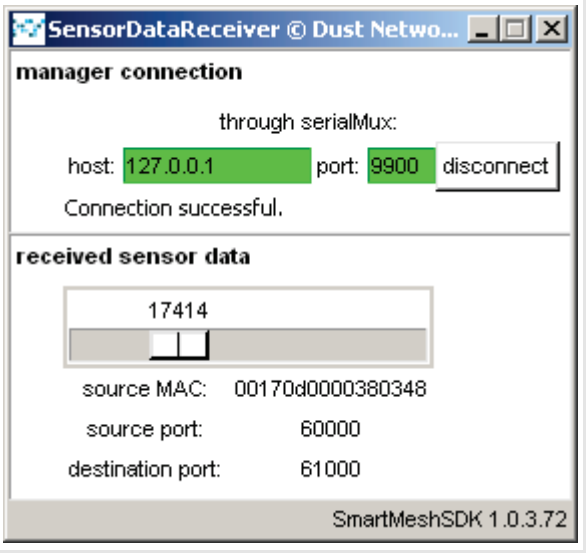
⚠ SmartMesh IP マネージャ側には、同時に 2 種類のアプリケーションに接続します。このため、SerialMux 経由での接続が必要です。

マネージャへのデータ送信

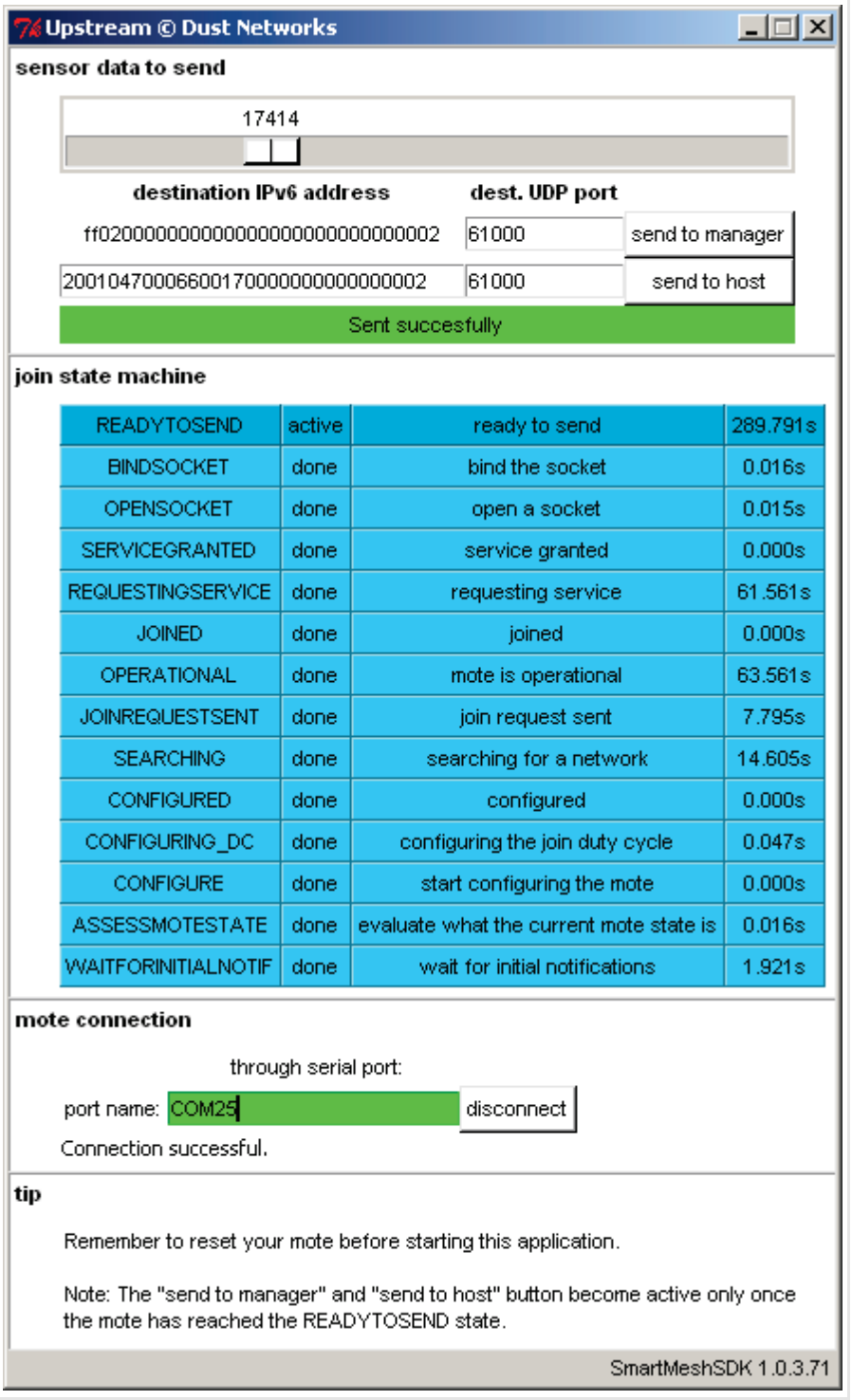
下記の手順は、「アップストリーム通信」チュートリアルに記載した手順と同じです。

1. SmartMesh IP モートをセットアップします。
    1. SmartMesh IP モートに正しいネットワーク ID が設定されており、**スレーブ・モード**で実行されていることを確認します。
    2. 電源を入れ直すか、または APIExplorer で *reset* コマンドを実行して、SmartMesh IP モートをリセットします。
    3. SmartMeshSDK ディレクトリに移動してから、*/win/* に移動します。
    4. *Upstream.exe* プログラムをダブルクリックします。アプリケーションが開きます。
    5. SmartMesh IP モートの API ポート番号を入力して、**connect** をクリックします。
    6. Upstream アプリケーションによって、SmartMesh IP モートのステート・マシンが以下のとおりに遷移します。
        - 初期の *boot* イベント通知を待機します。
        - ジョインのデューティ・サイクルを設定します。
        - *join* コマンドを発行します。SmartMesh IP モートがネットワークの検索を開始します。
        - ネットワークにジョインすると、SmartMesh IP モートはサービスを要求します。
        - サービスが提供されると、モートのステートが *READYTOSEND* に変わり、sensor data to send フレームが使用可能になります。
2. SmartMesh IP マネージャをセットアップします。
    1. SmartMeshSDK ディレクトリに移動してから、*/win/* に移動します。
    2. *SensorDataReceiver.exe* プログラムをダブルクリックします。アプリケーションが開きます。
    3. manager connection フレームでオプションを選択し、SmartMesh IP マネージャに接続します。
3. SmartMesh IP モートから SmartMesh IP マネージャにデータを送信します。
    1. UpStream アプリケーションで SmartMesh IP モートに接続し、スライダの値を変更します。これはセンサによって収集される値を再現したものです。
    2. send to manager ボタンをクリックします。以下の UDP パケットが SmartMesh IP モートから SmartMesh IP マネージャに送信されます。
        - UDP source port: *61000*
        - UDP destination port: *61000*（または UpStream アプリケーションで設定されている値）
        - パケットには、UpStream アプリケーション上のスライダの値を表す 2 バイトのデータが含まれます。
    3. SmartMesh IP マネージャに接続された SensorDataReceiver アプリケーションがパケットを受け取ると、received sensor data フレームにデータが表示されます。

SmartMesh IP マネージャ側



SmartMesh IP モート側

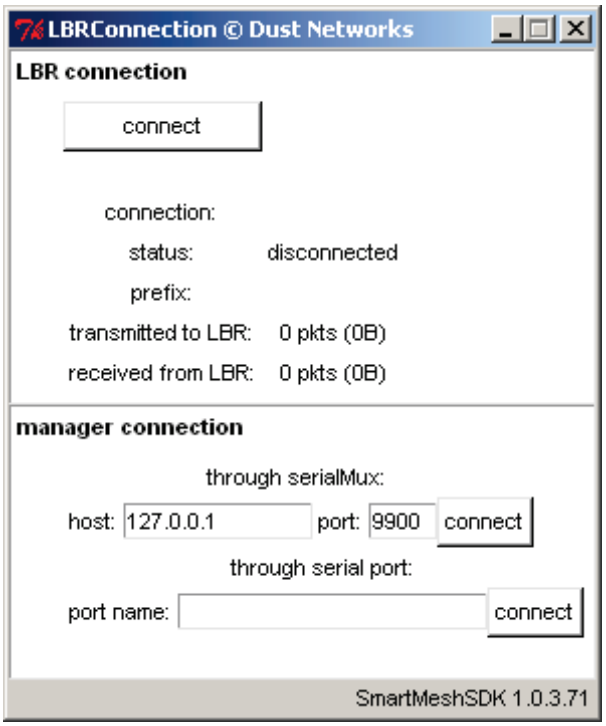


send to host ボタンを使用すると、任意の IP アドレスにデータを送信することができます。この機能には LBR 経由での転送が必要であり、詳しくは別のセクションで説明します。

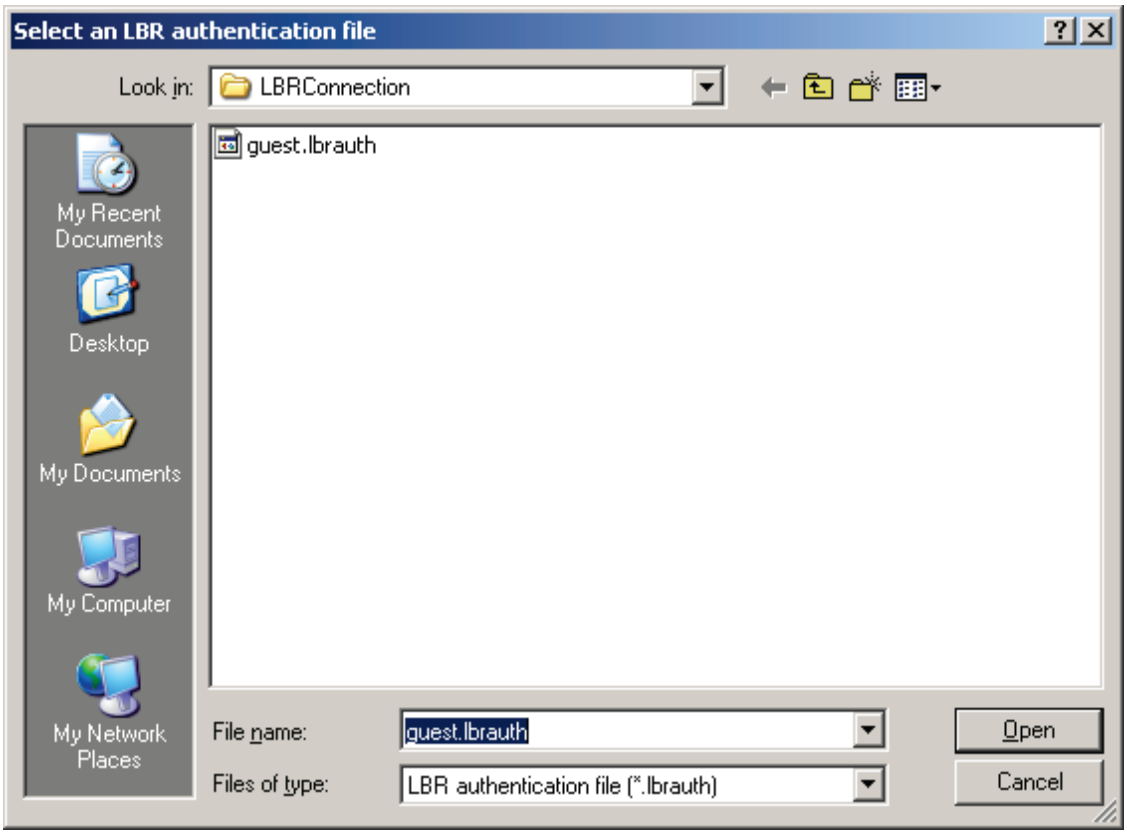
LBR へのマネージャの接続

Low-power Border Router（LBR）はインターネット上のサーバーであり、6LoWPAN ヘッダと IPv6 ヘッダとの間で圧縮/復元を実行します。詳しくは、「**Low-power Border Router**」を参照してください。

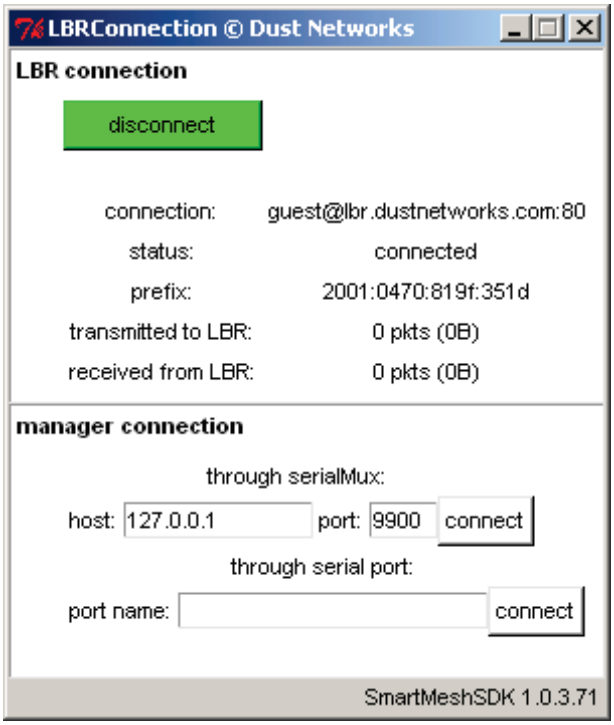
1. SmartMesh SDK ディレクトリに移動してから、*/win/*に移動します。
2. *LBRConnection.exe* スクリプトをダブルクリックします。アプリケーションが開きます。



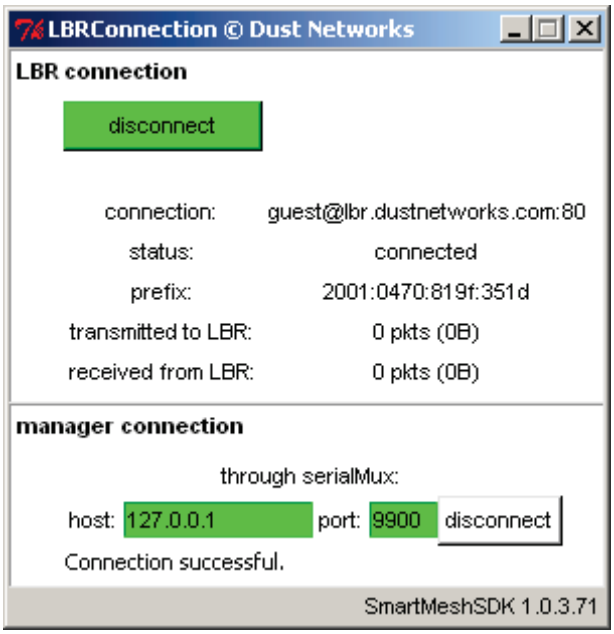
3. **LBR connection** フレームで **connect** をクリックすると、ファイル参照ウィンドウが表示されます。



4. *guest.lbrauth* ファイルを選択して **open** をクリックします。このファイルには、LBR のアドレスと認証に必要な資格情報が含まれています。
5. ボタンが緑色に変わり、LBR によって割り当てられた IPv6 プリフィックス（ここでは *2001:0470:819f:351d*）がフィールドに表示されます。このプリフィックスを書き留めておきます。

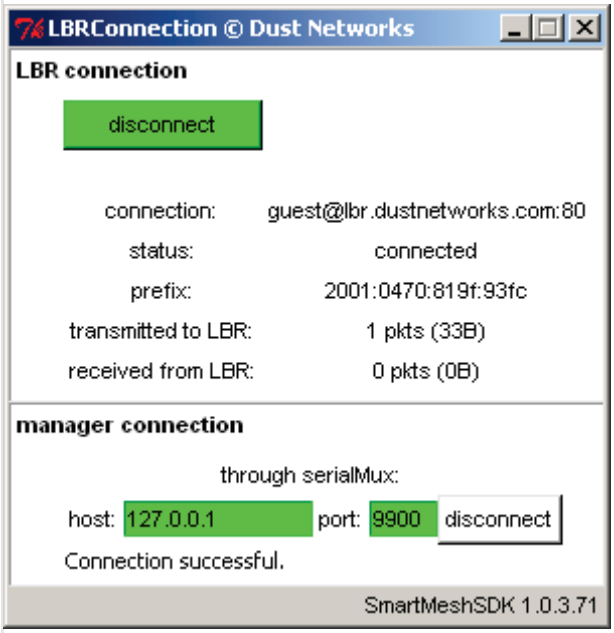


6. **manager connection** フレームを使用して SmartMesh IP マネージャに接続します。

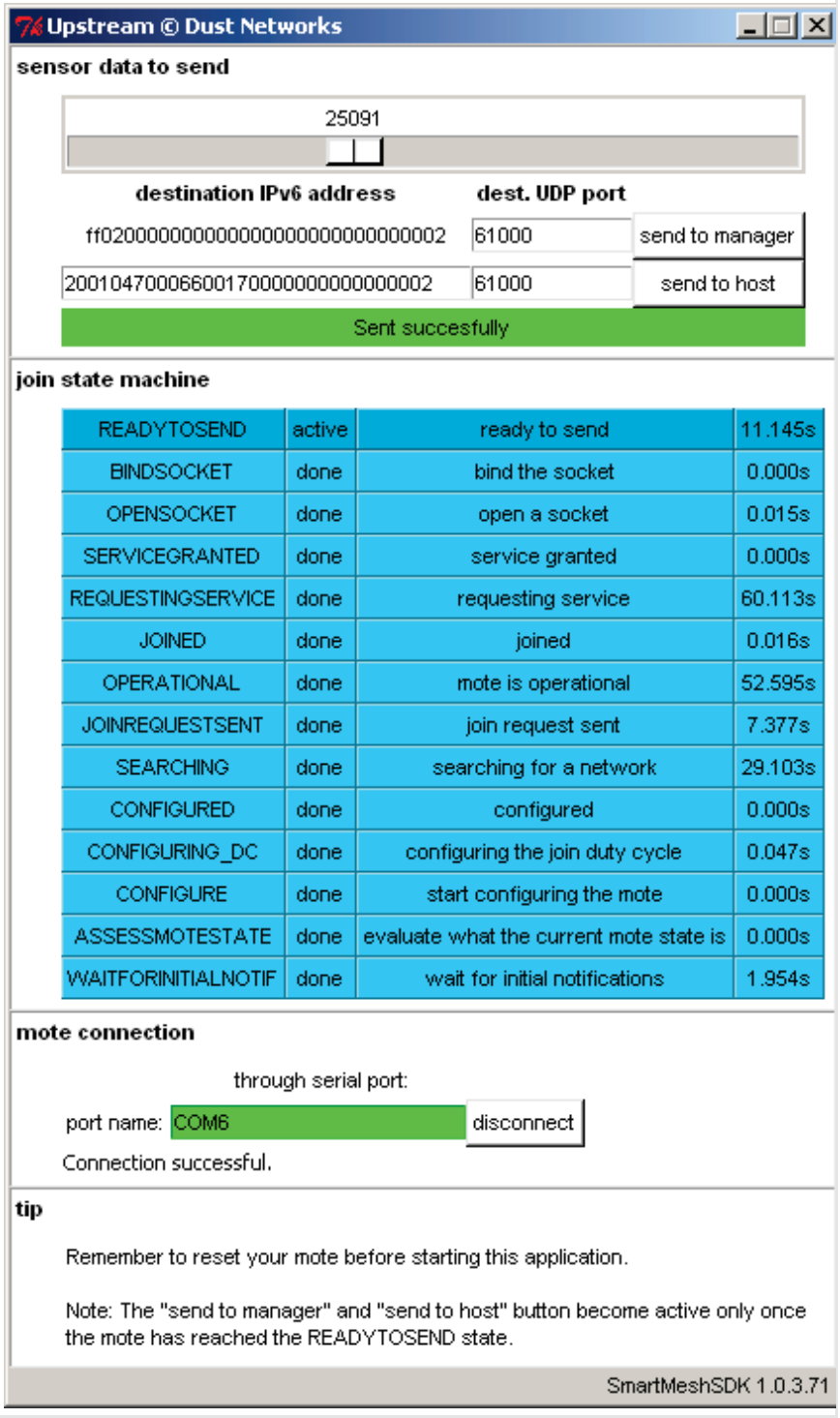


motedata.dustnetworks.com へのデータの送信

1. ブラウザで http://motedata.dustnetworks.com/ページを開きます。12 個のゲージに、最後にシステムに接続した 12 個の SmartMesh IP モートによって送信されたデータが表示されています。
2. UpStream アプリケーションで、以下の事項を確認します。
    1. 2 番目の IPv6 宛先アドレスが、*20010470006600170000000000000002*（*motedata.dustnetworks.com* サーバーの IPv6 アドレス）に設定されている。
    2. 2 番目の宛先 UDP ポートが *61000*（*motedata.dustnetworks.com* サーバーがリスニングしている UDP ポート）に設定されている。
3. Upstream アプリケーションで、**sensor data to send** のスライダを任意の値（ここでは *25091*）に設定し、**send to host** ボタンをクリックします。
4. これにより、以下の順番で処理が実行されます。
    1. SmartMesh IP モートが、IPv6 アドレス *20010470006600170000000000000002* の UDP ポート *61000* に対して、スライダで選択した値をペイロードとして含む UDP パケットを送信します。
    2. パケットが SmartMesh IP マネージャに到着すると、SmartMesh IP マネージャは自身が宛先ではないことを認識して、LBRConnection アプリケーションにパケットを渡します。
    3. LBRConnection アプリケーションはパケットを LBR に転送します。**transmitted to LBR** カウンタの値が増加します。
    4. LBR は 6LoWPAN から IPv6 にパケットを復元し、このパケットを IPv6 インターネットに送信します。
    5. IPv6 インターネットは、*motedata.dustnetworks.com* サーバーにパケットを転送します。
    6. *motedata.dustnetworks.com* サーバーは、受け取ったデータを保存します。
    7. ブラウザに表示された http://motedata.dustnetworks.com/ページが定期的にリフレッシュされ、データを表示したゲージが更新されます。

http://motedata.dustnetworks.com/	**SmartMesh IP マネージャ側**
	

SmartMesh IP モート側



Upstream アプリケーションでは、**send to manager** ボタンを使用するとマネージャにデータが送信され（SensorDataReceiver アプリケーションにデータを表示）、**send to host** ボタンを使用するとインターネットにデータが送信されます（http://motedata.dustnetworks.com/にデータを表示）。

これらのデータ・ストリームは互いに無関係であり、インターネットに送信されたデータが SensorDataReceiver アプリケーションに表示されることはなく、その逆もまた同様です。

一般的な問題

アプリケーションがモートに接続されない

- モートの電源は入っていますか。
- アプリケーションがデバイスの API ポートに接続されていますか。
- このポートにあらかじめ接続しているアプリケーションがほかにありませんか。

以下の表に、全てのデバイスに使用できるシリアル設定を示します。

デバイス	シリアル・ポート番号	用途	ボーレート	データ・ビット	パリティ	ストップ・ビット
SmartMesh IP マネージャ	3 番目*	CLI	9600	8	N	1
	4 番目*	API	115200**	8**	N**	1**
SmartMesh IP モート	3 番目*	CLI	9600	8	N	1
	4 番目*	API	115200**	8**	N**	1**

* FTDI ドライバによって作成されたシリアル・ポート
** デフォルト値

モートがネットワークにジョインしない

- ジョインのデューティ・サイクルとネットワークの状態によっては、SmartMesh IP モートがジョインするまでに何分かかかる場合があります。
- SmartMesh IP モートに設定されたネットワーク ID が、SmartMesh IP マネージャと同じであることを確認します。

9 Low-power Border Router

9.1 Low-power Border Router とは

Low-power Border Router（LBR）は、SmartMesh IP ネットワークからインターネットに接続するためのネットワーク・デバイスです。LBR は、SmartMesh IP マネージャとインターネットの間に配置されます。インターネットの IPv6 パケット形式を SmartMesh IP ネットワークの 6LoWPAN パケット形式に変換して、インターネット上のコンピュータ（インターネット・ホスト）と SmartMesh IP モートが通信できるようにします。

LBR は、以下の 3 つの役割を果たします。

- **接続**：SmartMesh IP モートからインターネット上のサーバーにデータを送信し、インターネット上のホストから SmartMesh IP モートにデータを送信できるようにします。
- **圧縮**：データがインターネットと SmartMesh IP ネットワークの間でやり取りされるとき、圧縮/復元エンジンが IPv6 から 6LoWPAN への変換を実行します。
- **アドレス割り当て**：IPv6 アドレスのプールを管理することで、それぞれの SmartMesh IP モートにグローバルなアクセスが可能な一意の IPv6 アドレスを設定します。

「Low-power Border Router」という用語は、IETF work group ROLL によって RFC6550 内に定義されています。

LBR は、以下の 2 つのモードで稼働するコンピュータ・プログラムです。

- **スタンドアロン・モード**では、SmartMesh IP マネージャに接続されたコンピュータで実行され、1 つの SmartMesh IP ネットワークに対応します。
- **サーバー・モード**では、インターネット上の任意の場所にあるサーバー上で実行され、複数の SmartMesh IP ネットワークからの接続を受け入れます。

9.2 ドキュメントの構成

- 概要：
- デモ用リソースの使用：デモ用 LBR に接続して、モートからインターネットにデータを送信する方法を紹介します。デモ用リソースを使用すると、何もインストールせずに LBR を試すことができます。
- インストール：LBR のインストール方法を説明します。LBR の機能を試すことのみが目的である場合は、デモ用リソースの使用を使用することをお勧めします。
- ユーザー・ガイド：ユーザー管理を通じて LBR を管理する方法について説明します。
- CLI ガイド：LBR のコマンド・ライン・インタフェース（CLI）のリファレンス・ガイドとなる詳しい説明を提供します。

9.3 概要

9.3.1 LBR の目標

SmartMesh IP ネットワークは IPv6 に対応しているため、それぞれの SmartMesh IP モートに IPv6 アドレスを割り当てることができ、交換されるパケットは 6LoWPAN 規格（IPv6 の圧縮バージョン）に準拠しています。

LBR は、SmartMesh IP マネージャをインターネットにつなぎます。リンクが確立されると、SmartMesh IP モートは生成したデータを直接インターネット・ホストに送信することができます。別々の SmartMesh IP モートが、それぞれ異なるホストにデータを送信できます。同様に、インターネット・ホストはネットワーク内の個々の SmartMesh IP モートにデータを送信できます。

つまり、SmartMesh IP モートをインターネット上のホストとまったく同じように使用できます。

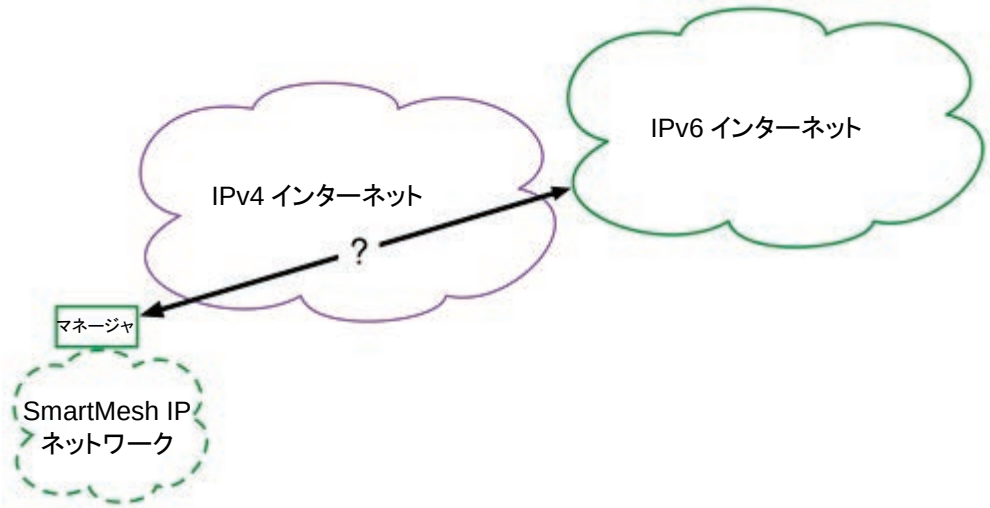
> 6LoWPAN 規格と LBR の概念を開発したのは、全てのインターネット関連規格を支える標準化団体の Internet Engineering Task Force です。

9.3.2 サービス

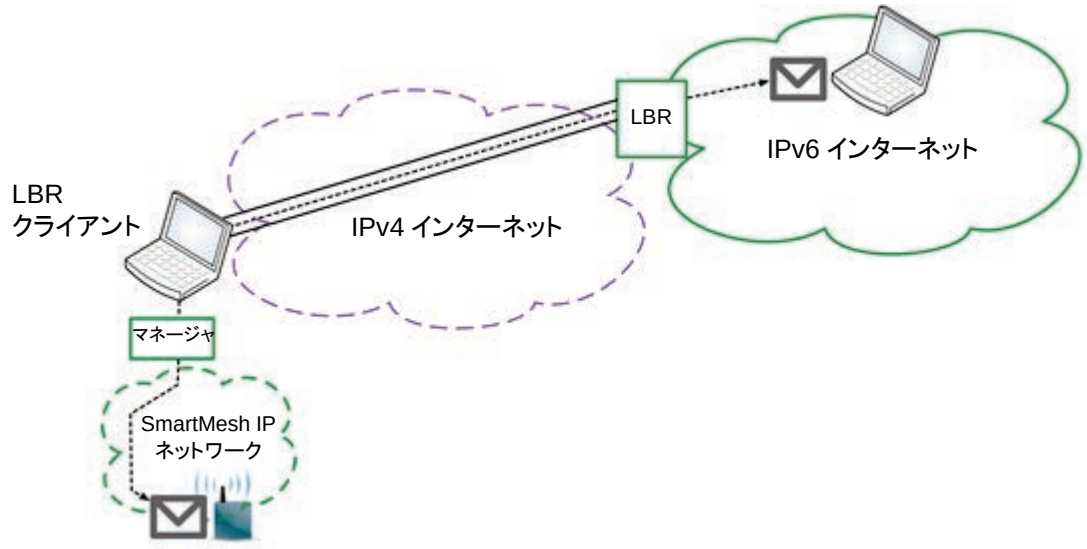
LBR は、以下の 3 つの役割を果たします。

- **接続**：SmartMesh IP モートからインターネット上のサーバーにデータを送信し、インターネット上のホストから SmartMesh IP モートにデータを送信できるようにします。
- **圧縮**：データがインターネットと SmartMesh IP ネットワークの間でやり取りされるとき、圧縮/復元エンジンが IPv6 から 6LoWPAN への変換を実行します。
- **アドレス割り当て**：IPv6 アドレスのプールを管理することで、それぞれの SmartMesh IP モートにグローバルなアクセスが可能な一意の IPv6 アドレスを設定します。

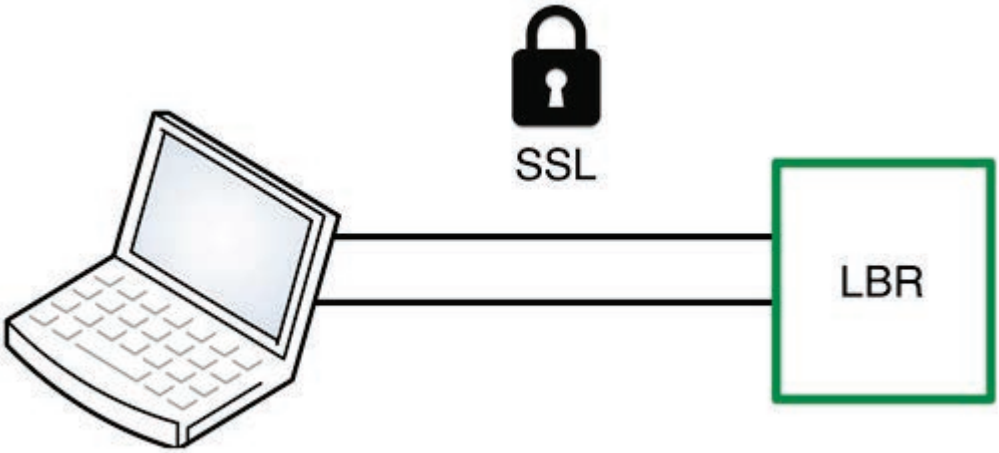
接続



現在のインターネットは、IPv4 という旧バージョンと IPv6 という新バージョンの 2 種類で構成されています。SmartMesh IP ネットワークは IPv6 に対応していますが、現在のインターネットの大半は依然として IPv4 バージョンで稼働しています。LBR は両方のバージョンを認識するため、IPv4 インターネットのみがサポートされている場所にある SmartMesh IP マネージャでも LBR に接続できます。

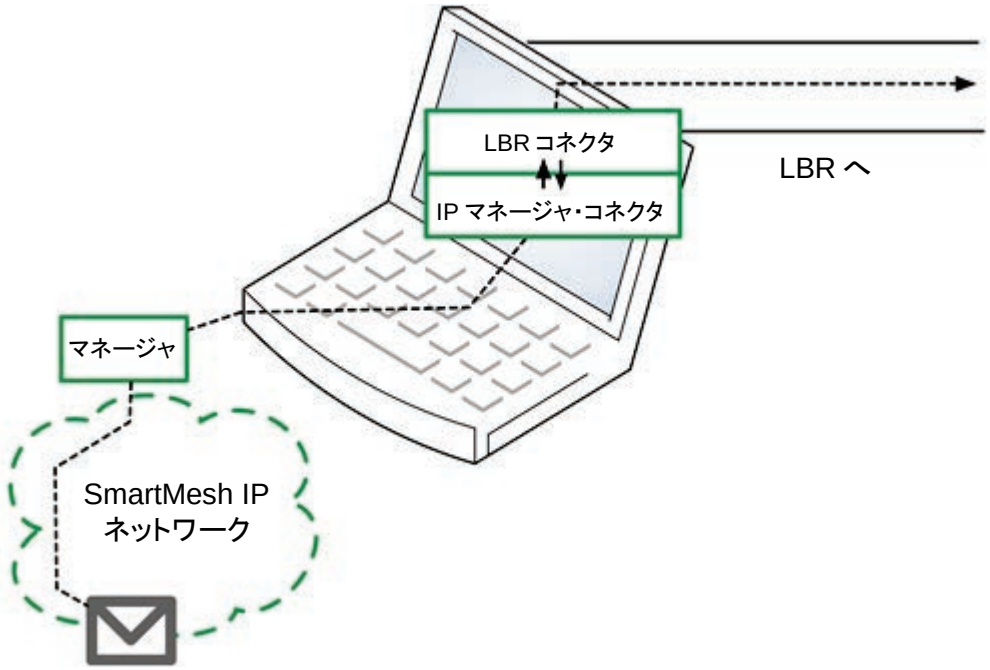


SmartMesh IP マネージャが LBR に接続されると、SmartMesh IP ネットワークと IPv6 インターネットの間で交換される IPv6 パケットは、SmartMesh IP マネージャと LBR 間の接続を介して「トンネリング」されます。



LBR クライアント

機密性、データ整合性、認証能力を維持するため、LBR クライアント（SmartMesh IP マネージャに接続）と LBR の間のセッションは Secure Sockets Layer（SSL）を介して保護されます。

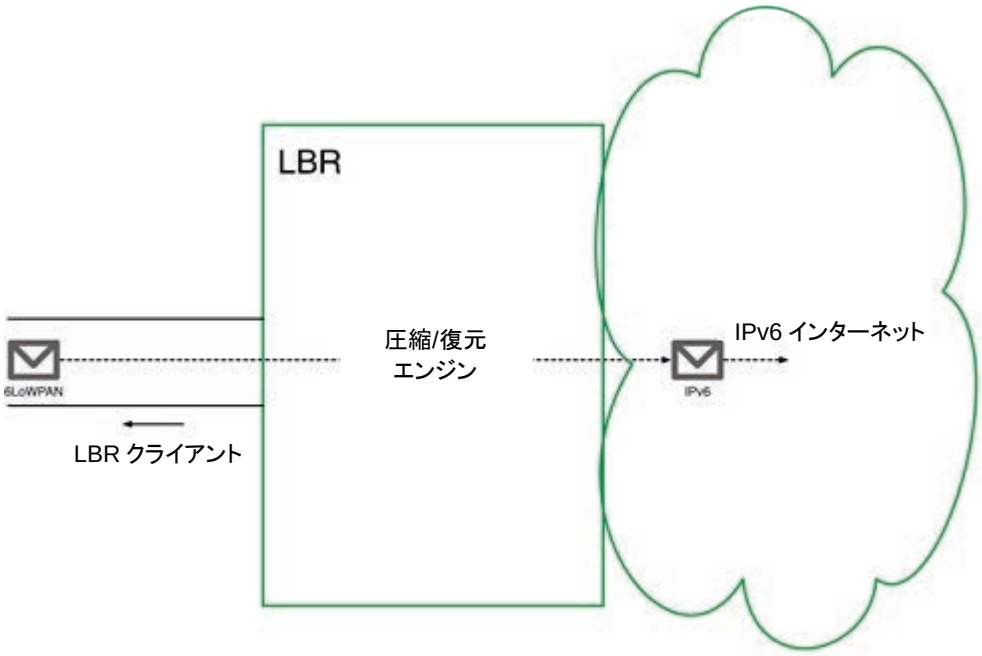


LBR クライアントは、SmartMesh IP マネージャに物理接続されたコンピュータ上で実行されるコンピュータ・プログラムであり、以下の 2 つのコンポーネントで構成されています。

- *IpMgrConnector* は、SmartMesh IP マネージャに接続します。
- *lbrConnector* は、LBR に接続します。

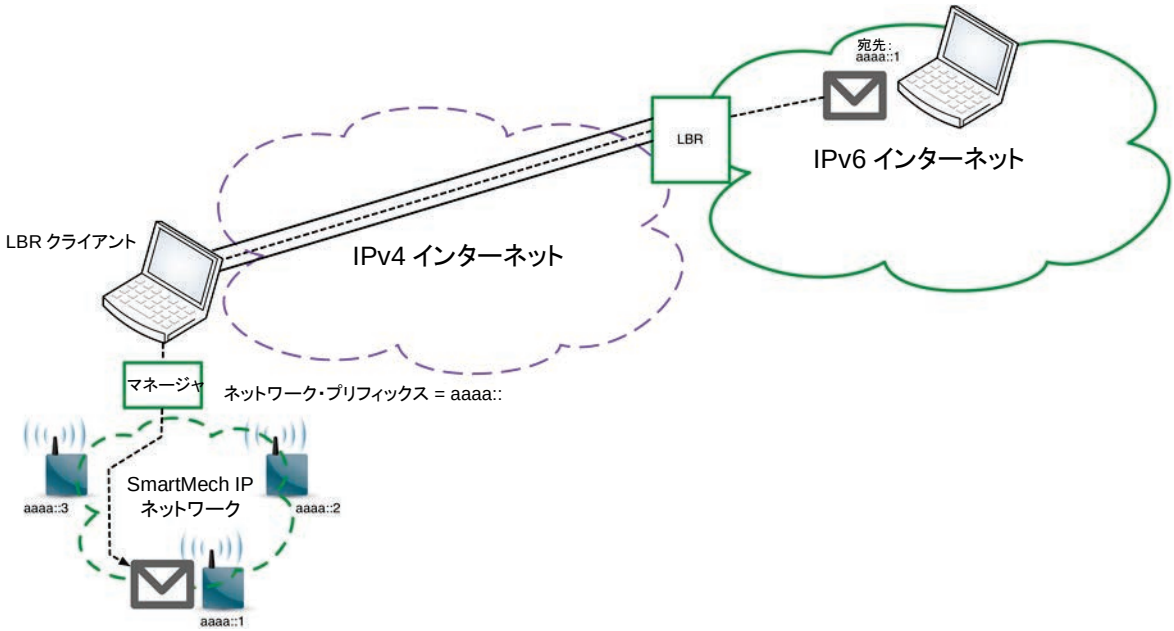
LBR クライアント・プログラムは SmartMesh SDK に含まれています。

圧縮



効率を高めるため、SmartMesh IP ネットワークでは、IPv6 プロトコルの圧縮バージョンである 6LoWPAN プロトコルが使用されます。LBR に搭載された圧縮/復元エンジンは、SmartMesh IP ネットワークとインターネットの間でデータが交換される過程で、6LoWPAN から IPv6 への変換を実行します。

アドレス割り当て



サーバー・モードで実行されている場合、LBR は IPv6 アドレスのプールを管理して、接続された複数の SmartMesh IP ネットワークのそれぞれにアドレスを割り当てます。クライアントが LBR に接続すると、LBR は認証後に、ネットワーク「プリフィックス」をクライアントに割り当てます。ネットワーク・プリフィックスは、SmartMesh IP ネットワーク内の各ノードが持つ IPv6 アドレスのネットワーク部分です。

SmartMesh IP モートはこのプリフィックスを使用して、ネットワーク内の各 SmartMesh IP モートの IPv6 アドレスを自動的に構成します。それぞれの SmartMesh IP モートに対して、グローバルにアクセス可能な一意の IPv6 アドレスが割り当てられます。

9.4 デモ用リソースの使用

ここでは、デモ用 LBR に接続して、モートからインターネットにデータを送信する方法を紹介します。デモ用リソースを使用すると、何もインストールせずに LBR を試すことができます。

9.4.1 実行方法

「インターネット統合」チュートリアルの手順に従って、下記のデモ用リソースに接続します。LBR をインストールする必要はありません。

9.4.2 デモ用リソース

lbr.dustnetworks.com

以下の LBR が、デモを目的としてサーバー・モードでセットアップされています。

DNS 名	*lbr.dustnetworks.com*
TCP ポート	*80*
IPv4 アドレス	*67.203.88.56*
IPv6 アドレス	*2001:470:1f04:1ca6::2* （省略しない場合は、*200104701f041ca60000000000000002*）
モートのプリフィックス*	*2001:470:819f*

* LBR によって管理された/48 IPv6 プリフィックスです。LBR に接続された各 SmartMesh IP ネットワークが、*2001:470:819f:xxxx/64* という形式のプリフィックスを受け取ります。*xxxx* はユーザー・アカウントによって異なります。

この LBR は TCP ポート 80 をリスニングしていますが、**Web サーバーではない**ため、ブラウザからアクセスすることはできません。

この LBR が受け入れるのは、ゲスト・アカウントだけです。

motedata.dustnetworks.com

以下のデータ・サーバーがデモを目的として用意されています。

DNS 名	*motedata.dustnetworks.com*
URL	http://motedata.dustnetworks.com/
IPv4 アドレス	*67.203.88.55*
IPv6 アドレス	*2001:470:66:17::2* （省略しない場合は、*20010470006600170000000000000002*）
UDP ポート*	*61000*

*SmartMesh IP モートから送信されるデータを受け取るために、サーバーがリスニングする UDP ポートです。

このサーバーは以下の機能を実行します。

- Upstream アプリケーションから送信されたデータを受け取って、保存するデーモン・プロセス
- 直前の 12 個の SmartMesh IP モートから受け取った最新データを表示する静的 Web ページ

9.5 インストール

ここでは、LBR のインストール方法を紹介します。LBR の機能を試すことのみが目的である場合は、デモ用リソースを使用することをお勧めします。

9.5.1 前提条件

オペレーティング・システム

LBR は、最近のほとんどの Linux バージョン上で動作します。また、以下のサービスを使用できる必要があります。

- LBR プロセスが、**tun/tap** 仮想カーネル・ネットワーク・デバイスを作成/破棄できる必要があります。LBR はクライアントに接続するたびに新しい *tun* インタフェースを作成し、クライアントの接続が切断されると、このインタフェースを破棄します。
- LBR プロセスが、**TCP ポート 80**（LBR が接続をリスニングするデフォルト・ポート）にソケットをバインドできる必要があります。
- **IPv6** がサポートされており、特に以下に対応している必要があります。
  - IPv6 転送が有効化されていること
  - LBR プロセスが、管理対象の各 *tun* インタフェースに任意の IPv6 アドレスを割り当てられること

- LBR プロセスから、以下の表に記載したシステム・コマンドを呼び出せる必要があります。

コマンド	説明
ping6	ICMPv6 エコーのリクエストおよびレスポンスを送受信するユーティリティ
ifconfig	LBR が管理する *tun* インタフェースの IPv6 アドレスを設定するユーティリティ
route	Linux カーネルのルーティング表を管理するユーティリティ

- LBR プロセスには、LBR インストール・ディレクトリの *bin/dustlbr/temp/* に対する書き込み権限が必要です。

最近の Linux バージョンの大半は、標準で、上記に示したオペレーティング・システム要件に準拠しています。

Python

LBR プログラムを実行するには、Python 2.6 または Python 2.7 がインストールされている必要があります。

IPv6 コネクティビティ

LBR は/48 IPv6 プリフィックスを管理しており、接続する SmartMesh IP ごとに 1 つの/64 プリフィックスに分割します。

/48 プリフィックスの入手先は次のとおりです。

- ローカル・ネットワークで IPv6 がサポートされている場合は、ネットワーク管理者
- Hurricane Electric などの IPv6 トンネル・プローカ

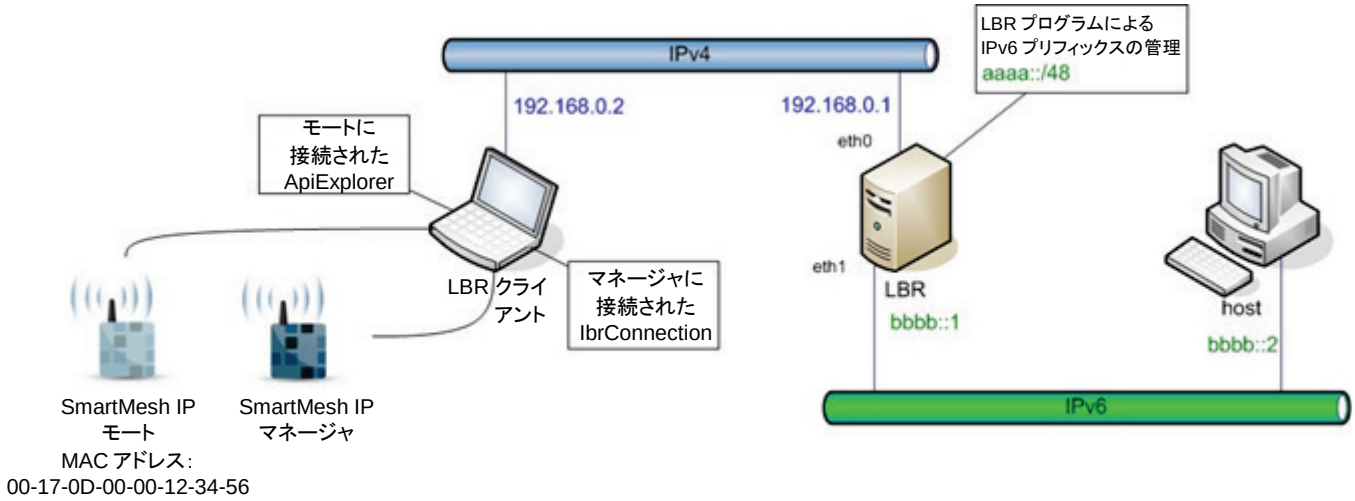
この/48 は LBR に転送される必要があります。つまり、このプリフィックスに属するアドレスに向けた全てのトラフィックは、最終的に LBR に到着します。

9.5.2 インストール手順

LBR をインストールしてその動作を確認するには、以下の手順に従います。この手順では、以下の処理を実行します。

- SmartMesh IP マネージャから LBR への接続と、SmartMesh IP ネットワークの IPv6 プリフィックスの取得
- SmartMesh IP モートからインターネット・ホストへのデータ送信
- インターネット・ホストから SmartMesh IP モートへのデータ送信

トポロジ



上の図は、インストール手順で使用する接続とアドレス割り当てを示したものです。必要に応じて適切なアドレスに変更してください。

設定を確認するには、以下の 2 台のマシンが必要です。

- SmartMesh IP マネージャに接続して、LBRConnection アプリケーションを実行する **LBR クライアント**
- IPv6 に接続され、SmartMesh IP モートとデータを交換する**ホスト**

⚠ 上の図のテスト用セットアップでは、LBR と LBR クライアントが同じイーサネット・リンク上に配置されており、LBR とホストも同一のイーサネット・リンク上に配置されています。これは必要条件ではありません。本番セットアップでは 3 つのマシン全てが別々の場所に配置されます。

LBR のインストール

IPv6 転送の有効化

Debian を使用する場合:

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

Ubuntu を使用する場合:

/etc/sysctl.conf ファイルを編集して、以下の行のコメントを外します。

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

⚠ 場合によって、この変更を反映するために、ネットワーク・サービスの再起動またはマシンのリブートが必要になります。

変更が反映されたことを確認します。

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

インタフェースの構成

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

接続の確認

ping コマンドを使用して、以下に示すマシン間が相互接続されていることを確認します。

- LBR から
  - LBR クライアント
  - ホスト
- LBR クライアントから
  - LBR
- ホストから
  - LBR

LBR プログラムのインストール

LBR-1.0.0.2.zip ファイルを解凍します。本書では、生成されたフォルダを LBR ルート・フォルダと呼びます。

LBR プログラムの起動

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

LBR コマンド・プロンプトが表示されます。

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

LBR クライアントから LBR への接続

LBR クライアントで以下の手順を実行します。

1. SmartMesh SDK のインストール・ディレクトリに移動します。
2. *bin/LBRConnection/guest.lbrauth* ファイルを開きます。
3. *lbrAddr* フィールドに LBR の IPv4 アドレス（ここでは *192.168.0.1*）を入力し、ファイルを保存します。
4. *LBRConnection.py* をダブルクリックして、LBRConnection アプリケーションを起動します。
5. LBR connection フレームで *connect* をクリックし、*guest.lbrauth* ファイルを選択します。アプリケーションが LBR に接続されます。
6. manager connection フレームで SmartMesh IP マネージャに接続します。

LBR によって、プリフィックス *aaaa:0000:0000:abcd* が SmartMesh IP ネットワークに割り当てられます。*abcd* は、LBR がランダムに割り当てるサブプリフィックスです。

モートからホストへのデータ送信

ホストで以下の手順を実行します。

1. netcat ユーティリティを起動して、UDP ポート 61000 に送信される IPv6 パケットをリスニングします。

```
nc -6lu 61000
```

LBR クライアントで以下の手順を実行します。

1. SmartMesh SDK のインストール・ディレクトリに移動します。
2. APIExplorer アプリケーションを起動します。
3. アプリケーションを SmartMesh IP モートに接続します。

4. 「インターネット統合」の手順に従って、以下のパケットを送信します。
   - 宛先アドレス(**destIP** フィールド):*bbbb0000000000000000000000000002*
   - 宛先ポート(**destPort** フィールド):*61000*
   - パケット・ペイロード:*706f69*(ASCII 文字列の「poi」に相当)

SmartMesh IP モート側で send ボタンをクリックすると、ホスト側の netcat ユーティリティに「poi」と出力されます。

ホストからモートへのデータ送信

LBR クライアントで以下の手順を実行します。

1. APIExplorer アプリケーションを使用して *getParameter.macAddress* コマンドを実行し、SmartMesh IP モートの MAC アドレスを読み取ります(ここでは、*00170d0000123456*)。
2. 取得した MAC アドレスを SmartMesh IP ネットワークのプリフィックスの末尾に追加すると、SmartMesh IP モートの IPv6 アドレスになります。ここでの例を具体的に説明します。
   1. SmartMesh IP ネットワークのプリフィックスは、*aaaa00000000abcd* です。
   2. SmartMesh IP モートの MAC アドレスは、*00170d0000123456* です。
   3. したがって、SmartMesh IP モートの IPv6 アドレスは、*aaaa00000000abcd00170d0000123456* になります。
3. SmartMesh IP モートのソケットが開いており、UDP ポート *60000* にバインドされていることを確認します。
4. APIExplorer アプリケーションは開いたままにしておきます。

ホストで以下の手順を実行します。

1. netcat ユーティリティを使用して、SmartMesh IP モートのポート *60000* に文字列「poi」を送信します。

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

ホスト側の *nc* ユーティリティで Enter キーを押すと、SmartMesh IP モートが以下の *receive* 通知を受け取ります。

- *socketId*:3(UDP ポート *60000* にバインドされたソケット)
- *srcIP*:*bbbb0000000000000000000000000002*(ホスト・コンピュータの IPv6 アドレス)
- *srcPort*::*nc* アプリケーションに使用される(ランダムな)UDP ポート
- *payload*:*706f69*(ASCII 文字列「poi」の 16 進表現)

以上で、Low-power Border Router のインストールとテストは終了です。

9.6 ユーザー・ガイド

このページでは、ユーザー管理を通じて LBR を管理する方法について説明します。

LBR がすでにインストールされていることを前提としています。まだインストールしていない場合は、「インストール」に記載された手順に従ってください。

9.6.1 セキュリティ・レベル

ここで説明するセキュリティ・レベルが適用されるのは、LBR と LBR クライアントの間のセッションのみです。SmartMesh IP モートとインターネット・ホストの間のエンドツーエンド・セッションには適用されません。

サポートされるセキュリティ・レベルは以下のとおりです。

レベル	名前	説明
0	*none*	LBR と LBR クライアントの間のセッションが TCP セッションになります。ユーザーの認証は必要ありません。このセキュリティ・レベルを使用するのはゲスト・アカウントです。
1	*password*	LBR と LBR クライアントの間のセッションが TCP セッションになります。TCP セッションが確立された後、**平文で**送信されるパスワードによってユーザーが認証されます。
2	*ssl*	LBR と LBR クライアントの間のセッションが SSL（Secure Sockets Layer）セッションになります。ユーザーと LBR は、送信鍵と秘密鍵に基づくセキュアなハンドシェイクによって認証されます。LBR と LBR クライアントの間で送信されるデータは暗号化されます。

使用例：

- 推奨されるのは、最も高いセキュリティを提供するセキュリティ・レベル 2（*ssl*）です。
- セキュリティ・レベル 1（*password*）は、SSL を使用できない場合にのみ使用します。
- セキュリティ・レベル 0（*none*）は、テスト用のゲスト・アカウントでのみ使用します。

9.6.2 ユーザー・アカウントの種類

LBR がサポートするユーザー・アカウントは 2 種類あります。

ゲスト・アカウント

ゲスト・アカウントを使用して LBR に接続する場合、以下の点に関しては正規アカウントと同じになります。

- ネットワークに IPv6 プリフィックスが割り当てられる
- ネットワーク内の SmartMesh IP モートがインターネット上のホストとデータを交換できる

ゲスト・アカウントの制約事項は以下のとおりです。

- LBR クライアントと LBR の間の接続がセキュリティ保護されない
- ユーザーがゲストとして接続するたびに、ばらばらなランダムのプリフィックスがネットワークに割り当てられる

ゲスト・アカウントを使用して LBR に接続するには、以下のパラメータを指定します。

ユーザー名	セキュリティ・レベル
guest	0(*none*)

複数のゲストが同時に LBR に接続できます。

正規アカウント

正規のユーザー・アカウントを使用するには、LBR 管理者が該当するユーザーを LBR 上のユーザー・データベースに追加する必要があります。

正規ユーザー・アカウントを識別する要素は次のとおりです。

- 一意のユーザー名
- 一意のサブプリフィックス(2 バイト)セキュリティ・レベル
- ユーザー認証用の資格情報

資格情報の種類はユーザーのセキュリティ・レベルに応じて、以下のように異なります。

- レベル 0:なし
- レベル 1:パスワード
- レベル 2:LBR クライアントの送信鍵/秘密鍵の組み合わせ、LBR の送信鍵

9.6.3 LBR の鍵生成データのインストール

セキュリティ・レベル 2(*ssl*)を持つユーザーを追加する場合、その前にこのステップを実行しておく必要があります。

このステップでは、送信鍵/秘密鍵の組み合わせを LBR で生成してインストールします。

この鍵生成データは「PEM」（詳しくは、RFC 1422 を参照）形式にする必要があります。PEM は、Base 64 エンコード形式にヘッダ行とフッタ行を付けたものです。以下の手順では、OpenSSL モジュール（通常、全ての Linux ディストリビューションにインストール済み）を使用していますが、その他の同等の方法を使用することもできます。

1. LBR で、*lbr/bin/dust*lbr/keys/ディレクトリに移動します。
2. 以下のコマンドを入力します。

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

3. プロンプトに従って、LBR についてのさまざまな情報を入力します。この情報は鍵生成データ内に保存され、LBR に接続した全てのユーザーが参照できる状態になります。
4. このコマンドを実行すると、以下の 2 つのファイルが生成されます。
   - LBR の秘密鍵を含むファイル：*serverkey.pem*
   - LBR の送信鍵を含むファイル：*servercert.pem*（自己署名証明書）

⚠ これらの鍵には、上記コマンドの *-days* オプションで指定された存続期間が設定されます。アプリケーションに合わせて適切な値を設定してください。

ファイルを開いて、実際に PEM 形式になっているかどうかを確認できます。

```
-----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-----
```

9.6.4 ユーザーの追加

LBR 認証ファイル.lbrauth

SmartMesh SDK の LBRConnection アプリケーションを使用して LBR に接続するとき、ユーザーが LBR 認証ファイルを選ぶ必要があります。以下の手順では、ユーザーの種類に合わせて認証ファイルを構築する方法を説明します。

LBR 認証ファイルは、多数の「*key = value*」の組み合わせで構成されます。

> ⚠ LBR 認証ファイルでは、等号(=)の両側に少なくとも 1 つの空白を入れる必要があります。以下に例を示します。
>
> - 「*key=value*」は誤った指定方法です。
> - 「*key = value*」は正しい指定方法です。

ゲスト・ユーザー

LBR の構成

ゲスト・ユーザーを受け入れるために、LBR に特別な設定を行う必要はありません。

LBR 認証ファイルの作成

LBRConnection アプリケーションでの使用を目的としてクライアントに配布する LBR 認証ファイルは、以下のようになります。

```
# 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
```

セキュリティ・レベル 0 を持つ正規ユーザー

ここでは、以下のユーザーの作成方法を説明します。

ユーザー名	サブプリフィックス	セキュリティ・レベル	認証情報
user_seclevel_0	*ab00*	0(*none*)	*なし*

LBR の構成

LBR で以下のコマンドを入力し、新規ユーザーを追加します。

```
> add user_seclevel_0 ab00
```

デフォルトで新規ユーザーにはセキュリティ・レベル 0 が設定されるため、その他の設定は必要ありません。

LBR 認証ファイルの作成

LBRConnection アプリケーションでの使用を目的としてクライアントに配布する LBR 認証ファイルは、以下のようになります。

```
# 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
```

セキュリティ・レベル 1 を持つ正規ユーザー

ここでは、以下のユーザーの作成方法を説明します。

ユーザー名	サブプリフィックス	セキュリティ・レベル	認証情報	
user_seclevel_1	*ab01*	1 (*password*)	パスワード	*user_password*

LBR の構成

LBR で以下のコマンドを入力し、新規ユーザーを追加します。

```
> add user_seclevel_1 ab01
 > passwordset user_seclevel_1 user_password
 OK.
 > seclevel user_seclevel_1 password
 OK.
```

LBR 認証ファイルの作成

LBRConnection アプリケーションでの使用を目的としてクライアントに配布する LBR 認証ファイルは、以下のようになります。

```
# 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
```

セキュリティ・レベル 2 を持つ正規ユーザー

ここでは、以下のユーザーの作成方法を説明します。

ユーザー名	サブプリフィックス	セキュリティ・レベル	認証情報	
user_seclevel_2	*ab02*	2 (*ssl*)	LBR の送信鍵	*servercert.pem* ファイルから生成
			LBR クライアントの送信鍵	OpenSSL を使用して生成
			LBR クライアントの秘密鍵	OpenSSL を使用して生成

ユーザーの送信鍵/秘密鍵の生成

はじめに、LBR クライアントの送信鍵/秘密鍵の組み合わせを表す文字列を生成します。

1. OpenSSL がインストールされたマシンで、以下のコマンドを入力します。

```
openssl req -new -x509 -days 365 -nodes -out clientcert.pem -keyout clientkey.pem
```

2. プロンプトに従って、クライアントについてのさまざまな情報を入力します。この情報は鍵生成データ内に保存され、LBR クライアントが接続した LBR から参照できる状態になります。
3. このコマンドを実行すると、以下の 2 つのファイルが生成されます。
   - LBR クライアントの秘密鍵を含むファイル:*clientkey.pem*
   - LBR クライアントの送信鍵を含むファイル:*clientcert.pem*(自己署名証明書)

⚠ これらの鍵には、上記コマンドの *-days* オプションで指定された存続期間が設定されます。アプリケーションに合わせて適切な値を設定してください。

ファイルを開いて、実際に PEM 形式になっているかどうかを確認できます。

```
-----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-----
```

LBR にこれらの鍵を入力するか、または LBR 認証ファイルに書き込むために、ファイルの内容を 1 つの文字列に変換する必要があります。具体的には、以下の要素を削除します。

- 改行コード
- ヘッダ行とフッタ行

上記処理から得られる文字列は、以下のようになります。ここでは、表示上の理由から末尾を切り捨てています。

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

```
MIICXQIBAAKBgQCtR2w+MFGgBpp5iBwzNxUYhB8uoLsXMfrq9aofSgsYlHER0oP67m/qkD/0DMsOxL0nrF... <truncated>
```

```
MIICsDCCAhmgAwIBAgIJAPMne+4vew7eMA0GCSqGSIb3DQEBBQUAMEUxCzAJBgNVBAYTAkFVMRMwEQYDVQ... <truncated>
```

LBR の構成

ユーザーを認証するために LBR 側に指定する必要があるのは、ユーザーの**送信鍵のみ**です。以下の文字列を 1 行で入力します。ここでも、表示上の理由から末尾を切り捨てています。

```
> add user_seclevel_2 ab02
 > publickeyset user_seclevel_2 MIICsDCCAhmgAwIBAgIJAOBABlW4RBllMA0GCSqGSIb3DQEBBQU... <truncated>
 > seclevel user_seclevel_2 ssl
 OK.
```

LBR 認証ファイルの作成

LBR 認証ファイルには、**ユーザーの送信鍵/秘密鍵**と **LBR の送信鍵**が含まれる必要があります。ここでも、サイズの大きい鍵は表示用に末尾を切り捨てています。

```
# 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>
```

9.6.5 ユーザーの管理

LBR の CLI を使用すると、LBR プログラムの実行中にユーザーを管理することができます。

このセクションでは、CLI コマンドを使用してユーザーを管理する方法を簡単に説明します。各コマンドの詳細については、「CLI ガイド」を参照してください。

ステータス

users コマンドを使用すると、接続済みユーザーの概要が表示されます。

```
> 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
```

ユーザー名を指定すると、そのユーザーについての統計情報を含む詳細情報が表示されます。

```
> 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
```

ロギング

LBR は LBR コアだけでなく、*bin/dustlbr/logs/*フォルダ内の各ユーザーのアクティビティをロギングします。

- *system.log* には、LBR コアのロギング情報が含まれます。
- *user_*.log* ファイルには、特定のユーザーに関するロギング情報が含まれます。

```
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
```

loglevel コマンドを使用すると、各ユーザーのログ・レベルを指定できます（詳しくは、「CLI ガイド」を参照）。

接続の解除

CLI コマンドの *disconnect* を使用すると、LBR からユーザーの接続を強制的に解除することができます。

ユーザーの削除

CLI コマンドの *remove* を使用すると、ユーザー・データベースからユーザーを削除できます。

ユーザーを削除すると、ユーザーに関連付けられた全ての統計情報も削除されます。

9.6.6 バックアップとリカバリ

全てのユーザー情報はユーザー・データベースに保存され、定期的に *bin/dustlbr/userDB.pkl* ファイルにバックアップされます。

LBR プログラムは起動時のこのファイルを読み取って、最後にファイルにバックアップされたユーザー・データベースを復元します。強制的にバックアップを実行するには、CLI コマンドの *backup* を使用します（詳しくは、「CLI ガイド」を参照）。

9.7 CLI ガイド

このセクションでは、LBR のコマンド・ライン・インタフェース（CLI）のリファレンス・ガイドとなる詳しい説明を提供します。

ここからは、それぞれの CLI コマンドについて説明します。

9.7.1 add

説明
ユーザーを追加します。

構文
add <username> <subprefix>

パラメータ

パラメータ	説明
username	新規ユーザーのユーザー名
subprefix	4 文字の 16 進数で表現されるクライアントのサブプリフィックス（2 バイト、例：*0b12*）

コマンド例
add myuser 0bc2

9.7.2 backup

説明

ユーザー・データベースを *userDB.pkl* ファイルにバックアップします。

構文

backup

パラメータ

パラメータ	説明

コマンド例

backup

9.7.3 disconnect

説明
現在接続しているユーザーの接続を解除します。

構文
disconnect <username>

パラメータ

パラメータ	説明
username	接続を解除するユーザーのユーザー名

コマンド例
disconnect myuser

9.7.4 help

説明
使用できるコマンドの一覧を表示します。

構文
help

パラメータ

パラメータ	説明

コマンド例
> 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

9.7.5 loglevel

説明

特定のユーザーのログレベルを変更します。

構文

loglevel <username> <loglevel>

パラメータ

パラメータ	説明
username	対象となるユーザー名、または全ユーザーを対象とする場合は *all*
loglevel	設定するログレベル（*debug*、*info*、*warning*、*error*、*critical* のいずれか）

コマンド例

9.7.6 passwordremove

説明

ユーザーに関連付けられたパスワードを削除します。

構文

passwordremove <username>

パラメータ

パラメータ	説明
username	対象となるユーザー名

コマンド例

9.7.7 passwordset

説明

ユーザーにパスワードを設定します。

構文

passwordset <username> <password>

パラメータ

パラメータ	説明
username	対象となるユーザー名
password	設定するパスワード

コマンド例

9.7.8 publickeyremove

説明

ユーザーに関連付けられた送信鍵を削除します。

構文

publickeyremove <username>

パラメータ

パラメータ	説明
username	対象となるユーザー名

コマンド例

9.7.9 publickeyset

説明

ユーザーに関連付ける送信鍵を設定します。

構文

publickeyset <username> <public_key>

パラメータ

パラメータ	説明
username	対象となるユーザー名
public_key	設定する送信鍵

コマンド例

9.7.10 quit

説明

LBR アプリケーションを終了します。

構文

quit

パラメータ

パラメータ	説明

コマンド例

9.7.11 remove

説明

ユーザー・データベースからユーザーを削除します。

構文

remove <username>

パラメータ

パラメータ	説明
username	対象となるユーザー名

コマンド例

9.7.12 seclevel

説明

ユーザーに関連付けるセキュリティ・レベルを設定します。

構文

seclevel <username> <level>

パラメータ

パラメータ	説明
username	対象となるユーザー名
レベル	設定するセキュリティ・レベル（*none*、*password*、*ssl* のいずれか）

コマンド例

9.7.13 status

説明

LBR のステータスを出力します。

構文

status

パラメータ

パラメータ	説明

コマンド例

9.7.14 users

説明

接続済みのユーザーの概要情報を出力します。*users <name>*コマンドを使用すると、特定ユーザーの詳細を出力します。

構文

users [username]

パラメータ

パラメータ	説明
username	対象となるユーザー名

コマンド例

9.7.15 version

説明
LBR のバージョンを出力します。

構文
version

パラメータ

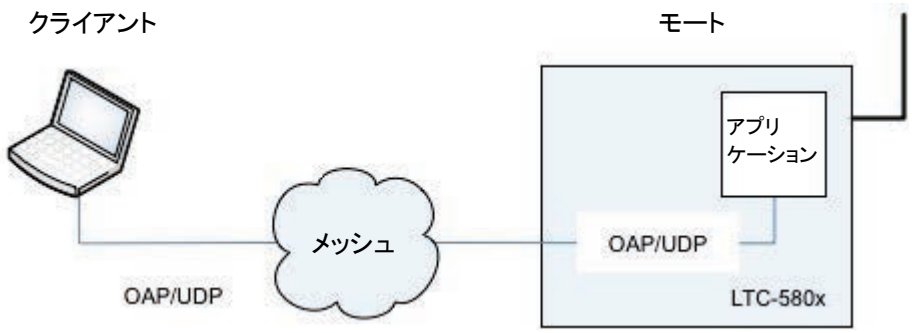
パラメータ	説明

コマンド例

10 オンチップ・アプリケーション・プロトコル

このセクションでは、オンチップ・アプリケーション・プロトコル（OAP）について説明します。OAP は RESTful スタイルのプロトコルであり、モート上で稼働するアプリケーションに対する外部クライアントの接続方法とパケット交換方法をデモンストレーションするために開発されたものです。OAP は UDP 経由で実行され、その概念は CoAP と似ています。OAP はセッション指向のプロトコルで、アクノリッジ制御パケットと非アクノリッジ制御パケットの 2 種類があります。一度に 1 つのクライアントのみが、セッションを使用してモートにリクエストを送信できます（下図を参照）。アクノリッジ制御トラフィックは、クライアントからモートへの一方向のみで送信されます。非アクノリッジ制御トラフィックは、モートから（1 つまたは複数の）クライアントへの一方向のみで送信されます。

OAP は UDP ポート *0xF0B9* を介して実行されます。これは最も圧縮しやすい 6LoWPAN ポート・レンジにあり、最大限のペイロードが使用可能になります。OAP はネットワーク・スタックに含まれていないので、OAP を完全に実装している場合はアプリケーション内で OAP を使用することもできます。



ここからは、以下のセクションごとに説明します。

- プロトコル - 接続と一般的なデータ構成について説明します。
- SmartMesh IP モートに含まれる OAP - SmartMesh IP モート・アプリケーションに含まれる各種の OAP モジュールについて説明します。
- OAP サンプル - よく使用される OAP の交換サンプルを紹介します。

> ⚠ OAP コマンドを終了するには、モートが**マスター・モード**で稼働している必要があります。
>
> スタータ・キット（DC9021A および DC9022A）に付属しているモートは、デフォルトでマスター・モードになっています。消費電力を最小限に抑えるため、**マスター・モード**での動作時には、使用しないデジタル入力（D0～D3）を解除することをお勧めします。

10.1 プロトコル

10.1.1 パケット構成

全ての OAP パケットは共通の構成を持ち、2 バイトのヘッダに可変長のペイロードが続きます。

Control	Id	Payload
1 バイト	1 バイト	0～n バイト

Control フィールド

Control フィールドは以下のサブフィールドで構成されます。

フィールド	ビット位置	説明
Transport	0	0=非アクノリッジ制御、1=アクノリッジ制御
Response	1	0=リクエスト、1=レスポンス
Sync	2	0=通常、1=再同期接続
Reserved	3～7	予約済み、0 に設定

Id フィールド

Id フィールドは以下のサブフィールドで構成されます。

フィールド	ビット位置	説明
Sequence number	0～3	パケット・Sequence
Session ID	4～7	Session ID

10.1.2 通信

OAP は以下の 2 つの通信パターンをサポートしています。

- クライアントからモートへの、信頼性の高いリクエスト/レスポンス通信
- モートからクライアントへのベストエフォート通知

リクエスト/レスポンス・トラフィックを使用して通信を開始するためには、クライアントからモートへの接続を確立する必要があります。OAP はパケットの順序と確実な配信を保証するために、Sequence を使用します。初めてのハンドシェイク中に最初の Sequence が設定され、新しいリクエスト/レスポンスがやり取りされるたびに番号が大きくなります。モートのレスポンスは常に、リクエスト・パケットのソース IP アドレスおよびポートに送信されます。通信時間と帯域幅を節約するため、最初のハンドシェイクにアプリケーション・ペイロードが含まれる場合があります。

クライアント：同期リクエストによる新規接続の開始

通信を確立するために、クライアントは同期リクエストを送信します（Control バイトで *Transport=1*、*Response=0*、*Sync=1* を指定、Sequence に*任意の番号*、Session ID に *0* を指定）。有効なレスポンスの場合、Control バイトに *Transport=1*、*Response=1*、*Sync=1* が設定されており、リクエストと同じ Sequence が含まれます。同期リクエストにリクエスト・ペイロードが含まれる場合、レスポンスにも対応するレスポンス・ペイロードが含まれます。クライアントが有効なレスポンスを受け取ると、接続が確立されたと見なされます。レスポンスの Session ID を、この後モートに送信する全てのリクエスト・パケットに含める必要があります。また、全てのレスポンス・パケットにこの Session ID が含まれることを確認します。パケット交換で使用された Sequence は、*last_sequence_number* として記録され、後続のリクエストが送信されるたびに増加します。

モート：新しい同期リクエストの処理

OAP プロトコルを理解するためには、受信した同期パケットをモートがどのように処理するかを確認すると良いでしょう。モート・アプリケーションは、起動時にランダムな *current_session_id* を選択して、OAP に関連付けられた UDP ポートのリスニングを開始します。Control バイトに *Reliable=1*、*Sync=1*、*Response=0* が設定されたパケットを受け取ると、以下の処理を実行します。

1. *current_session_id* を増やします。
2. 受け取ったパケットの Sequence を *last_sequence_number* に保存します。
3. パケットにペイロードが含まれる場合はこれを処理して、レスポンス・パケットにペイロードを設定します。ペイロードが含まれない場合は、レスポンス・パケットにペイロードを設定しません。
4. Control バイトに *Reliable=1*、*Sync=1*、*Response=1* を、Sequence に last_sequence_number を、Session ID に *current_session_id* を設定したレスポンス・パケットを送信します。

接続の使用

全てのリクエスト/レスポンスに新しい Sequence が付けられており、クライアントとモートの両方が 1 つの接続内で、パケット順序を厳格に守る必要があります。新しい Sequence を使用するには、その前の Sequence が割り当てられたパケットに対して、アクノリッジが返されている必要があります。新しいリクエスト・パケットを送信するには、クライアントが以下の処理を実行します。

1. *last_sequence_number* を増やします。
2. Control バイトに *Transport=Acknowledged*、*Sync=0*、*Response=0* を、Sequence に *last_sequence_number* を、Session ID に *current_session_id* を設定し、パケットを送信します。
3. 応答があるまで待機します。有効なレスポンスには、Control バイトに *Transport=Acknowledged*、*Sync=0*、*Response=1* が設定されており、Sequence として *last_sequence_number* と、正しい Session ID が指定されています。
4. 何の応答も受け取らなかった場合は、リクエストのコピーを再送信する必要があります。再送信できない場合、接続に失敗したと考えられます。クライアントが使用するタイムアウトはネットワークとトポロジによって異なり、本書の範囲外になりますが、少なくとも 3 回の再試行を推奨します。

非同期パケットに対するモートの受信ロジックを以下に示します。こちらもおそらく、プロトコルの理解に役立つでしょう。

- 受け取ったパケットに未知の Session ID が含まれる場合、モートはこれを破棄します。
- パケットに含まれる Sequence が *last_sequence_number*+1（次に受け取ることが予想されるパケット）である場合、

モートはレスポンスを送信し、*last_sequence_number* のコピーを更新します。レスポンス・ペイロードのコピーがキャッシュされます。

- パケットに含まれる Sequence が *last_sequence_number*（重複パケット）である場合、モートはペイロードを処理せずに、レスポンス・コピーのキャッシュを送り返します。
- パケットにその他の Sequence が含まれる場合、モートはこれを破棄します。

単純化のため、OAP プロトコルは同時に 1 つのクライアントのみとやり取りするように設計されています。複数のクライアントがモートに接続しようとすると、最後のクライアントが優先されます。
反対に、リクエスト/レスポンス・トラフィックに厳格な順序を適用する必要がない場合や、複数のクライアントが同じモートに対して並列でパケットを送信する場合、接続の確立と維持は不要であるか、不適切になる可能性があります。このような場合、クライアントは全てのリクエスト・パケットで同期ビットを設定します。こうすることで、モートはリクエスト/レスポンスの交換が 1 回限りの新しい接続としてこれを処理します。

接続の終了

接続には明確な終了はありません。通信の継続が必要なくなった場合、クライアントはいつでもパケット送信を中止することができます。ただし、モートは送信の中止を認識しないため、アプリケーションは通常どおりに稼働を続けます。

モートから送信される非アクノリッジ制御通知

非アクノリッジ制御トラフィック（Transport=Unacknowledged、Response=0）は、モートからクライアントへの一方向のみで送信され、通知を送信するために使用されます。パケットを送信する前に、アプリケーション・レベルで宛先 IP アドレスとポートを設定する必要があります。モートは非アクノリッジ制御トラフィックを受け取ると、これを破棄します。

10.1.3 OAP ペイロード

一般に、OAP の Payload フィールドに設定されて送信されるデータは、モートのアプリケーションによってサポートされる機能によって異なります。ただし、OAP のペイロードは全て共通の構造を持ちます。

リクエスト・パケット（Control バイトに *Transport=1*、*Response=0* を設定）のペイロードには順番に、コマンド ID、アドレス、変数リスト（このアドレスでアクセスされる変数で、Tag-Length-Value 形式でエンコードされている）が含まれます。アドレスと変数については、このセクションの後半で説明します。

コマンド	<アドレス>[<変数><変数><変数>...]
1 バイト	0～n バイト

レスポンス・パケット（Control バイトに *Transport=1, Response=1* を設定）のペイロードには、リクエストと同じコマンドの値が含まれますが、ヘッダにリターン・コード（RC）が含まれます。レスポンスに含まれるアドレスおよび変数リストは以下のとおりです。

コマンド	RC	<アドレス>[<変数><変数><変数>...]
1 バイト	1 バイト	0～n バイト

通知（Control バイトに *Transport=0* を設定）には、1 バイトのコマンド（通知）に続いてアプリケーション固有のペイロードが含まれます。

コマンド	通知データ
1 バイト	0～n バイト

Command フィールド

Command フィールドには、ペイロード・パケットの処理方法が指定されます。OAP に定義されているコマンドは以下のとおりです。

コマンド	値	送信元	説明
GET	0x01	クライアント	1 つまたは複数の変数の値をモートから取得する
PUT	0x02	クライアント	1 つまたは複数のモートの変数の値を更新する
POST	0x03	クライアント	指定された変数値を使用して、アプリケーション・オブジェクトを作成する
DELETE	0x04	クライアント	アドレスと変数値で指定されたオブジェクトを削除する
NOTIFICATION	0x05	モート	各種のモート通知

各コマンドの動作はアプリケーションとペイロードの内容によって異なりますが、RESTful アーキテクチャの一般原則に従います。

Return Code（RC）フィールド

Return Code フィールドには、OAP リクエスト・ペイロードの処理結果が含まれます。

リターン・コード	ID	説明
OK	0	リクエストが正しく処理された
RC_NOT_FOUND	1	オブジェクトが見つからない
RC_NO_RESOURCES	2	リクエストを完了するためのリソースが不足している
RC_UNK_PARAM	3	未知のパラメータ
RC_INV_VALUE	4	無効な値
RC_INV_ADDR	5	無効なアドレス
RC_ NO_SUPPORT	6	サポートされていない
RC_RD_ONLY	7	変数が読み取り専用である

リターン・コード	ID	説明
RC_WR_ONLY	8	変数が書き込み専用である
RC_FEWER_BYTES	9	想定されたバイト数に満たない
RC_TOO_MANY_BYTES	10	想定されたバイト数を超えている
RC_UNK_ERROR	11	未知のエラー
RC_EXEC_SIZE	12	コマンドを実行できない

⚠ OAP のリターン・コードと API コマンドから返されるリターン・コードを混同しないようにご注意ください。

10.1.4 Tag-Length-Value（TLV）形式のエンコーディング

アプリケーションのペイロードには、可変長の Tag-Length-Value フィールドが含まれます。一般的な TLV フィールドの形式は以下のとおりです。

タグ	長さ	値
1 バイト	1 バイト	*長さ*に指定されたバイト数

- タグは数値コードであり、特定のコンテキストに対してこのエンコードが表す項目を一意に識別します。タグ 0xFF には特別な意味があり、エンコードされたアドレスであることを示します。
- 長さは、値フィールドの長さをバイト数で示したものです。
- 値は可変長のバイト・データで、この部分のメッセージのデータを含みます。

値のデータ表現

TLV 形式でエンコードされる値には、以下の形式があります。

整数

整数値はビッグエンディアン順で送信されます。UINT8 および INT8 の値は、1 バイトです。UINT16 および INT16 の値は 2 バイトであり、UINT32 および INT32 は 4 バイトになります。

キャラクタ文字列

全てのキャラクタ文字列は末尾に「\0」バイトを付加して送信されます（null 終端）。
バイト文字列

バイト文字列は、バイトのストリームとして送信されます。

小数

小数は固定小数点で送信されます。長さと小数点の位置は、アプリケーションによって定義されます。

アドレス表現

OAP アドレスは、順序付けされた 1 バイトの数字の集合であり、さまざまなアプリケーション変数の位置を識別します。OAP アドレスの例には、3/2/1/6 や 1/1/1/25/1 があります。実際にアドレスが意味する内容は、アプリケーションのコンテキスト内でのみ意味を持ちます。

アドレスは特別なタグ値（0xFF）を使用して、TLV 形式でエンコードされます。TLV の値部分はアドレスに相当するバイト文字列であり、各バイトが個別の数字として扱われます。
例えば、1/4/0 というアドレスの TLV は以下のようになります。

Tag = 0xFF、Length = 3、Value = 0x01, 0x04, 0x00

10.1.5 OAP を使用したアプリケーションの対話操作

OAP をサポートするアプリケーションでは、操作と検索の対象となる変数を論理階層にマップする必要があります。これは、数多くの一般的な Web サイトでの、HTTP ベースの RESTful API を使用したリソース操作とよく似ています。見慣れた URL 表記を使用して、この階層について考えてみましょう。

2 つの GPIO ピンと 1 つのアナログ・チャネルを持つシンプルなデバイスについて考えます。アプリケーションでは、これらが 2 グループの変数として、gpio（id=0）と adc（id=1）で表現されるとします。リソース gpio/0（アドレス 0/0）は最初のピンを表し、gpio/1（アドレス 0/1）は 2 番目のピンを、adc/0（アドレス 1/0）は 1 つだけあるアナログ・チャネルを表します。

デバイスの検索と設定を行うには、グループごとに変数が必要です。

gpio の変数は以下のようになります（*x* には、どちらの gpio であるかによって、0 または 1 の値が入ります）。

変数	ID	アドレス	説明
direction	0	0/x/0	0=入力、1=出力
outval	1	0/x/1	0/1（出力ピン用）
inval	2	0/x/2	0/1（入力ピン用）

adc の変数は以下のようになります。

変数	ID	アドレス	説明
enable	0	1/0/0	0=無効、1=有効
voltage	1	1/0/1	チャネルから読み取られた値、8.8 固定小数点表示（2 バイト）

ここからは、OAP ペイロードの例をいくつか確認してみましょう。

GET gpio/0 variables – gpio/0 ピンに対する全ての変数の値を取得

gpio/0 のアドレスを TLV としてエンコードし、特別な「アドレス」タグの 0xff と、len=2、value=0x00 0x00 を設定します。OAP リクエストの Payload フィールドの内容は以下のようになります。

コマンド	アドレス
01	FF 02 00 00

予想されるレスポンス・ペイロードには、gpio/0 に対する全ての変数（direction（id=0）、outval（id=1）、inval（id=2））が含まれます。

コマンド	RC	アドレス	変数 1	変数 2	変数 3
01	00	FF 02 00 00	00 01 01	01 01 01	02 01 00

PUT gpio/1 outval=1 – gpio/1 ピンの出力レベルを高に設定

gpio/1 のアドレスをエンコードし、tag=0xff、len=2、value=0x00 0x01 を設定します。変数 outval には、tag=0x01、length=1、value=0x1 が設定されています。

OAP リクエストの Payload フィールドの内容は以下のようになります。

コマンド	アドレス	変数 1
02	FF 02 00 01	00 01 01

予想されるレスポンスは以下のようになります。

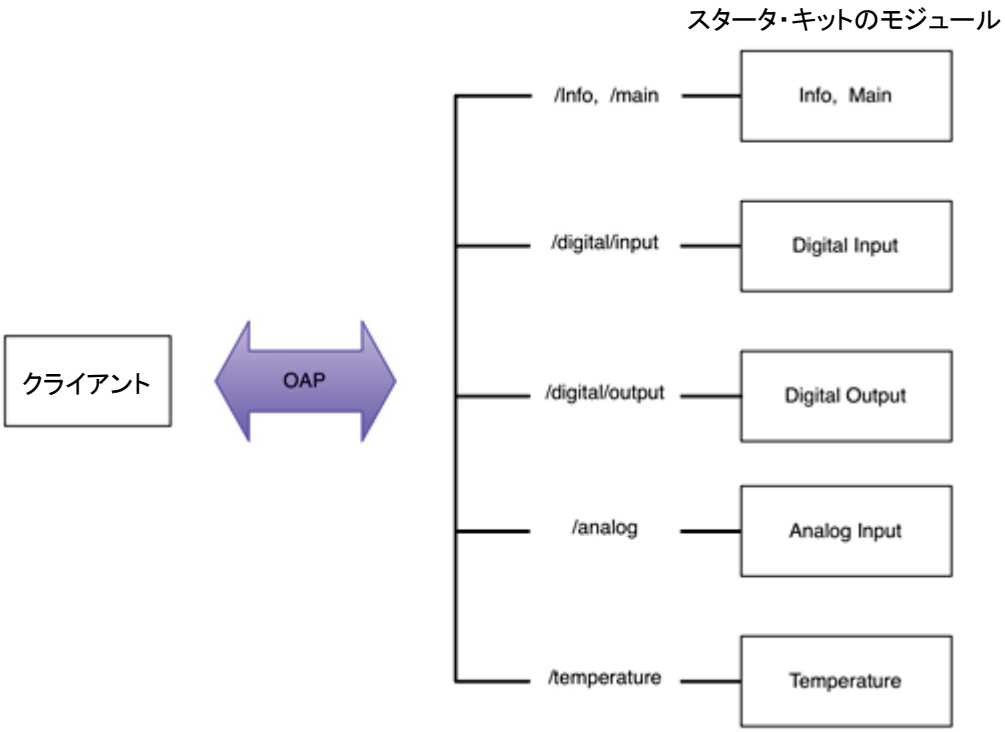
コマンド	RC	アドレス	変数 1
02	00	FF 02 00 01	00 01 01

レスポンスでは、設定された変数とその値が繰り返されています。

10.2 SmartMesh IP モートに含まれる OAP

10.2.1 概要

SmartMesh IP スタータ・キット（DC9021A）と SmartMesh WirelessHART スタータ・キッ(DC9022A）に含まれるモートには
アプリケーションが含まれており、マスター・モードで実行すると、LTC5800 プラットフォームの機能をデモンストレーションできます。このアプリケーションには複数のモジュールが含まれており、Stargazer（IP 用）または SmartMesh SDK に含まれるアプリケーションを介してアクセスおよび操作することができます。



以下の表に、各プラットフォームで実装されているアプリケーション・モジュールを示します。

	説明	SmartMesh IP でのサポート	SmartMesh WH でのサポート
/info	情報	あり	あり
/main	メイン	あり	あり
/digital/input	デジタル入力	あり	なし
/digital/output	デジタル出力	あり	なし
/analog	アナログ入力	あり	なし
/temperature	温度	あり	あり
/pkgen	パケット・ジェネレータ	あり	あり

アドレス指定可能な要素とピン配列

アプリケーション内のアドレス指定可能要素とそのアドレス識別子（かっこ内に記載）を以下に示します。機能グループ・アドレスの後に個別のピン・アドレスが付加されており、例えば、デジタル出力の D4 は、/digital_out/D4 または 3/0 というアドレスで識別されます。

- 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）

シグナル名	アドレス	LTC5800 のピン名	LTC5800 のピン番号	DC9018 のピン名	LTP5901/2 のピン名	LTP5901/2 のピン番号
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
インジケータ	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

info 要素には、アプリケーションを識別する情報が含まれています。OAP を実装する全てのデバイスは、この要素内の変数をサポートしている必要があります。

変数	ID	タイプ	アクセス	説明
swRevMaj	0	INT8U	R	アプリケーションのメジャー・ソフトウェア・リビジョン
swRevMin	1	INT8U	R	アプリケーションのマイナー・ソフトウェア・リビジョン
swRevPatch	2	INT8U	R	アプリケーションのパッチ番号
swRevBuild	3	INT16U	R	アプリケーションのビルド番号
appId	4	INT16U	R	一意のアプリケーション ID 0x0000-0x7FFF：予約済み 0x8000-0xFFFF：顧客による使用が可能 SmartMesh スタータ・キット= 0x0001
resetCounter	5	INT32U	R	モートのジョインカウンタと同じ
changeCounter	6	INT32U	R	前回のリセット以降に実施された構成変更の数（resetCounter と changeCounter を組み合わせると、最後のアクセス以降に変更が実施されたかどうかを特定できます）

/main

main 要素にはアドレス指定情報が含まれるため、特定のアドレスおよびポートにデータを送信するようにアプリケーションを構成することができます。この情報をスタータ・キット内で変更しないでください。変更した場合、モートが Stargazer または SmartMesh SDK と正しく相互作用しなくなります。

変数	ID	タイプ	アクセス	デフォルト値	説明
destAddr	0	INT8U[16]	r/w	FF02::02	モートからデータ・パケットを送信する宛先 IP アドレス
destPort	1	INT16U	r/w	F0B9	モートからデータ・パケットを送信する宛先 UDP ポート

/digital_in

この要素にはデジタル入力モジュールにアクセスするための変数が含まれます。基本アドレスの後にピン ID を付けると、個々のピンにアクセスできます（例：/digital_in/2 は D2 を指す）。

変数	ID	タイプ	アクセス	デフォルト値	説明
enable	0	INT8U	r/w	無効	0=無効、1=有効
rate	1	INT32U	r/w	10,000	サンプル・レート（ミリ秒）、dataFormat が「all」の場合のみ有効 最小レート：1000 ミリ秒 最大レート：300,000 ミリ秒
sampleCount	2	INT16U	r/w	1	dataFormat が「all」の場合に、パケット内に蓄積されるサンプルの数 （パケットに収まらない数が指定された場合も、パケットが一杯になるとモートはパケットを送信します）
dataFormat	3	INT8U	r/w	all	0='all' – サンプルを蓄積して全サンプルを送信する 1='on-change' – 値が変わった場合にパケットを送信する（迅速な変更のため、更新は 1 秒に 1 回だけに制限されます） 2='on-high' - 値が低から高に変わった場合にパケットを送信する（迅速な変更のため、更新は 1 秒に 1 回だけに制限されます） 3='on-low' - 値が高から低に変わった場合にパケットを送信する（迅速な変更のため、更新は 1 秒に 1 回だけに制限されます）
value	4	INT8U	r	n/a	読み取り時にピンの値を返す

/digital_out

この要素にはデジタル出力モジュールにアクセスするための全ての変数が含まれます。基本アドレスの後にピン ID を付けると、個々のピンにアクセスできます（例：/digital_out/1 は D5 を指す）。

変数	ID	タイプ	アクセス	デフォルト値	説明
value	0	INT8U	w	n/a	目的の値をピンに設定 0=ピンに 0 を設定 1=ピンに 1 を設定 2=ピンの値を 1 秒ごとに切り替え（ステータス LED でのみ有効）

/analog

この要素には ADC モジュールにアクセスするための変数が含まれます。基本アドレスの後に ID を付けると、個々のチャネルにアクセスできます（例：/analog/0 は A0 を指す）。

変数	ID	タイプ	アクセス	デフォルト値	説明
enable	0	INT8U	r/w	無効	0=無効、1=有効
rate	1	INT32U	r/w	10,000	サンプル・レート（ミリ秒）、 最小レート：1000 ミリ秒 最大レート：300,000 ミリ秒
sampleCount	2	INT8U	r/w	1	dataFormat が「all」の場合に、パケット内に蓄積されるサンプルの数（パケットに収まらない数が指定された場合も、パケットが一杯になるとモートはパケットを送信します） dataFormat が「stats」の場合に、統計集計で使用されるサンプルの数
dataFormat	3	INT8U	r/w	all	0='all' – サンプルを蓄積して全サンプルを送信する 1='stats' – *sampleCount* サンプルのそれぞれに対して、 最小/最大/平均を送信する
value	4	INT16U	r	n/a	読み取り時にチャネルの値（mV）を返す

/temperature

この要素には、内部の温度センサ・モジュールにアクセスするための変数が含まれます。

変数	ID	タイプ	アクセス	デフォルト値	説明
enable	0	INT8U	r/w	enabled	0=無効、1=有効
rate	1	INT32U	r/w	30,000	サンプル・レート（ミリ秒）、 最小レート：1000 ミリ秒 最大レート：300,000 ミリ秒
sampleCount	2	INT8U	r/w	1	dataFormat が「all」の場合に、パケット内に蓄積されるサンプルの数（パケットに収まらない数が指定された場合も、パケットが一杯になるとモートはパケットを送信します） dataFormat が「stats」の場合に、統計集計で使用されるサンプルの数
dataFormat	3	INT8U	r/w	all	0='all' – サンプルを蓄積して全サンプルを送信する 1='stats' – *sampleCount* サンプルのそれぞれに対して、 最小/最大/平均を送信する
Value	4	INT16S	r	n/a	0.01℃単位で現在の温度を返します （1.3 より前のモート・バージョンでは、1℃単位）。

/pkgen

パケット・ジェネレータ・モジュールを使用すると、指定した数のパケット（numPackets）を、指定したレート（rate）と指定したパケット・サイズ（packetSize）で送信できます。StartPID は、pkgen 通知で使用されます。pkgen は PUT コマンドを受け取ると、通知（type=pkgen）を生成し始めます。pkgen は値を保存しないため、クライアントは全ての変数を指定する必要があります。echo は、自動増分変数として使用されます。クライアントは初めに、PUT/echo=value を使用します。その後、クライアントが GET/echo をコールするたびに、echo の値が 1 ずつ増加します。モートがリセットされても、pkgen は通知の送信をリセットしません。pkgen による通知の送信を停止するには、numPackets=0 を指定した PUT リクエストを送信します。

変数	ID	タイプ	アクセス	デフォルト値	説明
echo	0	INT32U	r/w	0	受け取った echo を返す
numPackets	1	INT32U	w	0	送信するパケット数
rate	2	INT32U	w	10,000	パケット生成レート（ミリ秒）
packetSize	3	INT8U	w	80	送信するパケット・サイズ
startPID	4	INT32U	w	0	最初のパケット ID

10.2.2 通知

サンプル/レポート通知

フィールド	タイプ	説明
type	INT8U	0=サンプル
channel	TLV	アドレス形式でのデータ・ソース（例：/digital_in/1）
timestamp	UTC_TIME_L	レポート内の最初のサンプルのタイムスタンプ
rate	INT32U	サンプル間の間隔（ミリ秒）
numSamples	INT8U	パケット内のサンプル数
sampleSize	INT8U	各サンプルのサイズ（ビット）
samples[]	ビット	サンプル（ビット並び）

統計レポート（最小/最大/平均）

フィールド	タイプ	説明
type	INT8U	1=統計レポート
channel	TLV	アドレス形式でのデータ・ソース（例：/analog/1）
timestamp	UTC	統計収集開始時点のタイムスタンプ
rate	INT32U	サンプル間の間隔（ミリ秒）
numSamples	INT8U	統計収集期間
sampleSize	INT8U	サンプル・サイズ（各メトリックのサイズ、ビット表記）
stats	ビット	（各サンプル・サイズの）最小、最大、平均

アナログ・チャネルの値は mV 単位で送信されます。

温度は 0.01℃単位で送信されます。

デジタル変更通知

この通知は、デジタル I/O に変更があった場合に送信されます。

フィールド	タイプ	説明
type	INT8U	2=デジタル変更通知
Channel	TLV	ソース・チャネル
Timestamp	UTC	変更検出時点のタイムスタンプ
New value	INT8U	変更後の値（0 または 1）

PkGen 通知

この通知は、PkGen によってパケットが生成された場合に送信されます。

フィールド	タイプ	説明
type	INT8U	4=PkGen 通知
Channel	TLV	ソース・チャネル
Pid	INT32U	パケット ID（startPID から始まり、増加する）
StartPID	INT32U	最初の PID
numPackets	INT32U	パケット数
Payload	TLV	packetSize に設定されたサイズと一致する（例：00010203…）

10.3 OAP サンプル

ここで紹介するエンコーディング例は、モジュールに関するセクションで説明したように、モートに含まれるアプリケーションに適用されます。

10.3.1 INDICATOR_0 LED の点灯

モートにこのパケットを送信すると、インジケータ LED が点灯します。

OAP ヘッダ	コマンド ID	アドレス	ペイロード（可変 TLV）
05 00	02	FF 02 03 02	00 01 01

OAP ヘッダ

- コントロール：05（アクノリッジ制御リクエスト、再同期=1 は接続の確立に使用）
- ID：00（シーケンス=0、セッション=0）

OAP ペイロード

- コマンド：02（PUT）
- アドレス
  - アドレス TLV（長さ 2）
  - 03 02 は、/digital_out/INDICATOR_0 を示す
- タグ：00（Value 変数）
- 長さ：01
- 値：01=ピンに 1 を設定

10.3.2 温度サンプル通知

これは、温度測定が有効になっているモートから送信される温度通知の例です。

OAP ヘッダ	コマンド ID	通知タイプ	アドレス	タイムスタンプ	レート	サンプル数	サンプル・サイズ	サンプル
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 ヘッダ

- コントロール：00（非アクノリッジ制御リクエスト、通常同期）
- ID：03（シーケンス=3、セッション=0）

OAP ペイロード（通知）

- コマンド：05（通知）

- タイプ:00(サンプル)
- アドレス:FF 01 05(タグ、長さ、値=5(温度))
- UTC タイムスタンプ:00 00 00 00 53 16 60 93, 00 04 e5 77(エポック時間、μ 秒)
- レート:00 00 13 88(5000 ミリ秒)
- サンプル数:01
- サンプル・サイズ:10(16 ビット)
- サンプル:0a e0(2784、0.01℃単位)

10.3.3 アプリケーション情報の取得

OAP ヘッダ	コマンド ID	アドレス
05 00	01	FF 01 00

OAP ヘッダ

- コントロール:05(アクノリッジ制御リクエスト、再同期=1 は接続の確立に使用)
- ID:00(シーケンス=0、セッション=0)

OAP ペイロード

- コマンド:01(GET)
- アドレス
  - アドレス TLV(長さ 1)
  - 00 が情報に相当

レスポンス・ペイロード

OAP ヘッダ

- コントロール:05(アクノリッジ制御リクエスト、再同期=1 は接続の確立に使用)
- ID:00(シーケンス=0、セッション=x)
- RC:00

OAP ペイロード

- コマンド:01(GET)
- アドレス
  - アドレス TLV(長さ 1)
  - 00 が情報に相当
- swRevMajor、swRevMin、swRevPatch、swRevBuild、appId、resetCounter、changeCounter の値

OAP ヘッダ	アドレス	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

商標

Eterna、Mote-on-Chip、SmartMesh IP は、Dust Networks, Inc の商標です。Dust Networks ロゴ、Dust、Dust Networks、SmartMesh は、Dust Networks, Inc の登録商標です。LT、LTC、LTM、 は、Linear Technology Corp の登録商標です。第三者のブランド名および製品名は各社の商標であり、情報提供のみを目的として使用されています。

著作権

免責事項

本書は現状のまま提供され、明示、暗示を問わず一切の保証を行なわないものとします。かかる保証には、特定目的に対する商品性または適合性の黙示的保証が含まれますが、これに限定されません。

本書には技術的な誤りやその他の間違いが含まれる場合があります。訂正と改善は、新しいバージョンの文書に取り入れられる可能性があります。

リニアテクノロジーは、製品やサービスの適用または使用により発生する責任を負いかねます。また、間接的あるいは偶発的損害を含むがそれに限定されない、いかなる責任も負わないものとします。

リニアテクノロジーの製品は、誤動作がユーザーの深刻な人身傷害につながると合理的に予想できる生命維持装置、デバイス、またはその他のシステムでの使用、またはその機能不全により生命維持装置またはシステムの故障あるいはその安全性や有効性に影響すると合理的に予想できる生命維持装置またはシステムの重要な部品としての使用を目的として設計されていません。このような用途での使用を目的としてこれらの製品を使用または販売しているリニアテクノロジーの顧客は、顧客自身の責任でそれを行い、このような意図しないまたは不正な使用に関連する人身傷害または死亡に直接または間接的に起因するすべての主張、費用、損害、支出、および妥当な額の弁護士費用、また、かかるクレームでリニアテクノロジーに該当製品の設計または製造に関わる過失があったと主張される場合でも、これを完全に補償し、リニアテクノロジーとその役員、従業員、子会社、関連会社、および販売代理店に何ら損害を与えないことに同意するものとします。

リニアテクノロジーは、いつでも製品またはサービスに対する修正、変更、拡張、改良、その他の変更を行う権利を保有し、製品またはサービスを予告なく中止する権利を有します。顧客は、発注の前に最新の関連情報を入手し、その情報が最新で完全であることを確認する必要があります。すべての製品は、注文承諾時または販売時に提供される、販売に関する Dust Network の契約条件に従い販売されます。

リニアテクノロジーは、リニアテクノロジーの製品またはサービスが使用される組み合わせ、マシン、またはプロセスに関連するリニアテクノロジーの特許、著作権、マスクワーク権、その他のリニアテクノロジーの知的所有権に従って、明示か黙示かにかかわらず、ライセンスが付与されることを保証または主張するものではありません。第三者の製品またはサービスに関してリニアテクノロジーが送信した情報は、その製品またはサービスを使用するためのリニアテクノロジーからのライセンス提供、あるいはその保証または推奨を意味するものではありません。このような情報を使用する場合、第三者の特許または他の知的所有権に従って第三者からのライセンスが必要になるか、またはリニアテクノロジーの特許または他の知的所有権に従ってリニアテクノロジーからのライセンスが必要になります。

Dust Networks, Inc は、リニアテクノロジーの完全所有子会社です。