ユーザーズガイド

Sun Java™ Wireless Toolkit for CLDC

Version 2.5.2

Sun Microsystems, Inc.
www.sun.com

2007 年 9 月

コメントの送付: http://www.sun.com/hwdocs/feedback

原典: *User's Guide, Sun Java™ Wireless Toolkit for CLDC*

目次

はじめに

このマニュアルでは、Sun Java™ Wireless Toolkit for CLDC の操作方法について説明します。

対象読者

このマニュアルは、Sun Java™ Wireless Toolkit for CLDC を使用して MIDP (Mobile Information Device Profile) アプリケーションを作成する開発者を対象としています。このマニュアルは、MIDP プログラミングや、ツールキットでサポートされているオプション API のプログラミングに関するチュートリアルではありません。このマニュアルでは、読者が MIDP (Mobile Information Device Profile) と CLDC (Connected Limited Device Configuration) の使用方法を理解していることを前提にしています。

Java プログラミング言語をはじめて使用するときに支援が必要な場合は、次の URL を使用して、「New to Java Center」にアクセスしてください。

```
http://java.sun.com/learning/new2java/
```

MIDP プログラミングのクイックスタートについては、次の URL を使用して、Learning Path: Getting Started with MIDP 2.0 の内容をお読みください。

```
http://developers.sun.com/techtopics/mobility/learn/midp/midp20/
```

関連マニュアル

この節では、関連する Java Platform, Micro Edition (Java ME) の仕様を示します。Java ME は、これまで Java 2 Platform、Micro Edition、または J2ME の仕様名で呼ばれていました。仕様はもっとも信頼できる確定的なものですが、必ずしも利用しやすい情報ではありません。開発者向けのさまざまな記事については、Sun のモビリティーに関する次の Web サイトを参照してください。

`http://developers.sun.com/techtopics/mobility/`

表 P-1 関連マニュアル

内容	マニュアル名
Sun Java™ Wireless Toolkit for CLDC のカスタマイズ	Sun Java™ *Wireless Toolkit for CLDC* 基本カスタマイズガイド
リリースノート	Sun Java™ *Wireless Toolkit for CLDC* リリースノート
CLDC 1.0 - JSR 30	J2ME Connected Limited Device Configuration
MIDP 1.0 - JSR 37	Mobile Information Device Profile for the J2ME Platform
PDAP オプションパッケージ - JSR 75	PDA Optional Packages for the J2ME Platform
Bluetooth および OBEX - JSR 82	Java APIs for Bluetooth
MIDP 2.1 - JSR 118	Mobile Information Device Profile 2.0 (最終のリリース 2 は MIDP 2.1)
CLDC 1.1 - JSR 139	J2ME Connected Limited Device Configuration
MMAPI - JSR 135	Mobile Media API
J2ME Web Services - JSR 172	J2ME Web Services Specification
SATSA - JSR 177	Security and Trust Services APIs for J2ME
Location API - JSR 179	Location API for J2ME
SIP API - JSR 180	SIP API for J2ME
携帯 3D グラフィックス - JSR 184	Mobile 3D Graphics API for J2ME
JTWI - JSR 185	Java Technology for the Wireless Industry
WMA 2.0 - JSR 205	Wireless Messaging API (WMA)
CHAPI 1.0 - JSR 211	Content Handler API
SVG API - JSR 226	Scalable 2D Vector Graphics API for J2ME
Payment API - JSR 229	Payment API

表 P-1　関連マニュアル (続き)

内容	マニュアル名
Advanced Multimedia - JSR 234	Advanced Multimedia Supplements
Mobile Internationalization - JSR 238	Mobile Internationalization API
Java Binding for OpenGL® ES API - JSR 239	Java Binding for OpenGL® ES API
Mobile Service Architecture- JSR 248	*Mobile Service Architecture*

マニュアルの構成

このマニュアルは、以下の章で構成されています。

第 1 章では、Sun Java™ Wireless Toolkit for CLDC、およびこのツールキットで提供される開発機能の概要を説明します。

第 2 章では、MIDlet の作成および実行に必要な開発作業について説明します。

第 3 章では、ツールバーでプロジェクトを操作する方法について説明します。プロジェクトのプロパティーの調整方法、MIDlet の操作方法、転送レジストリの使用方法、およびプロジェクトのディレクトリ構造について説明します。

第 4 章では、エミュレータについて説明し、エミュレータのオプションの調整方法や、エミュレータのさまざまな機能の利用方法について説明します。

第 5 章では、メソッドプロファイラ、メモリーモニター、およびネットワークモニターを使用してアプリケーションのパフォーマンスを調べる方法について説明します。

第 6 章では、MIDlet スイートに署名する方法と、キーや証明書を管理する方法について説明します。

第 7 章では、ワイヤレスメッセージングアプリケーションを実行およびテストする場合のサポートについて説明します。

第 8 章では、Sun Java™ Wireless Toolkit for CLDC の Mobile Media API のサポートについて説明します。

第 9 章では、3D グラフィックスコンテンツの開発について説明します。

第 10 章では、ローカルファイルへのアクセスと、連絡先やカレンダなどの個人情報へのアクセスが、ツールキットでどのように実装されるかについて説明します。

第 11 章では、ツールキットの Bluetooth シミュレーション環境と OBEX シミュレーション環境について説明します。

第 12 章では、Web サービスのスタブジェネレータの使用方法について説明します。

第 13 章では、エミュレータの位置情報機能の操作方法について説明します。

第 14 章では、ツールキットの SATSA サポートについて説明します。

第 15 章では、ツールキットの SIP サポートについて説明します。

第 16 章では、ツールキットの Payment API 機能について説明します。

第 17 章では、Mobile Internationalization API のリソースの管理方法について説明します。

付録 A では、Sun Java™ Wireless Toolkit for CLDC に含まれているデモアプリケーションについて説明します。

付録 B では、Sun Java™ Wireless Toolkit for CLDC の機能をコマンド行から使用する方法について説明します。

付録 C では、Sun Java™ Wireless Toolkit for CLDC の各国語対応機能について説明します。

書体と記号について

表 P-2 に、このマニュアルでのフォントの使用規則を示します。

表 P-2 書体と記号について

書体または記号*	意味	例
`AaBbCc123`	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例。	`.login` ファイルを編集します。 `ls -a` を実行します。 `% You have mail.`
`AaBbCc123`	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して表します。	**マシン名`%` `su`** `Password:`
AaBbCc123	コマンド行の可変部分。実際の名前や値と置き換えてください。	`rm` *filename* と入力します。
『 』	参照する書名を示します。	『Solaris ユーザーマニュアル』

表 P-2　書体と記号について (続き)

書体または記号*	意味	例
「」	参照する章、節、または、強調する語を示します。	第 6 章「データの管理」を参照。 この操作ができるのは「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	`% grep '^#define \` `XV_VERSION_STRING'`
{*AaBbCc.dir*}	可変ファイル名およびディレクトリ。	このマニュアルでは、*toolkit* は Sun Java™ Wireless Toolkit for CLDC がインストールされているディレクトリを表します。*workdir* は、1-1 ページの 1.1.1 節「インストールディレクトリと作業用ディレクトリ」で説明したように、ユーザーの作業用ディレクトリを表します。

* 使用しているブラウザにより、これらの設定と異なって表示される場合があります。

Sun のオンラインマニュアル

次のサイトで、Java テクノロジに関連したテクニカルドキュメントを参照できます。

- `http://developer.sun.com/`
- `http://java.sun.com/javame/`

コメントをお寄せください

マニュアルの品質改善のため、お客様からのご意見およびご要望をお待ちしております。コメントは下記よりお送りください。

`developers.sun.com`

ご意見をお寄せいただく際には、下記のタイトルを記載してください。

『Sun Java Wireless Toolkit for CLDC ユーザーズガイド』

第1章

概要

このマニュアルでは、Sun Java™ Wireless Toolkit for CLDC の使用法を説明します。

Sun Java™ Wireless Toolkit for CLDC は、携帯電話などのワイヤレスデバイス用のアプリケーションを作成するための一連のツール群です。Sun Java™ Wireless Toolkit for CLDC は MIDP (Mobile Information Device Profile) 2.1 に準拠していますが、多数のオプションパッケージもサポートしているので、広範な機能を備えた開発ツールキットとなっています。

1.1 複数ユーザーの環境について

Sun Java Wireless Toolkit 2.5.2 for CLDC は、サポートされているバージョンの Windows または Linux を実行しているシステムにインストールできます。ホストマシンにアカウントを持つすべてのユーザーは、単独または同時にツールキットにアクセスできます。

1.1.1 インストールディレクトリと作業用ディレクトリ

Windows のパスではドライブ文字を含め、ディレクトリの区切り文字として円記号を使用します。Linux のパスでは、スラッシュを使用します。Linux のパスでは、~ は Linux ユーザーのホームディレクトリを表します。

複数のユーザーをサポートするために、ツールキットによってインストールディレクトリが作成され、コピー元として使用されます。このマニュアルでは、変数 *workdir* でツールキットの作業用ディレクトリを表し、*toolkit* でインストールディレクトリを表します。各ユーザーの個人用ファイルは、`j2mewtk` という別の作業用ディレクト

リで管理されます。このディレクトリには、インストールされたバージョンごとにサブディレクトリが作成されます。*workdir* のデフォルトの場所は、一般的に次のいずれかの場所になります。

Windows:	C:¥Documents and Settings¥*User*¥j2mewtk¥2.5.2 (*User* はユーザーのアカウント名)
Linux:	~/j2mewtk/2.5.2 (~ はユーザーのホームディレクトリ)

Windows と Linux での主な相違点は、このマニュアルで説明しています。一方のオペレーティングシステムしか説明していない場合は、前述したパスの違いを適用することができます。

1.1.2 作業用ディレクトリのファイル

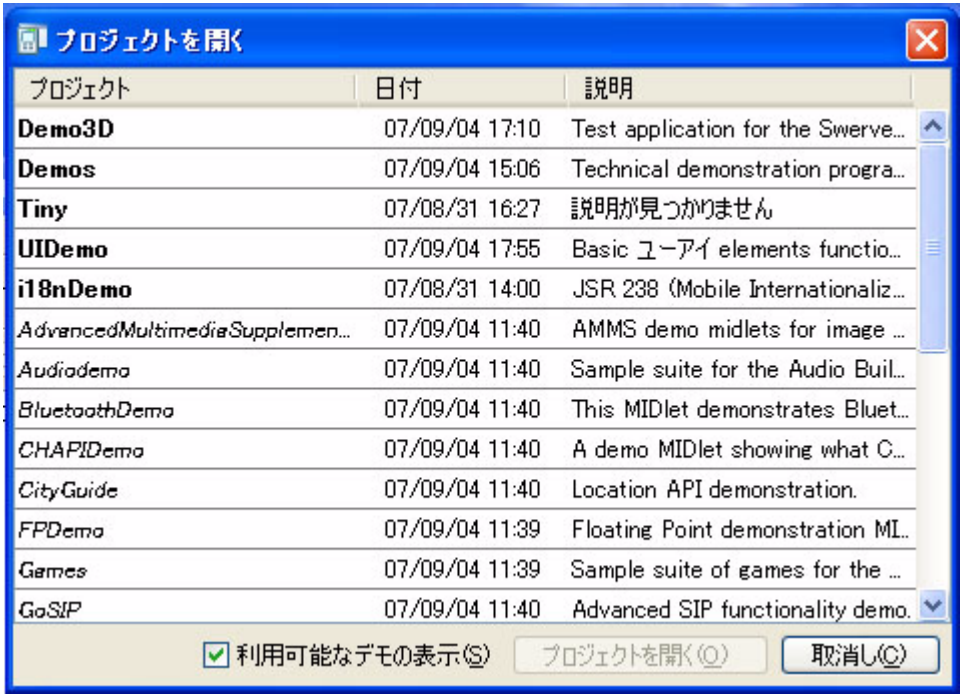
インストール時に、インストーラはインストールディレクトリから各ユーザーの作業用ディレクトリにファイルのサブセットをコピーします。作業用ディレクトリの内容は次のとおりです。

- .settings - `security.properties` の初期設定を含みます。
- appdb - アプリケーションデータベース全体が、作業用ディレクトリにコピーされます。
- apps - 作業用ディレクトリに空の `apps` ディレクトリが作成されます。デモプロジェクトを開くと、プロジェクトが自動的にこの場所にコピーされます。新しいプロジェクトを作成するときに (2-1 ページの 2.1 節「プロジェクトについて」)、プロジェクトはここに保存されます。
- wtklib - エミュレータのプロパティーファイル (第 4 章を参照) と、HTTP と WMA サーバーおよびクライアントの状態の情報が含まれます。
- sessions - 監視ツールのデフォルトの保存先です (第 5 章を参照)。最初は空です。

デモアプリケーションのすべてのソースコードは、*toolkit*¥`apps` にあり、各デモは独自のプロジェクトディレクトリを使用します。*toolkit*¥`apps` に保存されているアプリケーションはいずれも、すべてのユーザーに表示されます。

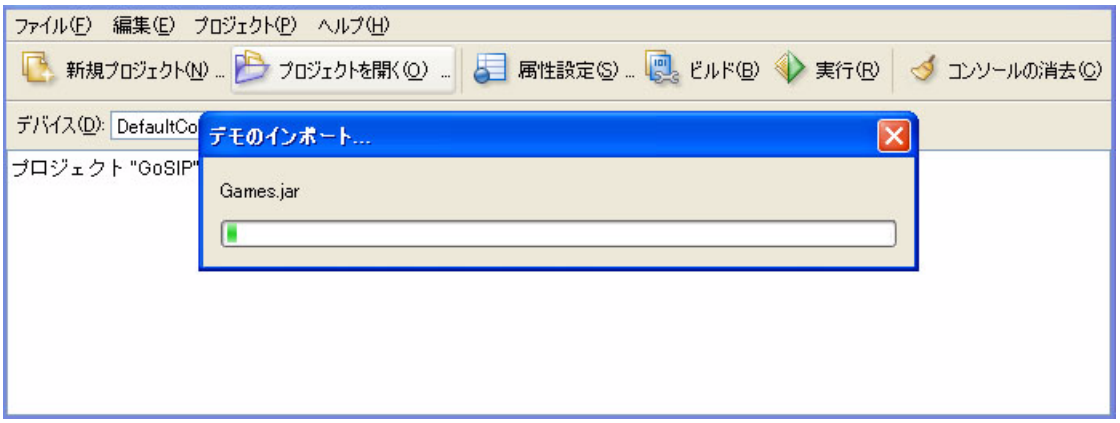
たとえば、Windows の `MobileMediaAPI` デモアプリケーションのソースコードは、*toolkit*¥`apps¥MobileMediaAPI¥src` にあります。「プロジェクトを開く」コマンドを使用すると、インストールディレクトリのプロジェクトと作業用ディレクトリのプロジェクト (存在する場合) を表示できます。図 1-1 に示すように、ローカルディレクトリにあるプロジェクトはボールド書体で表示され、インストールディレクトリにあるプロジェクトはイタリック書体で表示されます。ローカルファイルだけを表示するには、**「利用可能なデモの表示」**チェックボックスのチェックマークを解除します。

図 1-1　　ローカルプロジェクトをボールド書体で表示したプロジェクトリスト



インストールディレクトリのプロジェクトを開く場合は、アプリケーションのコピーが *workdir*¥apps にインポートされます。ほとんどの場合、インポートプロセスに気付くことはありませんが、低速なマシンでは図 1-2 に示すような進捗ダイアログが表示されることがあります。どのカスタマイズも、ユーザーのローカルコピーに行うことをお勧めします。

図 1-2　*toolkit* から *workdir* へのプロジェクトのインポート



1.2　クイックスタート

すぐに使い始めるには、Sun Java™ Wireless Toolkit for CLDC に含まれているデモアプリケーションを試してください。

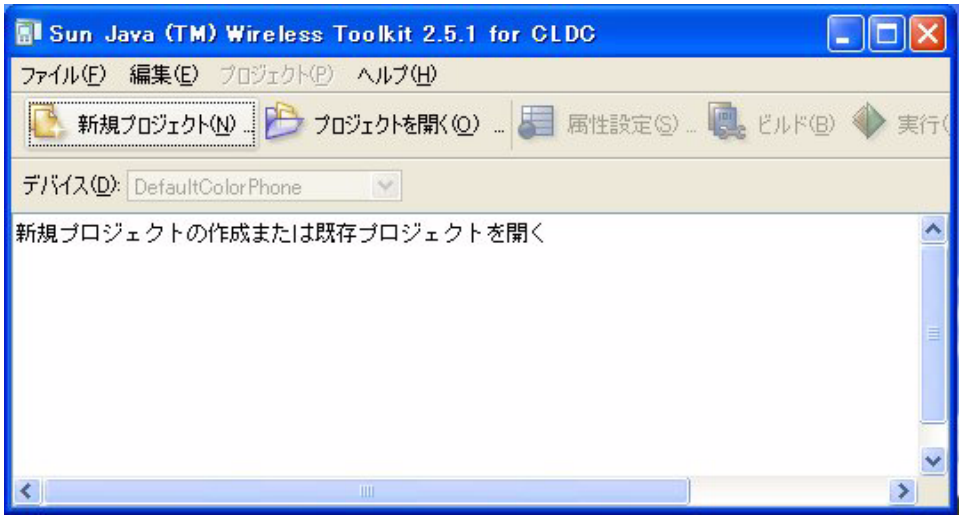
デモアプリケーションを実行するには、次のようにツールキットを起動します。

- Microsoft Windows で、「スタート」->「プログラム」->「Sun Java Wireless Toolkit 2.5.2 for CLDC」->「Wireless Toolkit 2.5.2」を選択します。[1]
- Linux では、*toolkit*`/bin` ディレクトリに移動します。ツールキットを実行するには、`./ktoolbar` と入力します。

図 1-3 のようなウィンドウが開きます。

1. Microsoft Windows の設定によっては、「スタート」->「プログラム」の代わりに、「スタート」->「すべてのプログラム」を選択する必要があります。

図 1-3　　ツールキットのユーザーインタフェース



「プロジェクトを開く」ボタンをクリックして、使用可能なすべてのアプリケーションの一覧を表示します。図 1-1 に示すように、イタリックの名前はインストールディレクトリに保存されているアプリケーションを表し、ボールドの名前は作業用ディレクトリに保存されているアプリケーション (もしあれば) を表します。これらのアプリケーションの 1 つを選択し、ダイアログの「プロジェクトを開く」ボタンをクリックします。このプロジェクトを初めて開く場合は、1-2 ページの 1.1.2 節「作業用ディレクトリのファイル」で説明したように、作業用ディレクトリにコピーが作成されます。

アプリケーションが開いたら、「実行」ボタンをクリックします。エミュレータが表示され、デモアプリケーションが実行されます。

ほとんどのデモアプリケーションは説明不要ですが、一部については追加の指示があります。一部のデモでは、実行の代わりに「プロジェクト」->「OTA 経由で実行」を使用する必要があります。デモの一般的な注意点と説明については、付録 A を参照してください。

1.3　ツールキットのコンポーネント

Sun Java™ Wireless Toolkit for CLDC の主要なコンポーネントは次の 3 つです。

- ユーザーインタフェースは、MIDP アプリケーションの作成にかかわる多くのタスクを自動化します。
- 「エミュレータ」は、携帯電話のシミュレーションです。MIDP アプリケーションのテストに役立ちます。

- 「ユーティリティー」のコレクションは、テキストメッセージングコンソールや暗号化ユーティリティーなど、その他の便利な機能を提供します。

ユーザーインタフェースから、アプリケーションのビルド、エミュレータの起動、ユーティリティーの起動を行うことができます。あるいは、エミュレータやユーティリティーを単独で実行することもできます。これは多くの状況で役立ちます。たとえば、MIDP アプリケーションのデモを行う場合は、エミュレータを単独で実行すると便利です。

その他に必要なツールは、ソースコードを編集するためのテキストエディタだけです。

1.4 ツールキットの機能

Sun Java™ Wireless Toolkit for CLDC は次のような主要機能を備え、MIDP アプリケーションの作成をサポートします。

- **ビルドとパッケージ化** - ユーザーはソースコードを作成するだけで、残りの処理はツールキットに任せることができます。ボタンをクリックすると、ツールキットによってソースコードのコンパイル、クラスファイルの事前検証、および MIDlet スイートのパッケージ化が行われます。
- **実行と監視** - MIDlet スイートは、エミュレータで直接実行するか、実際のデバイスにアプリケーションをインストールするときと同様の手順でインストールすることができます。MIDlet の動作を分析するために、メモリーモニター、ネットワークモニター、およびメソッドプロファイラが用意されています。
- **MIDlet スイートの署名** - ツールキットには、MIDlet スイートに暗号で署名するツールが含まれています。このツールは、さまざまな保護ドメインで MIDlet の動作をテストするのに役立ちます。

1.5 ツールキットの更新

インストール時に、「製品アップデートの確認」機能を有効にするかどうかを選択できます。この機能は、7 日ごとにネットワークを使用して更新を確認します。更新が必要かどうかを判定するために、次の情報が収集されます。

- オペレーティングシステム
- コンピュータの地域と言語の設定
- 現在の Wireless Toolkit の日付とバージョン
- Wireless Toolkit によって生成されたランダムな一意のユーザー ID

このデータは製品を改善するために使用されます。
http://www.sun.com/privacy/index.html で、Sun のプライバシーポリシーをお読みください。

この機能を有効または無効にするには、「編集」->「環境設定」を選択し、「ネットワーク構成」をクリックします。パネルの下部で、「更新の確認」ボックスにチェックマークを付けるか、チェックマークを外します。

1.6 サポートされている技術

Sun Java™ Wireless Toolkit for CLDC では、Java Community Process™ (JCP™) プログラムによって定義された多くの標準アプリケーションプログラミングインタフェース (Application Programming Interfaces、API) がサポートされています。表 1-1 は、サポートされている API とその仕様の参照先を示しています。

表 1-1 サポートされている JCP プログラム API

JSR API	名前 URL
JSR 248 MSA 1.0	Mobile Service Architecture http://jcp.org/en/jsr/detail?id=248
JSR 185 JTWI 1.0	Java Technology for the Wireless Industry http://jcp.org/en/jsr/detail?id=185
JSR 139 CLDC 1.1	Connected Limited Device Configuration http://jcp.org/en/jsr/detail?id=139
JSR 118 MIDP 2.0	Mobile Information Device Profile http://jcp.org/en/jsr/detail?id=118
JSR 75 PIM and File	PDA Optional Packages for the J2ME Platform http://jcp.org/en/jsr/detail?id=75
JSR 82 Bluetooth および OBEX	Java APIs for Bluetooth http://jcp.org/en/jsr/detail?id=82
JSR 135 MMAPI 1.1	Mobile Media API http://jcp.org/en/jsr/detail?id=135
JSR 172	J2ME Web Services Specification http://jcp.org/en/jsr/detail?id=172
JSR 177 SATSA	Security and Trust Services API for Java ME http://jcp.org/en/jsr/detail?id=177

表 1-1　サポートされている JCP プログラム API (続き)

JSR API	名前 URL
JSR 179 Location	Location API for Java ME http://jcp.org/en/jsr/detail?id=179
JSR 180 SIP	SIP API for Java ME http://jcp.org/en/jsr/detail?id=180
JSR 184 3D グラフィックス	Mobile 3D Graphics API for J2ME http://jcp.org/en/jsr/detail?id=184
JSR 205 WMA 2.0	Wireless Messaging API http://jcp.org/en/jsr/detail?id=205
JSR 211 CHAPI	Content Handler API http://jcp.org/en/jsr/detail?id=211
JSR 226	Scalable 2D Vector Graphics API for J2ME http://jcp.org/en/jsr/detail?id=226
JSR 229	Payment API http://jcp.org/en/jsr/detail?id=229
JSR 234 AMMS	Advanced Multimedia Supplements http://jcp.org/en/jsr/detail?id=234
JSR 238 MIA	Mobile Internationalization API http://jcp.org/en/jsr/detail?id=238
JSR 239	Java Binding for OpenGL® ES API http://jcp.org/en/jsr/detail?id=239
JSR 248	Mobile Service Architecture http://jcp.org/en/jsr/detail?id=248

第2章

MIDlet スイートの開発

この章では、Sun Java™ Wireless Toolkit for CLDC を使用してアプリケーションを作成する方法について説明します。まずツールキットのプロジェクトについて説明してから、開発手順の説明に進みます。

MIDlet スイートアプリケーションの作成には、2 つの基本的な開発サイクルをよく使用します。1 つは、すばやく簡単に開発を行うことができるサイクルです。この方法は初期の開発で役立ちます。もう 1 つの開発サイクルでは、時間はかかりますが、より実際に近い総合的なテストを行うことができます。

章の終わりの方では、Sun Java™ Wireless Toolkit for CLDC で難読化ツールやデバッガなどの拡張開発ツールを使用する方法について説明します。最後の節では、Web サーバーを設定して MIDP アプリケーションを提供する方法について説明します。

2.1 プロジェクトについて

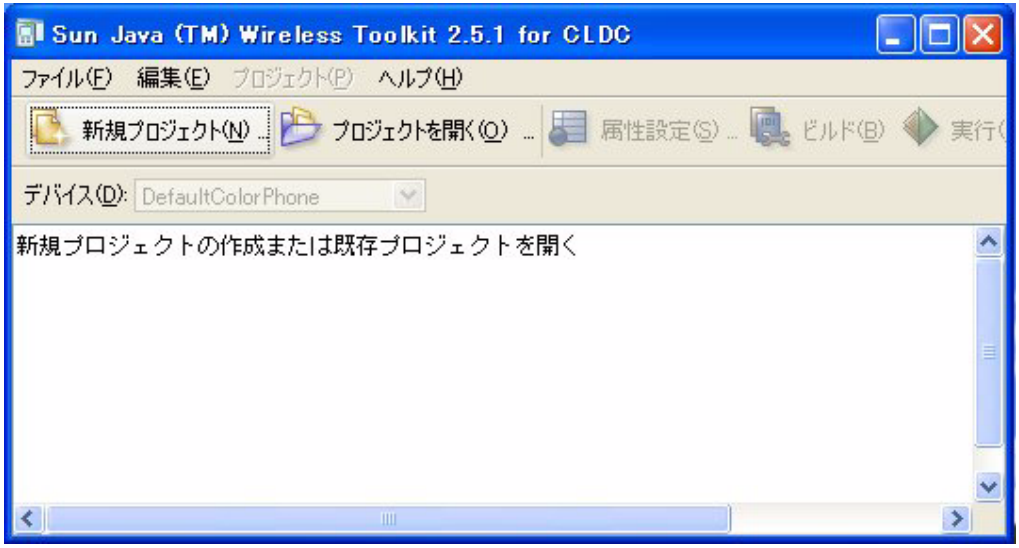
Sun Java™ Wireless Toolkit for CLDC では、MIDlet スイートは「プロジェクト」に分けて整理され、1 つのプロジェクトから最終的に 1 つの MIDlet スイートがビルドされます。プロジェクトには、Java ソースファイルやリソースファイルなど、MIDlet スイートのビルドに使用されるすべてのファイルと、MIDlet 記述子が含まれています。

Sun Java™ Wireless Toolkit for CLDC で一度に操作できるプロジェクトは 1 つだけです。新しいプロジェクトを作成するか、または既存のプロジェクトを開くことができます。

この章では、単純なサンプルプロジェクトを使用します。開発サイクルの各手順について読みながら、ツールキットで実際に作業します。

新しいプロジェクトを作成するには、まずユーザーインタフェースを起動します。Microsoft Windows で、「スタート」->「プログラム」->「Sun Java Wireless Toolkit 2.5.2 for CLDC」->「Wireless Toolkit 2.5.2」を選択します。[1]図 2-1 に示すように、ユーザーインタフェースが表示されます。

図 2-1 ツールキットのユーザーインタフェース



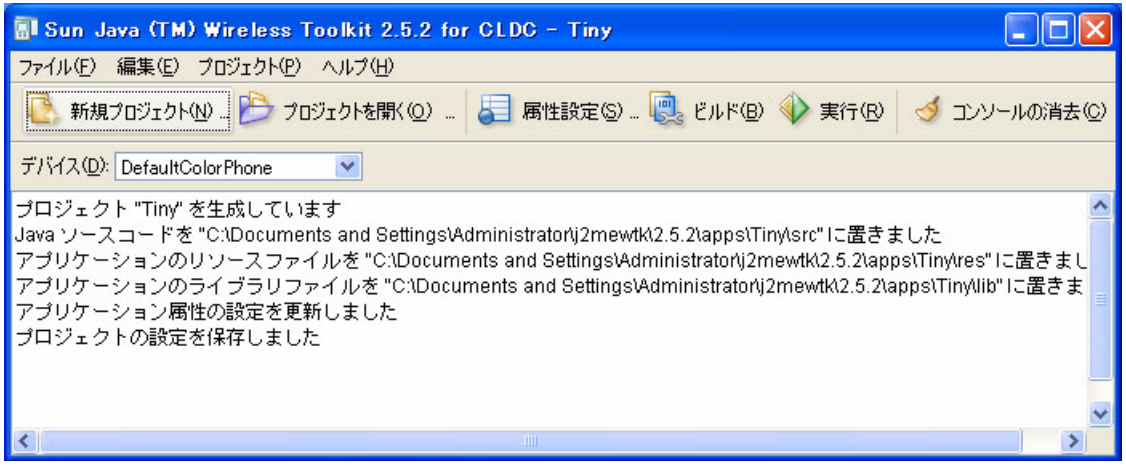
「新規プロジェクト」をクリックします。プロジェクトの名前と作成する MIDlet クラスの名前を入力するように求められます。名前を入力し、「プロジェクトの作成」をクリックします。

1. Microsoft Windows の設定によっては、「スタート」->「プログラム」の代わりに、「スタート」->「すべてのプログラム」を選択する必要があります。

図 2-2　　新しいプロジェクトの作成

「設定」ウィンドウが表示されます。選択する内容は、プロジェクトのビルド環境に影響を与えます。このサンプルではデフォルトのオプションをそのまま使用するので、「了解」をクリックしてこのウィンドウを閉じます。コンソールに、このプロジェクトのソースコードとリソースファイルの保存先を示すメッセージが表示されます。

図 2-3　　コンソールで指定されるファイルの場所



2.2 単純な開発サイクル

単純な開発サイクルは次のようなものです。

ソースコードの編集 -> ビルド -> 実行

1. **ソースコードの編集。**

   この手順では、アプリケーションで使用する Java ソースファイルおよびリソースファイルを作成します。

2. **ビルド (構築)。**

   ツールキットにより、Java ソースファイルのコンパイルと事前検証が行われます。

3. **実行。**

   コンパイル済みの Java クラスファイルをエミュレータ上で実行します。

ツールキットでソースファイルをコンパイルしようとしてエラーが発生した場合は、ソースファイルを編集し直します。エミュレータでアプリケーションをテストしているときにバグが見つかった場合は、ソースファイルを編集してバグを修正します。

ここまでで、ハイレベルでの単純な開発サイクルについて把握しました。この節の残りの部分では、Sun Java™ Wireless Toolkit for CLDC を使用して各手順を実施する方法について説明します。

2.2.1 ソースコードの編集

開発手順のうち、ソースコードの編集だけは、Sun Java™ Wireless Toolkit for CLDC で行えません。任意のテキストエディタを使用して、ソースコードファイルを作成および編集します。使いたいテキストエディタがない場合は、`http://jedit.org/` にある jEdit を試してみてください。

サンプルプロジェクトに従って作業している場合は、新しい Java テクノロジソースファイル `TinyMIDlet.java` を作成します。作成したファイルを、プロジェクトのソースディレクトリに保存します。たとえば、Windows の場合は *workdir*`¥apps¥Tiny¥src¥TinyMIDlet.java` に保存します。このファイルには、次のような簡単な MIDlet が含まれています。

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.MIDlet;

public class TinyMIDlet
    extends MIDlet
    implements CommandListener {
```

```
  public void startApp() {
    Display display = Display.getDisplay(this);

    Form mainForm = new Form("TinyMIDlet");
    mainForm.append("Welcome to the world of MIDlets!");

    Command exitCommand = new Command("Exit", Command.EXIT, 0);
    mainForm.addCommand(exitCommand);
    mainForm.setCommandListener(this);

    display.setCurrent(mainForm);
  }
  public void pauseApp () {}

  public void destroyApp(boolean unconditional) {}

  public void commandAction(Command c, Displayable s) {
    if (c.getCommandType() == Command.EXIT)
      notifyDestroyed();
  }
}
```

作業が完了したら、ファイルを保存します。

2.2.2 ビルド

次の手順では、ソースコードをビルドします。ツールキットを使用すると、この手順は非常に簡単です。

ユーザーインタフェースで、「ビルド」ボタンをクリックします。ソースファイルが正しい場所に保存されていれば、ツールキットがそのファイルを見つけてコンパイルします。コンパイルエラーはコンソールに表示されます。図 2-4 のようなエラーが表示された場合は、ソースコードを編集して、エラーを修正します。エラーがなくなると、コンソールにビルドが成功したことを示すメッセージが表示されます。

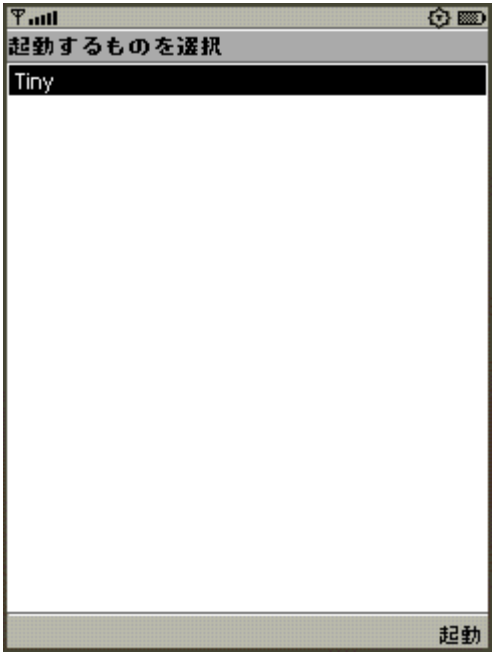
図 2-4　　ビルドに関するメッセージ

表面には現れませんが、ツールキットではコンパイル済みクラスファイルの事前検証も行われます。MIDlet クラスファイルを MIDP デバイスやエミュレータで実行するには、その前に検証する必要があります。この処理はツールキットによって実行され、メッセージは表示されません。事前検証の詳細は、CLDC 仕様を参照してください。

2.2.3　実行

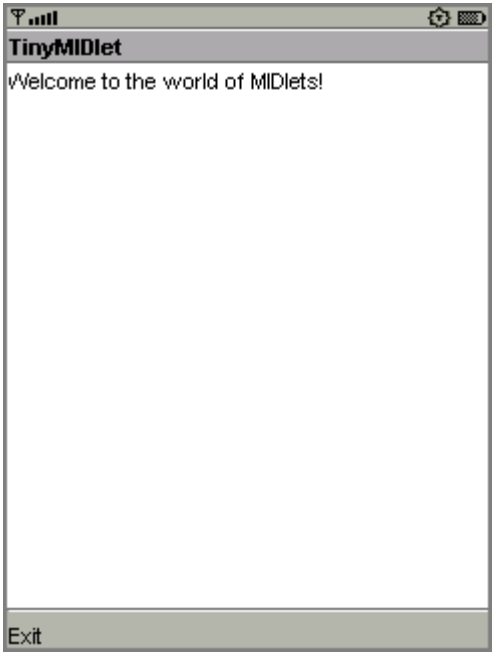
プロジェクトが正常にビルドされたら、エミュレータで実行してみることができます。「実行」ボタンをクリックします。エミュレータに、プロジェクト内のすべての MIDlet のリストが表示されます。

図 2-5　　プロジェクトの MIDlet のリスト



実行する MIDlet を選び、「起動」を選択します。`TinyMIDlet` の例に従って作業している場合、結果は図 2-6 のようになります。

図 2-6　動作中の TinyMIDlet



2.3　完全な開発サイクル

2 つ目の開発サイクルは少し複雑です。次のようなハイレベルの手順で構成されます。

ソースコードの編集 -> パッケージ化 -> インストール -> 実行

1. **ソースコードの編集。**
   これは、単純な開発サイクルの場合と同じです。
2. **パッケージ化。**
   この手順では、Sun Java™ Wireless Toolkit for CLDC によってソースファイルのコンパイルと事前検証が行われます (前述したビルドの手順と本質的に同じ)。続いて、Java クラスファイルとリソースファイルがバンドルされて、MIDlet スイートの Java Archive (JAR) ファイルと MIDlet スイート記述子が作成されます。
3. **インストール。**
   MIDlet スイートを実行するには、その前にインストールする必要があります。MIDlet スイートは、Sun Java™ Wireless Toolkit for CLDC のエミュレータまたは実際のデバイスにインストールできます。

4. **実行。**

単純な開発サイクルの場合と同様に、アプリケーションを実行して、バグがないかどうかをテストします。

完全な開発サイクルでも、最初の手順は単純な開発サイクルの場合と同じです。ソースコードの編集も通常どおりです。ビルドの手順は、パッケージ化の手順に組み込まれています。

完全な開発サイクルには、パッケージ化とインストールという 2 つの新しい手順が含まれています。アプリケーションをインストールしてから実行する方法と単純な開発サイクルでのアプリケーションの実行方法は大きく異なります。

2.3.1 パッケージ化

Sun Java™ Wireless Toolkit for CLDC では、MIDlet スイートをパッケージ化するタスクが自動化されます。パッケージ化の結果として、MIDlet 記述子と MIDlet スイートの JAR ファイルが作成されます。記述子は、MIDlet スイートに関する情報を含んでいる小さなテキストファイルです。JAR ファイルには、MIDlet スイートを構成するクラスファイルとリソースファイルが含まれています。デバイスで JAR ファイル全体をダウンロードする前に、記述子からアプリケーションの情報を取得できます。メモリーや帯域幅の少ないワイヤレス環境では、これは重要な考慮事項です。

ツールキットで MIDlet スイートをパッケージ化するには、「プロジェクト」->「パッケージ」->「パッケージを作成」を選択します。MIDlet スイートの記述子と JAR ファイルが生成され、プロジェクトの `bin` ディレクトリに置かれます。

パッケージ化には、ほかの手順が追加されることもあります。コード難読化ツールを使用すると、MIDlet スイートの JAR ファイルのサイズを縮小できます。この手法については、この章で後述します。さらに、ツールキットには、MIDlet スイートに暗号で署名できるようにするツールが用意されています。詳細は、第 6 章を参照してください。

2.3.2 インストール

MIDlet スイートを正しくテストするには、MIDlet スイートをツールキットのエミュレータまたは実際のデバイスにインストールします。ユーザーインタフェースの「実行」ボタンをクリックすると、MIDlet スイートはエミュレータにインストールされません。代わりに、MIDlet クラスがエミュレータで直接実行されます。

また、エミュレータは、実際のデバイスにアプリケーションを無線 (OTA) で転送してインストールするときと同様の手順で、アプリケーションをメモリーにインストールできます。Sun Java™ Wireless Toolkit for CLDC のエミュレータにアプリケーションをインストールするには、「プロジェクト」->「OTA 経由で実行」を選択します。

MIDlet クラスが直接実行されるのではなく、エミュレータのウィンドウが開き、アプリケーション管理ソフトウェア (Application Management Software、AMS) の開始画面が表示されます。エミュレータのソフトウェアは、MIDlet スイートを管理するために実際のデバイスで必要となるソフトウェアの種類の一例です。

図 2-7　　エミュレータの AMS の開始画面

「アプリ」を選択して、インストール済みアプリケーションのメインリストに移動します。「アプリケーションのインストール」を選択し、エミュレータの「SELECT」ボタンを押します。インストールするアプリケーションの URL を入力するように求められます。URL はあらかじめ入力されています。

図 2-8　URL のプロンプト

メニューから「進む」を選択して、インストールを開始します。URL で見つかったアプリケーションのリストがエミュレータに表示されます。1 つだけ選択し、メニューから「インストール」を選択します。最後にもう一度、インストールしてもよいかどうかを確認するメッセージが表示されます。

図 2-9　　インストールの確認

もう一度「インストール」を選択して、インストールを完了します。エミュレータのインストール済みアプリケーションのリストに戻ります。インストールしたばかりのアプリケーションもそのリストに表示されています。

図 2-10　アプリケーションメニュー

「OTA 経由で実行」を使用すると、ツールキットのエミュレータに MIDlet スイートを簡単にインストールでき、非常に便利です。転送レジストリ機能や、署名付き MIDlet スイートのインストール機能など、一部の機能はこの手法を使ってテストする必要があります。

実際のデバイスで MIDlet スイートをテストするには、最初にインストールする必要があります。その方法は、使用しているデバイスに大きく依存します。次の 2 つの方法が一般的に使用されます。

- アプリケーションを Web サーバーに配備してから、MIDP 2.0 仕様に記述された無線 (Over-The-Air、OTA) プロトコルを使って、サーバーからデバイスにアプリケーションを転送することができます。エンドユーザーがアプリケーションを購入またはインストールする手段として、この方法がもっとも多いと考えられます。
- Bluetooth、赤外線、シリアル接続などを使用して、デバイスに MIDlet スイートを転送できる場合もあります。この方法は、Web サーバーを実行する方法よりも簡単です。アプリケーションが OTA 経由でデバイスにインストールされるプロセスを知ることはできませんが、デバイス上でアプリケーションがどのように動作するかを確認することができます。

2.3.3 実行

アプリケーションをインストールしたら、リストからアプリケーションを選択し、メニューから「起動」を選択します。

図 2-11　インストール済みアプリケーションの起動

実際のデバイスでアプリケーションを実行する方法は、デバイス自体に大きく依存します。詳細については、デバイスのマニュアルを参照してください。

2.4 MIDlet スイートからのプロジェクトの作成

MIDlet スイートアーカイブ (`.jar` ファイル) と記述子 (`.jad` ファイル) から、Sun Java™ Wireless Toolkit for CLDC プロジェクトを作成することもできます。この方法は、ソースコードを利用できない場合でさえも、ツールキットのユーザーインタフェースから MIDlet スイートを実行するのに役立ちます。ツールキットを使用して、記述子で属性を簡単に操作したり、プロジェクトを実行し、ツールキットの監視ツール (第 5 章で説明) を使用してその動作を調査したりすることができます。

MIDlet スイートに基づいてプロジェクトを作成するには、「ファイル」-> 「JAD/JAR ファイルからプロジェクトを作成」を選択します。使用する記述子を選択し、「開く」をクリックします。記述子と JAR ファイルは、同じディレクトリに存在する必要があります。

2.5 難読化ツールの使用

「難読化ツール」は、クラスファイルのサイズを小さくするツールです。ダウンロード時間を最小限に抑え、製造元や通信事業者から指定される厳しい JAR ファイルのサイズ制限に従うために、MIDlet スイートはコンパクトである必要があります。難読化ツールの使用は、MIDlet スイートの JAR ファイルのサイズを小さく維持できる方法の 1 つ (唯一ではない) です。

開発サイクルのパッケージ化の手順で難読化ツールを使用できます。Sun Java™ Wireless Toolkit for CLDC に難読化ツールは含まれていませんが、ProGuard 難読化ツールを使用できるように設定されています。必要な作業は、ProGuard をダウンロードし、ツールキットで検索可能な場所に置くことだけです。

ProGuard は、GNU 一般公衆利用許諾契約書 (GPL) の条件に従って公開されています。利用許諾契約書の条件に同意する場合は、ProGuard を無償でダウンロードして使用できます。

2.5.1 ProGuard のインストール

ProGuard をインストールするには、次の手順を実行します。

1. **ProGuard の Web サイト** `http://proguard.sourceforge.net/` **にアクセスします。**
2. **最新バージョンをダウンロードします。**
3. **ProGuard のインストールディレクトリの** `lib` **ディレクトリにある** `proguard.jar` **ファイルを、***toolkit*`¥bin` **ディレクトリに圧縮解除して入れます。**

2.5.2 ProGuard の使用

ProGuard のインストール後に、「プロジェクト」->「パッケージ」->「難読化パッケージを作成」を選択すると、ProGuard を使用できます。

場合によっては、難読化ツールの動作を制御するスクリプトファイルを用意する必要があります。たとえば、`Class.forName()` を使用してクラスをロードする場合は、クラス名を変更しないように ProGuard に指示する必要があります。

スクリプトファイルを呼び出すには、`ktools.properties` ファイルのコピーを変更できるようにしてください。

- *toolkit*`/wtklib/`*os*`/ktools.properties` **を** *workdir*`/wtklib/`*os*`/ktools.properties` **にコピーします。**

  テキストエディタでスクリプトファイルを作成し、プロジェクトのメインディレクトリに保存します。スクリプトファイルの詳細については、ProGuard のマニュアルを参照してください。次に、このファイルの見つけ方をツールキットに指示します。*workdir*`/wtklib` にコピーした `ktools.properties` ファイルを編集します。次のような行を追加します。

  `obfuscate.script.name:` *scriptfile*

  scriptfile は、スクリプトファイルに使用した名前で置き換えてください。このファイルの編集内容を有効にするには、ツールキットをいったん終了して再起動する必要があります。

2.6 デバッガの使用

アプリケーションの実行手順では、アプリケーションをデバッガで実行することもできます。デバッガを使用すると、実行中のアプリケーションをより詳細に監視したり、ブレークポイントを設定したり、変数を調べたりできます。

デバッガはユーザー自身で用意する必要があります。Java SE プラットフォームの `jdb` デバッガなど、任意のデバッガを使用できます。デバッガを使用する場合、一般的には Sun Java™ Wireless Toolkit for CLDC に対応した Sun Java Studio Mobility ソフトウェアなどの統合開発環境 (IDE) を使用できます。詳細は、`http://www.sun.com/software/products/jsmobility/` を参照してください。

最初に、「プロジェクト」->「デバッグ」を選択します。デバッガでエミュレータに接続するために使用する TCP/IP ポート番号を入力します。「デバッグ」をクリックします。エミュレータが起動し、デバッガからの接続待ちの状態になります。

デバッガを起動し、指定したポートに接続します。リモートデバッガを設定し、このデバッガを TCP/IP を使用してリモートモードで動作させる必要があります。詳細については、デバッガのマニュアルを参照してください。

Debugging MIDlets に、Sun Java™ Wireless Toolkit for CLDC で `jdb` を使用する場合の情報が記載されています。これは `http://developers.sun.com/techtopics/mobility/midp/questions/jdb/` で入手できます。

2.7 Web サーバーへのアプリケーションの配備

MIDP 2.0 の仕様には、Over The Air User Initiated Provisioning Specification という仕様が含まれています。この仕様には、MIDlet スイートをデバイスに無線 (OTA) で転送する方法が記載されています。Sun Java™ Wireless Toolkit for CLDC のエミュレータを使用して、このようなシナリオをテストできます。

パッケージ化した MIDP アプリケーションを Web サーバーにリモートで配備するには、Java Application Descriptor (JAD) ファイルの `MIDlet-Jar-URL` プロパティーを JAR ファイルの URL に変更します。この URL は絶対パスで指定する必要があります。たとえば、次のようになります。

```
MIDlet-Jar-URL: http://your.server.com/midlets/example.jar
```

次に、Web サーバーが JAD ファイルと JAR ファイルの正しい MIME タイプを使用できるようにします。

- MIDlet スイートの記述子については、拡張子 `.jad` を MIME タイプ `text/vnd.sun.j2me.app-descriptor` に対応付けます。
- MIDlet スイートの JAR ファイルについては、拡張子 `.jar` を MIME タイプ `application/java-archive` に対応付けます。

Web サーバーの設定方法は、使用するソフトウェアによって異なります。

エミュレータは、OTA 配信中のデバイスの動作を実装します。エミュレータを使用して、サーバーからデバイスへ MIDlet スイートを配信する完全なプロセスをテストおよびデモできます。必要な作業は、エミュレータの AMS を起動することだけです。「OTA 経由で実行」オプションを使用したことがあるユーザーは、すでに AMS の操作に慣れているかもしれません。

エミュレータの AMS を起動する方法は 2 つあります。

- Microsoft Windows で、「スタート」->「プログラム」->「Sun Java Wireless Toolkit 2.5.2 for CLDC」->「OTA Provisioning」を選択します。
- コマンド行で、次のコマンドを実行します。

Windows:	*toolkit*¥bin¥emulator -Xjam
Linux:	*toolkit*/bin/emulator -Xjam

AMS のプロンプトに従って、アプリケーションをインストールします。この手順は、この章で説明した「OTA 経由で実行」オプションの場合とほぼ同じですが、アプリケーションをインストールするサーバーの URL を入力する必要があります。

第3章

プロジェクトの使用

前章では、MIDP の開発サイクルで Sun Java™ Wireless Toolkit for CLDC がどのように役立つかについて説明しました。この章では、プロジェクトの使用法について、より詳細に説明します。次のような内容が含まれます。

- プロジェクト用のターゲット API の選択
- MIDlet のリストを含む、MIDlet スイートの属性の操作
- プロジェクトのディレクトリ構造についての理解
- プロジェクトへのサードパーティ製ライブラリの組み込み

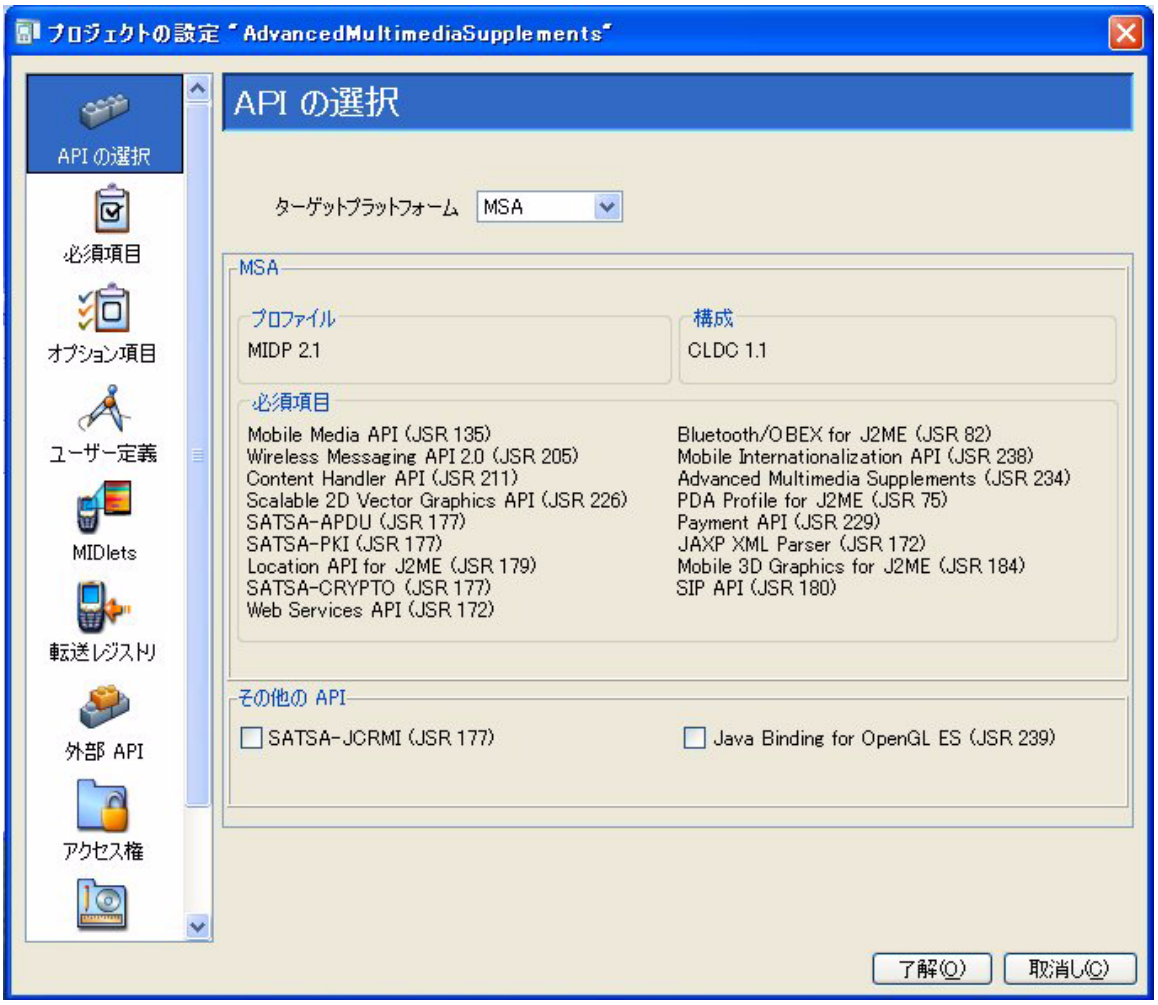
3.1 API の選択

各プロジェクトは、API セットに基づいてビルドされます。Sun Java™ Wireless Toolkit for CLDC では、多数の API がサポートされています。完全なリストは 1-7 ページの 1.6 節「サポートされている技術」に示しています。ツールキットでは、アプリケーションを開発するときに、アプリケーションが実行されるターゲットデバイスの種類に応じて API のサブセットを選択できます。

たとえば、ツールキットは JSR 184 「Mobile 3D Graphics API」に対応していますが、開発するアプリケーションでその API を使用しない場合もあります。プロジェクトの「API の選択」にある設定を使って、使用する API だけを選択できます。

この機能について確認するには、ツールキットを起動し、プロジェクトを開きます。「属性設定」をクリックすると、図 3-1 に示すウィンドウが表示されます。

図 3-1　「プロジェクトの設定」ウィンドウ



「API の選択」区画では、「ターゲットプラットフォーム」の設定に応じて、この区画の残り部分に表示される内容が変わります。適切なターゲットプラットフォームを選択し、下にあるコントロールを使って選択内容を微調整します。たとえば、「Java Technology for the Wireless Industry」の JSR に準拠するデバイス向けのアプリケーションを開発する場合は、コンボボックスで「JTWI」を選択します。次に、下にあるコントロールを使用して、CLDC のバージョンを指定し、オプションの API を選択します。

選択した内容は、ツールキットでソースコードをコンパイルする際に適用されます。

注 – API の選択は、エミュレータには適用されません。エミュレータは利用可能なすべての API を常にサポートしています。プロジェクト設定での API の選択は、プロジェクトのビルドだけに適用されます。基本的に、API の選択に応じて、ツールキットでソースファイルのコンパイルと実行前検証に使用されるクラスパスが選択されます。

3.2 MIDlet スイートの属性の変更

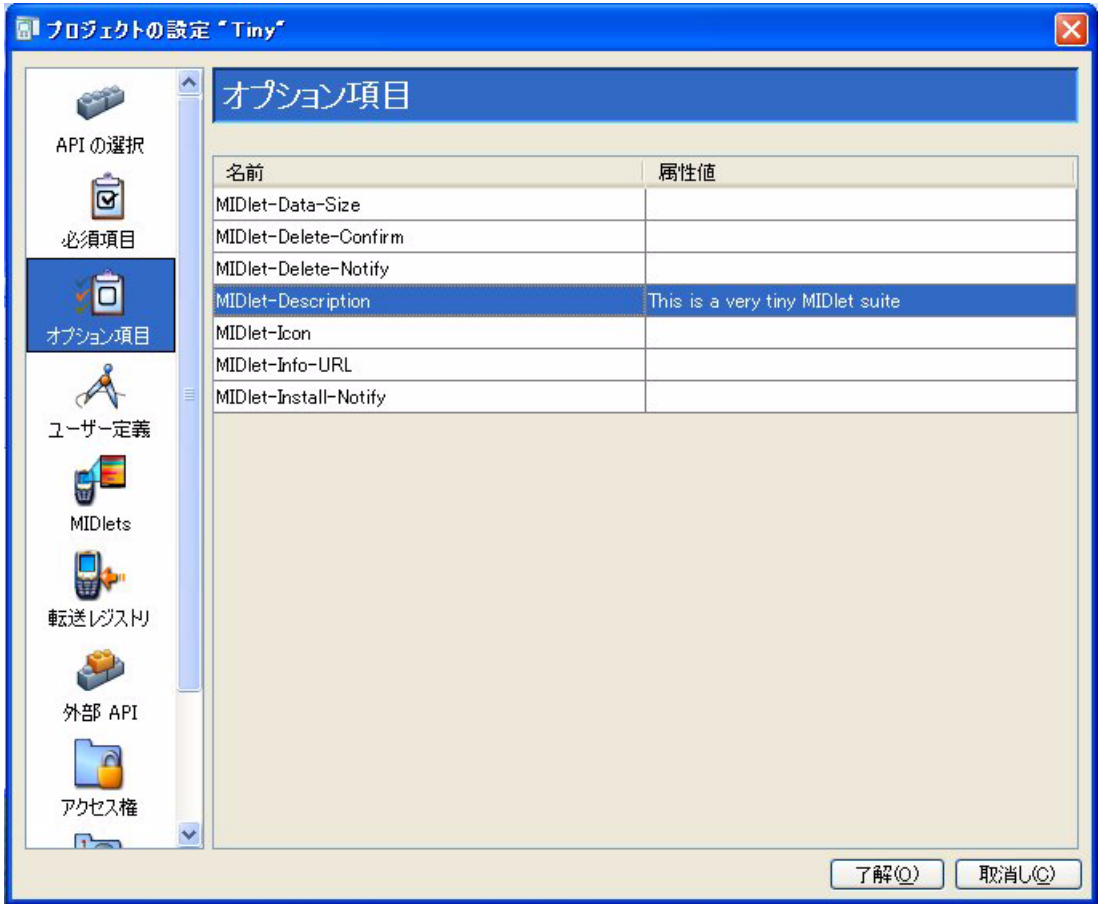
「プロジェクトの設定」ウィンドウでは、MIDlet スイートの属性も制御できます。これらは、MIDlet スイートの JAR ファイルのマニフェストファイルと記述子に保存されます。

属性を確認するには、プロジェクトを開き、「属性設定」ボタンをクリックします。設定ウィンドウの左にあるアイコンバーには、「必須項目」、「オプション項目」、および「ユーザー定義」の 3 つの属性アイコンが表示されます。

必須属性とオプション属性の定義については、MIDP 2.0 仕様の最終リリース 2 (MIDP 2.1) を参照してください。詳細の大部分は Sun Java™ Wireless Toolkit for CLDC で適切に処理されます。開発の初期段階では、属性について気にする必要はありません。アプリケーションが安定して動作するようになり、実際のデバイスへの配備や市場投入を検討し始めた段階で、属性値を調整します。

「必須項目」または「オプション項目」区画で値を調整するには、変更する属性キーの横にあるセルをクリックします。新しい値を入力します。

図 3-2　MIDlet スイートの属性の編集



新しいユーザー定義属性を作成するには、「ユーザー定義」アイコンをクリックします。「追加」ボタンをクリックしてプロパティーの名前と値を入力し、「了解」をクリックします。

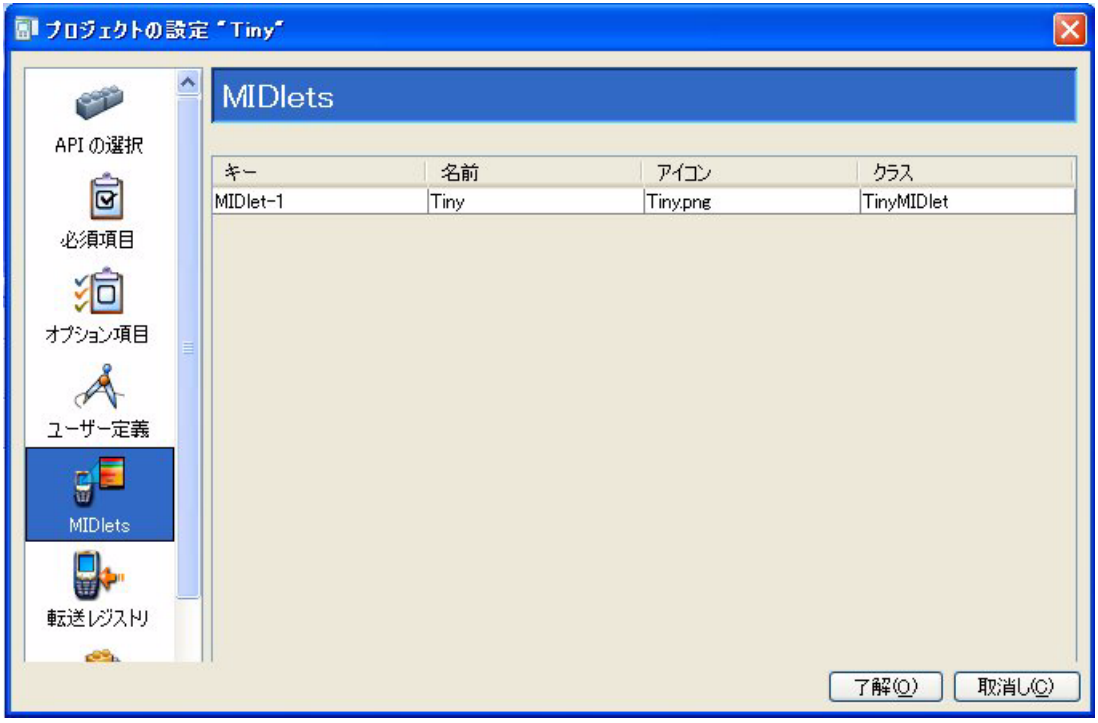
ユーザー定義のプロパティー値を編集するには、必須属性やオプション属性の場合と同様に、キーの横にある値の列をクリックします。

属性を削除するには、属性を選択して「削除」をクリックします。

3.3 MIDlet の操作

プロジェクト設定では、現在の MIDlet スイートプロジェクトに含まれる MIDlet の追加や変更を行うこともできます。この機能について確認するには、ツールキットを起動し、既存のプロジェクトを開きます。「属性設定」をクリックし、「MIDlets」アイコンを選択します。プロジェクト内の MIDlet のリストが表示されます。新しいプロジェクトを作成した直後は、ツールキットによって自動的に最初の MIDlet エントリが入力されています。

図 3-3　プロジェクトの MIDlet のリスト



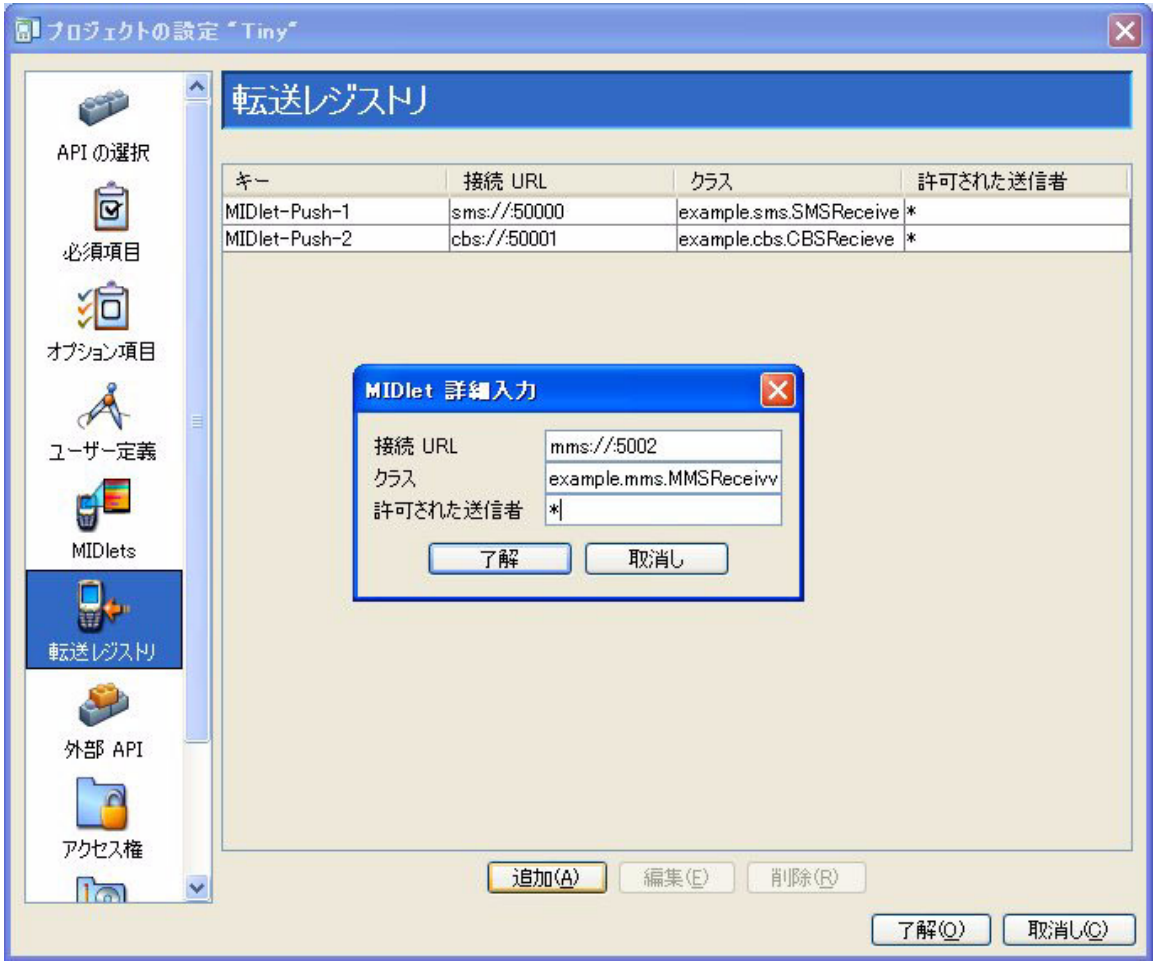
新しい MIDlet を追加するには、「追加」をクリックします。名前、アイコンファイル名、およびクラス名を入力します。アイコンファイル名は空白のままでもかまいません。MIDlet エントリの値を変更したり、エントリを削除したりするには、「編集」ボタンまたは「削除」ボタンを使用します。

MIDlet の名前は、MIDlet スイートが起動された順に表示されます。順序を変更するには、MIDlet を選択し、「上へ」または「下へ」をクリックします。

3.4 転送レジストリの使用

プロジェクト設定を使用して、MIDlet スイートの転送レジストリの設定を操作することもできます。「属性設定」をクリックし、「転送レジストリ」アイコンを選択します。

図 3-4 プロジェクトの転送レジストリの設定



転送レジストリにエントリを追加するには、「追加」をクリックし、接続 URL、MIDlet クラス、および許可された送信者の値を入力して、「了解」をクリックします。エントリを編集するには、エントリを選択し、「編集」ボタンをクリックします。転送レジストリのエントリを削除するには、エントリを選択し、「削除」をクリックします。

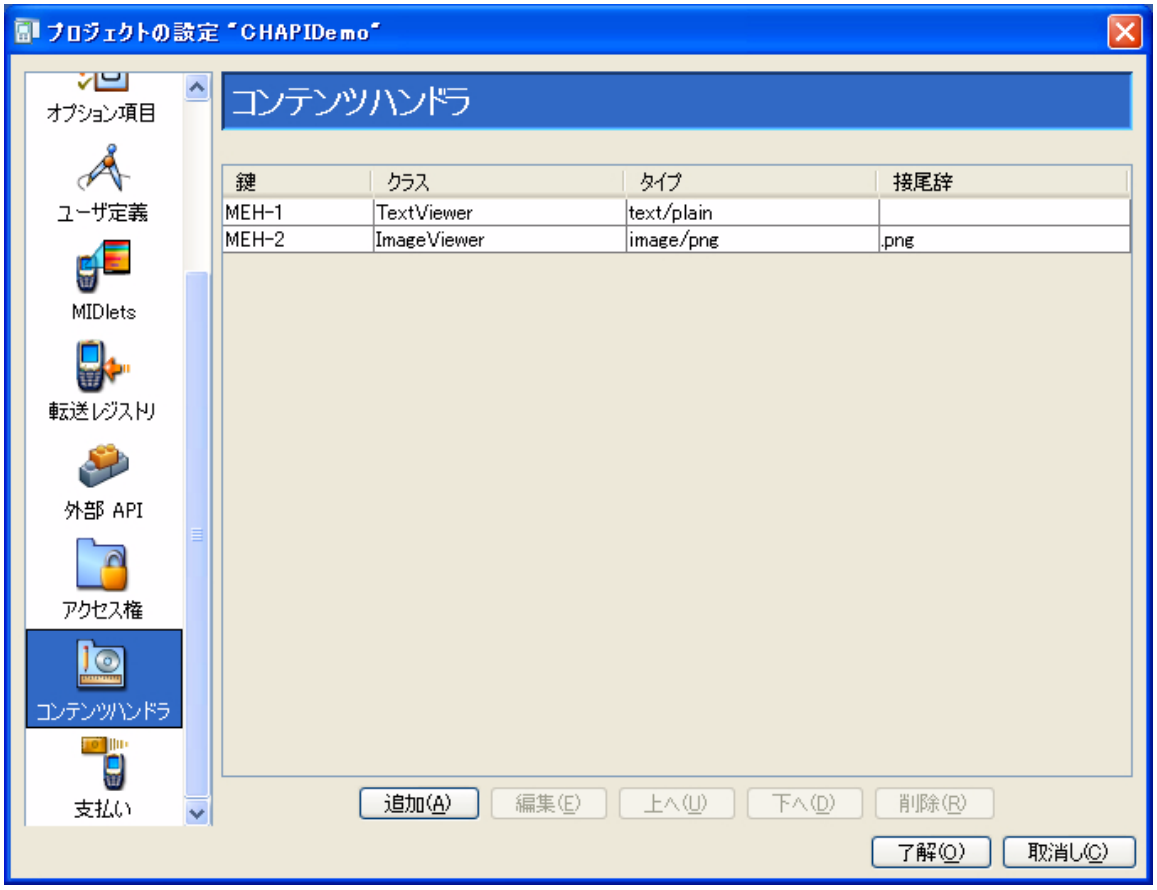
アプリケーション用に転送レジストリのエントリを作成する場合は、適切なアクセス権も必ず入力してください。詳細は、第 6 章を参照してください。

3.5 コンテンツハンドラの設定

Sun Java™ Wireless Toolkit for CLDC では、JSR 211 で定義されている「Content Handler API」(CHAPI) をサポートします。CHAPI の基本的な概念は、コンテンツ (ファイル) の着信に応じて MIDlet を起動できるということです。最近の携帯電話は、SMS、赤外線、Bluetooth、電子メール、およびその他の方法を使用してコンテンツを受信できます。ほとんどのコンテンツには、コンテンツ形式が対応付けられます。CHAPI は、特定のコンテンツの種類に応じて MIDlet を起動できるシステムを指定します。

プロジェクトでのコンテンツハンドラの設定を変更するには、「属性設定」をクリックして、「コンテンツハンドラ」区画を選択します。

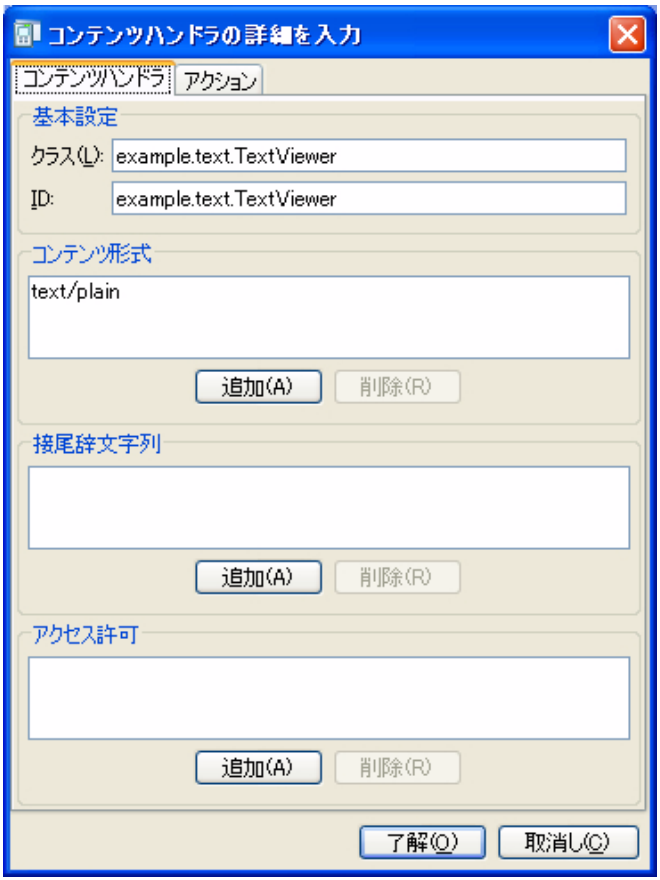
図 3-5　　コンテンツハンドラの設定



リストの各行は、コンテンツハンドラの設定を示しています。この例では、2 つのコンテンツハンドラが設定されています。1 つは `TextViewer` 用、もう 1 つは `ImageViewer` 用です。新しいコンテンツハンドラを作成する場合は「追加」をクリックし、既存のコンテンツハンドラを編集する場合は「編集」をクリックします。コンテンツハンドラの順序を調整するには、コンテンツハンドラを選択して「上へ」および「下へ」ボタンを使用します。コンテンツハンドラをリストから削除するには、「削除」をクリックします。

コンテンツハンドラを追加または編集するときには、詳細ウィンドウが表示されます。

図 3-6　　コンテンツハンドラの詳細設定



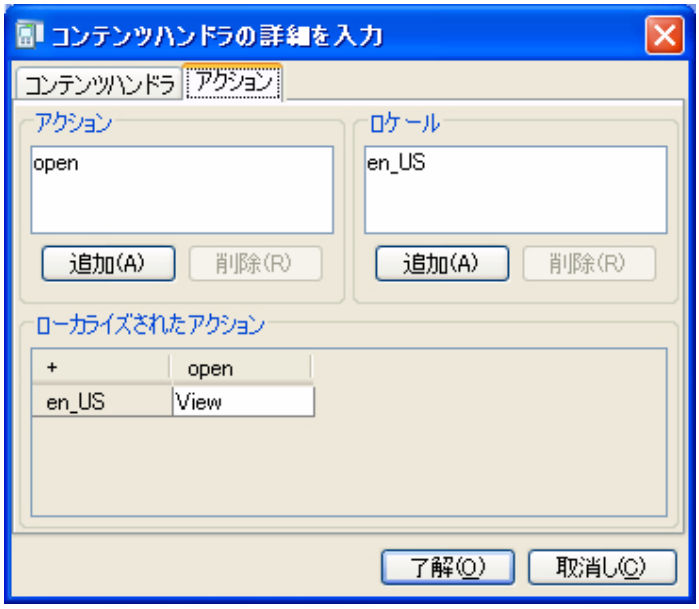
「クラス」フィールドに、MIDlet の名前を入力します。「ID」には、コンテンツハンドラの呼び出しやアクセスの制御に使用できる識別文字列を入力します。

「コンテンツ形式」は、このコンテンツハンドラで処理するコンテンツ形式のリストです。「追加」および「削除」ボタンを使用して、リストを管理します。「接尾辞文字列」は、明示的なコンテンツ形式の代用となる URL 接尾辞のリストです。「アクセス許可」は、このコンテンツハンドラにアクセスできるほかのコンテンツハンドラを示す、コンテンツハンドラ ID のリストです。リストが空の場合、すべてのコンテンツハンドラがこのコンテンツハンドラにアクセスできます。

コンテンツハンドラにはアクションが関連付けられ、これにより呼び出しアプリケーションはコンテンツの処理方法を選択できます。たとえば、イメージビューアコンテンツハンドラには、画像を元のサイズで表示するためのアクションや、画像を表示可

能な画面領域に拡大縮小するアクションなどが含まれます。「コンテンツハンドラの詳細を入力」ウィンドウで「アクション」タブをクリックして、コンテンツハンドラのアクションを編集します。

図 3-7　コンテンツハンドラのアクション



「アクション」リストには、このコンテンツハンドラのアクションの内部名が表示されます。「ロケール」は、人間が判読できるアクション名を指定するすべてのロケールのリストです。「ローカライズされたアクション」は、人間が判読できるさまざまなロケールのアクション名を含むグリッドです。各ロケールが行で指定され、アクションは列で示されます。特定のロケールの、人間が判読できるすべてのアクション名を、1 つの行で確認することができます。

3.6 プロジェクトのディレクトリ構造

プロジェクトには、標準のディレクトリ構造があります。プロジェクト自体は、apps サブディレクトリ内のディレクトリとして表されます。たとえば、Windows では demos プロジェクトは、*workdir*¥apps¥demos に格納されます。次の表で、プロジェクトのディレクトリ構造を説明します。

表 3-1 プロジェクトのディレクトリ構造

ディレクトリ	説明
bin	プロジェクトをパッケージ化すると、MIDlet スイートの記述子と JAR ファイルがこのディレクトリに置かれます。このディレクトリには、パッケージ化されていないマニフェスト情報も格納されます。「OTA 経由で実行」を使用する場合は、内部的に使用される HTML ファイルが置かれる場合もあります。
classes	このディレクトリは、コンパイル済みのクラスファイルを保存するために使用されます。
lib	このディレクトリには、このプロジェクトに組み込むサードパーティ製のライブラリを置きます。
res	このディレクトリには、画像、サウンド、およびその他のリソースファイルを置きます。これらはパッケージ化されて、MIDlet スイートの JAR ファイルのルートに置かれます。
src	このディレクトリには、ソースファイルを置きます。
tmpclasses	このディレクトリは、ツールキットで使用されます。
tmplib	このディレクトリは、ツールキットで使用されます。

このほかに、プロジェクトディレクトリには、プロジェクトに関する情報を含む project.properties ファイルもあります。

一時ディレクトリや一時ファイルをプロジェクトから削除するには、「プロジェクト」->「消去」を選択します。

3.7 サードパーティ製ライブラリの使用

Sun Java™ Wireless Toolkit for CLDC では、サードパーティ製のライブラリをアプリケーションに組み込むことができます。サードパーティ製のライブラリを使用すると、自分で機能を構築する必要がなくなるので、開発時間を短縮できます。ただし、MIDlet スイートの JAR ファイルのサイズに注意してください。

アプリケーションにサードパーティ製のライブラリを使用すると、そのライブラリのサイズだけ JAR ファイルのサイズが増加します。難読化ツールを使用すると、コードサイズを縮小できます。優れた難読化ツールでは、ライブラリの中で使用されていない部分をすべて除去することも可能です。ただし、難読化ツールを使用しても、多くの場合は、サードパーティ製のライブラリの方がユーザーが自分で作成したカスタムコードよりも大きくなります。開発時間の短縮と MIDlet スイートの JAR ファイルのサイズの兼ね合いを見極める必要があります。

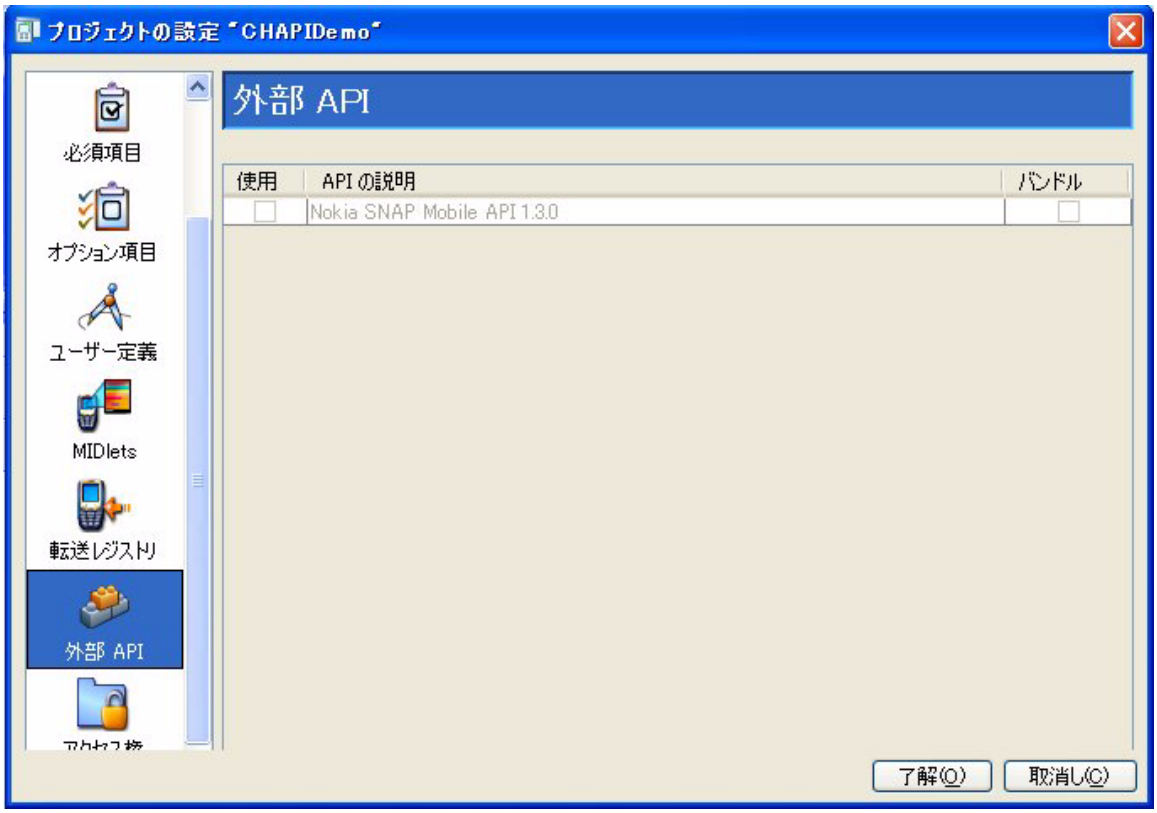
Sun Java™ Wireless Toolkit for CLDC には、サードパーティ製のライブラリを組み込むための方法が 2 つ用意されています。プロジェクト設定の「外部 API」区画を使用すると、プロジェクトにライブラリを組み込んだり、解除したりすることが簡単にできます。また、ライブラリを特定の場所に配置して、1 つまたはすべてのプロジェクトで利用するようにできます。

3.7.1 外部 API の使用

組み込む API を指定するには、「属性設定」をクリックし、「外部 API」アイコンをクリックします。利用可能な外部 API のリストが表示されます (リストが空の場合もあります)。

ビルドの際に API をクラスパスに追加するには、「使用」の列のボックスにチェックマークを付けます (図 3-8 を参照)。API をアプリケーションにバンドルする場合は、「バンドル」ボックスにもチェックマークを付けます。アプリケーションを配備するデバイスに選択した外部 API が存在しないことがわかっている場合は、アプリケーションに API をバンドルする必要があります。

図 3-8　外部 API の選択



3.7.2　1 つのプロジェクトに使用するサードパーティ製ライブラリ

プロジェクトの `lib` ディレクトリに置かれているライブラリファイルはすべて、プロジェクトのビルドおよびパッケージ化で組み込まれます。ライブラリは、Java テクノロジクラスの JAR ファイルまたは ZIP ファイルでなければなりません。

したがって、Windows でインストールを行い、`Tiny` という名前のアプリケーションの場合、クラスライブラリは *workdir*`¥apps¥Tiny¥lib` になります。プロジェクトをビルド、実行、デバッグ、およびパッケージ化する場合には、その `lib` ディレクトリのクラスファイルが使用されます。

3.7.3 すべてのプロジェクトに使用するサードパーティ製ライブラリ

デバイスによっては、インストールされるすべての MIDlet スイートにそのデバイスのライブラリを使用できることがあります。たとえば、製造元が自社製のすべてのデバイスに追加の API を用意している場合などです。このような場合は、アプリケーションのビルドやテストを行うときに、これらのライブラリを使用できるとよいでしょう。ただし、ライブラリはあらかじめデバイス上に用意されているので、そこにインストールする MIDlet スイートのパッケージにはライブラリを組み込まないようにします。

このようにするには、ライブラリを *workdir*¥apps¥lib ディレクトリに置きます。このディレクトリにあるライブラリは、すべてのプロジェクトで使用できます。

3.8 Wireless Toolkit の設定

ツールキットには上級者向けの設定オプションがあります。ktools.properties ファイルのコピーを編集することで、これらのオプションを使用できます。このファイルは、次の場所にあります。

Windows:	*toolkit*¥wtklib¥Windows¥ktools.properties
Linux:	*toolkit*/wtklib/Linux/ktools.properties

ktools.properties を *workdir*/*os*/wtklib にコピーし、この節で説明するように変更します。

- コンソールのフォントの変更
- アプリケーションディレクトリの設定
- javac のエンコーディングプロパティーの設定
- リビジョン制御システムの使用

ktools.properies に対する変更は、ツールキットを次に起動したときに有効になります。

3.8.1 コンソールのフォントの変更

2 つのプロパティーを編集することで、コンソール (およびその他のテキスト領域) で使用されるフォントを変更できます。ここでは、フォントを 20 ポイントの Times New Roman に変更する例を示します。

```
font.JTextArea=Times New Roman
font.size.JTextArea=20
```

デフォルトのフォントおよびサイズに戻すには、これらのプロパティーを削除します。

3.8.2 アプリケーションディレクトリの設定

Sun Java™ Wireless Toolkit for CLDC のデフォルト設定では、アプリケーションは作業用ディレクトリの `apps` サブディレクトリに格納されます。`ktools.properties` ファイルに次の行を追加すると、アプリケーションの格納先のディレクトリを変更することができます。

`kvem.apps.dir:` *application-directory*

Windows の場合、ディレクトリパスのすべての円記号 (¥) の前に、円記号をもう 1 つ追加する必要があります。また、空白文字を含むディレクトリのパスは指定できません。

たとえば、アプリケーションディレクトリを `D:¥dev¥midlets` に設定するには、次の行を追加します。

```
kvem.apps.dir: D:¥¥dev¥¥midlets
```

Linux のパスは、通常どおりに指定できます。

3.8.3 javac のエンコーディングプロパティーの設定

デフォルト設定では、`javac` は実行中の Java SE プラットフォームのエンコーディングセットを使用します。デフォルトのソースファイルエンコーディングを上書きする方法については、付録 C を参照してください。

3.8.4 リビジョン制御システムの使用

`filterRevisionControl` プロパティーを使用すると、SCCS、RCS、CVS の各リビジョン制御システムで作成された補助ファイルをツールキットに認識させたり、無視させたりすることができます。

補助ファイルを認識させたり、無視させたりするには、ktools.properties に次の行を追加します。

```
kvem.filterRevisionControl: true
```

これにより、ツールキットではリビジョン制御ファイルがソースファイルまたはリソースファイルとして扱われなくなります。たとえば、src¥SCCS¥s.MyClass.java というファイルは、SCCS リビジョン制御ファイルとみなされ、Java テクノロジソースファイルとしては扱われなくなります。

第4章

エミュレータの使用法

Sun Java™ Wireless Toolkit for CLDC のエミュレータは、デスクトップコンピュータ上で MIDP デバイスをシミュレートします。エミュレータを使用すると、アプリケーションが MIDP 環境でどのように動作するかを確認でき、デスクトップコンピュータ上だけですべての開発サイクルを行うことができます。

エミュレータは、特定のデバイスを表すものではなく、サポートされている API を正しく実装するものです。

4.1 エミュレータのスキン

「スキン」とは、エミュレータの表面にある薄い層のことで、エミュレータの外観、画面の特徴、入力コントロールなどを指定するものです。Sun Java™ Wireless Toolkit for CLDC には、さまざまな種類のデバイスを表すスキンが付属しています。

表 4-1 エミュレータのスキン

名前	画面サイズ	キャンバスサイズ	Colors	入力
DefaultColorPhone	240 x 320	240 x 289	4096	ITU-T
DefaultGrayPhone	180 x 208	180 x 177	4096	ITU-T
MediaControlSkin	180 x 208	180 x 177	4096	ITU-T
QwertyDevice	636 x 235	540 x 204	4096	Qwerty

必要に応じて、独自のエミュレータスキンを作成することもできます。詳細は、『基本カスタマイズガイド』を参照してください。

4.2 エミュレータのコントロール

エミュレータは、標準的なデスクトップウィンドウ内で携帯電話のように動作し、外観も似ています。この節では、エミュレータを制御する方法を説明します。説明や図は `DefaultColorPhone` スキンに基づいていますが、どのスキンの動作も同様です。

図 4-1　`DefaultColorPhone` エミュレータスキン

ボタンをマウスでクリックすることで、ボタンの押し下げをシミュレートできます。ほとんどのボタンにショートカットキーが割り当てられているので、通常はそれを使用する方が簡単です。キーボードの 0 ～ 9 の数字キーは、エミュレータの 0 ～ 9 のボタンに対応します。次の表では、いくつかのショートカットキーについて説明します。

表 4-2　ショートカットキー

エミュレータのボタン	キーボードキー
左ソフトウェアボタン	F1
右ソフトウェアボタン	F2
POWER ボタン	Esc
SELECT ボタン	Enter

テキストの入力方法は、実際のデバイスでの入力方法と同じです。文字を入力するには、目的の文字が表示されるまで数字キーを繰り返し押します。たとえば、数字キー 5 を 2 回押すと、文字 K を入力できます。テキストの入力中にアスタリスクキー (*) を押すと、大文字、小文字、数字、記号を切り替えることができます。現在のモードは、画面上部のインジケータに表示されます。シャープ記号 (#) を押すと、空白を入力できます。

代わりに、キーボードで直接タイプしてテキストを入力することもできます。この入力方法は便利ですが、エンドユーザーのほとんどはこの方法を利用できないということを忘れないでください。

また、テキスト領域で情報をコピーしてペーストすることもできます。Ctrl キー + v キーを押すと、クリップボードからテキストボックスまたはテキストフィールドにテキストをペーストできます。テキストボックスまたはテキストフィールドの内容をコピーするには、Ctrl キー + c キーを押します。テキストフィールドの内容全体が、クリップボードにコピーされます。

4.3 エミュレータの環境設定

エミュレータの設定を調整して、特定のデバイスの動作により近づけたり、さまざまなリソース条件の下でアプリケーションをテストしたりできます。

4.3.1 ネットワークプロキシ

エミュレータでは、デスクトップコンピュータのネットワーク接続が使用されます。たとえば、HTTP 接続を行う MIDlet をエミュレータで実行する場合、エミュレータはデスクトップコンピュータのネットワーク設定を使用して HTTP 接続を試みます。

開発用コンピュータがファイアウォールの内にある場合は、HTTP 接続にプロキシサーバーを使用していることがあります。不明な場合は、ブラウザの設定を調べて、プロキシサーバーを使用しているかどうかを確認してください。

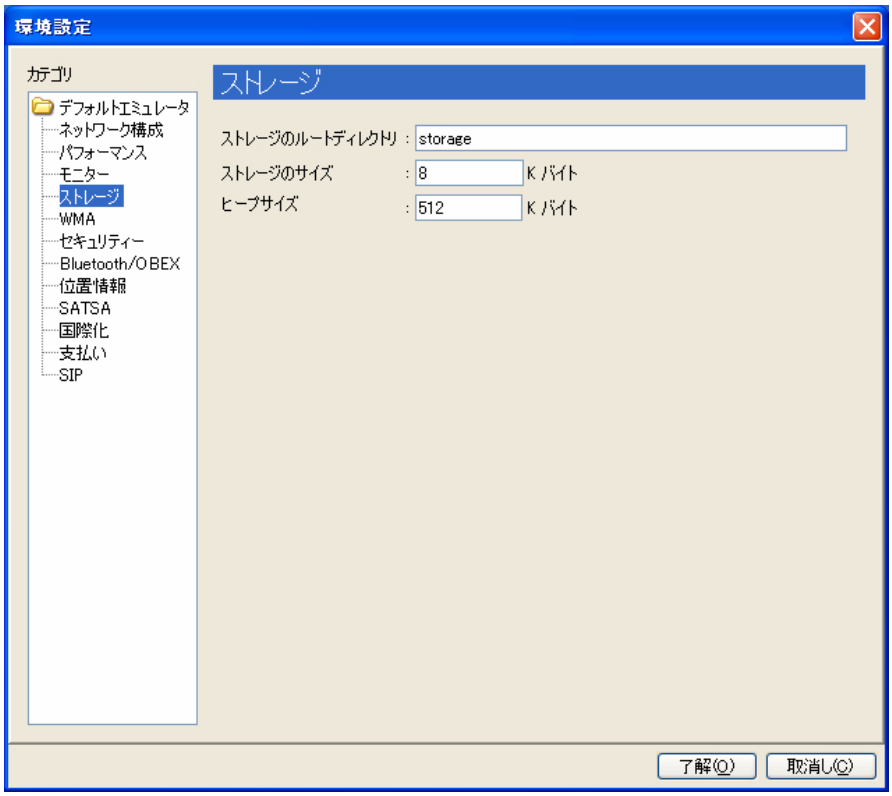
プロキシサーバーを使用している場合は、同じプロキシサーバーを使用するようにエミュレータを設定する必要があります。そのためには、「編集」->「環境設定」を選択します。「ネットワーク構成」区画で、使用するプロキシサーバーの名前とポート番号を入力します。使用する HTTP のバージョンも選択できます。

プロキシサーバーが HTTP 基本認証 (RFC 2617 を参照) を使用している場合は、「認証」にチェックマークを付け、ユーザー名とパスワードを入力します。

4.3.2 ストレージサイズ

各エミュレータに割り当てられた持続ストレージの量を設定または変更したり、ヒープサイズ (アプリケーションのオブジェクトに許可されるストレージの量) を変更したりすることができます。「編集」->「環境設定」を選択し、「ストレージ」の項目を選択します。

図 4-2　　ストレージの環境設定



4.3.2.1　持続ストレージ

エミュレータには持続的なストレージがあります。このストレージは、デフォルト設定では電話のスキンディレクトリの appdb サブディレクトリに置かれます。これらのファイルの拡張子は .db です。

たとえば、Windows では DefaultColorPhone エミュレータスキンの持続ストレージは、*workdir*¥appdb¥DefaultColorPhone¥manager_storage_settings.db に保存されます。

同じエミュレータスキンのインスタンスを同時に複数実行する場合、ツールキットでインスタンスごとに一意のファイルパスが生成されます。たとえば、Windows では DefaultColorPhone のインスタンスに、*workdir*¥appdb¥temp.DefaultColorPhone1、*workdir*¥appdb¥temp.DefaultColorPhone2 などのファイルパス名を割り当てられます。

注 – ファイル *workdir*¥appdb¥DefaultColorPhone¥in.use は、使用中としてマークされたストレージのルートの数を追跡します。エミュレータがクラッシュした場合は、in.use ファイルを削除する必要があります。

ツールキットでは、ストレージファイルの保存場所を変更したり、ストレージのサイズを制限したりできます。使用可能な持続ストレージが小さいときにアプリケーションの動作をテストする場合、この機能は役立ちます。

持続ストレージの設定を調整するには、「編集」->「環境設定」を選択し、左側の区画で「ストレージ」をクリックします。「ストレージのルートディレクトリ」フィールドに、持続ストレージに使用するディレクトリの名前を入力します。入力できるのは相対パスだけです。指定したディレクトリは、appdb サブディレクトリ内に作成されます。

デフォルトでは、1M バイト (1024K バイト) の持続ストレージを指定できます。K バイト単位で制限を入力できます。ストレージの実装では、アプリケーションの使用領域のほかに、いくらかのオーバーヘッドがあることに注意してください。たとえば、持続ストレージのサイズとして 8K バイトと入力した場合、アプリケーションデータとストレージオーバーヘッドの合計で 8192 バイトまで使用できます。

エミュレータの持続ストレージを消去するには、「ファイル」->「ユーティリティー」を選択します。「データベースの消去」ボタンをクリックして、持続ストレージを消去します。「データベースの消去」ボタンは、インストール済みのアプリケーションには影響しません。

4.3.2.2 ヒープサイズ

「ヒープ」とは、アプリケーションのオブジェクトが保存されるメモリーです。ヒープサイズを変更するには、「編集」->「環境設定」を選択し、「ストレージ」の項目を選択します (図 4-2 を参照)。デフォルトでは、ヒープサイズは 1M バイトです。最大ヒープサイズを設定して、実際のデバイスの条件により近いシミュレーションを行うことができます。「ヒープサイズ」フィールドに最大ヒープサイズを K バイト単位で入力します。

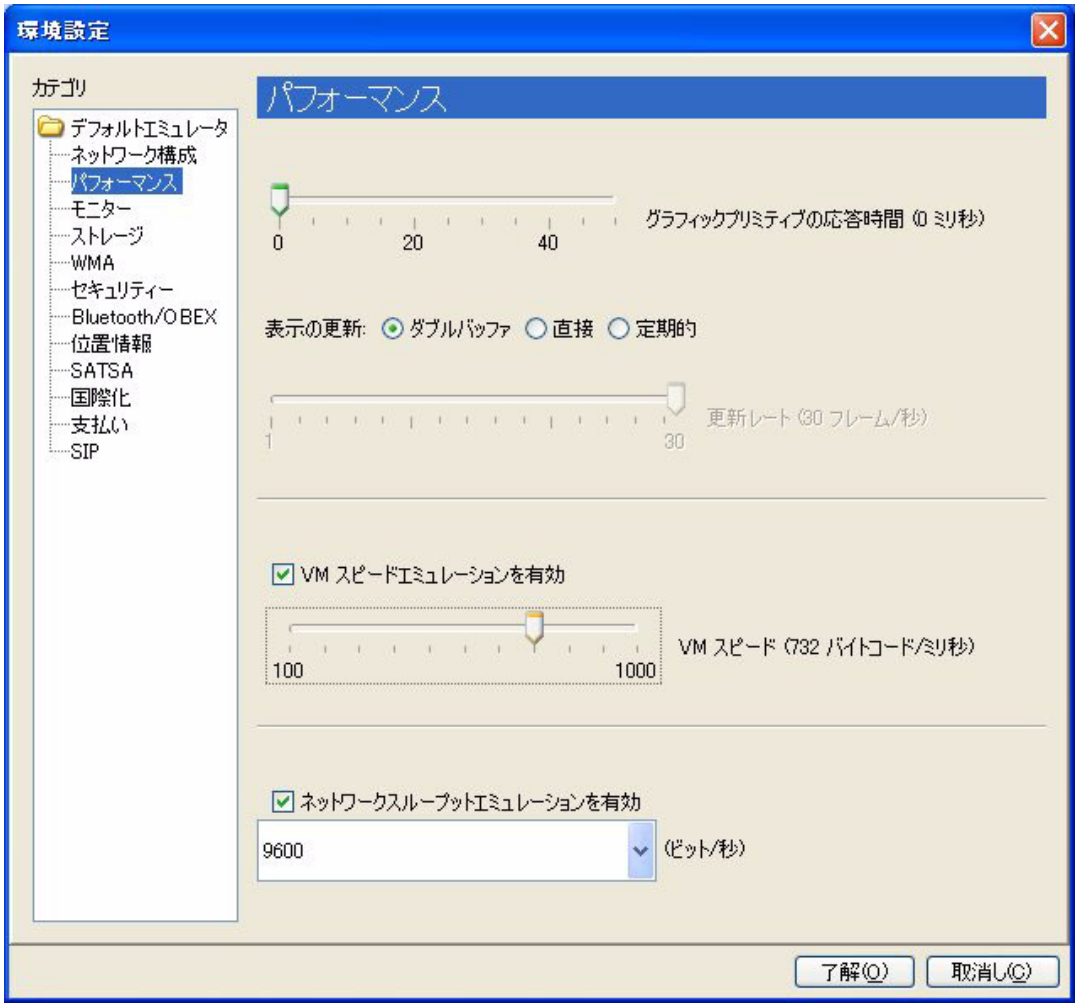
4.3.3 エミュレータのパフォーマンスの調整

エミュレータでは、ディスプレイやネットワーク接続など、デスクトップコンピュータのさまざまなリソースが使用されます。デスクトップベースのエミュレータに比べ、実際の MIDP デバイスでは、プロセッサは低速で、メモリは少なく、ネットワーク接続も低速であることが多く、ディスプレイの種類も異なる場合があります。

Sun Java™ Wireless Toolkit for CLDC では、実際のデバイスの制約された環境をシミュレートできます。エミュレータは実際のデバイスを表すものではありませんが、パフォーマンスの設定を調整することで、さまざまな実行条件の下でアプリケーションがどのように動作するかを確認できます。

「編集」->「環境設定」を選択し、左側の区画で「パフォーマンス」をクリックします。図 4-3 を参照してください。

図 4-3　エミュレータのパフォーマンスの調整



「グラフィックプリミティブの応答時間」を調整すると、`Graphics` クラスの描画メソッドをアプリケーションで呼び出した時点から実際に描画が行われるまでの時間が変化します。

画面の特徴を変更するには、「表示の更新」の種類から 1 つを選択します。「定期的」を選択した場合は、「更新レート」も指定する必要があります。

より低速な実際のデバイスをシミュレートするには、「VM スピードエミュレーションを有効」にチェックマークを付け、速度を選択します。

シミュレーションでのネットワークスピードを調整するには、「ネットワークスループットエミュレーションを有効」にチェックマークを付け、ビット/秒単位の速度を選択します。

4.4 一時停止と再開

MIDlet のライフサイクルは MIDP の仕様で定義されています。MIDlet の開始や停止は、デバイスによって行われます。さらに、着信などの外部イベントが発生したときに、デバイスで MIDlet を一時停止することもあります。

エミュレータでは、実行中の MIDlet の一時停止や再開を簡単に行うことができます。一時停止されたときのアプリケーションの動作をテストする場合に、この機能は非常に便利です。

エミュレータの実行中に、エミュレータウィンドウのメニューから「MIDlet」->「一時停止」を選択します。実行中の MIDlet が一時停止され、「着信...」のメッセージが画面に表示されます。

操作を再開するには、メニューから「MIDlet」->「再開」を選択します。

4.5 エミュレータの単独での実行

開発中は通常、「実行」ボタンをクリックするか、「プロジェクト」->「OTA 経由で実行」を選択して、ツールキットから直接エミュレータを実行することになります。ただし、テストやデモの目的で、エミュレータを単独で実行したい場合があります。この節では、いくつかの方法について説明します。Sun Java™ Wireless Toolkit for CLDC インストーラが作成するプログラムグループには、エミュレータを単独で実行するためのオプションがいくつか含まれています。

- アプリケーションを直接実行するには、「Run MIDP Application...」項目を選択します。これは、「実行」ボタンをクリックするのと似ています。ローカルディスク上の MIDlet 記述子ファイルの場所を指定するように求めるメッセージが表示されます。対応する MIDlet スイート JAR ファイルも存在している必要があります。

- エミュレータのアプリケーション管理ソフトウェア (AMS) を実行するには、「OTA Provisioning」を選択します。これは、ユーザーインタフェースの「OTA 経由で実行」の機能とほぼ同様です。エミュレータのウィンドウが開き、AMS の開始画面が表示されます。URL を入力することで、アプリケーションをインストールできます。
- エミュレータの環境設定を変更するには、ツールキットのプログラムグループから「環境設定」を選択します。「環境設定」ウィンドウが開きます。これは、ユーザーインタフェースで「編集」->「環境設定」を選択したときに開くウィンドウと同じです。
- Sun Java™ Wireless Toolkit for CLDC のユーティリティーも、ユーザーインタフェースを実行せずに単独で使用できます。「ユーティリティー」の項目を選択してください。
- 最後に、どのエミュレータスキンをデフォルトで使用するかを変更できます。「Default Device Selection」を選択し、使用可能なエミュレータスキンから 1 つを選択します。選択したスキンは、次にエミュレータを起動したときに使用されます。

コマンドプロンプトからエミュレータを実行することもできます。詳細は、付録 Bを参照してください。

4.6 サードパーティ製エミュレータの使用

デバイスの製造元やワイヤレス通信事業者などのサードパーティが、ツールキットと互換性のあるデバイスエミュレータを作成していることもあります。ほかのエミュレータをツールキットにインストールして、より多様な実装でアプリケーションを実行してみることができます。通常の手順では、サードパーティ製のエミュレータを解凍またはインストールし、そのディレクトリを *workdir*¥wtklib¥devices ディレクトリにコピーします。次にツールキットを起動したときに、このエミュレータが使用可能になります。

現在使用できるエミュレータの一部は、次のリストに記載されています。

http://developers.sun.com/techtopics/mobility/midp/articles/emulators/

第5章

アプリケーションの監視

Sun Java™ Wireless Toolkit for CLDC には、アプリケーションの動作を監視するためのツールがいくつか用意されています。これらのツールは、コードのデバッグや最適化を行うときに役立ちます。

- 「プロファイラ」には、アプリケーション内の各メソッドについて、使用頻度と実行時間が表示されます。
- 「メモリーモニター」には、アプリケーション実行時のメモリー使用率が表示されます。
- 「ネットワークモニター」には、アプリケーションによって送受信されたネットワークデータが表示されます。HTTP、HTTPS、SMS、CBS などの多数のネットワークプロトコルがサポートされています。
- 「トレース」では、低レベルの情報がツールキットのコンソールに出力されます。

注 – 監視機能を使用すると、アプリケーションの実行速度が低下することがあります。

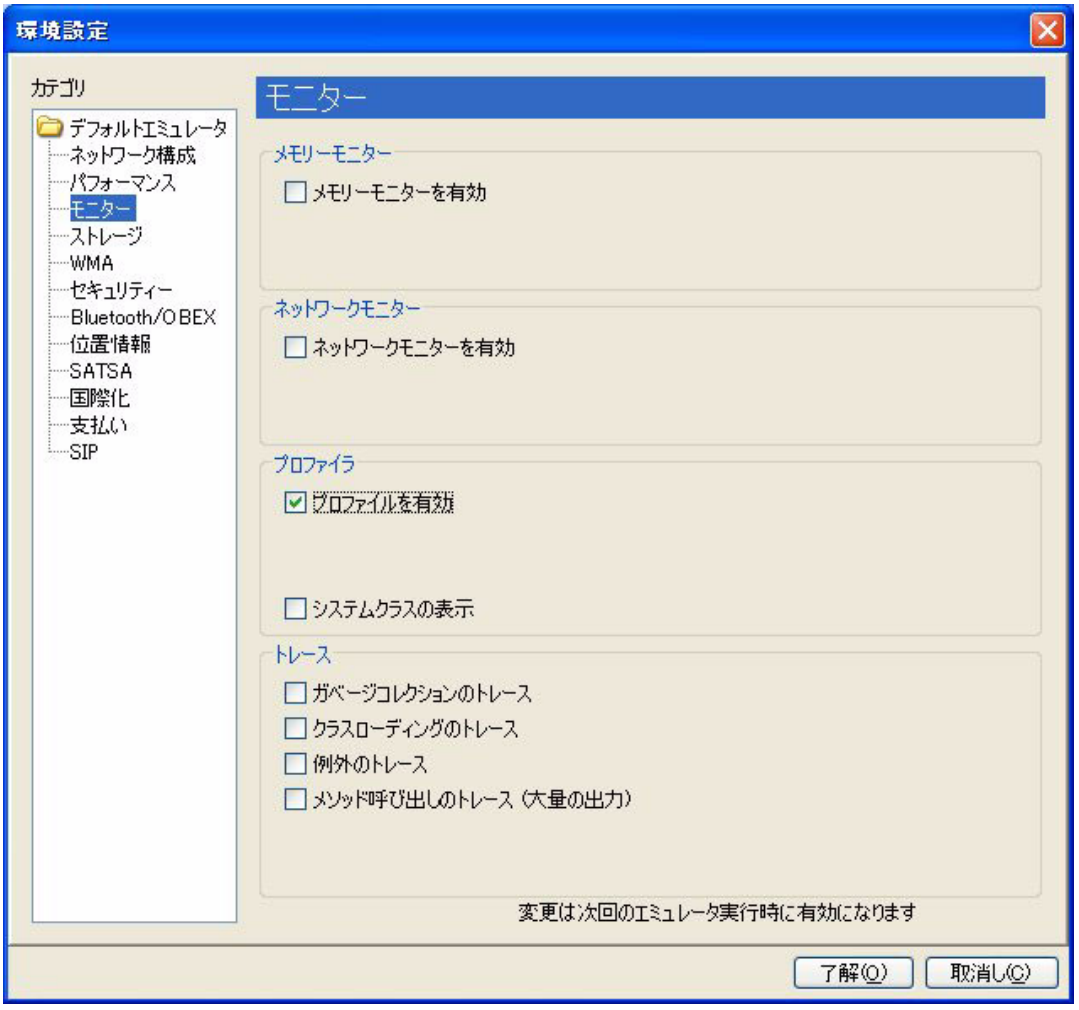
5.1 プロファイラの使用

プロファイラでは、アプリケーション内の各メソッドが追跡されます。アプリケーションの 1 回の実行について、各メソッドに費やされた時間と各メソッドが呼び出された回数が計算されます。アプリケーションの実行を終了し、エミュレータを停止すると、プロファイラのウィンドウが開きます。そこでメソッド呼び出しの情報を閲覧できます。

プロファイラを有効にするには、「編集」->「環境設定」を選択します。図 5-1 を参照してください。左側の区画で「モニター」をクリックします。右側の区画で、「プロファイルを有効」にチェックマークを付けます。すべてのシステム実装メソッドに

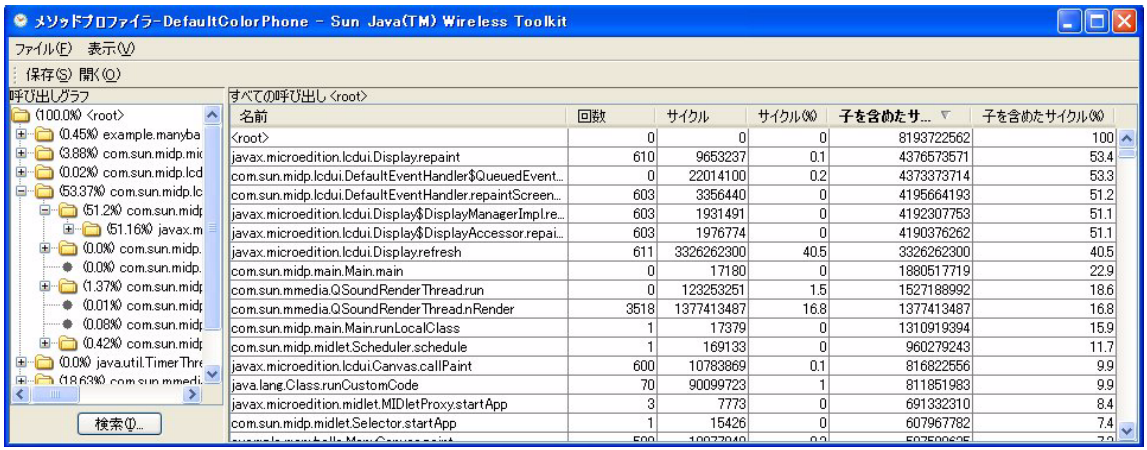
ついてプロファイル情報を表示するには、「システムクラスの表示」にチェックマークを付けます。それ以外の場合は、アプリケーションのメソッドへの呼び出しを含んでいるシステムメソッドだけがプロファイラに表示されます。「了解」をクリックします。

図 5-1　プロファイラの有効化



続いて、「実行」ボタンをクリックしてアプリケーションを起動します。通常のようにアプリケーションを操作します。操作が完了したら、エミュレータを停止します。プロファイラのウィンドウが開き、アプリケーション内のすべてのメソッド呼び出しに関する情報が表示されます。

図 5-2　　メソッドプロファイラ



プロファイラには、次の 2 種類の情報が表示されます。

- メソッドの関係が、「呼び出しグラフ」と呼ばれる階層リストで表示されます。
- プロファイラの右側には、各メソッドとその下位メソッドについて、実行時間と呼び出された回数が表示されます。

注 – エミュレータから取得されるプロファイル値は、実際のデバイスでの値を反映するものではありません。

5.1.1　呼び出しグラフ

呼び出しグラフは、メソッド呼び出しの階層を示します。ほかのメソッドを呼び出すメソッドは、フォルダとして表されます。メソッドをダブルクリックすると、そのメソッドで呼び出されるほかのメソッドを確認できます。ほかのメソッドをまったく呼び出さないメソッドは、グレーの円で表されます。

特定のクラス名やメソッド名を検索できます。「検索」をクリックし、名前を入力します。呼び出しグラフで現在選択されているものから、末尾に向かって検索が実行されます。呼び出しグラフ全体を検索するには、「折り返す」にチェックマークを付けてから、「検索」をクリックします。

呼び出しグラフで別のノードをクリックするたびに、そのノードのメソッドに関する詳細がプロファイラの右側に表示されます。

5.1.2 実行時間と呼び出された回数

プロファイラウィンドウの右側には、メソッドに関する詳細情報が表示されます。メソッドの名前、呼び出された回数、およびエミュレータでそのメソッドに費やされた時間を確認できます。実行時間は、4 つの方法で表されます。

- 「サイクル」は、そのメソッド自体に費やされたプロセッサ時間を示します。
- 「サイクル (%)」は、合計実行時間に対して、そのメソッド自体に費やされた時間の割合を示します。
- 「子を含めたサイクル」は、メソッド自体とそのメソッドで呼び出されたメソッドに費やされた時間を示します。
- 「子を含めたサイクル (%)」は、合計実行時間に対して、メソッド自体とそのメソッドで呼び出されたメソッドに費やされた時間の割合を示します。

いずれかの列をクリックすると、その列を基準にしてソートできます。もう一度クリックすると、昇順ソートと降順ソートを切り替えることができます。

右側の区画には、呼び出しグラフで現在選択されているノードについて、そこに含まれているメソッドが表示されます。すべてのメソッドを表示するには、呼び出しグラフで <root> ノードをクリックします。

5.1.3 プロファイラ情報の保存およびロード

プロファイラのセッションを保存するには、プロファイラウィンドウの「保存」ボタンをクリックします。ファイル名を指定します。

プロファイラのセッションをロードするには、「ファイル」->「ユーティリティー」を選択します。「プロファイラ」をクリックし、「起動」を押します。ファイルを選択すると、すべてのセッション情報がロードされたプロファイラのウィンドウが開きます。

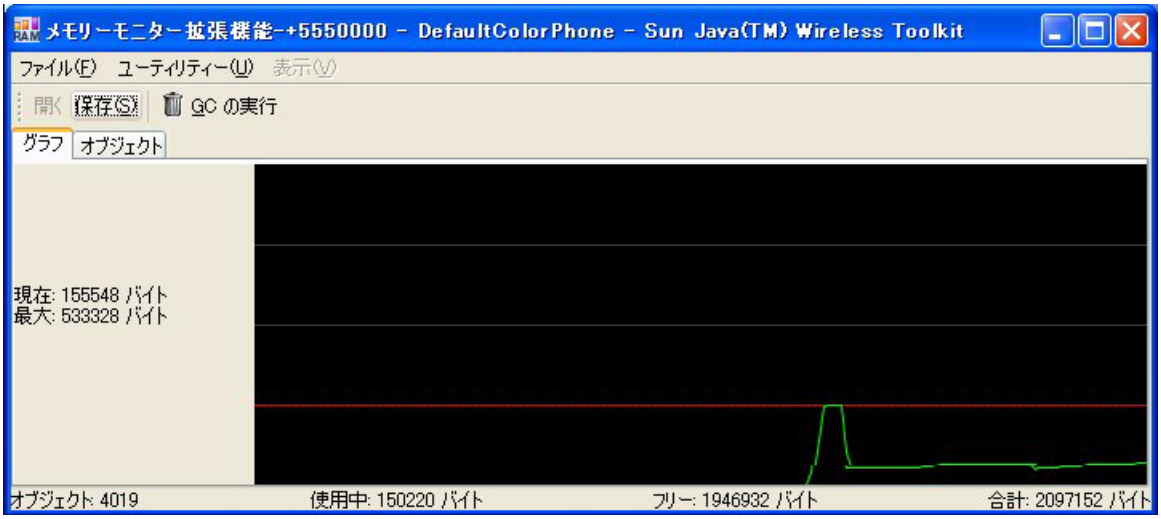
5.2 メモリーモニターの使用

多くの MIDP デバイスでは、メモリーの量が限られています。Sun Java™ Wireless Toolkit for CLDC には、アプリケーションのメモリー使用率を簡単に調べることのできるメモリーモニターが含まれています。アプリケーションで使用されるメモリーの合計量と、オブジェクトごとのメモリー使用率を示す詳細リストを確認できます。

メモリーモニターを有効にするには、「編集」->「環境設定」を選択します。左側の区画で「モニター」をクリックします。「メモリーモニターを有効」にチェックマークを付けます。

次にエミュレータを実行したときに、メモリーモニターのウィンドウが開き、アプリケーションのメモリー使用率の経時変化を示すグラフが表示されます。メモリーモニターを使用すると、作成されるすべてのオブジェクトが記録されるので、アプリケーションの起動が遅くなります。

図 5-3　メモリーモニターのグラフ



メモリーモニターのグラフには、次の情報が表示されます。

- **現在** - アプリケーションで使用された現在のメモリー量です。
- **最大** - プログラムの実行が開始された以降に使用された最大のメモリー量です。グラフ内では、赤の破線で示されます。
- **オブジェクト** - ヒープ内のオブジェクト数です。
- **使用中** - 使用されたメモリーの量です。
- **フリー** - 未使用で、使用できる状態のメモリーの量です。
- **合計** - 起動時に使用できる状態のメモリーの合計量です。

ヒープサイズを変更するには、「編集」->「環境設定」を選択し、「ストレージ」タブ選択します。詳細は、第 3 章を参照してください。

システムでガベージコレクションを実行させるには、「GC の実行」をクリックします。

注 – エミュレータで観察されるメモリー使用率は、実際のデバイスでのメモリー使用率と正確に一致するわけではありません。エミュレータは実際のデバイスを表すものではありません。サポートされている API を 1 つの方法で実装したものです。

アプリケーション内のオブジェクトについて詳細を表示するには、メモリーモニターウィンドウの「オブジェクト」タブをクリックします。

図 5-4　メモリーモニターのオブジェクトの表示

次の列を持つ表が表示されます。

- **名前** - オブジェクトのクラス名です。
- **実行中** - インスタンスの数です。ガベージコレクションの対象となるものも含まれます。
- **合計** - アプリケーションの開始以降に割り当てられたオブジェクトの合計数です。
- **合計サイズ** - オブジェクトで使用されるメモリーの合計量です。
- **平均サイズ** - オブジェクトの平均サイズです。アクティブなインスタンスの数で合計サイズを割って計算されます。

列のタイトルをクリックすると、その列を基準にしてソートできます。

特定のクラス名を検索するには、メモリーモニターウィンドウのメニューから「表示」->「検索」を選択します。

5.2.1 メモリーモニターの情報の保存およびロード

メモリーモニターのセッションを保存するには、「保存」ボタンをクリックします。ファイル名を指定します。

メモリーモニターのセッションをロードするには、「ファイル」->「ユーティリティー」を選択します。「メモリーモニター」をクリックし、「起動」を押します。ファイルを選択すると、すべてのセッション情報がロードされたメモリーモニターのウィンドウが開きます。

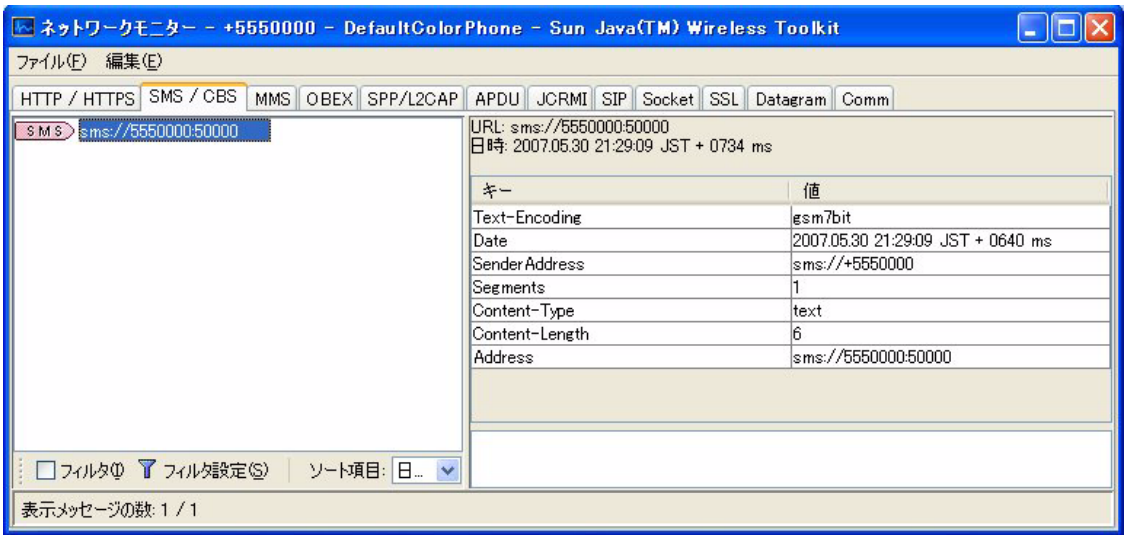
5.3 ネットワークモニターの使用

MIDP アプリケーションでは、少なくとも HTTP ネットワーク接続を行うことができますが、ほかにもさまざまなネットワーク接続が可能です。ネットワークモニターを使用すると、アプリケーションによってネットワーク上で送受信されている情報を簡単に確認できます。ネットワークでの対話をデバッグする場合や、ネットワークトラフィックの最適化方法を検討する場合に、この機能は便利です。

ネットワークモニターを有効にするには、「編集」->「環境設定」を選択します。左側の区画で「モニター」をクリックします。「ネットワークモニターを有効」にチェックマークを付けます。

次にエミュレータを実行したときに、ネットワークモニターのウィンドウが表示されます。

図 5-5　　ネットワークモニター



アプリケーションでネットワーク接続が行われると、その接続に関する情報が取り込まれて表示されます。図は、HTTP 要求と応答を示しています。

左側には、メッセージの階層とメッセージの一部分が表示されます。メッセージまたはメッセージの一部分をクリックすると、その詳細がネットワークモニターの右側に表示されます。メッセージまたはメッセージの一部分をダブルクリックすると、メッセージを展開または折りたたむことができます。

メッセージ本文は、16 進値の生データおよび対応するテキストで表示されます。

注 – すでに送信の準備に入っているメッセージを調べることもできます。不完全なメッセージは、メッセージツリー内でボールドで強調表示されます。

5.3.1　メッセージのフィルタ機能

フィルタは、ネットワークトラフィックの一部分を調べるときに役立ちます。フィルタ設定は、使用するネットワークプロトコルだけに適用されます。

フィルタを使用するには、「フィルタ設定」をクリックします。必要に応じてフィルタ設定を変更します。

表 5-1　ネットワークモニターのフィルタ設定

ネットワーク プロトコル	フィルタ設定
HTTP/HTTPS	URL、ステータス行、ヘッダ、本文など、HTTP メッセージのさまざまな部分と一致するテキストを入力します。たとえば、「URL」フィールドに `slashdot` と入力すると、URL に `slashdot` が含まれるメッセージだけが表示されます。
SMS/CBS	照合するプロトコル、メッセージタイプ、および方向を指定できます。送信者、受信者、およびメッセージの内容と照合するテキストも入力できます。
MMS	方向、送信者、受信者、CC 受信者、および BCC 受信者と照合するテキストを入力します。件名、コンテンツ ID、コンテンツの場所、MIME タイプ、およびエンコーディングに基づいてフィルタリングすることもできます。
OBEX、SPP/L2CAP	URL またはヘッダーの内容を使ってフィルタリングできます。
APDU、JCRMI	URL またはメッセージの内容に基づいてフィルタリングします。
SIP	利用できません。
Socket、SSL、Datagram、Comm	接続文字列 (URL) または内容のどちらかと照合するテキストを入力します。

フィルタ設定の入力が完了したら、「了解」をクリックしてネットワークモニターに戻ります。「フィルタ」チェックボックスにチェックマークが付きます。これは、フィルタが使用されていることを示します。フィルタを無効にしてすべてのメッセージを表示するには、チェックマークを解除します。

5.3.2　メッセージのソート

メッセージツリーを特定の順序で並べ替えるには、「ソート」コンボボックスをクリックし、条件を選択します。

- **日時** - 送信または受信した時間順に、メッセージがソートされます。
- URL - URL アドレスの順に、メッセージがソートされます。複数のメッセージに同じアドレスが定義されている場合は、時間順にソートされます。
- **接続** - 通信の接続によってメッセージがソートされます。複数のメッセージで同じ接続を使用している場合は、時間順にソートされます。このソートでは、要求、およびそれに関連する応答によってメッセージをグループ化して表示することができます。
- ソートのパラメータは、選択するメッセージプロトコルによって異なります。たとえば、時間順のソートはソケットメッセージには必要ありません。

5.3.3 ネットワークモニターの情報の保存およびロード

ネットワークモニターのセッションを保存するには、ネットワークモニターウィンドウのメニューから、「ファイル」->「保存」を選択するか、「ファイル」->「別名保存」を選択します。ファイル名を指定します。

ネットワークモニターのセッションをロードするには、「ファイル」->「ユーティリティー」を選択します。リストから「ネットワークモニター」を選択し、「起動」を押します。ファイルを選択すると、すべてのセッション情報がロードされたネットワークモニターのウィンドウが開きます。

5.3.4 メッセージツリーの消去

ネットワークモニターからすべてのメッセージを削除するには、ネットワークモニターのメニューから「編集」->「消去」を選択します。

第6章

セキュリティーと MIDlet の署名

MIDP 2.0 (JSR 118) には、保護ドメインに基づく総合的なセキュリティーモデルが含まれています。MIDlet スイートは保護ドメインにインストールされ、そこで保護された機能へのアクセスが決定されます。MIDP 2.0 仕様には、公開鍵暗号方式を使って MIDlet スイートの検証と認証を行うための推奨手順も含まれています。

明確な情報については、MIDP 2.0 仕様を参照してください。Sun Java™ Wireless Toolkit for CLDC を使用した MIDlet 署名の概要については、次の URL で Understanding MIDP 2.0's Security Architecture を参照してください。
`http://developers.sun.com/techtopics/mobility/midp/articles/permissions/`

公開鍵暗号方式に関する基礎知識については、次の URL で MIDP Application Security 1: Design Concerns and Cryptography を参照してください。
`http://developers.sun.com/techtopics/mobility/midp/articles/security1/`

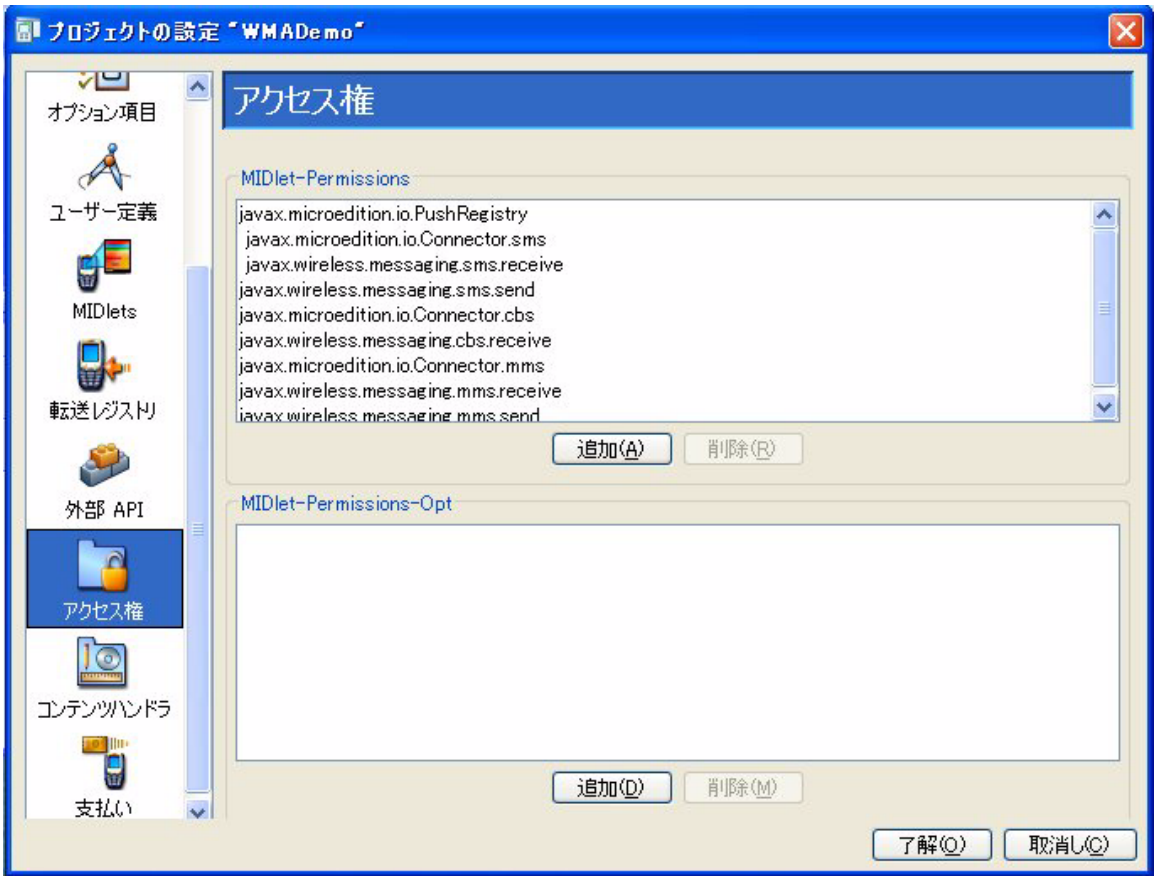
この章では、保護ドメイン、アクセス権、および MIDlet への署名に関する Sun Java™ Wireless Toolkit for CLDC のサポートについて説明します。

6.1 アクセス権

ネットワークへの接続など、セキュリティーにかかわる処理を MIDlet で実行するには、MIDlet にアクセス権が付与されている必要があります。アクセス権には固有の名前があります。MIDlet スイートに特定のアクセス権が必要な場合は、MIDlet スイート記述子の属性を使って指定できます。

これらのアクセス権属性をプロジェクトに追加するには、「属性設定」ボタンをクリックします。「アクセス権」アイコンを選択します。「MIDlet-Permissions」ボックスに、MIDlet に必須のアクセス権が表示されます。「MIDlet-Permissions-Opt」ボックスには、オプションのアクセス権が表示されます。

図 6-1　MIDlet スイートのアクセス権



どちらかのボックスにアクセス権を追加するには、「追加」をクリックし、アクセス権を選択します。アクセス権を削除するには、それを強調表示してから「削除」をクリックします。

6.2　セキュリティーポリシーの選択

Sun Java™ Wireless Toolkit for CLDC では、JSR 185 (Java Technology for the Wireless Industry) と JSR 248 (Mobile Service Architecture (MSA)) の両方で定義されたセキュリティーポリシーがサポートされます。保護ドメインの詳細は、6-3 ページの 6.2.1 節「MSA 保護ドメイン」および 6-4 ページの 6.2.2 節「Java Technology for the Wireless Industry 保護ドメイン」で説明します。

エミュレータで使用するセキュリティーポリシーを選択するには、「編集」->「環境設定」を選択し、「カテゴリ」リストで「セキュリティー」を選択します。「セキュリティーポリシー」コンボボックスで、「MSA」または「JTWI」のいずれかを選択します。利用可能なセキュリティーポリシーのいずれかを選択します。

「OTA 経由で実行」を使用する場合、パッケージ化された MIDlet スイートは、エミュレータに直接インストールされますが、その際いずれかの保護ドメインに配置されます。エミュレータは、公開鍵暗号方式を使用して、インストールされた MIDlet スイートの保護ドメインを決定します。

MIDlet スイートに署名がない場合は、デフォルトの保護ドメインに置かれます。MSA と JTWI では、デフォルト設定が異なります。6.2.1 項および 6.2.2 項を参照してください。署名付き MIDlet は、署名キーの証明書チェーンのルート証明書に関連付けられている保護ドメインに配置されます。

たとえば、Respectable Software という架空の会社が、暗号で署名された MIDlet スイートを配布するとします。Respectable Software は、Super-Trustee という架空の認証局から署名キーペアを購入します。Respectable Software は、署名キーを使って MIDlet スイートに署名し、MIDlet スイートとともに証明書を配布します。MIDlet スイートがエミュレータまたはデバイスにインストールされるときに、Respectable Software の証明書が検証されます。この検証には、Super-Trustee のルート証明書のコピーが使用されます。次に、Respectable Software の証明書を使って、MIDlet スイートの署名が検証されます。すべての検査が正常に終了した場合、デバイスまたはエミュレータで MIDlet スイートがインストールされ、Super-Trustee のルート証明書に関連付けられている保護ドメインに配置されます。ほとんどの場合は、`identified_third_party` です。

ツールキットには、MIDlet スイートへの署名、キーの管理、およびルート証明書の管理を行うためのツールが用意されています。

6.2.1 MSA 保護ドメイン

ツールキットでは、MSA に対して次の 5 つの保護ドメインがサポートされます。

- `unidentified_third_party` - アプリケーションの発信元や信頼性が確認できない場合、そのアプリケーションに高いレベルのセキュリティーが適用されます。アプリケーションがセキュリティーにかかわる処理を行おうとするたびに、ユーザーの確認が求められます。
- `identified_third_party` - 暗号化証明書によって発信元が確認された MIDlet で使用されます。アクセス権は自動で付与されませんが、ユーザーが確認を求められる頻度は `unidentified_third_party` ドメインほど高くありません。
- `manufacturer` - 資格が製造業者のルート証明書に基づいている MIDlet スイートで使用されます。
- `minimum` - このドメインの MIDlet に対して、すべてのアクセス権が拒否されます。

- maximum - このドメインの MIDlet に対して、すべてのアクセス権が付与されます。

エミュレータでアプリケーションを実行するために「実行」ボタンをクリックすると、デフォルト設定では unidentified_third_party 保護ドメインでコードが実行されます。

6.2.2 Java Technology for the Wireless Industry 保護ドメイン

Sun Java Wireless Toolkit には、次の 4 つの保護ドメインがあります。

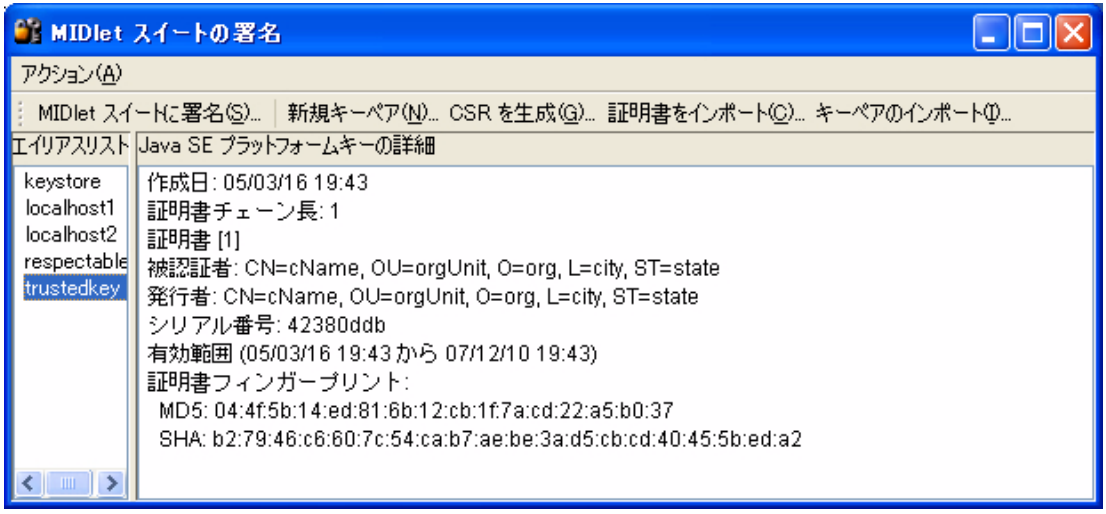
- untrusted - アプリケーションの発信元や信頼性が確認できない場合、そのアプリケーションに高いレベルのセキュリティーが適用されます。アプリケーションがセキュリティーにかかわる処理を行おうとするたびに、ユーザーの確認が求められます。
- trusted - このドメインの MIDlet に対して、すべてのアクセス権が付与されます。
- minimum - このドメインの MIDlet に対して、すべてのアクセス権が拒否されます。
- maximum - このドメインの MIDlet に対して、すべてのアクセス権が付与されます (trusted と同等)。

エミュレータでアプリケーションを実行するために「実行」ボタンをクリックすると、デフォルト設定では untrusted 保護ドメインでコードが実行されます。

6.3 MIDlet スイートへの署名

MIDlet スイートに署名するには、最初にパッケージ化を行う必要があります (「プロジェクト」->「パッケージ」を選択)。続いて、「プロジェクト」->「署名」を選択します。署名ウィンドウが表示されます (図 6-2)。

図 6-2　　MIDlet スイートの署名ウィンドウ



使用するキーを「エイリアスリスト」で選択し、「MIDlet スイートに署名」ボタン (緑色の三角形) をクリックします。

6.4　キーの管理

MIDlet の署名ウィンドウは、キーの管理にも使用できます。テストの目的で、キーペアを作成して MIDlet に署名できます。デバイスに配備するには、デバイスが認識する認証局から署名キーペアを取得する必要があります。既存の Java SE プラットフォームキーストアからキーをインポートすることもできます。

6.4.1　新しいキーペアの作成

まったく新しいキーペアを作成するには、「新規キーペア」をクリックします。キーの別名とキーペアに関連付ける情報の入力を求めるメッセージが表示されます。

図 6-3　　新しいキーペアの作成

「作成」をクリックすると、保護ドメインの選択を求めるメッセージが表示されます。作成したキーペアと保護ドメインとの関連は不透明に思えますが、次のような意味があります。

- 作成されたキーペアを使って、自己署名付きのルート証明書がツールキットで作成されます。
- エミュレータのルート証明書リストにルート証明書が追加されます。
- ツールキットでは、ルート証明書と保護ドメインを関連付ける必要があります。

この新しいキーで署名された MIDlet スイートをインストールすると、次のような処理が行われます。

- MIDlet スイート記述子の証明書チェーンが検査されます。この例では、証明書チェーンは単一の自己署名付きルート証明書です。
- 内部リストで証明書チェーンのルートが検索されます。ルート証明書はキーペアの作成時に追加されているので、この検索は成功します。
- これが有効な証明書と見なされ、この証明書を使って MIDlet スイートの署名が検証されます。
- ユーザーが選択した保護ドメインに MIDlet スイートがインストールされます。

6.4.2 実際のキーの取得

Sun Java™ Wireless Toolkit for CLDC 環境でキーペアを作成して MIDlet に署名する機能は、テストのためだけに使用する機能です。実際のデバイスでアプリケーションを実行するときは、デバイスが認識する認証局から署名キーペアを取得する必要があります。

実際のキーを使用して MIDlet スイートに署名する手順は、次のとおりです。

1. **新しいキーペアを作成します。**

   Sun Java™ Wireless Toolkit for CLDC では、前述したように、MIDlet の署名ウィンドウで「新規キーペア」を押してこの操作を実行できます。

2. **証明書署名要求 (CSR) を生成します。**

   a. **署名ウィンドウで「CSR を生成」を押します。**

   b. **CSR ファイルの場所を変更する場合は、新しいパスを入力するか、「参照」を押して新しいファイルの場所を選択します。**

   c. **「作成」を押して、CSR ファイルを書き込みます。**

      CSR が書き込まれると、成功したことを示すメッセージが表示されます。

3. **CSR を認証局 (CA) に送信します。**

   CA では、識別情報を確認するための詳細情報が必要です。また、生成される証明書に応じて認証局に料金を支払う必要もあります。

   CA は、識別情報を確認して支払いを受け取ると、公開鍵を証明する証明書を送信します。

4. **MIDlet の署名ウィンドウで「証明書のインポート...」を押して、証明書を Sun Java™ Wireless Toolkit for CLDC にインポートします。**

   これで、独自の非公開鍵を使用して MIDlet スイートに署名することができます。Sun Java™ Wireless Toolkit for CLDC により、署名と証明書が MIDlet スイートに配置されます。

6.4.3 既存のキーペアのインポート

MIDlet 署名に使用するキーが、Java SE プラットフォームキーストアに含まれている場合があります。この場合、MIDlet スイートに署名するには、署名キーを Sun Java™ Wireless Toolkit for CLDC にインポートする必要があります。そのためには、MIDlet の署名ウィンドウで「キーペアのインポート」をクリックします。Java SE プラットフォームキーストアが含まれているファイルを選択します。インポートするキーペアの別名を選択するように要求されます。続いて、キーストアにインポートしたキーペアを識別するための別名を入力します。最後に、キーペアのルート証明書に対して保護ドメインを選択する必要があります。

6.4.4 キーペアの削除

MIDlet の署名ウィンドウからキーペアを削除するには、そのキーペアの別名を選択してから、「アクション」->「選択を削除」を選択します。

6.5 証明書の管理

この節では、Sun Java™ Wireless Toolkit for CLDC を使用してエミュレータのルート証明書リストを管理する方法について説明します。

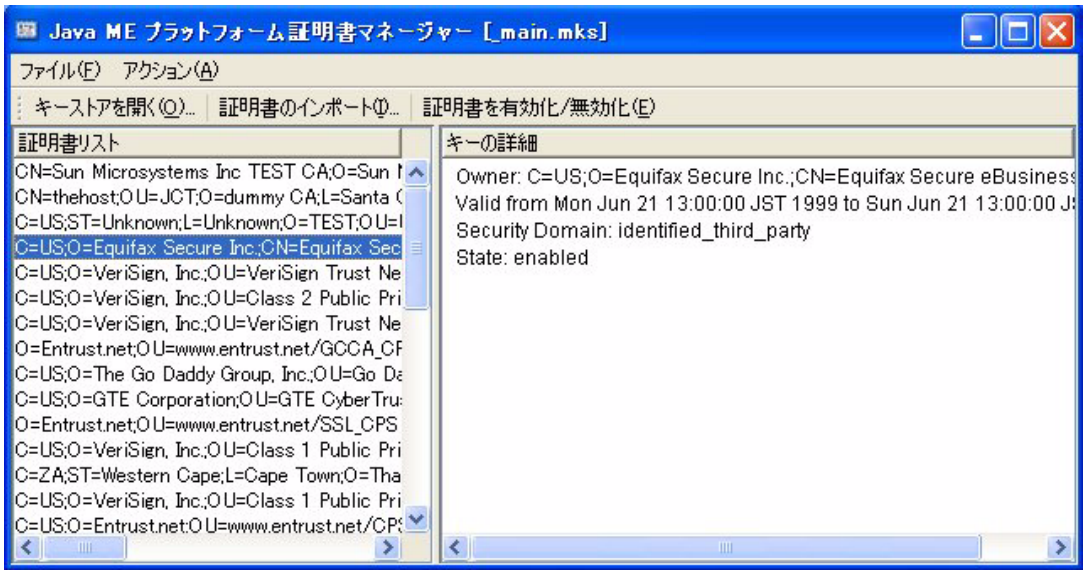
実際のデバイスにも同様のルート証明書リストがありますが、通常は、ユーザーがこれらを変更することはできません。実際のデバイスにアプリケーションを配備するときは、認証局によって発行された署名キーを使用し、その認証局のルート証明書がそのデバイス上に存在している必要があります。それ以外の場合、デバイスはアプリケーションを検証できません。

アプリケーションの開発中に、ツールキットの証明書管理ユーティリティーを使用すると、エミュレータのルート証明書リストをテストの目的で簡単に操作できます。

「ファイル」->「ユーティリティー」を選択します。「証明書の管理」を選択して「起動」を押すと、証明書管理ウィンドウが表示されます。小さいキーストア `_main.mks` が表示されます。このキーストアは、`appdb` ディレクトリに存在します。

`appdb` ディレクトリには、`keystore.ks` と `serverkeystore.ks` も含まれます。Java ME プラットフォーム証明書マネージャーでは `*.ks` ファイルを開けませんが、6-9 ページの 6.5.2 節「証明書のインポート」で説明するように、これらのキーストアから証明書をインポートできます。

図 6-4　　証明書マネージャー



ウィンドウの左部分の「証明書リスト」には、各証明書が 1 行で表示されます。証明書をクリックすると、ウィンドウの右側部分に、証明書の詳細と関連付けられた保護ドメインが表示されます。

6.5.1　証明書の有効化と無効化

証明書は有効または無効にすることができます。この方法は、証明書をキーストアから削除することなく、一時的に使用不可にする場合に便利です。証明書を有効または無効にするには、リストで証明書を選択して「証明書を有効化/無効化」を押します。操作を確認するメッセージが表示されます。「はい」を選択して、処理を続行します。

6.5.2　証明書のインポート

証明書ファイルまたは Java SE プラットフォームのキーストアファイルから、証明書をインポートできます。

ファイルから証明書をインポートするには、証明書マネージャーウィンドウの「証明書のインポート」をクリックします。証明書ファイルの場所を指定してから、その証明書に関連付けられている保護ドメインを選択します。

Java SE プラットフォームキーストアから証明書をインポートするには、証明書マネージャーウィンドウのメニューから「アクション」->「Java SE プラットフォーム証明書をインポート」を選択します。まず、証明書の保護ドメインを選択します。次に、キーストアファイルを選択し、キーストアのパスワードを入力します。最後に、インポートする証明書の別名を選択します。

6.5.3 証明書の削除

リストから証明書を削除するには、証明書を選択してから、「アクション」->「選択を削除」を選択します。

6.6 USB トークンのサポート

USB トークンにより、パスワードで保護された携帯用ストレージで公開鍵と非公開鍵および証明書を利用できます。Java SE PKCS#11 ネイティブインタフェースは、PKCS#11 準拠のネイティブドライバを備えた USB トークンへのアクセスをサポートします。ドライバがインストールされるときに、PKCS#11 ライブラリもインストールされます。Windows では、ライブラリは win32.DLL です。

この節では、Windows プラットフォームで USB トークンをインストールして使用する手順の例を示します。

注 – USB トークンを Linux ドライバで十分にテストしていないため、Linux はサポート対象外です。PKCS#11 準拠のネイティブドライバを利用できる場合は、Linux でも USB トークンが動作する可能性があります。

この節の残りの部分では、USB トークンのサポートに必要なインストールと設定の処理を、手順に従って説明します。

6.6.1 USB トークンドライバのインストール

すべてのアプリケーションを終了します。

1. `http://downloads.geotrust.com/TCSPIKEY0407203016.exe` **にアクセスします。**
2. **「ファイルのダウンロード」ダイアログボックスが表示されたら、「保存」をクリックします。**

   実行可能ファイルを保存するディレクトリを記録しておきます。

3. **実行可能ファイルを選択してダブルクリックし、Crypto Token のインストールを開始します。**

   インストールの指示に従ってトークンを挿入すれば、インストールを完了できます。

4. **Windows の「ハードウェアの追加ウィザード」が起動します。**

   指示に従い、すべてデフォルトの設定を使用します。

5. **ウィザードが完了したら、「はい」を選択してコンピュータを再起動します。**

USB トークンのパスフレーズの再設定

この手順は、新しい USB トークンのみで有効です。すべての USB トークンには、デフォルトのパスフレーズとして PASSWORD (すべて大文字) が設定されています。このパスフレーズを再設定してください。

1. **Windows のタスクバーで「スタート」をクリックします。「すべてのプログラム」->「GeoTrust Token」->「iKey 2000 Series Software」->「PassPhrase Utility」を選択します。**

2. **「Update Passphrase」をクリックします。**

   新しいパスフレーズを入力する前に、「古いパスフレーズ」を入力するように指示されます。

3. **パスフレーズを再設定します。**

   GeoTrust および Cingular は、パスフレーズに 8 文字以上の組み合わせを使用することを推奨しています。

USB トークンの管理

GeoTrust ドライバのインストールディレクトリに移動します。`CIPUtils.exe` を実行し、USB トークンの内容を管理します。

6.6.2 USB トークンの使用

「MIDlet スイートに署名」ダイアログで、USB トークンにアクセスできます。USB トークンを接続してドライバをインストールし、「ファイル」->「Load keystore」->「from USB Token」を選択します (または Ctrl キー + T キーを押す)。USB トークンがパスワードで保護されている場合、パスワードの入力を求めるメッセージが表示されます。トークンが正しくロードされると、すべてのエイリアスとキーの詳細が表示されます。6-4 ページの 6.3 節「MIDlet スイートへの署名」で説明するように、キーを選択して署名に使用できます。

USB トークンからキーストアをロードしようとしたときに、エラーメッセージが表示される場合があります。

- USB トークンにアクセスできない場合は、「USB token or driver might be unplugged or invalid.」というエラーが表示されることがあります。トークンが接続されていることを確認してください。コンピュータの別の USB ポートも試してみてください。
- ネイティブライブラリが見つからない場合は、ドライバと一緒にインストールされた DLL ライブラリのパスを入力するよう求められます。通常、このファイルはドライバをインストールしたディレクトリにあります。

第7章

Wireless Messaging API の使用法

Sun Java™ Wireless Toolkit for CLDC では、高度なシミュレーション環境による Wireless Messaging API (WMA) がサポートされています。WMA 1.1 (JSR 120) では、MIDlet で Short Message Service (SMS) メッセージや Cell Broadcast Service (CBS) メッセージを送受信できます。WMA 2.0 (JSR 205) は、MMS メッセージにも対応しています。

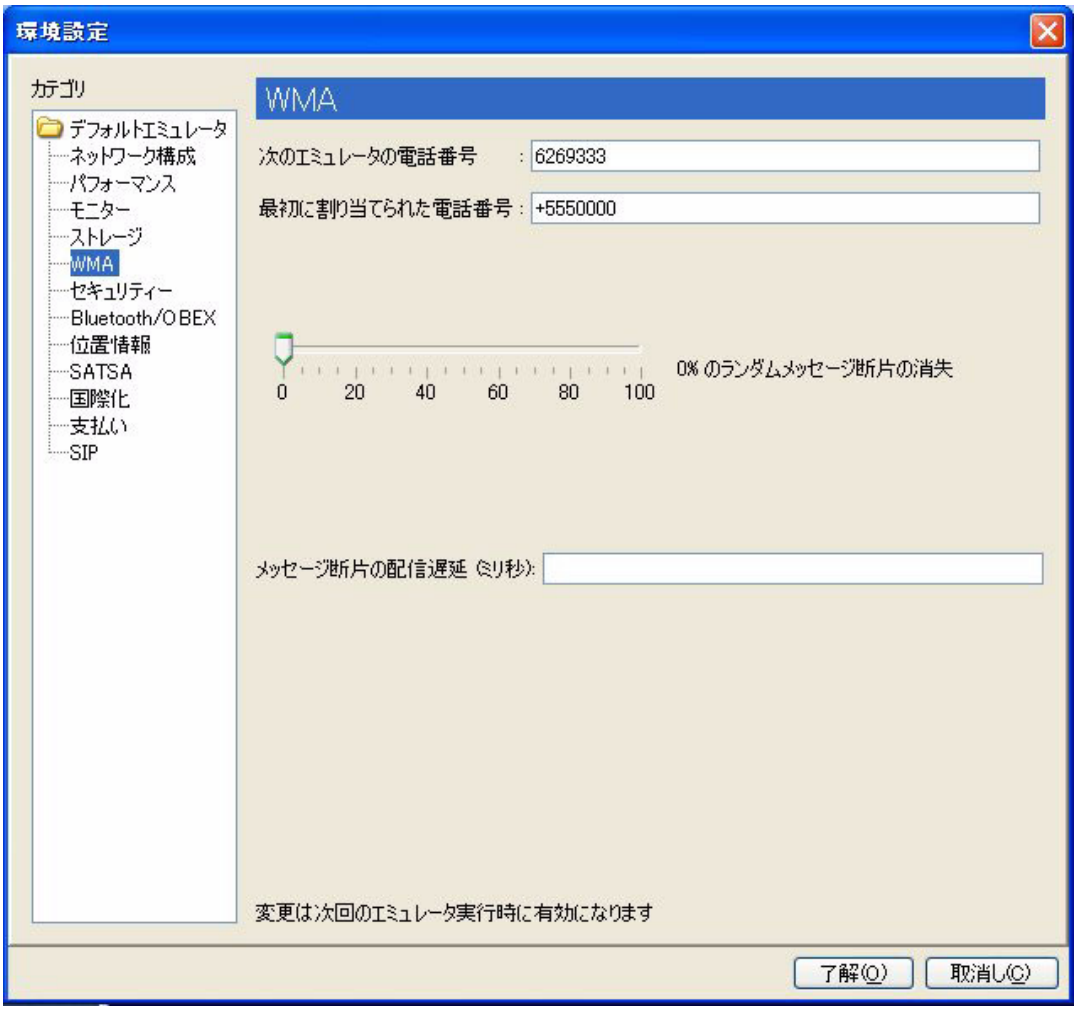
この章では、WMA アプリケーションを開発するためのツールについて説明します。まず、エミュレータの WMA サポートを設定する方法を示します。次に、WMA コンソールについて説明します。これは、WMA アプリケーションのテストに役立つユーティリティーです。最後に、ネットワークモニターの WMA サポートについて概略を説明します。

7.1 エミュレータの電話番号の設定

実行中のエミュレータインスタンスごとにシミュレーション用の電話番号があり、エミュレータウィンドウのタイトルバーに表示されます。電話番号は WMA メッセージのアドレスとして使用されるので重要です。デフォルト設定では、最初のエミュレータインスタンスに電話番号 +555000 が割り当てられます。以降のエミュレータインスタンスには連番が割り当てられ、+5550001、+5550002、+5550003 のようになります。

割り当てられた電話番号を変更するには、「編集」->「環境設定」を選択し、左側の区画で「WMA」を選択します。

図 7-1　WMA の環境設定



「次のエミュレータの電話番号」フィールドは、名前のとおりです。このフィールドに番号を入力すると、次のエミュレータインスタンスにこの番号が割り当てられます。

「次のエミュレータの電話番号」フィールドが空白の場合や、その番号がすでに使用されている場合、次のエミュレータインスタンスには「最初に割り当てられた電話番号」が割り当てられます。以降のインスタンスには連番が割り当てられます。

たとえば、「次のエミュレータの電話番号」に +6269333 を入力し、「最初に割り当てられた電話番号」に +5550000 を入力した場合を考えてみましょう。4 つのエミュレータインスタンスを起動すると、それぞれの番号は +6269333、+5550000、+5550001、+5550002 となります。

7.2 信頼性の低いネットワークのシミュレート

長いメッセージの送信では、メッセージが分割されて断片が個別に送信され、受信側でこれらの断片が再構築されます。ワイヤレスネットワークで発生する障害の一部を Sun Java™ Wireless Toolkit for CLDC でシミュレートできます。先と同じように、「編集」->「環境設定」を選択し、「WMA」を選択します。

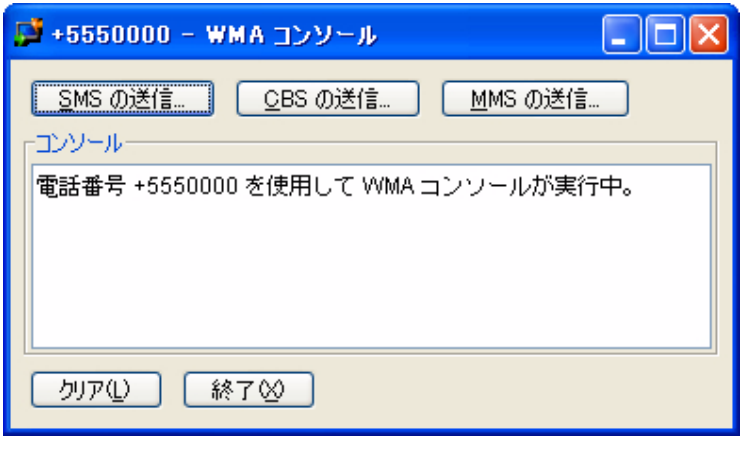
メッセージ断片の一部が失われる場合をシミュレートするには、「n% のランダムメッセージ断片の消失」スライダを使用して、消失割合を調整します。メッセージ断片の送信から受信までの遅延時間をシミュレートするには、「メッセージ断片の配信遅延」フィールドに遅延時間をミリ秒単位で入力します。

7.3 WMA コンソールによるメッセージの送信

WMA コンソールは、メッセージの送信や受信を簡単に実行できるユーティリティーです。たとえば、WMA コンソールを使用して、エミュレータ上で実行中の MIDlet に SMS メッセージを送信できます。

WMA コンソールを起動するには、「ファイル」->「ユーティリティー」を選択します。「WMA コンソール」をクリックして、「起動」を押します。

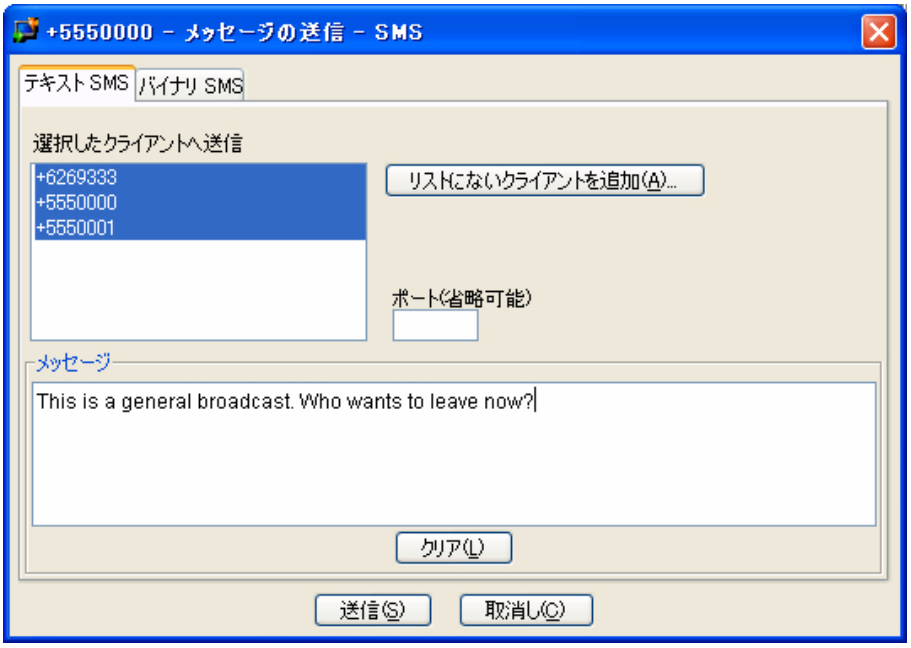
図 7-2　WMA コンソール



7.3.1 SMS テキストメッセージの送信

SMS テキストメッセージを送信するには、「SMS の送信」をクリックします。送信ウィンドウが表示されます。

図 7-3　テキストメッセージの送信

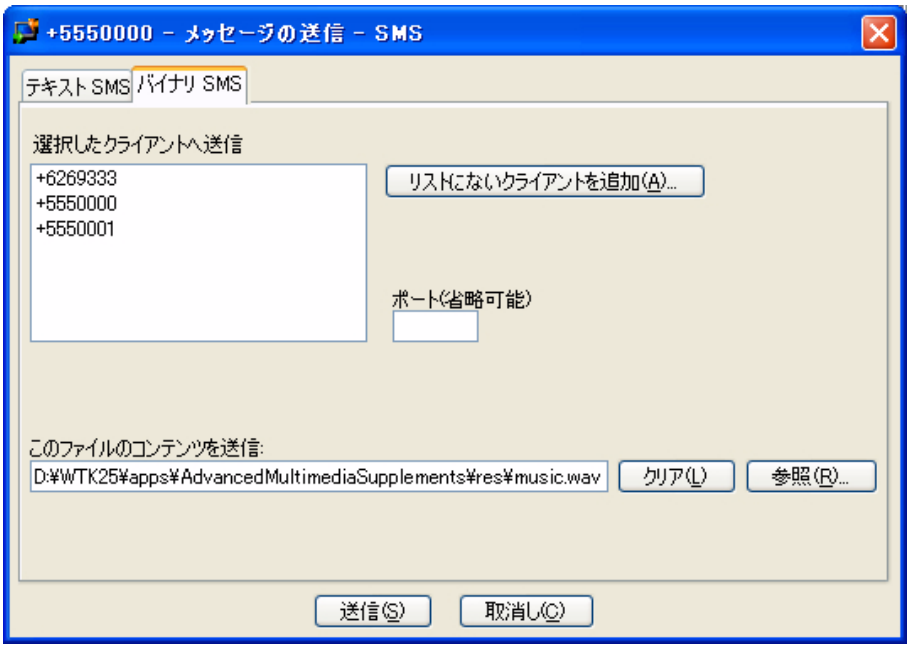


ウィンドウには、実行中のすべてのエミュレータインスタンスの電話番号が自動的に一覧表示されます。送信先を選択します。複数の送信先を選択するには、Ctrl キーを押しながらクリックします。必要に応じてポート番号を入力します。メッセージを入力し、「送信」をクリックします。

7.3.2 SMS バイナリメッセージの送信

WMA コンソールを使用して、ファイルの内容をバイナリメッセージとして送信できます。「SMS の送信」をクリックして、送信ウィンドウを開きます。「バイナリ SMS」タブをクリックします。

図 7-4　　バイナリメッセージの送信

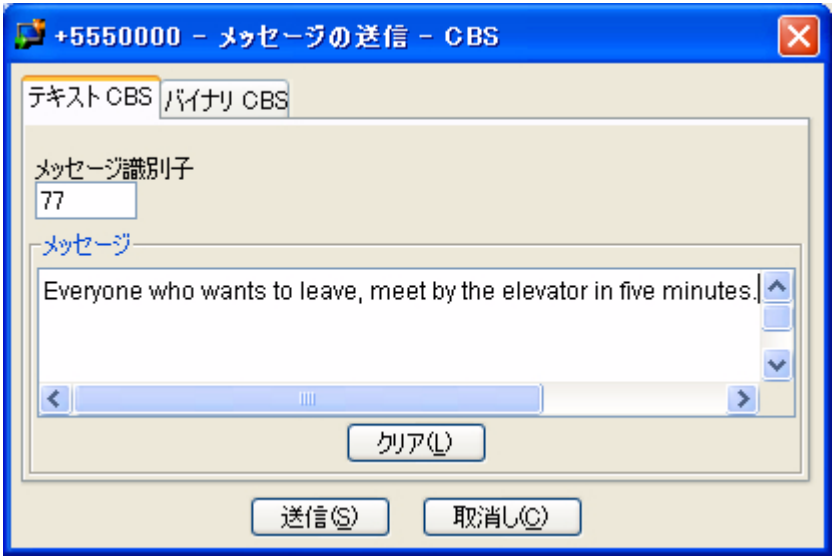


SMS テキストメッセージを送信する場合と同様に、受信者を選択します。ファイルのパスを直接入力するか、「参照」をクリックしてファイル選択ダイアログを開きます。

7.3.3　CBS テキストメッセージまたは CBS バイナリメッセージの送信

CBS メッセージの送信手順は、SMS メッセージの送信手順と似ていますが、受信者を選択する必要はありません。CBS テキストメッセージまたは CBS バイナリメッセージを送信するには、WMA コンソールの「CBS の送信」をクリックします。「送信」ウィンドウが表示されます。

図 7-5　CBS メッセージの送信

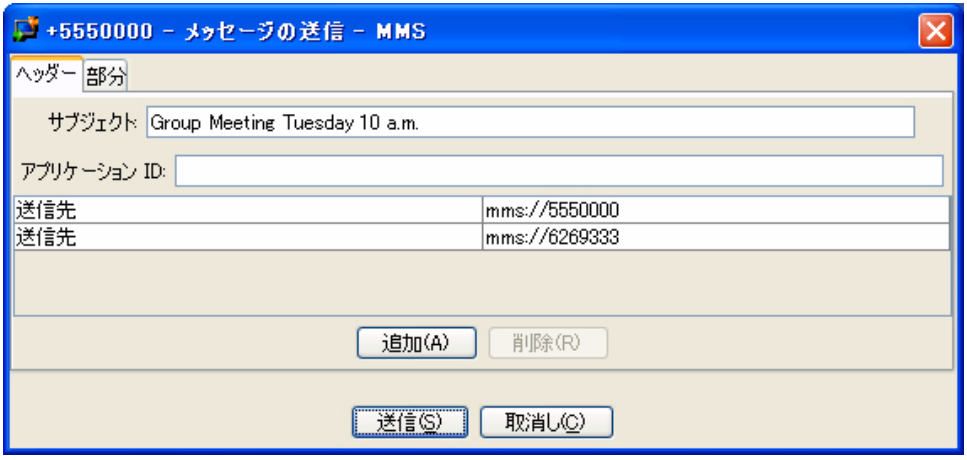


7.3.4　MMS メッセージの送信

MMS メッセージは、主に画像やサウンドなど、1 つまたは複数のファイルで構成されます。MMS メッセージは、複数の受信者に送信できます。WMA コンソールから MMS メッセージを送信するには、「MMS の送信」ボタンをクリックします。

MMS メッセージ作成ウィンドウには 2 つのタブがあり、一方では受信者、もう一方では内容を指定します。まず、件名と受信者を入力します。さらに受信者を追加するには、「追加」ボタンをクリックします。たとえば、+5550001 という番号が割り当てられた実行中のエミュレータにメッセージを送信する場合は、「送信先」行に `mms://+5550001` と入力します。受信者を削除するには、その行を選択してから、「削除」をクリックします。

図 7-6　　MMS メッセージの受信者の追加



メッセージにメディアファイルを追加するには、「部分」タブをクリックします。メッセージに部分を追加するには、「追加」をクリックします。部分を削除するには、部分を選択し、「削除」をクリックします。

図 7-7　　MMS メッセージへの部分の追加

7.4 WMA コンソールでのメッセージの受信

WMA コンソールでメッセージを受信することもできます。WMA コンソールウィンドウのタイトルバーには、その電話番号が表示されます。エミュレータ上で実行中のアプリケーションから、WMA コンソールにメッセージを送信できます。

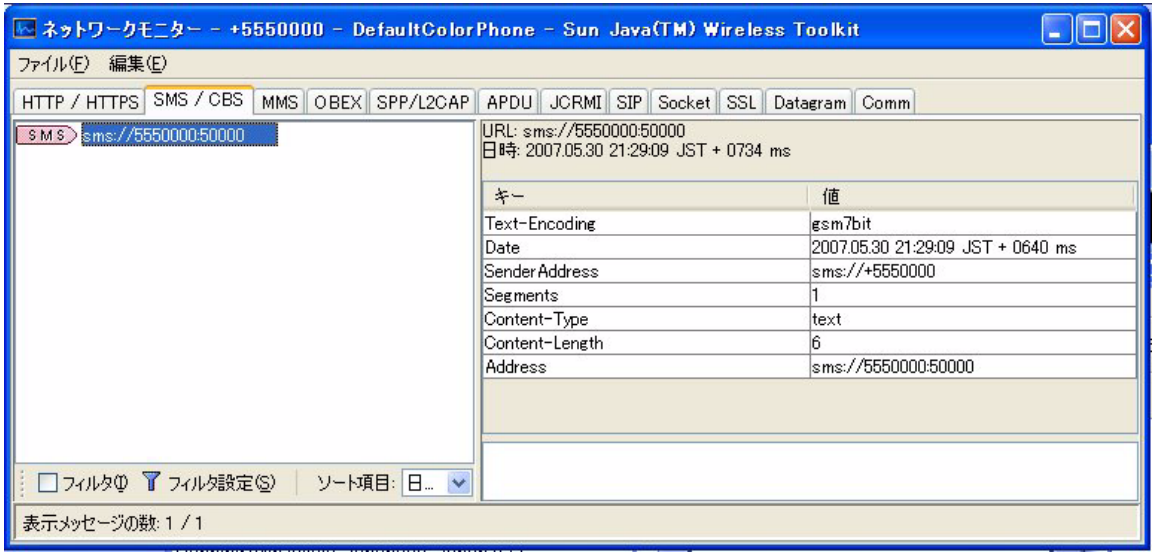
WMA コンソールで受信されたメッセージは、コンソールのテキスト領域に表示されます。

7.5 WMA でのネットワークモニターの使用

ネットワークモニターについては、第 5 章で詳しく説明しています。ネットワークモニターを使用して、エミュレータで送受信される WMA メッセージを追跡できます。

WMA メッセージを表示するには、「SMS/CBS」タブまたは「MMS」タブをクリックします。メッセージとその断片に関する情報が、ネットワークモニターの左側の区画に表示されます。メッセージまたはメッセージの断片をクリックすると、その詳細が右側の区画に表示されます。

図 7-8 ネットワークモニターを使用した WMA メッセージの表示



第8章

Mobile Media API の使用法

JSR 135 「Mobile Media API」 (MMAPI) は、オーディオやビデオなどの時間ベースのメディアの描画や取り込みを行う標準 API です。この API は、さまざまなデバイスでサポートされているメディア形式、プロトコル、および機能に対して柔軟性を持つように設計されています。MMAPI を使用したプログラミングについては、次の記事を参照してください。

- Mobile Media API Overview

http://developers.sun.com/techtopics/mobility/apis/articles/mmapi_overview/

- The J2ME Mobile Media API

http://developers.sun.com/techtopics/mobility/midp/articles/mmapioverview/

8.1 サポートされている形式とプロトコル

エミュレータの MMAPI 実装では、次のメディア形式がサポートされます。

表 8-1 サポートされている MMAPI メディア形式

MIME タイプ	説明
audio/amr	Adaptive Multi-Rate
audio/midi	MIDI ファイル
audio/sp-midi	Scalable Polyphony MIDI
audio/x-tone-seq	MIDP 2.0 トーンシーケンス
audio/x-wav	WAV PCM サンプリングオーディオ
image/gif	GIF 89a (アニメーション GIF)
video/mpeg	MPEG ビデオ
video/vnd.sun.rgb565	ビデオの取り込み

8.2 Adaptive Multi-Rate (AMR) コンテンツ

Sun Java™ Wireless Toolkit for CLDC では、Adaptive Multi-Rate (AMR) コンテンツ (`http://www.ietf.org/rfc/rfc3267.txt`) のサポートがシミュレートされます。ツールキットは AMR コンテンツをデコードできませんが、要求時には AMR コンテンツ用のプレーヤーを返します。

8.2.1 Windows

Windows では、AMR ファイルは通常の WAVE ファイルに変換され、Qsound に渡されます。Windows バージョンは 3GPP 実装とやり取りを行うため、AMR ファイルを再生するために必要な操作はありません。

8.2.2 Linux

Linux の AMR サポートは、3GPP AMR Narrow Band (AMR-NB) リファレンス実装のデコーダーと、SOX オーディオプロセッサに基づきます。AMR Wide Band はサポートされません。

8.2.2.1 AMR サポートの有効化

AMR サポートを有効にするには、次の手順に従います。

1. **3GPP から提供されている AMR-NB リファレンス実装を入手します。**

   利用可能なバージョンがいくつかあります。次のサイトでも入手できます。

   `http://www.3gpp.org/ftp/Specs/archive/26_series/26.073/26073-530.zip`

2. **パッケージに含まれている makefile を開きます。**

   「CFLAGS =」で始まる行を探します。

   a. `DMMS_IO` **オプションを追加します。**

   b. `pedantic-errors` **オプションを削除します。**

   保存してコンパイルします。問題が発生する場合は、O3 (または O2) 最適化フラグも削除してみてください。バイナリのサイズは約 2 倍になります。

3. **リファレンス実装をビルドするには、`make VAD=VAD1` と入力します。**

   コンパイルが完了したら、`decoder` という名前のバイナリファイルが生成されているはずです。これが AMR-NB デコーダーです。

4. **環境変数 `AMR_DECODER` に、デコーダーのパスを設定します。**

   たとえば、デコーダーのパスが `~/amr` の場合は、次のように指定します。

   ```
   export AMR_DECODER=~/amr/decoder
   ```

5. **`decoder` ファイルの実行アクセス権を設定します。**

   デコーダーと同じディレクトリで、次のように入力します。

   ```
   chmod 555 ./decoder
   ```

6. **デコーダーを次のようにテストします。**

   A-31 ページの A.13 節「MobileMediaAPI」で説明しているように、MobileMediaAPI デモの Simple Player を実行します。メインメニューから「Simple Player」を選択し、次に「AMR Narrow Band [jar]」を選択します。必ずスピーカーをオンにしてください。

8.2.2.2 AMR 形式のサポート

AMR-NB の複製の一部で問題が発生する場合もあります。AMR-NB コーデックデータには、少なくとも次の 2 つのファイル形式があります。

- 「AMR ファイルストレージ形式」で保存された `.AMR` ファイル

  これは、`draft-ietf-avt-rtp-amr-10.txt` の Sec. 6.2 で仕様化されています。この形式は、Ericsson AMR 変換ツールに含まれています。ドラフトは、RFC 3267 の初期の形式です。これらのファイルは、Ericsson AMR ツールと Nokia Series 60 の電話機で処理されます。この形式には、`#!AMR\n` のヘッダーがあり、ビッグエンディアンでエンコードされています。

- 「AMR Interface Format 2」で保存された `.COD` ファイル

  この形式は、3GPP TS 26.101 の Appendix A で仕様化されています。これらは、3GPP TS 26.104 浮動小数点リファレンスコーデックソースパッケージによりエンコードおよびデコードされます。これらのファイルにはヘッダーがなく、リトルエンディアンでエンコードされます。

これらの形式を相互に変換するには、次の Python スクリプトを使用してください。

```
http://www.connactivity.com/~eaw/amrwork/amrconv.py
```

8.3 MediaControlSkin の使用

Sun Java™ Wireless Toolkit for CLDC には、MediaControlSkin というエミュレータスキンが付属しています。このスキンは主に、マルチメディアの再生と制御に使用されます。このスキンのボタンには、再生、停止、音量増、音量減などのコマンドを表す記号が付いています。MediaControlSkin がどのように役立つかを確認するには、MobileMediaAPI デモアプリケーションで使用してみてください。

8.4 メディアの取り込み

Sun Java™ Wireless Toolkit for CLDC のエミュレータでは、オーディオとビデオの取り込みがサポートされています。オーディオの取り込みには、エミュレータを実行しているシステムの取り込み機能が使用されます。

ビデオの取り込みには、シミュレートされたカメラ入力が使用されます。

オーディオとビデオの取り込み方法を示す MobileMediaAPI デモアプリケーションで、詳細とソースコードを確認してください。

8.5 正常に動作する MIDlet

MIDlet には、MIDP の仕様で定義されたライフサイクルがあります。MIDlet は、着信などのイベントによって一時停止することもあります。正常に動作する MIDlet は、一時停止したときに重要なデバイスリソースを解放し、MIDlet が再開されたときにこのリソースを再度割り当てるか、再起動します。MMAPI の領域では、MIDlet が一時停止したときに、コンテンツを描画している Player を停止します。

MIDlet を一時停止したときに、実行中の Player が停止されないと、Sun Java™ Wireless Toolkit for CLDC からコンソールにメッセージが出力されます。この機能は、MobileMediaAPI デモアプリケーションの Pausing Audio Test MIDlet を使用してテストできます。詳細は、付録 Aを参照してください。

実行中のエミュレータごとに、警告メッセージが 1 度だけ出力されます。

8.6 着信音

A-31 ページの A.13.1 節「Simple Tones」および A-32 ページの A.13.2 節「Simple Player」で説明するように、MMAPI を使用して着信音を再生できます。複数の着信音の形式が一般的に使用されています。着信音は、ダウンロードまたは自作できます。

8.6.1 着信音のダウンロード

着信音ファイルは、多くのインターネットサイトからダウンロードできます。次に例を示します。

- `http://www.surgeryofsound.co.uk/`
- `http://www.convertyourtone.com/`
- http://www.filmfind.tv/ringtones/

8.6.2 着信音の形式

この節では、いくつかの形式の例を示します。

- RTTTL (Ringing Tones text transfer language 形式) は、次のサイトで説明されています。

  `http://www.convertyourtone.com/rtttl.html`

- Nokia Composer

  ベートーベンの交響曲第 9 番は、Nokia Composer 形式では次のようになります。

  ```
  16g1,16g1,16g1,4#d1,16f1,16f1,16f1,4d1,16g1,16g1,16g1,16#d1,
  16#g1,16#g1,16#g1,16g1,16#d2,16#d2,16#d2,4c2,16g1,16g1,16g1,
  16d1,16#g1,16#g1,16#g1, 16g1,16f2,16f2,16f2,4d2
  ```

- Ericsson Composer

  ベートーベンのメヌエット ト長調は次のようになります。

  ```
  a b + c b + c b + c b + C p + d a B p + c g A
  p f g a g a g a g A p b f G p a e F
  ```

  ベートーベンの交響曲第 9 番の主題は次のようになります。

  ```
  f f f # C # d # d # d C p f f f # c # f #f # f f +# c + #
  c + # c # A ff f c # f # f # f f + # d + # d + # d
  ```

- Siemens Composer Format

  ガジェット警部のテーマは次のようになります。

```
C2(1/8) D2(1/16) Dis2(1/8) F2(1/16) G2(1/8)
P(1/16) Dis2(1/8) P(1/16) Fis2(1/8) P(1/16)
D2(1/8) P(1/16) F2(1/8) P(1/16) Dis2(1/8)
P(1/16) C2(1/8) D2(1/16) Dis2(1/8) F2(1/16)
G2(1/8) P(1/16) C3(1/8) P(1/16) B2(1/2) P(1/4)
C2(1/8) D2(1/16) Dis2(1/8) F2(1/16) G2(1/8) P(1/16)
Dis2(1/8) P(1/16) Fis2(1/8) P(1/16) D2(1/8) P(1/16)
F2(1/8) P(1/16) Dis2(1/8) P(1/16) C3(1/8) B2(1/16)
Ais2(1/8) A2(1/16) Gis2(1/2) G2(1/8) P(1/16) C3(1/2)
```

- Motorola Composer

  ベートーベンの交響曲第 9 番は次のようになります。

```
4 F2 F2 F2 C#4 D#2 D#2 D#2 C4 R2 F2 F2 F2 C#2 F#2 F#2
F#2 F2 C#+2 C#+2 C#+2 A#4 F2 F2 F2 C2 F#2 F#2 F#2 F2
D#+2 D#+2 D#+2
```

- Panasonic Composer

  ベートーベンの交響曲第 9 番は次のようになります。

```
444** 444** 444** 1111* 4444** 4444** 4444** 111*
0** 444** 444** 444** 1111** 4444** 4444** 4444**
444** 11** 11** 11** 6666* 444** 444** 444** 111**
4444** 4444** 4444** 444** 22** 22** 22**
```

- Sony Composer

  ベートーベンの交響曲第 9 番は次のようになります。

```
444****444****444****111#*****444#****444#****444#****
111*****(JD)0000444****444****444****111#****444#****
444#****444#****444****11#****11#****11#****666#*****
444****444****444****111****444#****444#****
444#****444****22#****22#****22#****
```

第9章

Mobile Graphics の使用

この章では、グラフィックスコンテンツの操作について概要を説明します。Sun Java™ Wireless Toolkit for CLDC には、対話型 2D および 3D グラフィックスの総合的な機能を提供する 3 つの API が用意されています。

- JSR 184 「Mobile 3D Graphics API for J2ME」では、低レベル API および高レベルなシーングラフ API による 3D グラフィックス機能が提供されます。この章では、JSR 184 の概要と一般的な操作のガイドラインを説明します。
- JSR 226 「Scalable 2D Vector Graphics API for J2ME」では、高度な対話型 2D コンテンツの描画がサポートされます。
- JSR 239 「Java Bindings for OpenGL® ES」では、オープンスタンダード OpenGL® ES グラフィックス API に対する Java 言語インタフェースが提供されます。

9.1 Mobile 3D Graphics API の使用

JSR 184 は、J2ME 用の Mobile 3D Graphics (M3G) API を定義する仕様です。この API は、CLDC や MIDP 準拠のデバイスに適したコンパクトなパッケージで 3D 機能を提供します。この API では、2 つの方法で 3D グラフィックスコンテンツを表示できます。「即時モード」の API を使用すると、アプリケーションで 3D 要素を直接作成および操作できます。このモードの上に、「シーングラフ」または「リテインモード」と呼ばれる API があります。これを使用すると、あらかじめ作成されている 3D シーン全体をロードして表示できます。アプリケーションでは、リテインモード API と即時モード API のどちらか適切な方を使用するか、両方を組み合わせて使用することができます。JSR 184 仕様では、シーングラフのファイル形式 (`.m3g`) も定義されています。

詳細については、次の JSR 184 仕様を参照してください。
`http://jcp.org/en/jsr/detail?id=184`

JSR 184 には、新世代の 3D アプリケーションを作成できる、CLDC/MIDP デバイス用の標準 API が用意されています。これに対し、即時モード API には、軽量な標準 3D グラフィックス API である OpenGL ES との互換性があります。OpenGL ES の詳細については、`http://khronos.org/` を参照してください。

9.1.1 即時モード

即時モードは、3D グラフィックスコンテンツをアルゴリズムによって生成するアプリケーションに適しています。たとえば、科学理論の視覚化や統計のグラフ化を行う場合などです。3D オブジェクトの作成と操作は、アプリケーションで直接実行されます。

即時モードの実例については、`Demo3D` アプリケーションの `Life3D` MIDlet を参照してください。

9.1.2 リテインモード

ゲームなどの多くのアプリケーションでは、リテインモードと呼ばれるシーングラフ API が使用されます。この場合、グラフィックデザイナーやグラフィックアーティストは、3D モデリングソフトウェアを使用してシーングラフを作成します。シーングラフは JSR 184 ファイル形式で保存されます。このシーングラフファイルはアプリケーションにバンドルされます。アプリケーションの実行時に、シーングラフ API を使ってファイルがロードされ表示されます。

アプリケーションでは、ロードしたシーングラフの各部分を操作して、文字をアニメーション表示したり、ほかの効果を作成したりできます。基本的には、できるだけ多くの作業をモデリングソフトウェアで行うようにします。実行時にアプリケーションでは、シーングラフの各部分をグラブしたり操作したりできます。アニメーションパスやその他の効果が含まれている場合もあります。

リテインモードの実例については、`Demo3D` アプリケーションの `retainedmode` MIDlet を参照してください。

9.1.3 品質と速度の兼ね合い

MIDP の開発における難問の 1 つは、デバイスの環境に制約があることです。デスクトップコンピュータに比べて、MIDP デバイスのプロセッサは低速で、メモリも少量です。この問題は 3D グラフィックスにも関連します。さまざまな実装に対応するため、JSR 184 仕様には 3D シーンをできるだけ効率よく表示するためのさまざまなメカニズムが用意されています。

その 1 つは「スコーピング」と呼ばれる手法です。この手法では、3D グラフィックスの実装に対して、どのような場合にオブジェクト間の相互作用を無効にするかを指示できます。たとえば、家のシーングラフを定義する場合、地階のライトが 2 階の寝室の外観に影響を与えないよう、スコーピングを使って指定できます。スコーピングを使用すると、シーンの表示に必要な計算回数が減少し、実装での処理が簡単になります。

ただし、3D シーンの描画速度を向上させるには、品質の面でいくらか妥協するのが最良の方法です。Mobile 3D Graphics API には「描画のヒント」が含まれており、品質面で妥協して描画速度を向上させる方法をアプリケーションで参照できます。

9.1.4 Mobile 3D Graphics コンテンツの作成

モバイル用 3D アプリケーションのほとんどは、リソースファイル内のシーングラフを使ってオブジェクト、シーン、および文字を記述します。通常は、プログラマではなくグラフィックデザイナーやグラフィックアーティストが、標準の 3D モデリングツールを使用してシーングラフを作成します。

一部のベンダーは、コンテンツ作成ツールや、ファイルを JSR 184 形式に変換するツールを提供しています。Superscape (`http://superscape.com/`) もこのようなベンダーの 1 つです。

アプリケーションで即時モード API を使って 3D グラフィックスコンテンツの作成や操作を行うことは、比較的難しい作業です。したがって、ほとんどのアプリケーションでは、できる限りシーングラフファイルが利用されます。設計時にできるだけ多くの情報をシーングラフファイルに含めることで、アプリケーションでの実行時の処理がかなり簡素化されます。

9.2 Scalable Vector Graphics コンテンツの描画

Scalable Vector Graphics (SVG) は、World Wide Web Consortium で定義された標準です。これは、高度な対話型 2D グラフィックスを記述するための XML 文法です。

Sun Java™ Wireless Toolkit for CLDC エミュレータでは、JSR 226 「Scalable 2D Vector Graphics API for J2ME」をサポートします。JSR 226 は、SVG コンテンツをロード、操作、描画、および再生するための Java ME の API です。SVG Tiny は、高度な対話型アニメーション 2D コンテンツを記述するための、コンパクトで強力な XML 形式です。

テキストエディタでも SVG コンテンツを作成することはできますが、多くの場合は作成ツールを使用します。ここでは、3 つのツールを紹介します。

- BeatWare Mobile Designer - `http://www.beatware.com/products/md_golive.html`
- Ikivo Animator - `http://www.ikivo.com/animator/`
- Adobe Illustrator CS2 - `http://www.adobe.com/products/illustrator/main.html`

SVG コンテンツを使用する Java ME アプリケーションでは、ユーザーのディスプレイの画面解像度とフォームファクタに適合したグラフィックス効果を作成できます。

SVG の画像は、2 つの方法でアニメートできます。1 つは、説明的アニメーションを使用する方法です。もう 1 つは、API 呼び出しを通して SVG 画像のパラメータ (色や位置など) を継続的に変更する方法です。説明的アニメーションについては、A-52 ページの A.21.3 節「Play SVG Animation」で説明します。

9.3 OpenGL® ES の概要

JSR 239 では、OpenGL® for Embedded Systems (OpenGL® ES) と EGL の 2 つの API を結び付ける Java プログラミング言語バインディングが定義されています。OpenGL® のサブセットである OpenGL® ES は、3D グラフィックスの標準 API であり、デスクトップコンピュータ上で広く使用されています。EGL は、標準的なプラットフォームのインタフェース層です。OpenGL® ES と EGL のどちらも、Khronos Group (`http://khronos.org/opengles/`) によって開発されています。

オブジェクト指向の JSR 184 では高度な機能が要求されますが、OpenGL® はハードウェアで高速化された 3D グラフィックスへのアクセスに適した低レベルなグラフィックスライブラリです。OpenGLESDemo サンプルプロジェクトのコードを参照してください。

第 10 章

PIM API と FileConnection API の使用法

Sun Java™ Wireless Toolkit for CLDC は、JSR 75 「PDA Optional Packages (PDAP) for the J2ME Platform」に対応しています。JSR 75 には、2 つの独立した API が含まれています。

- FileConnection オプションパッケージを使用すると、MIDlet でローカルデバイスのファイルシステムにアクセスできます。
- Personal Information Management (PIM) オプションパッケージには、連絡先リスト (アドレス帳)、カレンダ、予定表リストなどを操作するための API が含まれています。

この章では、FileConnection API と PIM API が Sun Java™ Wireless Toolkit for CLDC でどのように実装されるかについて説明します。

10.1 FileConnection API

実際のデバイスでは、FileConnection API は主に、デバイスのメモリーやメモリーカードに保存されているファイルにアクセスするために使用されます。

Sun Java™ Wireless Toolkit for CLDC のエミュレータでは、MIDlet で FileConnection API を使用して、デスクトップコンピュータのハードディスクに保存されているファイルにアクセスできます。

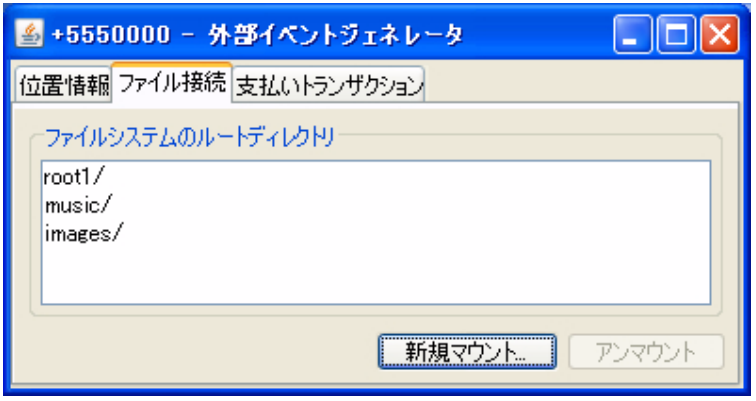
FileConnection を使ってアクセスできるのは、*workdir*¥appdb¥*skin*¥filesystem のサブディレクトリに保存されているファイルです。たとえば、DefaultColorPhone エミュレータスキンには root1 というルートディレクトリがインストールされており、ここには Readme というファイルと photos という空のディレクトリがあります。ファイルのフルパスは次のようになります。

Windows:	*workdir*¥appdb¥*skin*¥filesystem¥root1¥photos
Linux:	*workdir*/appdb/*skin*/filesystem/root1/photos

注 – 同じエミュレータスキンのインスタンスを同時に複数実行する場合、Sun Java™ Wireless Toolkit for CLDC では、インスタンスごとに一意のファイルパスが生成されます。たとえば、1 つ目のディレクトリの名前が DefaultColorPhone となり、2 つ目のインスタンスは DefaultColorPhone1 という名前になります。

filesystem の各サブディレクトリは「ルート」と呼ばれます。Sun Java™ Wireless Toolkit for CLDC には、ルートを管理するためのメカニズムが用意されています。エミュレータの実行中に、エミュレータウィンドウのメニューから「MIDlet」->「外部イベント」を選択します。ルートを追加および削除するためのユーティリティーウィンドウが表示されます。

図 10-1 ファイルシステムのルートの管理



アプリケーションでは、FileConnection API を使用して、マウント済みのルートとその内容にアクセスできます。

新しいルートディレクトリを追加するには、「新規マウント」をクリックし、ディレクトリの名前を入力します。FileConnection API からディレクトリにアクセスできないようにするには、リストでそのディレクトリを選択し、「アンマウント」をクリックします。

10.2 PIM API

Sun Java™ Wireless Toolkit for CLDC のエミュレータでは、連絡先、カレンダ、および予定表の情報は、デスクトップコンピュータのハードディスクに標準ファイルで保存されます。すべての情報は、*workdir*¥appdb¥*skin*¥pim に保存されます。このディレクトリは、実行中のすべてのエミュレータで共有されます。リストは、contacts、events、および todo というディレクトリのサブディレクトリに保存されます。たとえば、Contacts という連絡先リストは、次の場所に保存されます。

Windows:	*workdir*¥appdb¥*skin*¥pim¥contacts¥Contacts
Linux:	*workdir*/appdb/*skin*/pim/contacts/Contacts

リストディレクトリ内のアイテムは、vCard (.vcs) 形式または vCalendar (.vcf) 形式で保存されます (http://www.imc.org/pdi/ を参照)。連絡先は vCard 形式、カレンダと予定表は vCalendar 形式でそれぞれ保存されます。

第11章

Bluetooth API と OBEX API の使用法

Sun Java™ Wireless Toolkit for CLDC のエミュレータは、JSR 82 「Java APIs for Bluetooth」に対応しています。エミュレータは、転送レジストリとの統合が記述されているバージョン 1.1 の仕様に完全に準拠しています。JSR 82 には、2 つの独立した API が含まれています。

- Bluetooth API は、デバイス検出やデータ交換などを含む、Bluetooth ワイヤレスネットワークへのインタフェースを提供します。
- OBEX API では、アプリケーションで Bluetooth などの通信チャネル上で Object Exchange (OBEX) プロトコルを使用できます。

この章では、Bluetooth API と OBEX API が Sun Java™ Wireless Toolkit for CLDC でどのように実装されるかについて説明します。

11.1 Bluetooth シミュレーション環境

Sun Java™ Wireless Toolkit for CLDC のエミュレータを使用すると、実際の Bluetooth ハードウェアがなくても、Bluetooth を使用するアプリケーションの開発やテストを行うことができます。実行中のエミュレータに対して、ツールキットが Bluetooth 環境をシミュレートします。複数のエミュレータインスタンスは、Bluetooth API を使用して、相互に検出してデータを交換することができます。

実例については、付録 A の `BluetoothDemo` のマニュアルを参照してください。

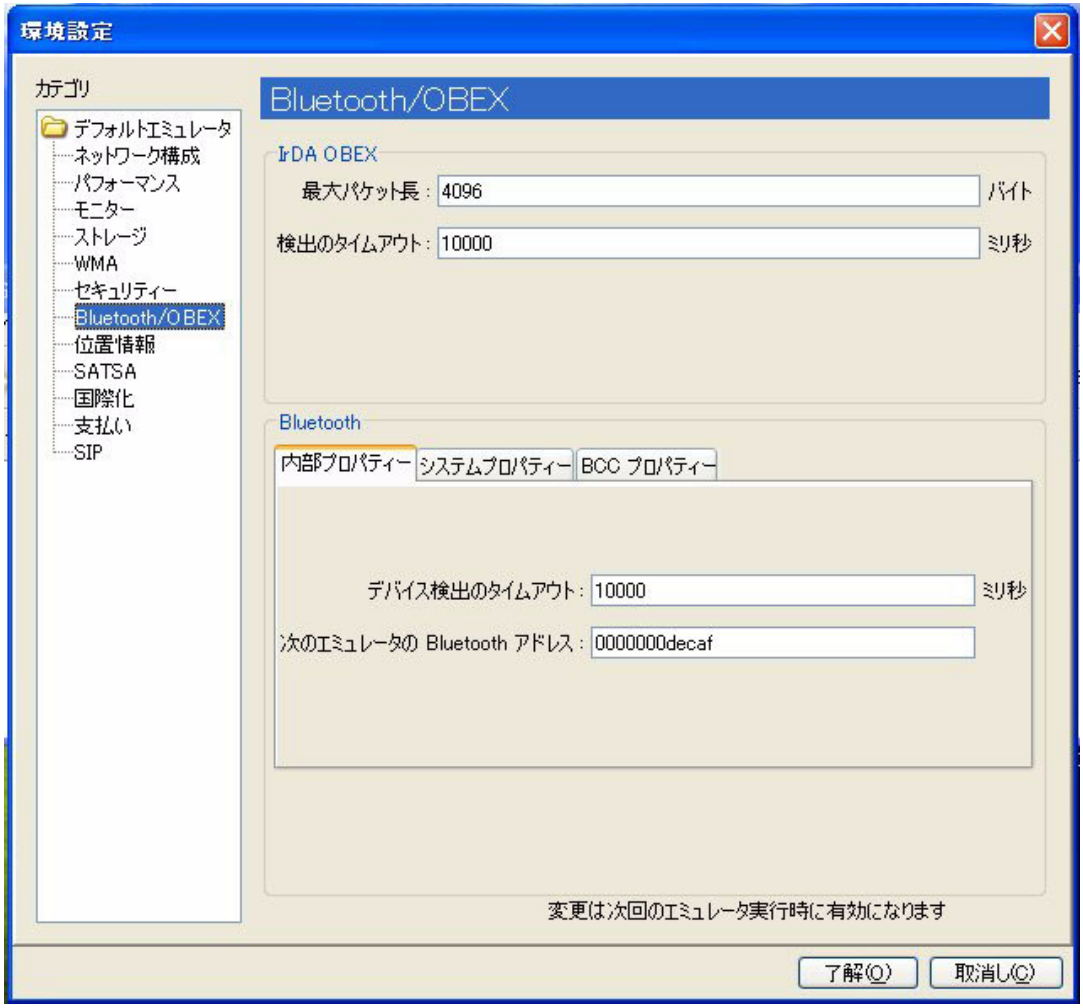
11.2 赤外線経由の OBEX

Sun Java™ Wireless Toolkit for CLDC は、シミュレートした Bluetooth 接続や赤外線接続を介した OBEX 転送を実装します。シミュレートされた赤外線接続は、IrDA 標準に準拠しています (`http://www.irda.org/` を参照)。実行中の複数のエミュレータ間で、赤外線転送をシミュレートできます。

11.3 OBEX と Bluetooth の環境設定

Sun Java™ Wireless Toolkit for CLDC では、Bluetooth と OBEX のシミュレーション環境を設定できます。「編集」->「環境設定」を選択し、「Bluetooth/OBEX」を選択して次のウィンドウを表示します。

図 11-1　Bluetooth と OBEX の環境設定



11.3.1　OBEX の環境設定

IrDA を実際に使用するデバイスは、待機することによってほかのデバイスを検出します。Sun Java™ Wireless Toolkit for CLDC のエミュレータでほかのデバイスを検出するために待機する時間は、環境設定ウィンドウの「IrDA OBEX」セクションの「検出のタイムアウト」フィールドで設定できます。値はミリ秒単位で入力します。

API レベルでは、検出タイムアウトの値により、`Connector.open("irdaobex://discover...")` の呼び出しがブロックされる時間が決まります。この時間が経過すると、この関数から戻されるか、例外がスローされます。

最大パケット長の設定は、エミュレータ間で送信される各パケット内のデータ量に影響を与えます。パケット長が小さいほど、パケット数が増加し、パケットのオーバーヘッドも増大します。

11.3.2 Bluetooth 内部プロパティー

環境設定ウィンドウの「Bluetooth」セクションの「デバイス検出のタイムアウト」では、シミュレートされた Bluetooth 環境でほかのデバイスを検出するためにエミュレータで待機する時間を、ミリ秒単位で指定します。

「次のエミュレータの Bluetooth アドレス」は、最初のエミュレータインスタンスに割り当てられる Bluetooth アドレスです。以降のエミュレータインスタンスには、自動的にアドレスが割り当てられます。

11.3.3 Bluetooth システムのプロパティー

環境設定の「Bluetooth」セクションの「システムプロパティー」タブには、`javax.bluetooth.LocalDevice` の `getProperty()` メソッドを使ってアプリケーションで取得できるプロパティーが含まれています。

Bluetooth のプロパティーについては、JSR 82 仕様で詳しく説明されています。

11.3.4 Bluetooth BCC のプロパティー

Bluetooth Control Center (BCC) は、Bluetooth の設定を制御します。デバイスによっては、Bluetooth の設定をカスタマイズするための GUI が用意されていることもあります。Sun Java™ Wireless Toolkit for CLDC では、Bluetooth 環境設定の「BCC プロパティー」タブを使って BCC を設定します。プロパティーは次のとおりです。

表 11-1 BCC のプロパティー

プロパティー	説明
Bluetooth のサポートを有効にする	このプロパティーが無効になっている場合は、`LocalDevice.getLocalDevice()` で `BluetoothStateException` がスローされるので、接続を作成できません。JSR 82 に対応しているが Bluetooth 機能が無効になっているデバイスで、アプリケーションの動作をテストする場合に役立ちます。
デバイス検出可能	このエミュレータをほかのエミュレータから検出可能にするかどうかを示します。
フレンドリ名	シミュレートされた Bluetooth 環境のエミュレータを表す、人間が判読できる名前です。この名前を空白のままにすると、エミュレータではフレンドリ名機能がサポートされません。
暗号化	接続の暗号化に対応するかどうかを指定します。`on` の場合は対応し、`off` の場合は対応しません。`force` の場合は、すべての接続に暗号化を義務付けます。詳細については、`RemoteDevice` の `encrypt()` メソッドのマニュアルを参照してください。
承認	「暗号化」に類似したプロパティーです。`RemoteDevice` の `authorize()` メソッドを参照してください。
認証	「暗号化」および「承認」に類似したプロパティーです。`RemoteDevice` の `authenticate()` メソッドを参照してください。

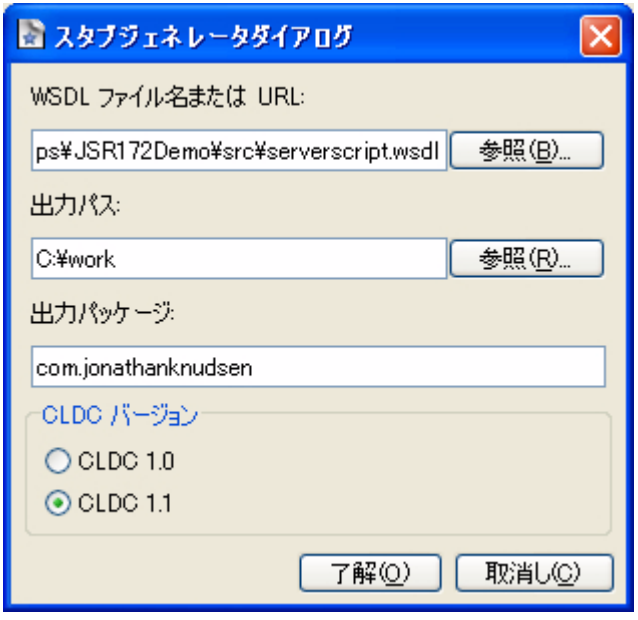
第 12 章

Web サービスの使用法

Sun Java™ Wireless Toolkit for CLDC のエミュレータは、JSR 172 「J2ME Web Services Specification」に対応しています。JSR 172 には、モバイルアプリケーションから Web サービスにアクセスするための API が用意されています。また、XML ドキュメントの構文解析を行うための API も含まれています。

Sun Java™ Wireless Toolkit for CLDC のスタブジェネレータを使用すると、Web サービスにアクセスするためのソースコードを自動的に作成できます。スタブジェネレータを起動するには、「ファイル」->「ユーティリティー」を選択します。「スタブジェネレータ」をクリックし、「起動」を押します。

図 12-1　Web サービスのスタブジェネレータ



「WSDL ファイル名または URL」フィールドに、アクセスする Web サービスの WSDL (Web Services Description Language) ファイルのパスを入力します。「出力パス」には、スタブファイルを保存する場所を指定します。「出力パッケージ」には、スタブファイルの Java プログラミング言語パッケージ名を指定します。最後に、CLDC 1.0 または CLDC 1.1 のどちらのスタブを生成するかを選択します。

「了解」をクリックして、スタブファイルを生成します。

第13章

Location API の使用法

JSR 179 「Location API」により、アプリケーションはデバイスの位置情報機能を利用できます。たとえば、一部のデバイスは GPS (Global Positioning System) ハードウェアを搭載しています。また、ワイヤレスネットワークから位置情報を受信できるデバイスもあります。Location API では、基礎となるテクニックに関係なく、位置情報に対する標準のインタフェースが提供されます。

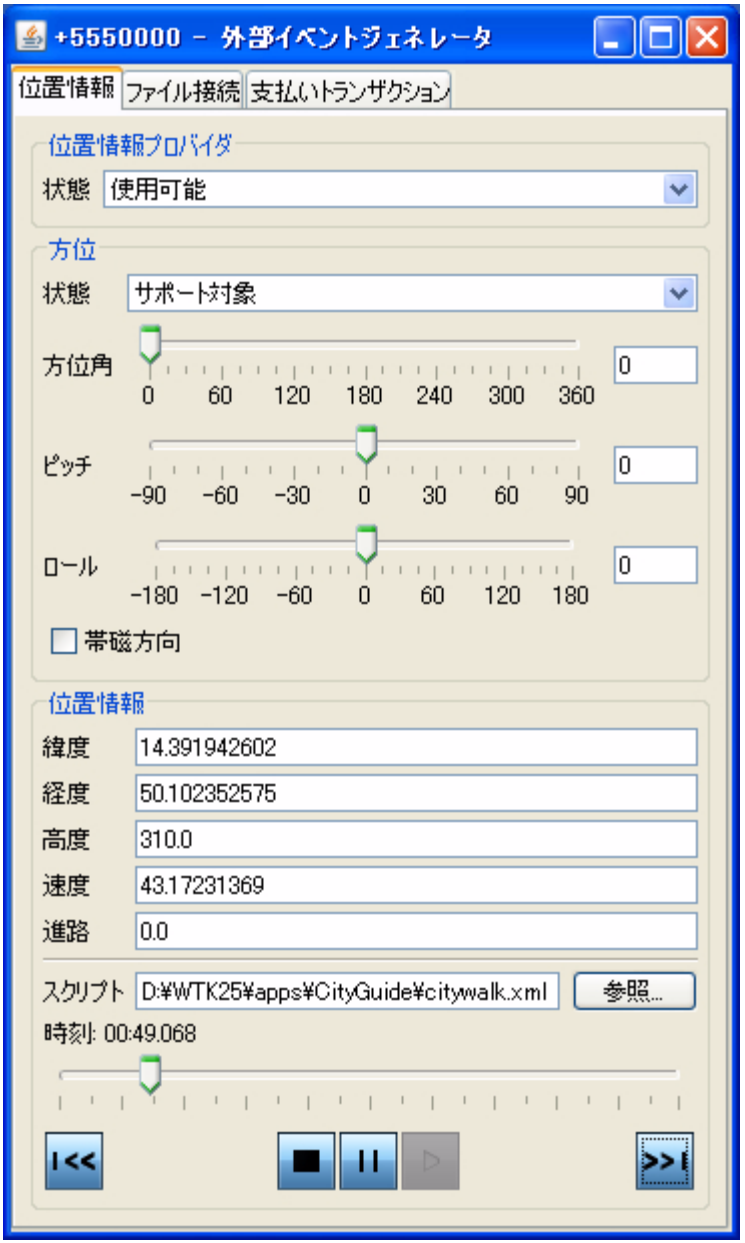
Location API では、「位置情報プロバイダ」が測位方式をカプセル化し、デバイスの位置に関する情報を提供します。アプリケーションは、精度や応答時間などの必要な条件を指定してプロバイダに要求を行います。適切な実装を利用できる場合、アプリケーションはこれを使用して、デバイスの実際の位置に関する情報を取得します。

Sun Java™ Wireless Toolkit for CLDC には、シミュレートされた位置情報プロバイダが含まれています。エミュレータの「外部イベント」ウィンドウを使用して、エミュレータの位置を指定できます。また、プロバイダのプロパティーを設定したり、ランドマークのデータベースを管理することもできます。

13.1 実行時のエミュレータの位置情報の設定

エミュレータが実行されている間のシミュレートされた位置を指定できます。これを行うには、エミュレータウィンドウのメニューから「MIDlet」->「外部イベント」を選択します。「位置情報」タブをクリックします。図 13-1 を参照してください。

図 13-1　エミュレータでの位置情報の制御



タブの「位置情報」の領域で、緯度、経度、高度、速度、および進路の値を入力できます。Location API を使用するアプリケーションは、これらの値をエミュレータの位置情報として受け取ることができます。

より詳細なテストでは、時間の経過に伴う移動を記述する位置情報スクリプトを設定できます。位置情報スクリプトは、位置のリスト (中間地点) と関連付けられた時間で構成される XML ファイルです。Sun Java™ Wireless Toolkit for CLDC は、位置情報スクリプトの中間位置の間を補間することで、エミュレータの現在の位置を決定します。ここで、開始位置 (`time="0"`) と 10 秒後の位置を指定する、簡単な位置情報スクリプトを示します。

```
<waypoints>
  <waypoint time="0"
            latitude="14" longitude="50" altitude="310" />
  <waypoint time="10000"
            latitude="14.5" longitude="50.1" altitude="215" />
</waypoints>
```

高度の値はメートル単位、時間の値はミリ秒単位です。

テキストエディタを使用して、位置情報スクリプトを作成します。作成したスクリプトは、「スクリプト」フィールドの横にある「参照」ボタンを押して指定して、外部イベントウィンドウにロードできます。その下にあるコントロールで、位置情報スクリプトの再生、一時停止、停止、およびスクリプトの先頭または末尾への移動を実行できます。時間スライダを特定の位置にドラッグすることもできます。

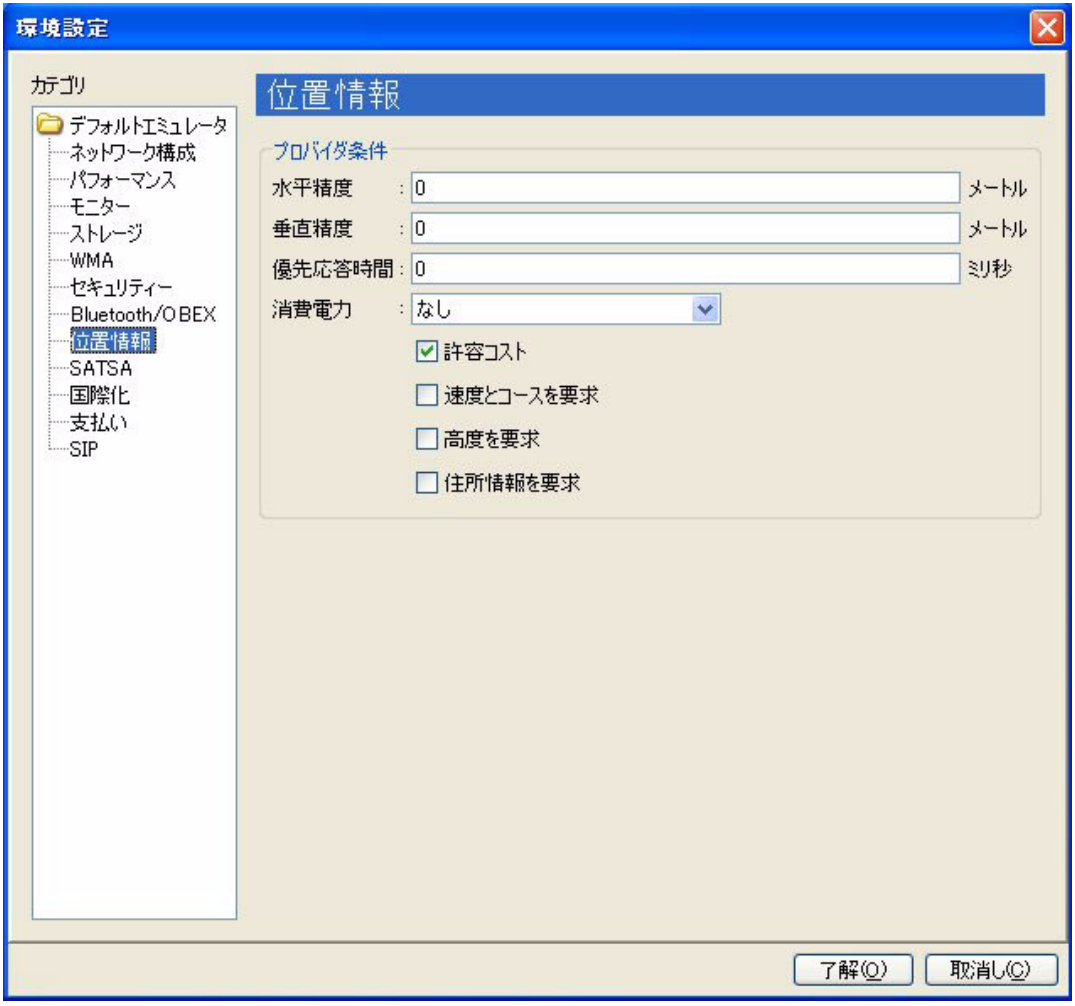
一部のデバイスでは方位の測定も可能です。この情報をアプリケーションで利用できるようにするには、「方位」の「状態」フィールドを「サポート対象」に変更し、方位角、ピッチ、およびロールの値を入力します。「磁極」チェックボックスは、方位角とピッチの値を、地球の磁界を基準にするか、真北と重力の方向を基準にするかを指定します。

予期していない状況をアプリケーションが処理する方法をテストするには、「位置情報プロバイダ」ボックスの「状態」フィールドを「一時的に使用不可」または「非稼働中」に変更します。アプリケーションがエミュレータの位置情報を取得しようとすると、例外がスローされ、アプリケーションがどのように応答するかを確認できます。

13.2 位置情報プロバイダの設定

Sun Java™ Wireless Toolkit for CLDC の位置情報プロバイダのプロパティーは、環境設定を使用して設定できます。ユーザーインタフェースで、「編集」->「環境設定」を選択し、「位置情報」をクリックします。

図 13-2　位置情報プロバイダの設定

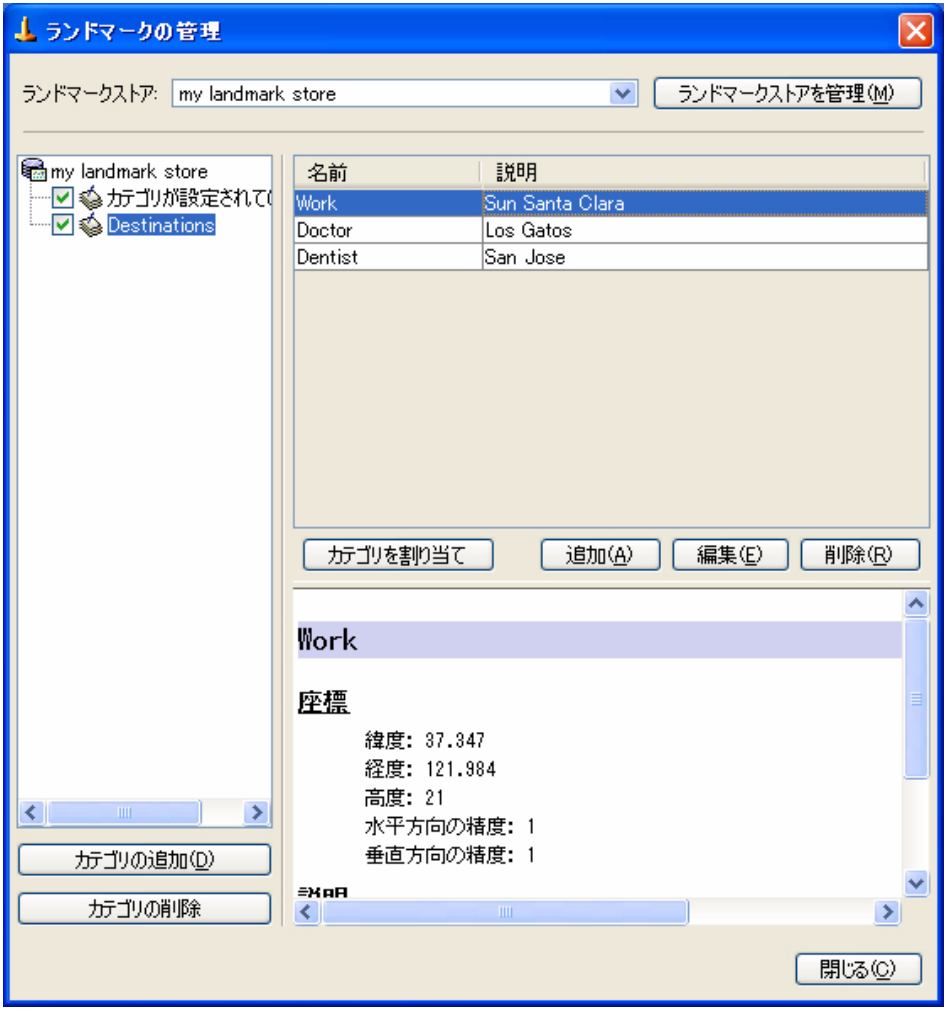


「位置情報」のフィールドで、ツールキットの組み込み位置情報プロバイダのプロパティーを指定できます。環境設定で指定するプロパティーは、アプリケーションが位置情報プロバイダに要求を行うときに使用する `Criteria` クラスに対応します。

13.3 ランドマークの設定

Sun Java™ Wireless Toolkit for CLDC エミュレータには、実際の多くのデバイスと同様に、「ランドマークストア」システムが含まれています。ランドマークストアは、名前などの情報と場所が関連付けられたもののコレクションです。ランドマークストアを管理するには、メニューから「ファイル」->「ユーティリティー」を選択し、「ランドマークの管理」を選択して「起動」を押します。

図 13-3　ランドマークマネージャー



ランドマークマネージャーには、1 つのランドマークストアの内容が表示されます。JSR 179 では少なくとも 1 つのランドマークストアが必要で、これはデフォルトストアと呼ばれます。別のランドマークストアを選択したり、新しいランドマークストアを作成するには、ウィンドウ上部にある「ランドマークストア」コンボボックスで選択します。

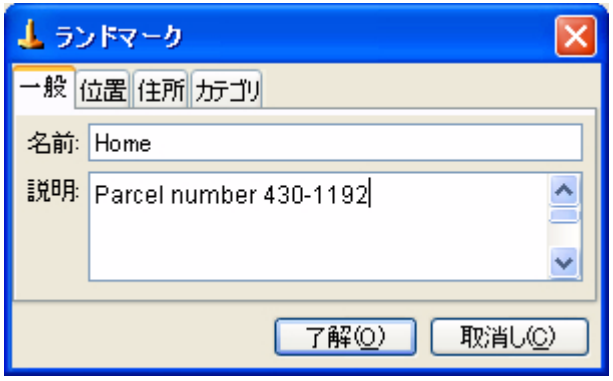
「ランドマークストアの管理」ボタンをクリックすると、ランドマークストアを追加または削除できます。ランドマークストアの名前は変更できません。

ランドマークは、ランドマークストアに固有な「カテゴリ」に関連付けることができます。現在のランドマークストアのカテゴリは、ウィンドウの左側の区画に表示されます。リストの下にあるボタンを使用して、カテゴリを追加または削除できます。カテゴリに一致するランドマークだけを表示する場合は、1 つまたは複数のカテゴリのチェックマークを外します。また、「カテゴリが設定されていません」にチェックマークを付けると、カテゴリが関連付けられていないランドマークを表示できます。

ランドマークマネージャーの右側の区画には、現在のランドマークストアにあるランドマークが一覧で表示されます。ランドマークをクリックすると、右側の区画の下部に詳細なリストが表示されます。

新しいランドマークを追加するには、「追加」をクリックしてフィールドに適切な情報を入力します。現在選択しているランドマークを変更する場合は、「編集」をクリックします。なお、現在選択しているランドマークを削除する場合は、「削除」をクリックします。

図 13-4　ランドマークの追加または編集



メインウィンドウの「カテゴリを割り当て」を使用して、ランドマークのカテゴリを指定することもできます。

第14章

SATSA の使用法

Security and Trust Services APIs (SATSA) は、小型デバイス上で動作するアプリケーションにスマートカードへのアクセスと暗号化の機能を提供します。JSR 177 (SATSA 仕様) では、オプションパッケージとして 4 つの API が定義されています。

- SATSA-APDU - アプリケーションが低レベルプロトコルを使用してスマートカードアプリケーションと通信できるようにします。
- SATSA-JCRMI - リモートオブジェクトプロトコルを使用してスマートカードアプリケーションと通信するための代替方法を提供します。
- SATSA-PKI - アプリケーションがスマートカードを使用してデータにデジタル署名したり、ユーザー証明書を管理できるようにします。
- SATSA-CRYPTO - メッセージダイジェスト、デジタル署名、および暗号をサポートする汎用の暗号化 API です。

Sun Java™ Wireless Toolkit for CLDC エミュレータでは、SATSA を完全にサポートします。この章では、Sun Java™ Wireless Toolkit for CLDC を使用して、アプリケーションで SATSA を操作する方法を説明します。

SATSA と小型デバイスでスマートデバイスを使用する方法の概要については、『SATSA Developer's Guide』(`http://java.sun.com/j2me/docs/satsa-dg/`) を参照してください。

Sun Java™ Wireless Toolkit for CLDC には、Java Card Platform Simulator が含まれています。これを使用して、Sun Java™ Wireless Toolkit for CLDC エミュレータのスロットでスマートカードをシミュレートできます。Java Card Platform Simulator は、次の場所にあります。

Windows:	*toolkit*¥bin¥cref.exe
Linux:	*toolkit*/bin/cref

以降では、単に `cref` と呼びます。

独自の Java Card アプリケーションを開発する必要がある場合は、`http://java.sun.com/products/javacard/` で Java Card Development Kit をダウンロードしてください。

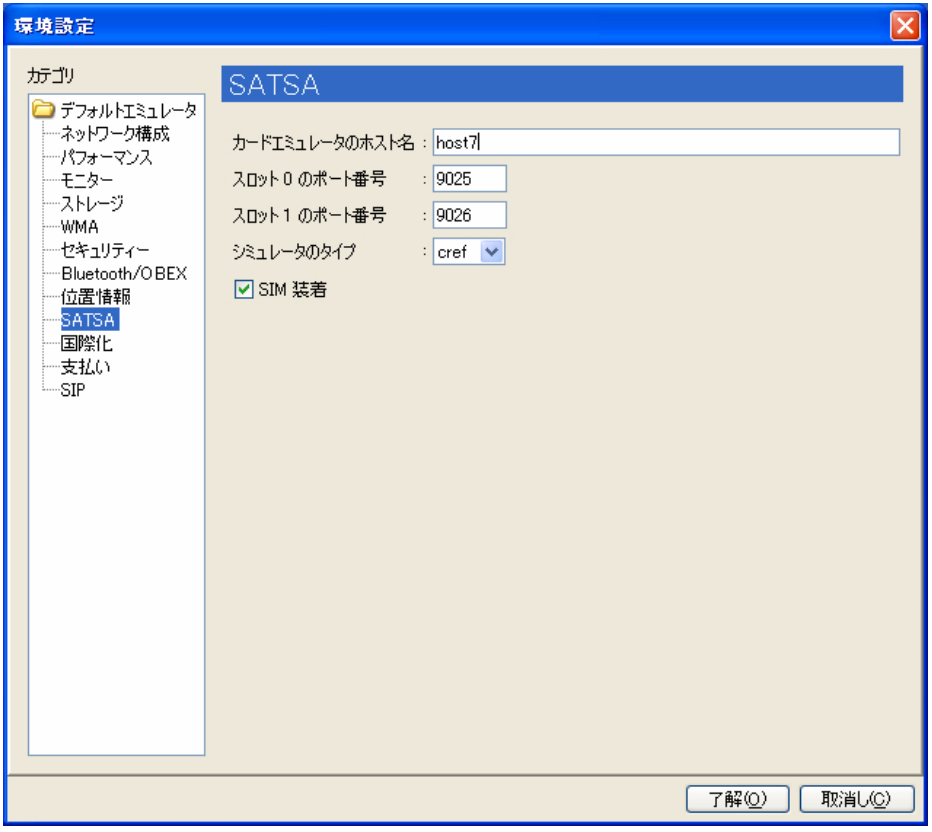
14.1 エミュレータのカードスロット

多くの場合、実際の SATSA デバイスには、スマートカードを収納する 1 つまたは複数のスロットが装備されます。SATSA を使用してスマートカードと通信するアプリケーションは、スロットとカードアプリケーションを指定する必要があります。

Sun Java™ Wireless Toolkit for CLDC エミュレータは実際のデバイスではないため、スマートカード用の物理的なスロットはありません。代わりに、ソケットプロトコルを使用してスマートカードアプリケーションと通信します。ソケットの通信先はスマートカードシミュレータか、実際のスマートカードハードウェアと通信するプロキシになります。

Sun Java™ Wireless Toolkit for CLDC エミュレータには、シミュレートされたスマートカードスロットが 2 つあります。各スロットには、スマートカードアプリケーションと通信するために使用されるプロトコルの端点を表すソケットが関連付けられています。各スロットにソケットのポート番号を設定できます。「編集」->「環境設定」を選択し、「SATSA」タブをクリックします。スロット 0 のデフォルトポートは 9025、スロット 1 は 9026 です。

図 14-1　スマートカードスロットのポート番号の設定



14.2 Java Card Platform Simulator の使用

Sun Java™ Wireless Toolkit for CLDC を使用して SATSA アプリケーションをテストする基本的な手順は、次のとおりです。

1. **Java Card プラットフォームアプリケーションで `cref` を起動します。**
2. **Sun Java™ Wireless Toolkit for CLDC エミュレータを起動します。**

   SATSA アプリケーションがスマートカードと通信を試みるときには、ソケット接続を使用して `cref` と通信します。

   したがって、必ず Sun Java™ Wireless Toolkit for CLDC の環境設定で指定したスロットのポート番号と同じポート番号を使って `cref` を起動することが重要です。

   たとえば、構築済みのメモリーイメージを使ってポート 9025 で `cref` を実行する場合は、コマンド行で次のように入力します。

   ```
   cref -p 9025 -i memory_image.eeprom
   ```

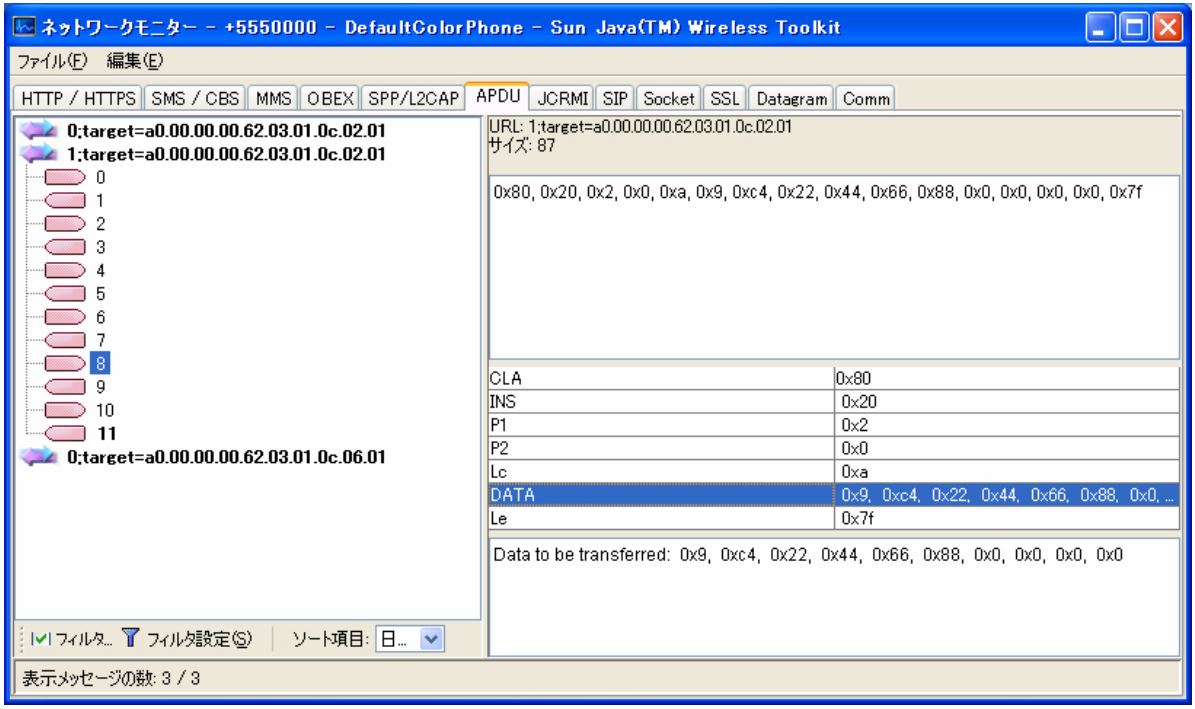
   Sun Java™ Wireless Toolkit for CLDC に含まれているデモアプリケーション `Mohair` では、SATSA の使用方法が説明されています。`Mohair` の実行手順の詳細については、付録 A を参照してください。

14.3 SATSA でのネットワークモニターの使用

Sun Java™ Wireless Toolkit for CLDC では、シミュレートされたスマートカードアプリケーションと交換したデータをネットワークモニターに表示できます。ネットワークモニターには、エミュレータとスマートカードシミュレータ間で交換された APDU (Application Protocol Data Unit) が表示されます。また、Java Card Remote Method Invocation (Java Card RMI) プロトコルを使用して交換されたデータも表示できます。ネットワークモニターの「APDU」および「JCRMI」タブに、スマートカードと交換したデータが表示されます。

ネットワークモニターは各 APDU を解析し、要求と応答のフィールドを表示します。

図 14-2　ネットワークモニターでの APDU の表示



14.4　アクセス制御の調整

アクセス制御のアクセス権と PIN プロパティーは、テキストファイルで指定できます。最初の APDU または Java Card RMI 接続が確立されると、実装は *workdir*¥appdb ディレクトリの acl_*slot-number* から ACL と PIN データを読み取ります。たとえば、スロット 0 のアクセス制御ファイルは、*workdir*¥appdb¥acl_0 です。ファイルがない場合またはファイルにエラーがある場合、このスロットのアクセス制御の検証は無効になります。

ファイルには、PIN プロパティーとアプリケーションのアクセス権に関する情報も指定できます。

14.4.1 PIN プロパティーの指定

PIN プロパティーは、アクセス制御ファイルの pin_data レコードで表されます。

```
pin_data {
    label string
    id number
    type        bcd | ascii | utf | half-nibble | iso
    min         minLength -
    stored      storedLength
    max         maxLength
    reference   byte
    pad         byte - optional
    flag        case-sensitive | change-disabled |
                unblock-disabled | needs-padding |
                disable-allowed | unblockingPIN
   }
```

14.4.2 アプリケーションのアクセス権の指定

アプリケーションのアクセス権は、アクセス制御ファイル (acf) のレコードで定義されます。

```
acf AID fnumbers separated by blanks {
    ace {
         root CA name
         ...
         apdu {
              eight numbers separated by blanks
                 ...
         }
         ...
         jcrmi {
                  classes {
                 classname
                     ...
                     }
                      hashModifier string
                      methods {
                 method name and signatiure
                     ...
                   }
         }
         ...
         pin_apdu {
```

```
                id number
            verify | change | disable | enable | unblock
          four hexadecimal numbers
                ...
        }
        ...
        pin_jcrmi {
                id number
            verify | change | disable | enable | unblock
          method name and signature
                ...
          }
      ...
      }
  ...
}
```

acf レコードはアクセス制御ファイルを表します。acf の後ろの AID は、アプリケーションを識別します。AID が指定されていない場合、エントリはすべてのアプリケーションに適用されます。acf レコードには、ace レコードを含めることができます。ace レコードがない場合、アプリケーションへのアクセスはこの acf によって制限されます。

ace レコードはアクセス制御エントリを表します。このレコードには、root、apdu、jcrmi、pin_apdu、および pin_jcrmi レコードを含めることができます。

root レコードには、1 つの CA 名を記述します。MIDlet スイートがこの CA によって発行された証明書を使用して承認されている場合、この ace はこの MIDlet へのアクセスを許可します。root フィールドが指定されていない場合は、識別された対象のすべてに ace が適用されることを示します。1 行に 1 つの主体を記述します。この行には、root と主体名だけを記述する必要があります。たとえば、次のように指定します。

```
root CN=thehost;OU=JCT;O=dummy CA;L=Santa Clara;ST=CA;C=US
```

apdu または jcrmi レコードは、APDU または Java Card RMI アクセス権を指定します。アクセス権レコードが指定されていない場合は、すべての操作が許可されます。

APDU アクセス権には、空白文字で区切られた 8 つの 16 進数の列が 1 つまたは複数含まれます。最初の 4 バイトは APDU コマンドを表し、残りの 4 バイトはマスクを表します。たとえば、次のように指定します。

```
apdu {
    0 20  0 82  0 20  0 82
    80 20  0  0 ff ff  0  0
}
```

Java Card RMI アクセス権には、ハッシュ修飾子 (オプション)、クラスリスト、およびメソッドリスト (オプション) に関する情報が含まれます。メソッドのリストが空の場合、アプリケーションはクラスのリストにあるすべてのリモートメソッドのインタフェースを呼び出すことができます。たとえば、次のように指定します。

```
jcrmi {
    classes {
             com.sun.javacard.samples.RMIDemo.Purse
    }
    hashModifier zzz
    methods {
        debit(S)V
        setAccountNumber([B)V
        getAccountNumber()[B
    }
}
```

すべての数値は 16 進数です。タブ、空白文字、復帰、および改行記号が、区切り文字として使用されます。{ および } の記号の前後では、区切り文字を省略できます。

pin_apdu および pin_jcrmi レコードには、PIN エントリメソッドに必要な情報が含まれます。これらの情報は、PIN 識別子、APDU コマンドヘッダー、またはリモートメソッド名です。

14.4.3 アクセス制御ファイルの例

```
pin_data {
  label   Unblock pin
  id        44
  type      utf
  min       4
  stored    8
  max       8
  reference 33
  pad       ff
  flag      needs-padding
  yflag     unblockingPIN
}
pin_data {
  label     Main pin
  id        55
  type      half-nibble
  min       4
  stored    8
  max       8
```

```
  reference 12
  pad       ff
  flag      disable-allowed
  flag      needs-padding
}

acf a0 0 0 0 62 ff 1 {
  ace {
        root CN=thehost;OU=JCT;O=dummy CA;L=Santa Clara;ST=CA;C=US

        pin_jcrmi {
            id 55
            verify enterPIN([B)S
            change changePIN([B[B)S
            disable disablePIN([B)S
            enable enablePIN([B)S
            unblock unblockPIN([B[B)S
           }
  }
}

acf a0 0 0 0 62 ee 1 {
  ace {
    root CN=thehost;OU=JCT;O=dummy CA;L=Santa Clara;ST=CA;C=US

    pin_apdu {
            id 55
            verify 1 2 3 1
            change 4 3 2 2
            disable 1 1 1 3
            enable 5 5 5 4
            unblock 7 7 7 5
        }
  }
}

acf a0 0 0 0 62 3 1 c 8 1 {
  ace {
        root CN=thehost;OU=JCT;O=dummy CA;L=Santa Clara;ST=CA;C=US

        jcrmi {
                  classes {
                      com.sun.javacard.samples.RMIDemo.Purse
              }
                  hashModifier xxx
                  methods {
                    setAccountNumber([B)V
                    getBalance()S
                    credit(S)V
```

```
                  }
  }
}
  ace {
        jcrmi {
                classes {
                     com.sun.javacard.samples.RMIDemo.Purse
                  }

                  debit(S)V
                  getAccountNumber()[B
              }
            }
       }
  }

acf a0 00 00 00 62 03 01 0c 02 01 {
  ace {
        root CN=thehost;OU=JCT;O=dummy CA;L=Santa Clara;ST=CA;C=US
        apdu {
                0 20  0 82  0 20  0 82
                80 20  0  0 ff ff  0  0
              }
        apdu {
                80 22  0  0 ff ff  0  0
              }
        }
  }
acf a0 00 00 00 62 03 01 0c 02 01 {

  ace {
    apdu {
        0 20 0 82 ff ff ff ff
    }
  }
}

acf a0 00 00 00 62 03 01 0c 06 01 {

  ace {
    apdu {
        0 20 0 82 ff ff ff ff
    }
  }
}
```

第 15 章

SIP の使用法

Sun Java™ Wireless Toolkit for CLDC では、プロキシサーバー、登録機関、およびネットワークモニターサポートを含む、SIP API for J2ME (JSR 180) がサポートされます。

SIP (Session Initiation Protocol) は、RFC 3261 で定義されています。この情報は、`http://www.ietf.org/rfc/rfc3261.txt` で入手できます。

SIP では、アプリケーションで通信を設定する標準的な方法が提供されます。アプリケーションでは、実際に実行する通信を決定します。SIP を使用して、インスタントメッセージング、テキストチャット、音声チャット、ビデオ会議、およびその他のセッションを設定できます。

15.1 登録機関とプロキシについて

SIP 登録機関により、クライアントアプリケーションはユーザー名と特定のネットワークアドレスを関連付けることができます。登録処理により、自分の場所を通知する方法がユーザーに提供されます。

SIP プロキシサーバーは、プロキシサーバーによる大規模なネットワークへの入口でしかありません。プロキシサーバーに到着する SIP メッセージは、適切な宛先にルーティングされます。通常、この宛先は別のプロキシサーバーか、デスクトップコンピュータやモバイルデバイスなどのエンドポイントです。SIP メッセージはデバイス間で直接送信できますが、通常はプロキシサーバーを通してルーティングされます。

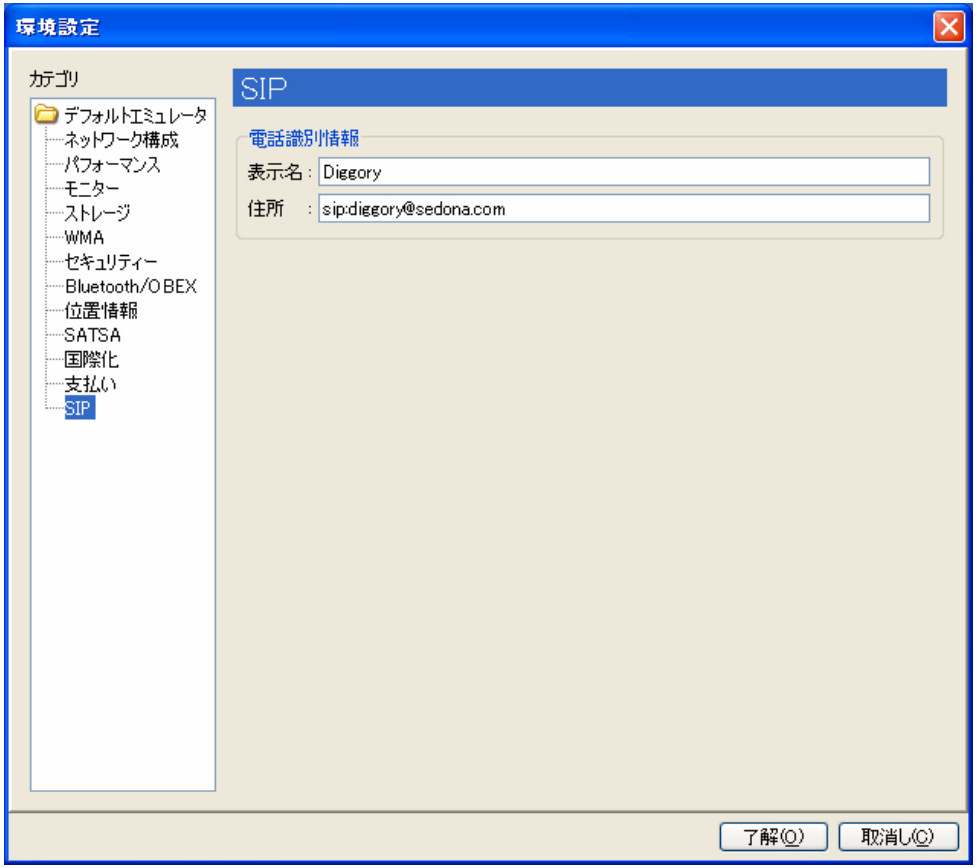
たとえば、Diggory が Polly とのテレビ会議を開始する場合を考えてみましょう。Polly は外出中で、その携帯電話から登録機関にメッセージが送信され、自分の名前と携帯電話のネットワークアドレスが関連付けられます。Diggory が Polly とのテレビ会議を設定しようとするときに、Diggory のアプリケーションでは SIP を使用して Polly の現在のネットワーク上の場所を登録機関に問い合わせます。

Sun Java™ Wireless Toolkit for CLDC には、SIP API を使用するアプリケーションのテストに使用できる、簡単な SIP プロキシと登録機関サーバーが用意されています。また、ツールキットを設定して、外部のプロキシサーバーと登録機関サーバーを使用することもできます。

15.2 SIP の設定

Sun Java™ Wireless Toolkit for CLDC の SIP 環境の設定を調整するには、「編集」->「環境設定」を選択し、「SIP」をクリックします。

図 15-1 SIP の設定



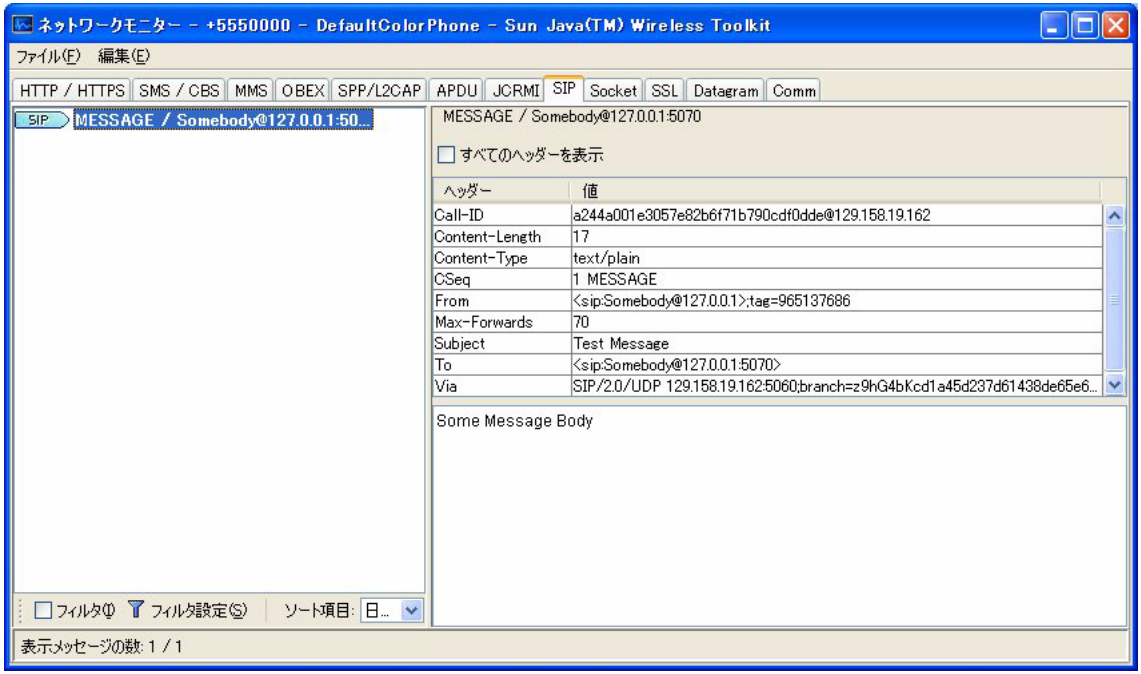
「表示名」と「アドレス」フィールドでは、それぞれ sip.display.name と sip.address のシステムプロパティーを設定します。アプリケーションでは、デバイスに関連付けられた識別情報を取得する標準的な方法として、これらのシステムプロパティーを使用できます。

15.3 ネットワークモニターでの SIP トラフィック

SIP API を使用して送受信されるネットワークデータは、ネットワークモニターで記録できます。ネットワークモニターについては、第 5 章で詳しく説明しています。ネットワークモニターを使用して、エミュレータで送受信される SIP メッセージを追跡できます。

「SIP」タブをクリックして SIP メッセージを表示します。SIP メッセージは、ネットワークモニターの左側の区画に表示されます。メッセージをクリックすると、その詳細が右側の区画に表示されます。

図 15-2　ネットワークモニターでの SIP メッセージ



15.4 SIP プロキシサーバーと登録機関

SIP アプリケーションを簡単に作成できるように、Sun Java™ Wireless Toolkit for CLDC には簡単な SIP プロキシサーバーと登録機関が用意されています。サーバーを起動するには、「ファイル」->「ユーティリティー」を選択します。リストから「SIP サーバーを起動」を選択し、「起動」を押します。SIP サーバーのコンソールウィンドウが表示されます。

図 15-3　SIP サーバーのコンソール

サーバーを停止するには、「停止」をクリックします。サーバーを起動するには、「起動」をクリックします。

サーバーの動作中は、左上の区画に登録機関で認識されているすべてのユーザーが表示されます。ユーザー名をクリックすると、ユーザーに関する詳細が右上の区画に表示されます。ウィンドウの下部の区画は、プロキシサーバーによって送受信される SIP メッセージが表示されるコンソールです。

サーバーが動作していないときは、サーバーのオプションを調整できます。サーバーを停止して「オプション」をクリックすると、オプションのウィンドウが表示されます (図 15-4 参照)。

図 15-4　　プロキシオプションの設定

「プロキシ」タブで、IP アドレスと、サーバーが着信メッセージを待機するポートを設定できます。待機中のデフォルトの場所 (リスニングポイント) として、SIP プロキシではポート 5060 がよく使われます。マルチユーザー環境で作業している場合は、ほかのユーザーがこのポートを使用していて、意図せず別のユーザーの SIP サーバーインスタンスに接続してしまう可能性があります (SIP サーバーには認証の機構がなく、TCP/IP ポートには自由にアクセスできます)。この場合は、別のポートを指定する必要があります。

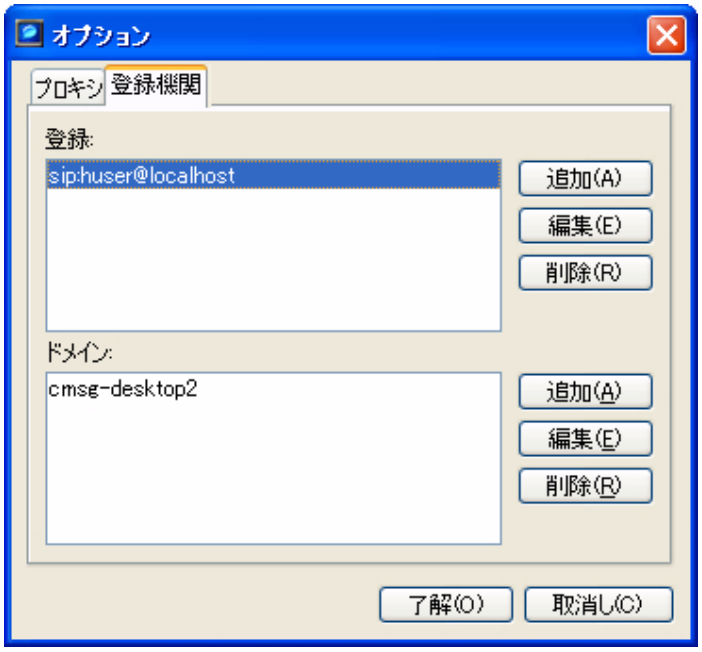
「追加」をクリックして、ポートとその種類を指定します。ポートを選択して「削除」をクリックすると、待機中のポートを削除できます。

接続するクライアントをサーバーに対して強制的に認証するには、「認証を使用」にチェックマークを付けます。使用される方式はダイジェスト認証です。この認証は、RFC 3261 の第 22.4 節で説明されています。SIP のダイジェスト認証は、HTTP のダイジェスト認証とほぼ同じです。

「登録機関」タブで、登録機関で認識されるユーザーとドメインを設定できます。上部のリストには、SIP サーバーを起動したときに自動的に登録される SIP ユーザーがリストされます。新しいユーザーの追加、既存のユーザーの編集、またはユーザーの削除を行えます。

また、この登録機関で管理されるドメインのリストを調整できます。「追加」を押すとドメインを追加できます。「編集」を押すと既存のドメイン名を編集できます。「削除」を押すとドメインを削除できます。

図 15-5　登録機関オプションの設定



第 16 章

Payment API の操作

JSR 229 「Payment API」を使用すると、アプリケーションはユーザーに代わって支払いを行うことができます。Payment API では、支払い「アダプタ」を通じてさまざまな支払い機構がサポートされます。Payment API を実装するデバイスには、1 つまたは複数のアダプタがあります。MIDlet スイートは、記述子属性を使用して、使用できる支払いアダプタの種類を指定します。

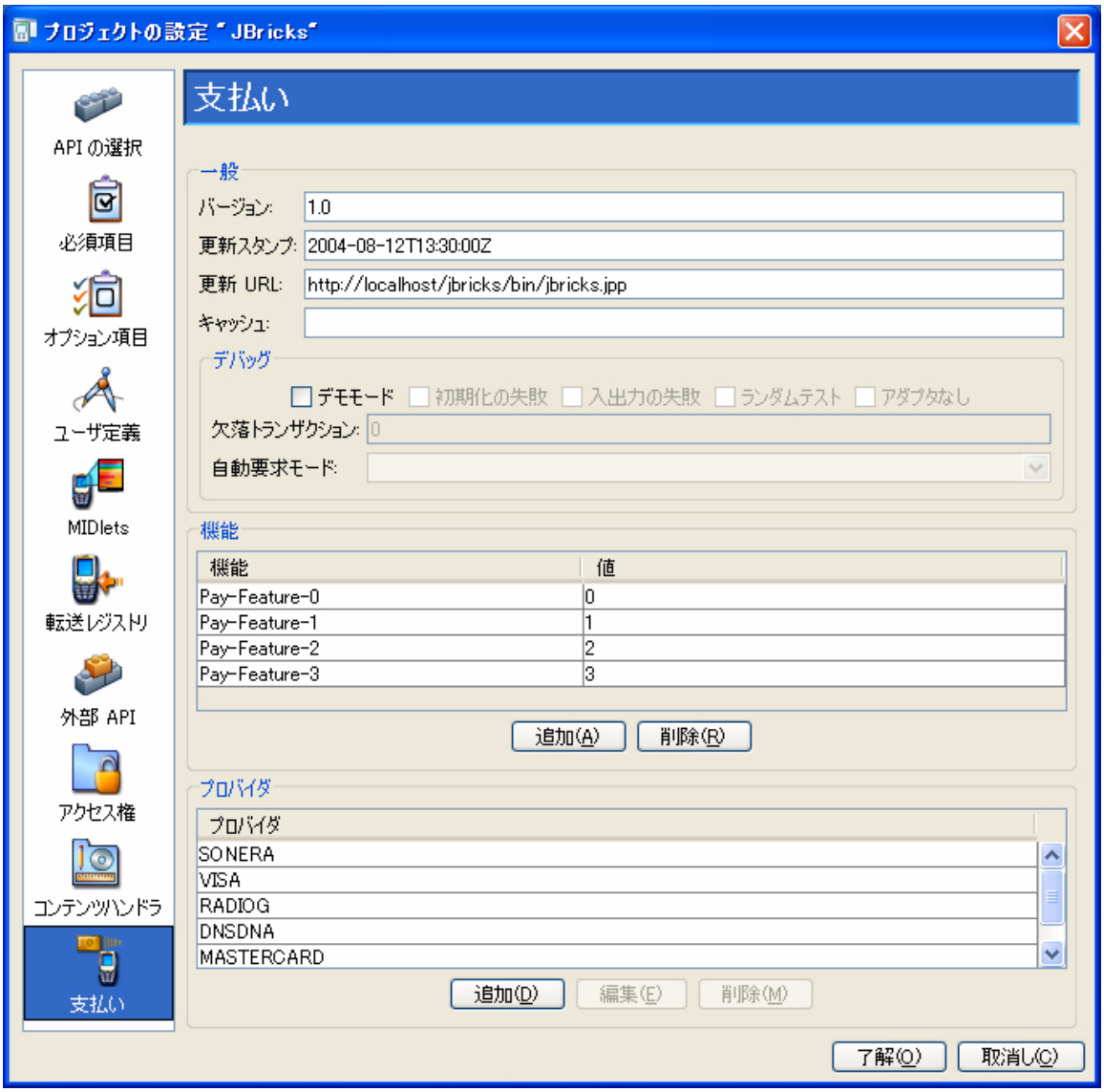
Sun Java™ Wireless Toolkit for CLDC のエミュレータが実装する Payment API には、Premium Priced での SMS (PPSMS) とクレジットカードによる支払いの両方をシミュレートする支払いアダプタのサンプルが付属しています。また、ツールキットにより、MIDlet の記述子と JAR ファイルマニフェストに必要な属性を簡単に設定できます。支払いコンソールにより、アプリケーションによって実行された支払いや試みられた支払いを簡単に追跡できます。

Payment API は配信および外部デバイスの支払い機構に密接に結び付けられ、支払いは信頼できる保護ドメインでのみ成功するため、常にツールキットの「OTA 経由で実行」を使用して Payment API アプリケーションのテストとデバッグを行なってください。詳細は、第 2 章を参照してください。

16.1 支払いのプロジェクト設定

プロジェクトの支払い属性を調整するには、「属性設定」をクリックし「支払い」アイコンを選択します。

図 16-1　支払いの設定



フィールドと値は JSR 229 「Payment API」の仕様で詳しく説明されています。

「一般」ボックスには、使用中の Payment API のバージョンと支払いの更新を検索する場所に関する情報があります。テストでは、プロジェクトディレクトリから直接更新ファイルを取得する `localhost` の URL (スクリーンショットを参照) を指定できます。

「デバッグ」ボックスには、アプリケーションのテスト中に役立つオプションがあります。各オプションについては、Payment API の仕様で説明されています。

「機能」ボックスには、アプリケーションで料金が発生する機能のリストが表示されます。これらの機能は、プロバイダごとに一覧表示される価格情報に対応します。機能のリストは、「追加」および「削除」ボタンを使用して変更できます。

「プロバイダ」ボックスには、このアプリケーションで使用できる支払いプロバイダが一覧表示されます。支払いを行うときに、エミュレータ (またはデバイス) は使用できる支払いアダプタの 1 つを、アプリケーションに対して表示されるプロバイダのいずれかに対応付けます。プロバイダのリストは、「追加」、「編集」、および「削除」ボタンを使用して変更できます。プロバイダを追加または編集する場合は、次のウィンドウが表示されます。

図 16-2　支払いプロバイダの編集

これらのフィールドも、Payment API の仕様で詳しく説明されています。

「価格情報」ボックスには、定義された支払い機能が 1 行ずつ表示されます。価格タグの値を編集するには、対応する「値」列のセルをダブルクリックします。

16.2 支払い属性の直接編集

支払い属性は、拡張子が .jpp の「支払い更新ファイル」に保存されます。詳細については、仕様を参照してください。Sun Java™ Wireless Toolkit for CLDC には、プロジェクトの設定と独立して、支払い更新ファイルを簡単に編集するためのユーティリティーが用意されています。

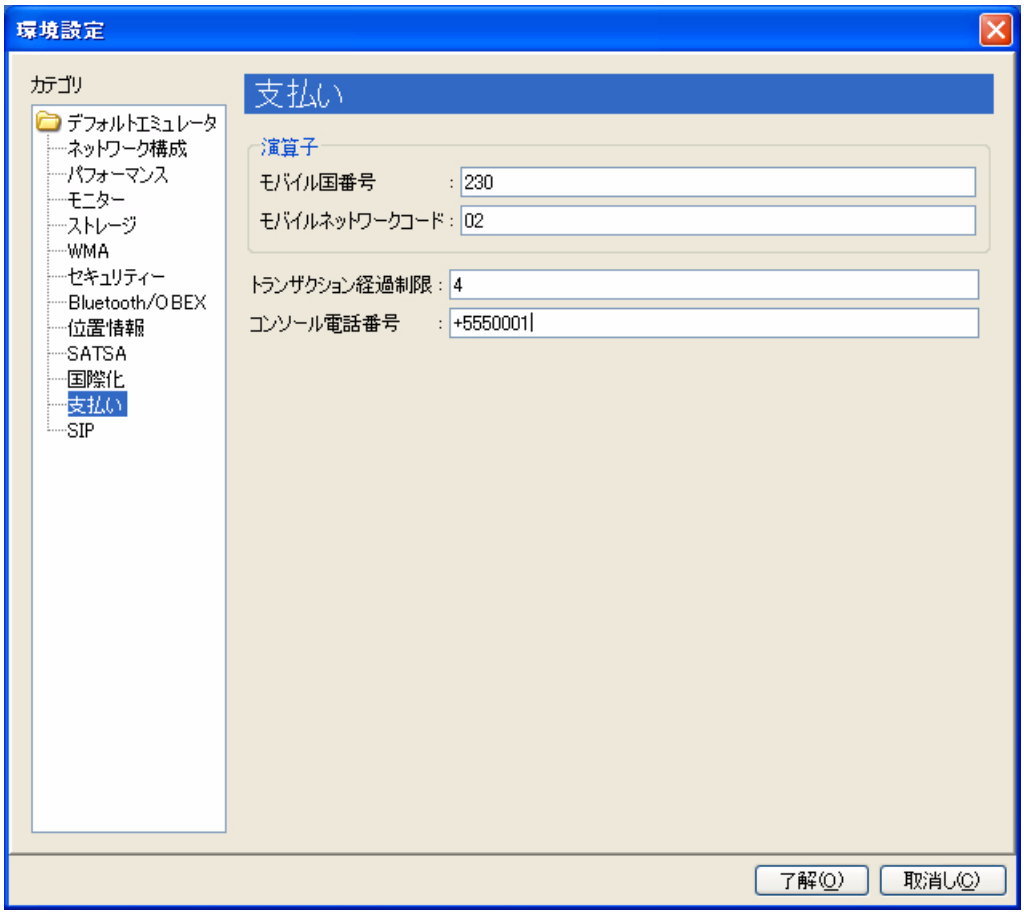
ユーティリティーを実行するには、「ファイル」->「ユーティリティー」を選択し、「支払い編集ダイアログ」を選択して「起動」を押します。編集する支払い更新ファイルを選択するように求めるメッセージが表示されます。ファイルを選択すると、プロジェクト設定の「支払い」セクションとほぼ同じ内容のウィンドウが表示されます。支払い編集ユーティリティーにはデバッグ設定がありません。

支払い更新設定を編集し、完了したら「了解」を押します。支払い更新ファイルは暗号で署名されるため、ファイルの署名に使用できる鍵のリストが表示されます。使用する鍵を選択して、「支払い更新ファイルに署名」を押します。

16.3 支払いの環境設定

ツールキットの Payment API 設定を調整するには、「編集」->「環境設定」を選択し、「支払い」をクリックします。

図 16-3　支払いの環境設定



「モバイル国番号」(MCC) と「モバイルネットワークコード」(MNC) は、PPSMS 支払いプロバイダと組み合わせて使用されます。これらを組み合わせて、MCC と MNC はデバイスのワイヤレス通信事業者を識別します。支払い時には、プロジェクト設定のプロバイダのリストから一致するプロバイダを検索するために MCC と MNC が使用されます。エミュレータは実際のデバイスではないため、MCC と MNC に適切な値を入力してキャリアをシミュレートできます。詳細については、Payment API の仕様を参照してください。

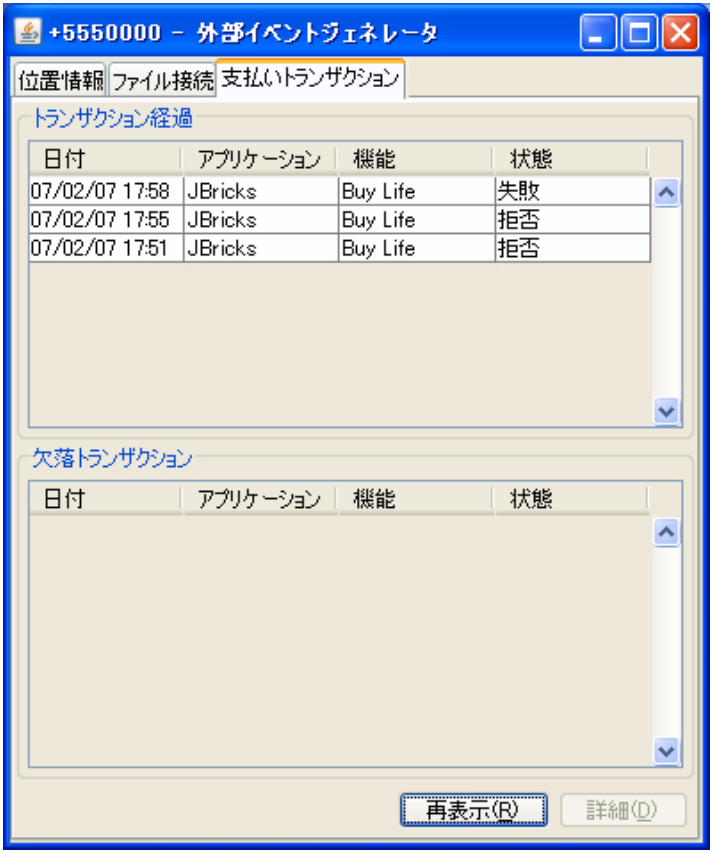
「トランザクション経過制限」は、エミュレータに記録されるトランザクションの数を決定します。この設定は、外部イベントウィンドウに表示されるリストの長さ (後述) と、アプリケーションが取得できる過去のトランザクションの数に影響します。

最後に、「コンソール電話番号」は、支払いコンソールのシミュレートされた電話番号を決定します。これについては後述します。

16.4 トランザクション履歴の表示

エミュレータは、実際のデバイスと同様に支払いトランザクションを追跡します。トランザクションの履歴を確認するには、エミュレータウィンドウのメニューから「MIDlet」->「外部イベント」を選択します。「支払いトランザクション」タブをクリックします。

図 16-4　支払いトランザクションの表示



さらに支払いを行なったあとに「更新」をクリックすると、リストが更新されます。トランザクションを選択して「詳細」をクリックすると、別のウィンドウにすべての詳細が表示されます。

外部イベントウィンドウには、エミュレータを OTA 経由で実行している間に行われたトランザクションだけが表示されます。OTA を使用せずにトランザクションを完了することもできますが、これらのトランザクションは表示されません。実際に近い支払いのシミュレーションでは、常に「OTA 経由で実行」を使用してアプリケーションをテストしてください。

16.5 支払いの監視

Sun Java™ Wireless Toolkit for CLDC では支払いコンソールが提供され、サンプルの支払いアダプタを通過する支払いを簡単に確認できます。支払いサービスプロバイダコンソールを起動するには、「ファイル」->「ユーティリティー」->「支払いコンソール」を選択します。

図 16-5　支払いコンソール

また、ネットワークモニターを使用してトランザクションを表示できます。クレジットカードのトランザクションは HTTPS を使用して処理され、PPSMS トランザクションでは SMS が使用されます。ネットワークモニターの詳細は、第 5 章を参照してください。

第17章

Mobile Internationalization API の使用法

JSR 238 「Mobile Internationalization API」は、複数の言語で表示され、複数の国で使用されるアプリケーション向けに設計されています。国 (または地域) と言語の組み合わせを、「ロケール」と呼びます。

JSR 238 の中心となる概念は「リソース」です。これは、特定のロケールに適した文字列、画像、またはその他のオブジェクトです。たとえば、ヨーロッパで配布されるアプリケーションには、イタリア国内のイタリア語を話すユーザー、スイス国内のイタリア語を話すユーザー、スペイン国内のスペイン語を話すユーザー、ポルトガル国内のスペイン語を話すユーザーなどに向けたリソースが含まれます。

リソースは、JSR 238 で定義された形式でファイルに保存されます。リソースファイルは、MIDlet スイートの JAR ファイルの一部としてバンドルされます。Sun Java™ Wireless Toolkit for CLDC には、リソースファイルの作成と管理作業を簡単に行えるようにするリソースマネージャーが用意されています。

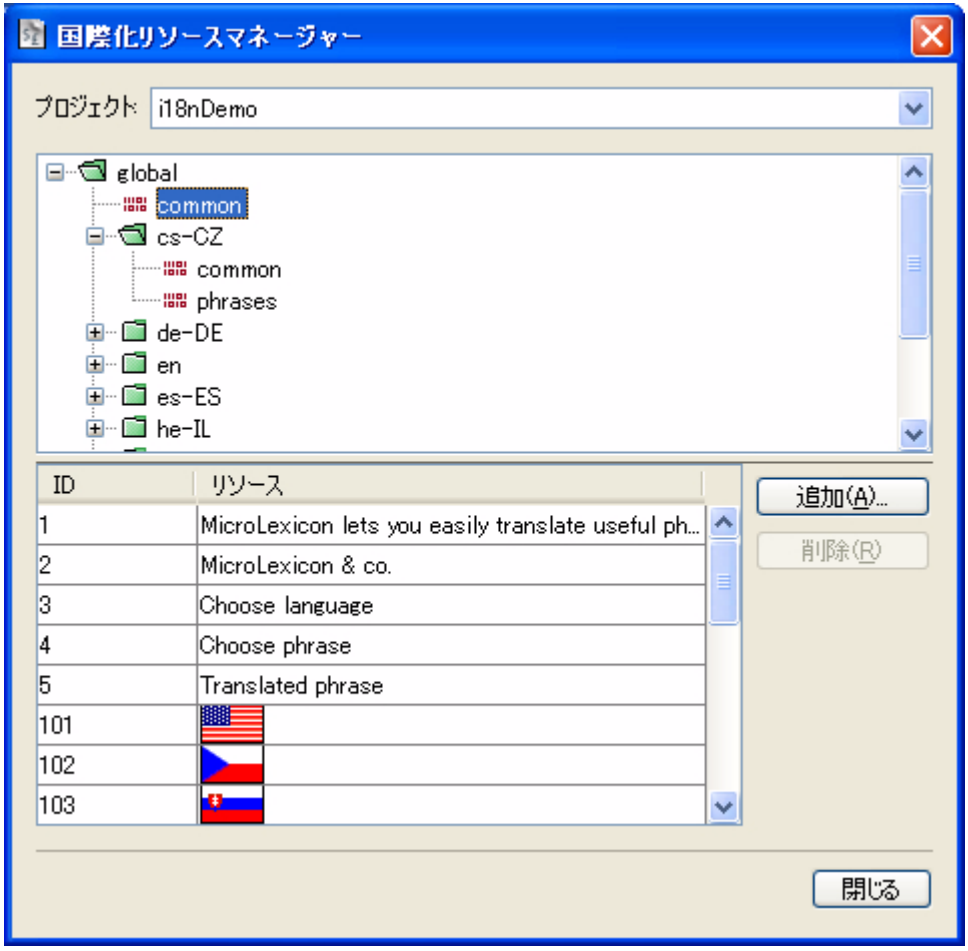
17.1 エミュレータのロケールの設定

デバイスのロケールは、システムプロパティー `microedition.locale` に格納されます。エミュレータのロケールを変更するには、「編集」->「環境設定」を選択し、「国際化」を選択します。コンボボックスからロケールを選択するか、直接入力します。

17.2 アプリケーションリソースの表示

リソースマネージャーを起動するには、「ファイル」->「ユーティリティー」を選択します。「国際化リソースマネージャー」を選択し、「起動」をクリックします。

図 17-1 リソースマネージャー



最初に、「プロジェクト」メニューからプロジェクトを選択します。選択したプロジェクトのすべてのリソースが、ウィンドウ内に表示されます。リソースマネージャーを見るのが初めての場合は、`i18nDemo` デモアプリケーションのリソースを確認してください。このアプリケーションには、多くの興味深いリソースが含まれています。

+ 記号をクリックすると、ディレクトリを展開できます。- 記号をクリックすると、ディレクトリが折りたたまれます。

その他のほとんどの操作は、ディレクトリまたはリソースファイルを右クリックすることで実行できます。

ウィンドウの上部には、アプリケーションにあるリソースファイルの階層が表示されます。ウィンドウの下部には、リソースファイルの内容が表示されます。図 17-1 では、最上位の common リソースファイルの内容を表示しています。

17.3 ロケールの操作

ロケールは、最上位の global ディレクトリ以下のディレクトリで表されます。ロケールディレクトリにはリソースファイルが含まれ、リソースファイルにはアプリケーションで使用できる実際のリソースが保存されています。

ロケールは、MIDP 2.0 仕様で説明されているように、標準の言語と国のコードによって表されます。たとえば、pt-BR はブラジル国内でポルトガル語を話すユーザーを表します。

ロケールディレクトリを追加するには、最上位の global ディレクトリを右クリックし、「ロケールを追加」を選択します。コンボボックスからロケールを選択するか、ロケールを直接入力して、「了解」をクリックします。

ロケールの名前を変更するには、ロケールディレクトリを右クリックして、「名称変更」を選択します。

ロケールとロケールに含まれているすべてのリソースファイルを削除するには、ロケールディレクトリを右クリックして「削除」を選択します。

17.4 リソースファイルの操作

リソースファイルは、グローバル (最上位) にするか、ロケールディレクトリ内部に配置できます。新しいグローバルリソースファイルを作成するには、最上位の global ディレクトリを右クリックして、「新規リソースファイルを追加」を選択します。ファイルの名前を選択します。

リソースファイルの名前を変更するには、ファイルを右クリックして「名称変更」を選択します。

リソースファイルを削除するには、ファイルを右クリックして「削除」を選択します。

リソースファイル全体をコピー、カット、およびペーストできます。ファイルを右クリックして、「コピー」または「カット」を選択します。ロケールディレクトリ (または最上位の `global`) を右クリックし、「ペースト」を選択します。

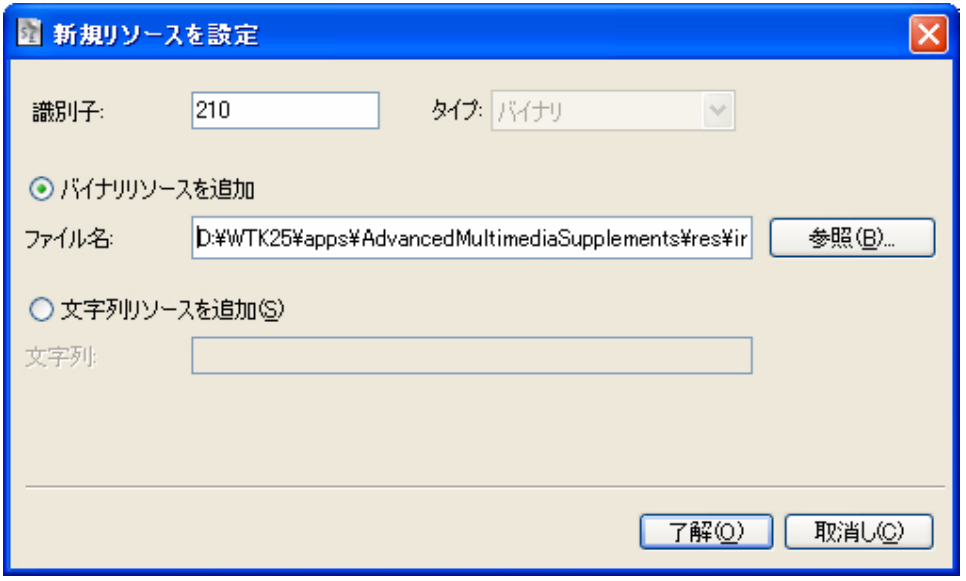
17.5 リソースの操作

リソースマネージャーウィンドウの上部でリソースファイルをクリックすると、そのファイルの内容がウィンドウの下部に表示されます。

新しいテキストリソースを追加するには、「追加」をクリックし、「文字列リソースを追加」を選択します。識別子は自動的に割り当てられますが、必要に応じて別の識別子を入力することもできます。「了解」をクリックします。

画像または別の種類のデータを追加するには、「追加」をクリックし、「バイナリリソースを追加」を選択します。追加するファイルは参照して指定できます。この場合も、識別子を変更できます。「了解」をクリックして、リソースを追加します。

図 17-2 画像リソースの追加



リソースを編集するには、リソースをダブルクリックします。文字列リソースに新しい値を入力できます。ファイルリソースをダブルクリックすると、別のファイルを選択できます。

付録 A

デモアプリケーション

この付録では、Sun Java™ Wireless Toolkit for CLDC にバンドルされているデモアプリケーションについて説明します。

A.1 概要

Sun Java™ Wireless Toolkit for CLDC には、エミュレータでサポートされている技術や API の一部を紹介するデモアプリケーションが含まれています。これらのデモアプリケーションの目的は、エミュレータの API 機能およびツールキット全体で拡張された機能について概要を示すことです。

表 A-1 に、このリリースに含まれているデモアプリケーションの一覧を示します。

ほとんどのデモアプリケーションは簡単に実行できます。A-4 ページの A.2 節「一般的な手順」では、デモを実行する手順を説明しています。補足のドキュメントがあるデモは、表 A-1 にリンクが示されています。リンクがない場合、デモは単純で (または独自の手順があり)、一般的な手順で実行できます。

デモアプリケーションのソースコードはすべて、*toolkit*/apps ディレクトリに用意されています。プロジェクトはサブディレクトリに配置されています。プロジェクトごとに src ディレクトリがあり、そこに Java プログラミング言語のソースコードがあります。たとえば、Windows でツールキットが C:¥WTK2.5.2 にインストールされている場合、WMADemo の SMS 送信 MIDlet (example.sms.SMSSend) のソースコードは、C:¥WTK2.5.2¥apps¥WMADemo¥src¥example¥sms¥SMSSend.java にあります。1-2 ページの 1.1.2 節「作業用ディレクトリのファイル」で説明したように、プロジェクトを開くと、*workdir*/apps ディレクトリにプロジェクトがコピーされます。

表 A-1　デモアプリケーション

デモアプリケーション	API	説明	特殊な手順
AdvancedMultimedia Supplements	JSR 234 Advanced Multimedia Supplements	3D オーディオ、残響、画像処理、およびカメラ制御のデモを示します。	A.3
Audiodemo	MMAPI 1.1	アニメーションとオーディオのミキシングや再生など、オーディオ機能をデモします。	
BluetoothDemo	JSR 82 Bluetooth	Bluetooth を使用したデバイス検出とデータ交換をデモします。	A.4
CHAPIDemo	JSR 211 CHAPI	MediaHandler も使用するコンテンツビューアです。	A.5
CityGuide	JSR 179 Location API	現在位置に基づいてランドマークを表示する市街図を示します。	A.6
Demos	MIDP 2.0	アニメーション、カラー、ネットワーク、財務などのさまざまなサンプルが含まれています。	A.7
Demo3D	JSR 184 Mobile 3D Graphics	即時モードおよびリテインモードの両方で 3D グラフィックスの使用方法をデモする MIDlet が含まれています。	A.8
FPDemo	CLDC 1.1	浮動小数点計算を行う簡易電卓です。	
Games	MIDP 2.0	TilePuzzle、WormGame、および PushPuzzle が含まれています。	
GoSIP	JSR 180 SIP	SIP および SIP サーバーを使用したチャットの設定をデモします。	A.9
i18nDemo	JSR 238 Mobile Internationalization API	文字列のソート、数値の書式、および慣用句の翻訳機能が含まれます。	A.10
JBricks	JSR 229 Payment API	追加ライフやレベルの購入に Payment API を使用するゲームです。	A.11
JSR172Demo	Web サービス	JSR 172 API を使用して MIDlet から Web サービスに接続する方法をデモします。	A.12

表 A-1　デモアプリケーション (続き)

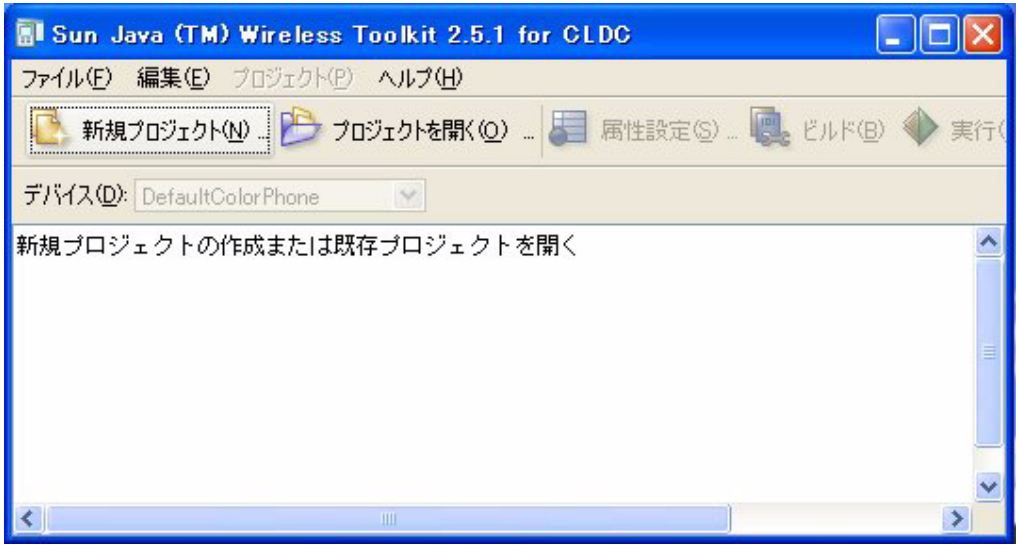
デモアプリケーション	API	説明	特殊な手順
MobileMediaAPI	MMAPI 1.1	トーンシーケンス、MIDI 再生、サンプリングオーディオ再生、ビデオなど、MMAPI の機能をデモします。	A.13
NetworkDemo	MIDP 2.0	データグラムとシリアル接続の使用方法を示します。	A.14
ObexDemo	JSR 82 OBEX	IrDA 経由の OBEX を使用したデータ転送をデモします。	A.15
OpenGL® ES Demo	JSR 239 OpenGL® ES	OpenGL® ES を使用して 3D グラフィックスを作成する方法を示します。	
PDAPDemo	JSR 75 PIM と FileConnection	連絡先、カレンダ、および予定表のアイテムを操作する方法を示します。ローカルファイルへのアクセス方法をデモします。	A.16
Photoalbum	MIDP 2.0	さまざまな画像形式をデモします。	
SATSADemos	JSR 177 SATSA	スマートカードと SATSA のその他の機能との通信をデモします。	A.17
SATSAJCRMIDemo	JSR 177 SATSA	SATSA-Java Card RMI の使用方法を示します。	A.18
SIPDemo	JSR 180 SIP	SIP を使用する簡単なメッセージ交換です。	A.19
SnapMobileSample		ネットワークに接続されたコミュニティーゲームプレイ機能をデモします。	
SVGContactList	JSR 226 SVG API	異なるスキンでの連絡先リストの表示をデモします。	A.20
SVGDemo	JSR 226 SVG API	SVG を描画するさまざまな手法を示します。	A.21
UIDemo	MIDP 2.0	MIDP 2.0 の幅広いユーザーインタフェース機能を紹介します。	
WMADemo	WMA 2.0	SMS、CBS、および MMS の各メッセージを送受信する方法を示します。	A.22

A.2 一般的な手順

ほとんどのデモアプリケーションは、特別な準備を必要とせずに実行して起動できます。一部のデモでは、ツールキットの環境設定や設定を変更する必要があります。この節では、一般的な手順について説明します。

まず、ツールキットを実行します。Microsoft Windows の「スタート」メニューで、「スタート」->「すべてのプログラム」->「Sun Java Wireless Toolkit 2.5.2 for CLDC」->「Wireless Toolkit 2.5.2」を選択します。図 A-1 に示すように、ユーザーインタフェースが表示されます。

図 A-1　Wireless Toolkit のユーザーインタフェース



デモアプリケーションを開くために、「プロジェクトを開く」ボタンをクリックします。使用できるすべてのアプリケーションのリストが表示されます。1-2 ページの 1.1.2 節「作業用ディレクトリのファイル」で説明したように、インストールディレクトリ内のプロジェクトはイタリックで示され、作業用ディレクトリ内のプロジェクトはボールドで示されます (図 1-1)。インストールディレクトリ内のプロジェクトを開くと、プロジェクトが作業用ディレクトリにコピーされてから開きます。

アプリケーションが開いたら、ツールバーの「実行」ボタンをクリックできます。また、エミュレータへのインストールが必要な場合は、「プロジェクト」->「OTA 経由で実行」をクリックします。

デバイスエミュレータウィンドウが表示され、デモアプリケーションが実行されます。MIDlet のメニューがある場合は、ナビゲーションの矢印キーを使用して項目を選択し、「選択」を選択します。デモの実行中に、画面下部の左側または右側にあるソフトウェアボタンを押す必要が生じる場合があります。ソフトウェアボタンは、アプリケーションのインストールまたは起動、メニューの表示、終了、その他のアクションの実行などに使用します。一部のデモには、これらの手順が含まれています。

一部のデモアプリケーションでは、特定の設定や手順が必要です。たとえば、Web サービスを使用するデモをファイアウォールの内側で実行する場合は、エミュレータのプロキシサーバーを設定しなければデモは失敗します。

- 「編集」->「環境設定」を選択します。
- 「ネットワーク構成」アイコンを選択します。
- 「プロキシサーバーを使用」にチェックマークを付けます。
- プロキシサーバーのアドレスとポート番号をフィールドに入力します。

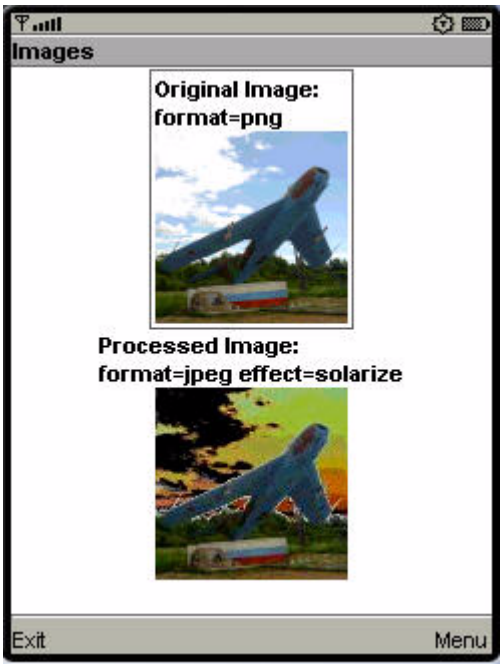
詳しい操作方法については、各デモの説明を参照してください。

A.3 Advanced Multimedia Supplements

この MIDlet スイートは、JSR 234 の Advanced Multimedia Supplements (AMMS) の機能をデモします。このデモは、次の MIDlet で構成されます。

- Image Effects - 標準的な画像処理操作を示します。メニューからエフェクトを選択します。元の画像の下に、処理された画像が表示されます。「Set Transfroms」などの一部の項目では、複数の操作を実行できます。「Done」ソフトウェアボタンをクリックして、各エフェクトを適用します。

図 A-2　　MIDlet での画像の処理



- Radio Tuner - ツールキットの組み込み HTTP サーバーを経由してプロジェクトディレクトリのオーディオファイルを読み取り、ラジオ受信機をシミュレートします。
- Camera - Advanced Multimedia Supplements でデバイスのカメラを制御する方法を示します。画面が (動画でシミュレートされた) カメラのファインダーになります。メニューにあるコマンドを使用して、カメラの設定を変更したり、スナップショットを撮影および管理することができます。
- Moving Helicopter - 静止した観測者の周りを飛ぶヘリコプターをシミュレートします。デモを最適に実行するには、ヘッドフォンを使用してください。シミュレーションでは、ヘリコプターの大きさ、ドップラー効果を使用するかどうか、観測者の向き (たとえば、正面を向いて立っているか、うつ伏せになっているか) など、多数のパラメータを制御できます。
- Music Effects - Advanced Multimedia Supplements の高度なオーディオ機能を示します。オーディオファイルのループ再生中に、ボリューム、パン、イコライザー設定、残響、およびコーラス設定を調整できます。

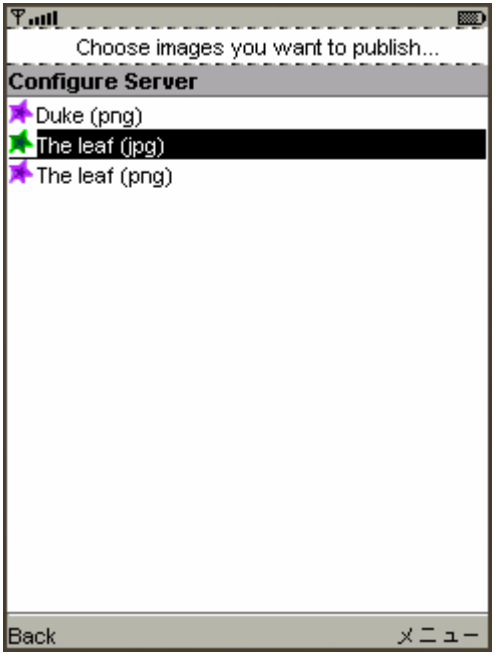
A.4 Bluetooth Demo

このアプリケーションには、JSR 82 の Bluetooth API の使用方法をデモする MIDlet が含まれています。

プロジェクト BluetoothDemo では、Bluetooth を使用してデバイス間で画像を転送する方法を示します。このデモアプリケーションの動作を確認するには、エミュレータインスタンスを 2 つ実行する必要があります。

1 つ目のエミュレータで Bluetooth Demo を起動し、「Server」を選択します。Bluetooth 接続を許可するかどうかを尋ねるメッセージが表示されます。「Yes」を選択します。サーバーが起動し、画像のリストが表示されます。最初は、どの画像も Bluetooth ネットワークで使用できないようになっています。画像を使用できるようにするには、画像を選択し、メニューから「Publish image」を選択します (または 1 を入力するかクリックします)。アイコンの色が紫から緑に変化し、画像が公開されたことを示します。

図 A-3 Bluetooth Demo のサーバーの実行



2 つ目のエミュレータで Bluetooth Demo を起動し、「Client」を選択します。MIDlet で画像を検索できる状態になり、そのメッセージが表示されます。「Find」ソフトウェアボタンをクリックします。もう一方のエミュレータが MIDlet で検出さ

れ、その画像リストが取得されます。リストから画像を 1 つ選択し、「Load」をクリックします。接続を許可するかどうかを尋ねるメッセージが表示されます。「Yes」を選択します。

- 信頼された保護ドメインでデモを実行している場合は、シミュレートされた Bluetooth を使用して画像が転送され、クライアントエミュレータに表示されます。
- 信頼された保護ドメインで実行していない場合は、1 つ目のエミュレータ (サーバー) に、クライアントからの接続を承認するかどうかを確認するメッセージが表示されます。「Yes」を選択します。クライアントエミュレータに画像が表示されます。

図 A-4　　シミュレートされた Bluetooth を使用して転送された画像

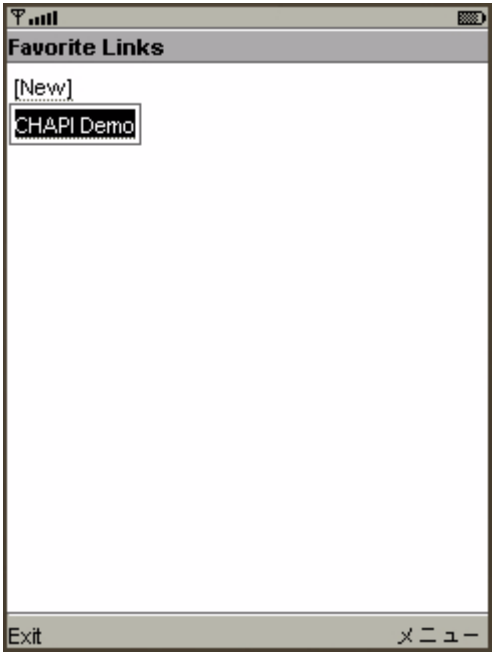
A.5 CHAPIDemo

CHAPIDemo はコンテンツブラウザです (JSR 211 を参照)。お気に入りのリストを保持して、さまざまなコンテンツを選択および表示できるようにします。

このデモでは、コンテンツハンドラレジストリを使用します。したがって、「実行」ボタンを使用する場合は、すべての機能を確認できません。代わりに、「プロジェクト」->「OTA 経由で実行」を使用して、アプリケーションをエミュレータにインストールします。この手順がわからない場合は、2-9 ページの 2.3.2 節「インストール」を参照してください。

インストールした CHAPIDemo は、アプリケーションリストに「Text Viewer」の名前で表示されます。これは、プレーンテキストのコンテンツハンドラである MIDlet です。「Text Viewer」を選択して、ソフトウェアボタンメニューから「起動」を選択します。お気に入りのリンクのリストが表示されます。

図 A-5 CHAPIDemo でのお気に入りのリンクの表示



ナビゲーションキーを使用して「CHAPIDemo」を強調表示し、エミュレータの「SELECT」を押します。通信時間を使用するかどうかを確認するメッセージが表示されます。「はい」ソフトウェアボタンを押します。さまざまな種類のコンテンツのリストが表示されます (図 A-6)。

図 A-6　　コンテンツリスト

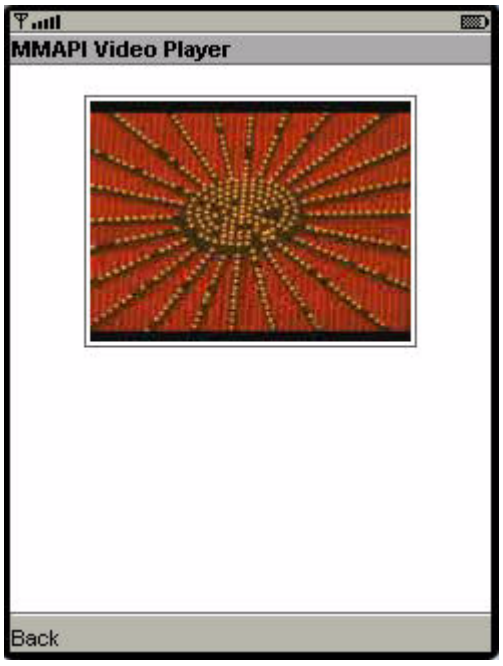
まず、`Duke.png` のいずれかを選択してみます。矢印キーを使用してリンクを強調表示し、「SELECT」を押してファイルを表示します。CHAPI を使用して、`ImageViewer` MIDlet が起動され、コンテンツが表示されます。

図 A-7　　PNG 画像の表示

別の種類のコンテンツでは、別のコンテンツハンドラ MIDlet スイート (MediaHandler) が必要です。このスイートを CHAPIDemo からインストールするには、`MediaHandler.jad` のリンクを選択します (図 A-6 に示すリストの最初の項目)。AMS が呼び出され、インストールの手順が実行されます。

MIDlet スイートをインストールすると、Text Viewer に表示されるほかの種類のコンテンツを表示できます。たとえば、「`http:video/test-mpeg.mpg`」を選択すると、図 A-8 に示す画像を含む、一連の画像が表示されます。

図 A-8　MediaHandler を使用した MPEG ムービーの表示



TextViewer および ImageViewer MIDlet のコンテンツハンドラの設定を確認するには、「属性設定」をクリックし、「コンテンツハンドラ」アイコンをクリックします。MediaHandler プロジェクトも調べてみてください。

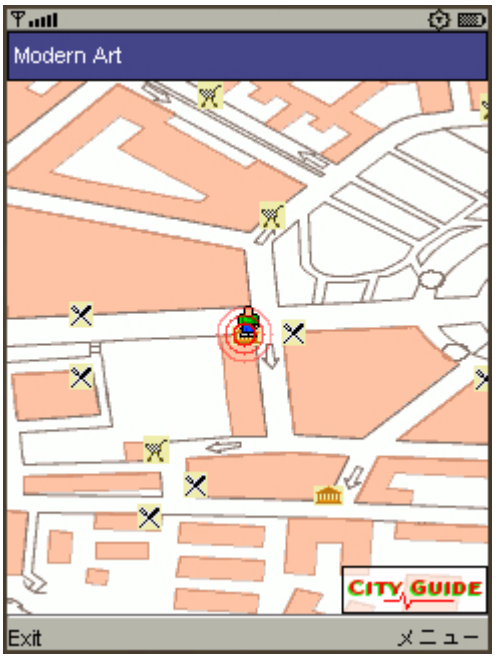
A.6　CityGuide

CityGuide は、Location API (JSR 179) の使用方法を示します。このデモは、歩行者の現在位置を市街図に重ねて表示します。歩行者が市内を移動すると、歩行者の近くにあるランドマークが強調表示および識別されます。このデモでは、歩行者の位置情報を、`citywalk.xml` という名前の XML スクリプトから取得しています。このイベントファイルは、デバイスの位置情報を送信します。詳細については、第 13 章を参照してください。

位置情報の入力要求が頻繁に発生するため、このデモは manufacturer (trusted) モードでの実行が適しています。manufacturer モードについては、6-3 ページの 6.2.1 節「MSA 保護ドメイン」を参照してください。ユーザーインタフェースで、「編集」->「環境設定」を選択し、「セキュリティー」を選択します。「セキュリティードメイン」で「Manufacturer」を選択します。

CityGuide プロジェクトを開いて実行します。エミュレータで、CityGuide MIDlet を起動します。「Next」をクリックして、地図ページを表示します。

図 A-9　　現在位置



エミュレータウィンドウのメニューから、「MIDlet」->「外部イベント」を選択します。「位置情報」タブで、参照ボタンをクリックします。
workdir¥apps¥CityGuide¥citywalk.xml のイベントファイルを選択します。

ウィンドウ下部にあるプレーヤーのボタンがアクティブになります。図 13-1 を参照してください。緑色の再生ボタン (右向きの三角形) を押して、スクリプトを実行します。

画面には、レストラン、美術館、店舗、および劇場の 4 種類のランドマークが表示されます。ランドマークの表示を調整するには、ソフトウェアボタンメニューを開き、「Settings」コマンドを選択します。図 A-10 を参照してください。ナビゲーションキーを使用してカテゴリを強調表示し、「SELECT」を使用して項目にチェックマークを付けるか、チェックマークを解除します。

地図上で強調表示されたランドマークの近くに来たときに、ソフトウェアボタンメニューを開き、「Detail」コマンドを選択して詳細情報を表示します。位置情報スクリプトの詳細については、第 13 章を参照してください。

図 A-10　位置情報の設定

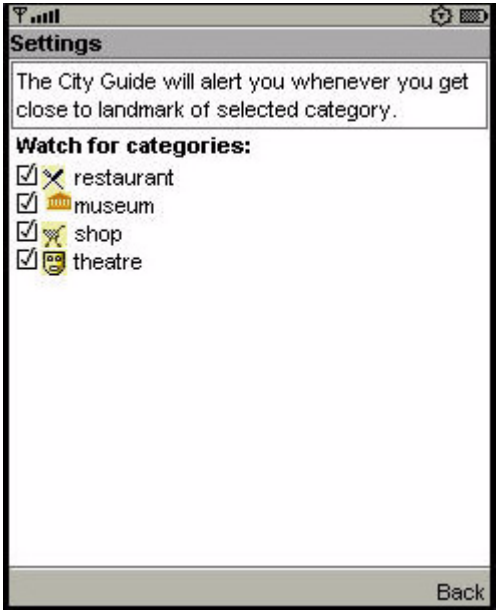
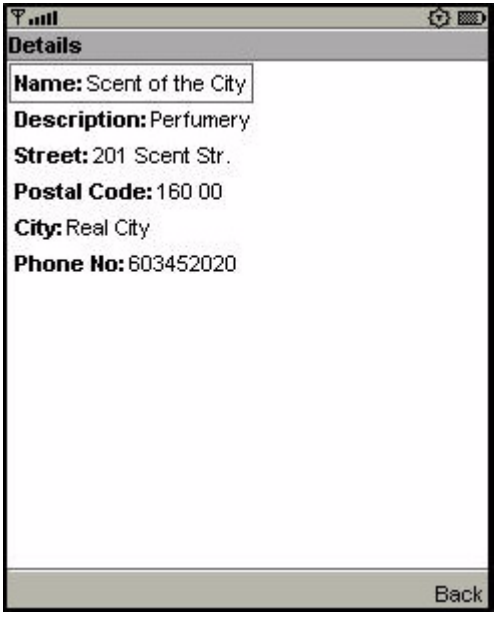


図 A-11　ランドマークの詳細



A.7 Demos

このデモには、さまざまな MIDP 機能を示すいくつかの MIDlet が含まれています。

A.7.1 Colors

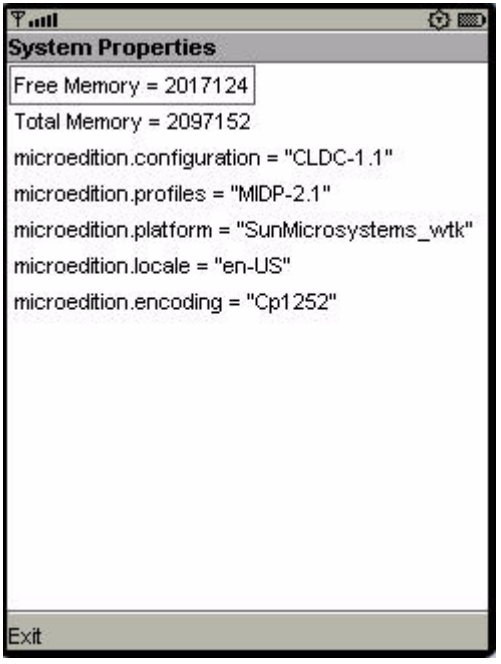
このアプリケーションでは、画面幅いっぱいの、大きな横長の矩形が表示されます。その下に、縦長の小さい 10 個の矩形が水平方向に表示されます。最後に、3 つの横長のカラーバーに青、緑、および赤 (RGB) の値が示されます。最初のメニューの選択に基づいて、値は 10 進数 (0 ～ 255) または 16 進数 (00 ～ ff) で表示されます。

- 変更する縦長のバーを選択するには、ナビゲーションの上矢印キーを使用してカラーバーに移動します。ナビゲーションの右矢印キーを使用して、カラーバーを強調表示します。大きな矩形が選択したバーの色になります。
- 上または下方向の選択矢印キーを使用して、変更する値 (赤、緑、または青) を選択します。左または右方向の矢印キーを使用して、選択した値を増減します。2 番目のメニュー項目で、増分を 4 (Fine) または 32 (Coarse) ステップに設定できます。
- 任意またはすべての縦長のバーの色を変更できます。

A.7.2 Properties

この MIDlet はプロパティーの値を表示します。図 A-12 に例を示します。

図 A-12　システムプロパティー



A.7.3 Http

このテストアプリケーションは、HTTP 接続を使用して Web ページを要求します。要求は、HTTP プロトコルの GET または POST メソッドを使用して発行されます。HEAD メソッドが使用された場合は、要求から head プロパティーが読み取られます。

デモ実行の準備

最初に、次の設定を確認します。

- 「環境設定」->「セキュリティー」に移動します。ポリシーを「JTWI」に、ドメインを「maximum」に設定します。
- 「環境設定」->「ネットワーク構成」で、HTTP のバージョンが 1.1 である必要があります。
- ファイアウォール内で実行している場合は、「環境設定」->「ネットワーク構成」を選択し、プロキシサーバーの情報を指定します。
- ウイルス対策ソフトウェアを実行している場合は、この MIDlet で特定の Web サイトとの接続を行えるように、ルールの作成が必要になることがあります。

デモの実行

Http MIDlet を起動します。デフォルト設定では、MIDlet は `http://www.yahoo.com` に接続を試みます。テストするには、「メニュー」ソフトウェアボタンを選択して 1、2、または 3 を選択し、URL をテストします。

Http Test では、取得できる情報が返されます。表示される情報が一画面に収まらない場合は、下矢印キーを使用して末尾までスクロールできます。情報の量は、要求の種類と、ページにある `META` 情報の量によって異なります。ボディ情報またはコンテンツを提供するには、ページで `CONTENT-LENGTH` を宣言している必要があります。これについては、RFC 2616 で説明されています。

メニューオプションの使用

「メニュー」ソフトウェアボタンを使用して、次の操作を実行できます。

- 1 を選択すると、選択したページから `GET` メソッドによって情報を取得します。
- 2 を選択すると、選択したページから `POST` 情報を取得します。
- 3 を選択すると、ページの `HEAD` 属性が表示されます。
- 4 を選択すると、Web ページの現在のリストが表示されます。新しいページを選択したり、リストに独自のページを追加したりできます。新しい URL を指定するには、「メニュー」ソフトウェアボタンで 4 を選択します。画面には `http://` が表示されます。URL の残りを入力し、末尾にスラッシュ (/) を入力する必要があります。たとえば、`http://www.internetnews.com/` のように指定します。「OK」ソフトウェアボタンを押します。「Http Test」画面に新しい URL が表示され、操作を要求するメッセージが表示されます。

A.7.4 FontTestlet

この MIDlet は、利用できるさまざまなフォント (プロポーショナル、標準、標準イタリック、ボールド、およびボールドイタリック) を表示します。メニューで 1 または 2 を選択すると、システムフォント (Sans Serif) とモノスペースフォントが切り替わります。

A.7.5 Stock

Http デモと同様、このサンプルは HTTP 接続を使用して情報を取得します。A-16 ページの A.7.3 節「Http」の手順と同じ準備を行います。

Demos プロジェクトを実行し、Stock MIDlet を起動します。

デフォルト設定では、画面の上部に空のティッカーバーが表示されます。ティッカーの下のメニューリストに、Stock Tracker、What If?、Alerts、および Settings の 4 つのアプリケーションが表示されます。最初の 3 つのアプリケーションを使用するには、株式シンボルを追加する必要があります。

A.7.5.1 Settings の使用

アプリケーションの機能を使用するには、アプリケーションで操作される株式シンボルを指定する必要があります。

ティッカーへの株式シンボルの追加

ティッカーに株式シンボルを追加するには、ナビゲーションの矢印キーを使用して「Settings」を選択します。

「Add Stock」を選択します。

株式シンボルの入力を求めるメッセージが表示されます。`SUNW` を入力し、「Done」ソフトウェアボタンを選択します。追加した株式とその現在値がティッカーに表示されます。さらに、IBM や HPQ などの株式シンボルも追加します。

更新間隔の変更

デフォルト設定では、更新間隔は 15 分です。「Updates」を選択して間隔を変更します。ナビゲーションの矢印キーを使用して、「Continuous」、「15 minutes」、「30 minutes」、「1 hour」、または「3 hours」のいずれかを選択します。「Done」ソフトウェアボタンを選択します。

株式の削除

「Remove a Stock」を選択します。追加してある株式のリストが表示されます。ナビゲーションキーを使用して、削除する 1 つまたは複数の株式を選択します。「Done」ソフトウェアボタンを選択します。

A.7.5.2 Stock Tracker

Stock Tracker には、追加した株式のリストとその現在の値が表示されます。選択した株式の追加情報も表示されます。たとえば、終値や高値と安値などが表示されます。

株式を選択して、「SELECT」を押します。

A.7.5.3 What If?

What If? アプリケーションでは、元の購入価格と保有株数の入力を求められます。アプリケーションは、現在の価格に基づいて利益と損失を計算します。

株式シンボルを選択します。

購入価格と株数を入力して、「Calc」を押します。

A.7.5.4 Alerts

このアプリケーションは、価格がユーザーの指定した値になったときに通知を送信します。

メインメニューから、「Alerts」を選択します。

「Add」を選択します。

株式を選択します。画面に、「`Alert me when a stock reaches`」というメッセージが表示されます。整数を入力します。

アラートは「Current Alerts」リストに表示されます。アラートを削除するには、「Remove」を押してアラートを選択します。「Done」ソフトウェアボタンを選択します。

指定した値になると、警告音が鳴り、メッセージが表示されます。たとえば、「*Symbol* has reached your price point of $*value* and is currently trading at $*current_value*」のようなメッセージになります。アラートは起動されると、「Current Alerts」リストから削除されます。

A.7.6 Tickets

このデモは、オンラインのチケットオークションアプリケーションの動作を示します。ホーム画面の一番上に沿ってチケットティッカーが表示されます。デフォルト設定では、「Choose a Band」フィールドに「Alanis Morrisette」が表示されます。

バンドを選択するには、バンド名を強調表示して「SELECT」を押します。下矢印キーを使用して別のバンド (たとえば、moby) を強調表示して、「SELECT」を押します。利用可能なオークションが表示されます。

入札するには、「メニュー」ソフトウェアボタンを選択して 2 を選択します。矢印キーを使用して、フィールドを移動します。各フィールドに必要な情報を入力します。「Next」ソフトウェアボタンを選択します。入札を確認するメッセージが表示されます。矢印キーを使用して「Submit」を強調表示し、「SELECT」を押します。確認番号が表示されます。「Bands」をクリックして、開始画面に戻ります。

アラートを設定するには、「メニュー」ソフトウェアボタンを選択して 3 を選択します。ナビゲーションの矢印キーを使用してフィールドを移動し、現在の入札額よりも高い値を入力します。「Save」ソフトウェアボタンを選択します。開始ページに戻ります。アラートの値を超える入札を行うと、アラートが起動されます。設定により、アプリケーションが更新を確認する頻度が決定されます。したがって、アラートが鳴るまでに数分かかる場合があります。

バンドを追加するには、「メニュー」ソフトウェアボタンを選択して 4 を選択します。バンド名を 1 つ入力するか、コンマ区切りの名前リストを入力します。「Save」ソフトウェアボタンを選択します。確認後、開始ページに戻ります。「Choose a Band」ドロップダウンメニューに、追加したバンドが表示されます。

注 – これはデモです。バンドを完全に記述するには、*workdir*¥apps¥Demos¥src¥example¥auction¥NewTicketAuction.java ファイルを編集する必要があります。

バンドを削除するには、「メニュー」ソフトウェアボタンを選択して 5 を選択します。バンド名に移動し、「SELECT」を選択してチェックボックスにチェックマークを付けます。複数のバンドを選択できます。「Save」ソフトウェアボタンを選択します。

ティッカーの表示、更新、アラートの音量、および日付についての現在の設定を表示するには、「メニュー」ソフトウェアボタンを選択して 6 を選択します。必要な場合は、矢印キーと選択キーを使用して、これらの値を変更します。「Save」ソフトウェアボタンを選択します。

A.7.7 ManyBalls

この MIDlet では、画面を移動する 1 つのボールが表示されます。上および下矢印キーを使用すると、ボールのスピード (fps) が増減します。右または左矢印キーを使用すると、ボールの数が増加または減少します。

A.8 Demo3D

このアプリケーションには、エミュレータでの JSR 184 「Mobile 3D Graphics API」への対応を示す 3 つの MIDlet が含まれています。

A.8.1 Life3D

Life3D は、よく知られている Game of Life の 3 次元版です。生きているセルは立方体で表されます。各セルには、斜め方向も含めて、最大 26 個の隣接するセルが存在します。アニメーションの各ステップで、隣接するセルが 4 個に満たないセルは孤独のために死に、隣接するセルが 5 個を超えるセルは過密のために死にます。隣接するセルがちょうど 4 個ある場合、空のセルは生きたセルに変わります。

ゲームボードは、すべての角度から見えるようにゆっくり回転します。

図 A-13 3 次元の Game of Life

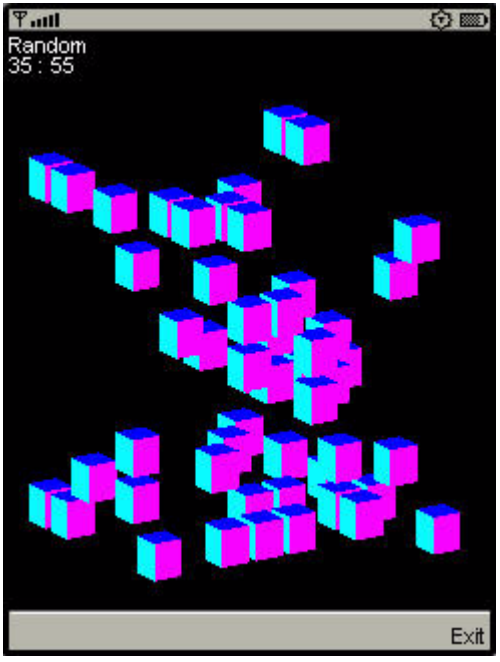


表 A-2 に示すキーパッドのボタンを使用してゲームを制御できます。

表 A-2 `Life3D` のコントロール

ボタン	説明
0	アニメーションを一時停止します。
1	アニメーションをデフォルトの速度で再開します。
2	アニメーションを速くします。
3	アニメーションを遅くします。

表 A-2　Life3D のコントロール (続き)

ボタン	説明
4	任意のリストから 1 つ前の事前設定を選択します。画面の上部に設定の名前が表示されます。
5	リストから次の事前設定を選択します。
*	ランダムな設定を生成し、安定するか死ぬまでアニメーション表示します。死んだ場合は、新しいランダム設定を生成します。

このサンプルのソースコードには、特に詳しい解説が付いています。*toolkit*¥apps¥Demo3D¥src¥com¥superscape¥m3g¥wtksamples¥life3d¥Life3D.java を参照してください。

A.8.2 PogoRoo

PogoRoo では、ホッピングに乗って飛び跳ねているカンガルーが表示されます。カンガルーは矢印キーで操作できます。上矢印キーで前進、下矢印キーで後退、左矢印キーと右矢印キーで方向変換させることができます。効果が現れるまでキーを押し続ける必要があります。

図 A-14　飛び跳ねているカンガルー

A.8.3 retainedmode

retainedmode MIDlet では、疲れ知らずのスケートボーダーを永久ループで表示するシーンファイルが再生されます。

図 A-15　疲れ知らずのスケートボーダー

A.9 GoSIP

GoSIP は、SIP (JSR 180) を使用するチャットアプリケーションです。SIP プロキシサーバーと登録機関を使用して通信を設定します。

最初に SIP サーバーを実行します。「ファイル」->「ユーティリティー」を選択します。「SIP サーバーを起動」を選択し、「起動」を押します。SIP プロキシサーバーウィンドウが表示されます。「開始」をクリックして、サーバーを実行します。

次に、エミュレータの 2 つのインスタンスで GoSIP アプリケーションを実行します。

図 A-16　SIP プロキシと登録機関

1 つ目のエミュレータで、Sippy A を起動します。プロキシホストの入力を求めるメッセージが表示されたら、ローカルマシン名または IP アドレスを入力します。「Next」を選択し、次に「Register」を選択します。SIP サーバーウィンドウには、エミュレータから送信された SIP メッセージが表示されます。登録されたユーザーのリストに `Sippy A` が表示されます。エミュレータから、通信先の Sippy B を招待するようにメッセージが表示されます。ここでは、まだ招待しません。

2 つ目のエミュレータで、Sippy B を起動します。前の手順と同じように、SIP プロキシのアドレスを入力し、「Next」および「Register」を選択します。Sippy B ユーザーが、SIP サーバーウィンドウに表示されます。

1 つ目のエミュレータで「Invite」を選択します。2 つ目のエミュレータに、呼び出し中であることが表示されます。「Answer」を選択して、チャットを開始します。どちらのエミュレータにも、「Talking」画面が表示されます。「Send」コマンドを使用してメッセージをやり取りできます。

メッセージのやり取りが終わったら、「Bye」を選択してチャットを終了します。

A.10 i18nDemo

この MIDlet スイートは、JSR 238 の Mobile Internationalization API を示します。String Comparator と Formatter の MIDlet は、異なるロケールで文字列のソートと数字の表示を適切に行う方法を示しています。3 つ目の MicroLexicon の MIDlet は、プラハ、ヘルツェリア、北京、ミラノ、またはその他の地域でビールを注文するときに便利な、慣用句の簡単な翻訳機能です。

注 – Sun Java™ Wireless Toolkit for CLDC のデフォルトのフォントでは、中国語と日本語はサポートされていません。これらの言語を使用するには、デモを実行する前に次の手順を実行してください。

1. 中国語または日本語をサポートする True Type フォントをインストールします。
2. *toolkit*¥wtklib¥devices¥*skin-directory*¥*skin*.properties を変更して、True Type フォントを指定します。

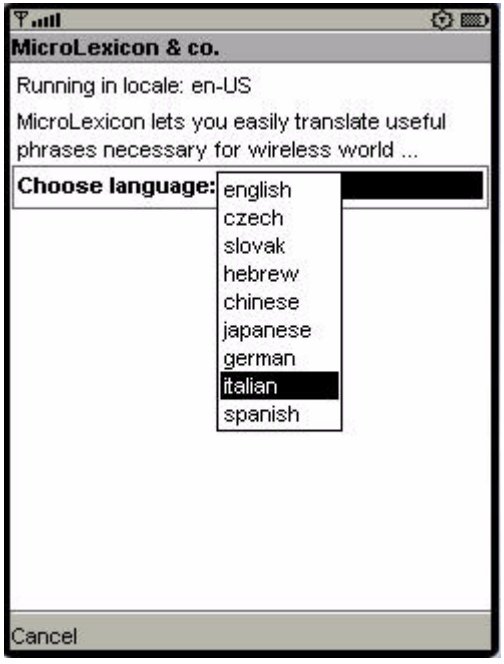
MIDlet を実行するには、「SELECT」ボタンを使用して MIDlet を強調表示し、画面右下のボタンを使用して MIDlet を起動します。

String Comparator MIDlet は、ロケールによる文字列 (都市名) のソート方法の違いを示します。MIDlet を起動します。画面右下のボタンを使用して、メニューを表示します。2 をクリックまたは入力して、「Sort - default」を選択します。リストがアルファベット順にソートされます。3 をクリックするか入力して、「Sort - slovak」を選択します。上にマークが付いた Z、またはマークの付いていない Z で始まる都市名で、ソートの違いを簡単に確認できます。「Exit」をクリックして、MIDlet のリストに戻ります。

2 つ目の Formatter の MIDlet は、時刻と数字をロケールに応じた形式で表示します。「Next」をクリックしていくと、4 つの画面がすべて表示されます。「Exit」をクリックして、MIDlet のリストに戻ります。

最後の MicroLexicon の MIDlet は、慣用句を別の言語に翻訳します。目標言語をリストから選択するには、ナビゲーションの矢印キーを使用して「Choose Language」を強調表示します。「SELECT」をクリックし、言語のドロップダウンメニューを表示します。ナビゲーションの矢印キーを使用して言語を選択し (図 A-17 を参照)、「SELECT」をクリックします。

図 A-17　目標言語の選択

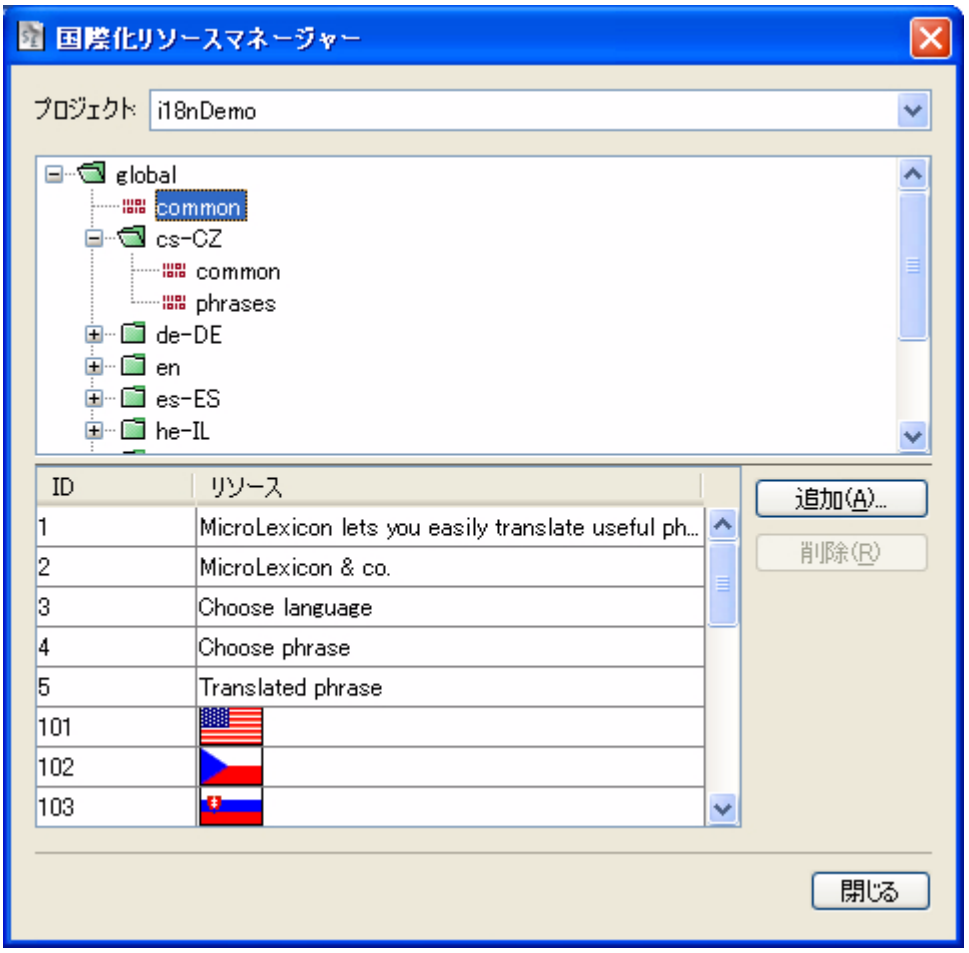


MicroLexicon では、語句のリストが表示されます。いずれかを強調表示して、エミュレータの「SELECT」ボタンを押します。目標言語の国旗と、翻訳された語句が表示されます。

元の言語を変更するには、「編集」->「環境設定」を選択します。「国際化」タブをクリックし、有効なロケール文字列を入力します。次にエミュレータおよび MicroLexicon を実行したときに、指定したロケールでテキストが表示されます (ロケールがサポートされている場合)。ロケールの例として `cs-CZ` があります。

MicroLexicon では、MIDlet リソースが使用されています。ツールキットを使用してアプリケーションをローカライズする方法を理解するには、「プロジェクト」->「国際化リソースマネージャー」を選択します。MicroLexicon で使用されている、テキストと画像のすべてのリソースが表示されます。リソースを編集してから MicroLexicon を実行するとどうなるかを確認できます。リソースは実行時にロードされるため、アプリケーションを再度ビルドする必要はありません。

図 A-18　国際化リソースマネージャー



リソース自体は *workdir*¥apps¥i18nDemo¥res¥global に保存されます。

A.11　JBricks

JBricks は、JSR 229 の Payment API の使用方法をデモするゲームです。ゲーム自体はブレイクアウトやアルカロイドなどのブロック崩しに似ています。JBricks では、追加ライフや新しいゲームレベルを購入できます。バックグラウンドで、Payment API が細部を処理しています。

JBricks の支払い機能を使用するには、「プロジェクト」->「OTA 経由で実行」を使用して、JBricks をエミュレータにインストールします。この手順がわからない場合は、2-9 ページの 2.3.2 節「インストール」を参照してください。

JBricks が Payment API をどのように使用しているかを確認するには、ゲームのメインメニューから「Buy Life」または「Buy Level」を選択します。続いて、ライフを 1 つ購入するか、割引価格で 3 つ購入するかを選択します。次の画面で、支払いの種類を選択できます。

図 A-19　支払いの種類の選択

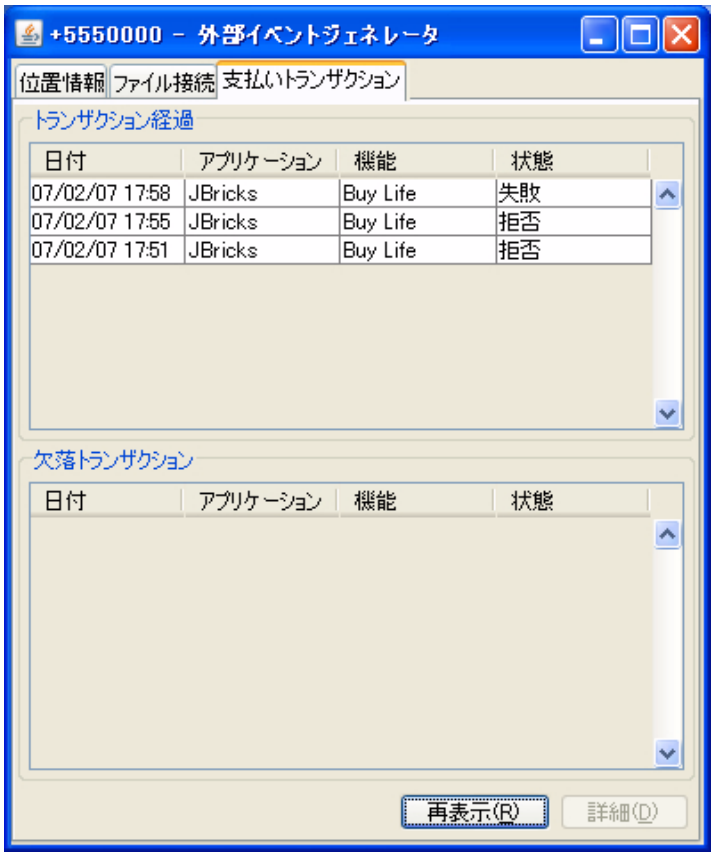
ナビゲーションの矢印キーを使用して、「Pay by」で始まる行を選択します。「SELECT」ボタンをクリックして、ドロップダウンメニューで使用可能なクレジットカードアダプタを確認します。ナビゲーションの矢印キーを使用して VISA アダプタを選択し、「SELECT」をクリックします。画面右下の「はい」をクリックして、処理を続行します。

続いて、クレジットカード情報を入力できます。有効な VISA カード番号 (たとえば、4111111111111111) と有効期限を入力します。

図 A-20　支払い情報の入力

エミュレータの現在のインスタンスに関するトランザクションを表示するには、「MIDlet」->「外部イベント」を選択し、「支払いトランザクション」タブをクリックします。エミュレータの特定のインスタンスに関するトランザクションが表示されます。

図 A-21　トランザクションの表示



また、ツールキットの支払いシステムを通過するすべてのトランザクションを表示できます。「ファイル」->「ユーティリティー」を選択し、「支払いコンソール」を選択します。コンソールのトランザクションは次のようになります。

```
PSP Console running, using phone number +5550001.
PSP Server running at https://localhost:-1
Received Payment Request from 127.0.0.1
   Credit card issued by: VISA
   Credit Card type: 0
   Credit Card Number: 4111111111111111
   Credit Card Holder: Jonathan Knudsen
   Feature ID: 3_lives
   Credit Card Verification Number (CCV): 123
   Payload: null
Response to 127.0.0.1
HTTP/1.1 200 OK
```

```
Content-Length: 0
Pay-Response: SUCCESSFUL
Pay-Timestamp: 1156282954734
```

A.12 JSR172Demo

JSR172Demo は、MIDlet から Web サービスにアクセスする方法を示します。Web サービスは、すでにインターネットサーバーで実行中です。ファイアウォール内にいる場合は、エミュレータのプロキシサーバーを設定する必要があります。「編集」->「環境設定」を選択し、「ネットワーク構成」を選択します。プロキシサーバーのアドレスファイルとポート番号をフィールドに入力します。サンプルをビルドして実行します。

JSR172Demo には、Server Script という MIDlet だけが含まれています。この MIDlet を起動し、プロンプトに従って操作します。Web サービスから取得されたシミュレーション用のニュース見出しを閲覧できます。

バックグラウンドの処理を確認するには、ネットワークモニターを使用します。

A.13 MobileMediaAPI

MobileMediaAPI アプリケーションには、ツールキットのマルチメディア機能を示す 4 つの MIDlet が含まれます。この節では、これらの MIDlet について説明し、アプリケーションでマルチメディアを使用するための追加情報を示します。

A.13.1 Simple Tones

Simple Tones サンプルは、対話型合成トーンの使用方法をデモします。サンプルを選択し、画面右下の「Play」をクリックします。

- Short Single Tone と Long Single Tone は、`Manager.playTone()` を使用して高さの異なる発信音を再生します。
- Short MIDI イベントは、対話型 MIDI デバイス (ロケータ `"device://midi"`) で和音を再生します。`MIDIControl` の `shortMidiEvent()` メソッドは、和音を鳴らすために使用されます。
- MMAPI Drummer デモを実行するには、数字キー (0 ～ 9) をクリックするか入力します。それぞれの数字キーで、異なるサウンドが再生されます。

A.13.2 Simple Player

Simple Player アプリケーションは、エミュレータのさまざまなオーディオおよびビデオ機能をデモします。このアプリケーションには、さまざまな形式のサンプルファイルが含まれています。また、エミュレータの持続ストレージや HTTP URL にあるファイルを再生できます。

プレーヤー部分では、汎用の `javax.microedition.media.Player` インタフェースが使用されます。プレーヤーには、継続時間、メディアの時間、およびメディアファイルを実行するためのコントロールが表示されます。ファイルにメタデータが含まれている場合は、作成者やタイトルなどの情報もプレーヤーで表示できます。MIDI ファイルの場合、カラオケ用テキストがファイルに含まれているときは、再生中にそのテキストが画面に表示されます。グラフィカルユーザーインタフェースのコントロールがある場合は、それらが画面に表示されます。これらのコントロールを使用するには、Simple Player でメディアサンプルの 1 つを選択し、「メニュー」ボタンをクリックします。表示されるコマンドの一覧から、目的のコマンドを選択します。

「Simple Player」を選択し、「起動」をクリックします。このデモアプリケーションには、次のメディアサンプルが含まれています。

- Bong は、短い WAV ファイルを再生します。いくつかの再生機能は調整可能です。これについては、このマニュアルで後述します。画面には、サウンドの長さが「分:秒.1/10 秒」という形式で表示されます。たとえば、00:17.5 のようになります。このオーディオサンプルは、MIDlet スイートの JAR ファイルに含まれているリソースファイルです。
- MIDI Scale は、サンプル音階を再生します。画面には、選択したミュージックファイルのタイトル、曲の長さ、再生の経過時間、および現在のテンポ (bpm 単位) が表示されます。この MIDI ファイルは、MIDlet スイートの JAR ファイルに格納されています。
- Simple Ring Tone は、ベートーベンの交響曲第 5 番の一節を再生します。画面には、選択したミュージックファイルのタイトル、曲の長さ、再生の経過時間 (秒および 1/10 秒単位)、および現在のテンポ (bpm 単位) が表示されます。この着信音ファイル (.jts 形式) は、MIDlet スイートの JAR ファイルに格納されています。
- WAV Music は、短いオーディオファイルを再生します。画面には、選択したオーディオファイルのタイトル、オーディオの長さ、再生の経過時間、および再生レート (% 単位) が表示されます。この WAV ファイルは HTTP サーバーから取得されます。
- MIDI Scale は、HTTP サーバーから MIDI ファイルを取得して再生します。
- Animated GIF サンプルは、1 から 5 までカウントするアニメーション GIF を表示します。このファイルは、MIDlet スイートの JAR ファイルに格納されています。
- デフォルトデバイスから Audio Capture を使用すると、マイクロフォンや接続されたデバイスからオーディオを取り込むことができます。取り込まれたサウンドはスピーカーで再生されます。ハウリングを防ぐには、ヘッドセットを使用してください。

- Video Capture Simulation は、カメラ付きデバイスで可能なビデオ入力の表示をシミュレートします。
- MPEG1 Video [http]。`http://java.sun.com/products/java-media/mma/media/test-mpeg.mpg` にある MPEG ビデオを再生します。
- `[enter URL]` を使用すると、任意の HTTP サーバーにあるメディアファイルを再生できます。ファイルを再生するには、挿入ポイントに有効な URL (たとえば、`http://java.sun.com/products/java-media/mma/media/test-wav.mpg`) を入力して、「OK」をクリックします。メディアを選択するために HTTP ディレクトリを開く場合は、URL の末尾にスラッシュを追加してください。

さらに、Simple Player は、RTTTL (Ringing Tones text transfer language) 形式の着信音の構文解析を行うことができます。RTTTL については、`http://www.convertyourtone.com/rtttl.html` を参照してください。

Simple Player には、メディアの再生を制御する一般的なコマンドが用意されています。コマンドは Simple Player のメニューから使用でき、一部のコマンドにはキーパッドのボタンが割り当てられています。次の表では、これらのコマンドについて説明します。

表 A-3 Simple Player のコマンド

コマンド	キー	説明
Mute/Unmute	0	ファイルの再生を継続したまま、サウンドをオフにします。このコマンドは「Unmute」と切り替わります。
Volume	* と #	音量を増減します。
META data		著作権情報、タイトル、トラックリストなど、メディアファイルで提供されている情報を表示します。
Stop in 5 seconds		オーディオの再生中にこれを設定すると、5 秒後に再生を一時停止します。
Rate	4 と 6	再生速度のレートを変更します。
Tempo		トーンシーケンスまたは MIDI ファイルのテンポを増減します。
Pitch	UP と DOWN	MIDI ファイルの音符の高さを上下します。
Start Recording/Stop Recording		オーディオの再生を録音します。録音したオーディオを含むファイルが、エミュレータが実行されているディレクトリに作成されます。ファイル名を指定しない場合は、recording.wav というファイルが作成されます。このコマンドは「Stop Recording」と切り替わります。
Step Frame	7 と 9	一度に 1 フレームずつ、ビデオファイル内を前後に移動します。
Play/Stop	2 と SELECT	メディアを開始または停止します。

表 A-3　Simple Player のコマンド (続き)

コマンド	キー	説明
Loop Mode		オーディオファイルの再生が終わるとすぐに再生を繰り返します。「Loop Mode」を 1 回クリックすると、オーディオファイルは 1 回だけ再生されます。もう 1 回クリックすると、ファイルが 3 回再生されます。3 回クリックすると、ファイルが繰り返し再生されます。4 回目のクリックで、単一再生に戻ります。
Skip	1 と 3	メディアファイルの長さの 5 パーセントだけ、前後にスキップします。サウンドトラックはビデオに同期します。
Rewind		オーディオ再生の開始位置に戻ります。
Stop and Rewind	5	再生を停止し、開始位置に巻き戻します。
Quick Help		コマンドとキーパッドボタンを一覧表示します。

これらのコマンドを使用できるかどうかは、Simple Player で再生中のメディアの種類に依存します。また、一部のコマンドは、キーパッドのボタンを使って呼び出すことができます。次の表では、コマンドの使用の可否、対応するキーパッドボタン、および MMAPI の関連クラスについて説明します。

コマンドおよび対応するキーパッドボタンの簡易リストは、Simple Player アプリケーション自体にも用意されています。メニューから「Quick Help」を選択してください。

A.13.3 Video

Video アプリケーションは、エミュレータでアニメーション GIF の再生やビデオの取り込みを行う方法を示します。カメラを備えた実際のデバイスでは、ビデオの取り込みを使用して、カメラに写っているものを確認できます。

アニメーション GIF やビデオの取り込みは、`Form` `Item` または `Canvas` を使用して実装できます。Video デモアプリケーションには、可能な組み合わせがすべて含まれています。Animated GIF - Form [jar] は、アニメーション GIF をフォームアイテムとして表示します。フォームには、現在の時間など、再生に関する情報が含まれています。「Snapshot」コマンドを選択すると、実行中のアニメーションのスナップショットを作成できます。スナップショットは、アニメーション GIF に続いてフォームに配置されます。

- Animated GIF - Canvas [jar] は、アニメーション GIF をキャンバスに表示します。アニメーションの進行状況が簡易インジケータに表示されます。「Snapshot」を選択すると、現在の表示の静止画像を取得できます。スナップショットがしばらく表示されたあと、アニメーションの表示に戻ります。

- Video Capture - Form は、カメラなどのソースからビデオを取り込み、それをフォームのアイテムとして表示する処理をシミュレートします。「Snapshot」コマンドを選択すると、取り込まれたビデオのスナップショットを作成できます。スナップショットは、ビデオの取り込みに続いてフォームに配置されます。
- Video Capture - Canvas は、カメラなどのソースからビデオを取り込み、それをキャンバスに表示する処理をシミュレートします。「Snapshot」を選択すると、現在の表示の静止画像を取得できます。スナップショットがしばらく表示されたあと、ビデオ取り込みの表示に戻ります。
- MPEG1 Video - Form、MPEG1 Video - Canvas

  MPEG1 アプリケーションは Web から MPEG を取得します。したがって、ファイアウォール内にいる場合は、エミュレータのプロキシサーバーを設定する必要があります。

  「編集」->「環境設定」を選択し、「ネットワーク構成」を選択します。「プロキシサーバーを使用」にチェックマークを付けます。プロキシサーバーのアドレスとポート番号をフィールドに入力します。このデモでは、「HTTP/1.0」を選択します。

  デモを再生するときに、WTK がデータを取得するまでに数秒かかります。
  MPEG1 デモは、それぞれ Video Capture - Form および Video Capture - Canvans と同じように動作します。

A.13.4 Pausing Audio Test

この MIDlet は、一時停止された MIDlet が実行中のプレーヤーを停止していない場合に、Sun Java™ Wireless Toolkit for CLDC が警告を生成する仕組みをデモします。MIDlet を起動したら、「Play」コマンドを選択してオーディオの再生を開始します。画面には、「Well-behaved」または「Not Well-Behaved」のいずれかの状態が表示されます。

エミュレータウィンドウのメニューから、「MIDlet」->「一時停止」を選択します。MIDlet が一時停止されます。ツールキットのコンソールにはメッセージは表示されません。エミュレータウィンドウのメニューから、「MIDlet」->「再開」を選択します。

ここで、「Misbehave」コマンドを選択します。再度 MIDlet を一時停止します。ツールキットのコンソールに、「`An active media (subtype Player) resource was detected while the MIDlet is paused. Well-behaved MIDlets release their resources in pauseApp()`」という警告が表示されます。

A.13.5 MobileMediaAPI の属性

MobileMediaAPI のアプリケーションには、次のような属性があります。これらの属性は、プロジェクト設定ダイアログボックスの「ユーザー定義」タブで変更できます。

表 A-4　MMAPI 固有の MIDlet 属性の説明

属性	説明
PlayerTitle-*n*	Simple Player MIDlet で再生される *n* 番目のメディアタイトルの名前です。
PlayerURL-*n*	Simple Player MIDlet で再生される *n* 番目のメディアタイトル (PlayerTitle-*n*) の場所です。
VideoTest-*n*	Video アプリケーションで再生される *n* 番目のメディアタイトルの名前です。
VideoTest-URL*n*	Video アプリケーションで再生される *n* 番目のメディアタイトル (VideoTest-*n*) の場所です。

A.14 Network デモ

このデモには、Socket Demo と Datagram Demo の 2 つの MIDlet があります。それぞれのデモでは、サーバーとクライアントの関係をエミュレートできるように、2 つのエミュレータインスタンスを実行する必要があります。

A.14.1 Socket Demo

エミュレータインスタンスを 2 つ実行します。一方をソケットサーバーとして実行し、もう一方をソケットクライアントとして実行します。

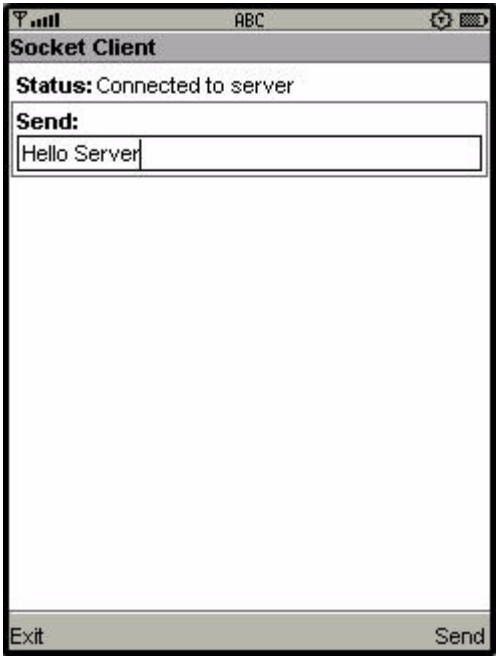
1 つ目のエミュレータで、アプリケーションを起動し、「Server peer」を選択します。「Start」を選択します。エミュレータから、デモでネットワークを通じてデータが送受信されることを示すメッセージが表示され、「Is it OK to use network?」という確認のメッセージが表示されます。「Yes」を選択します。ソケットサーバーにより、接続を待機していることを示す画面が表示されます。

2 つ目のエミュレータで、アプリケーションを起動し、「Client peer」を選択します。次に、「Start」を選択します。エミュレータから、デモでネットワークを通じてデータが送受信されることを示すメッセージが表示され、「Is it OK to use network?」という確認のメッセージが表示されます。「Yes」を選択します。ソケットクライアントにより、サーバーに接続されたことを示す画面が表示されます。ナビ

ゲーションの下矢印キーを使用して、「Send」ボックスを強調表示します。「Send」ボックスにメッセージを入力して、「Send」ソフトウェアボタンを選択します。

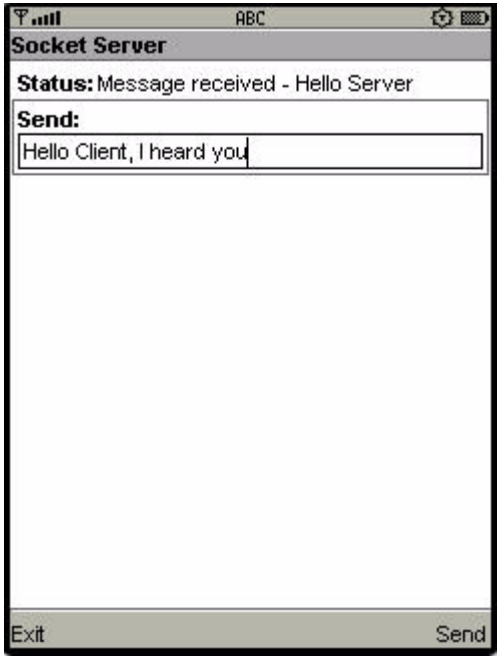
たとえば、クライアントで、「Send」ボックスに `Hello Server` と入力します(図 A-22 を参照)。「Send」ソフトウェアボタンを選択します。送信中、エミュレータでは青いライトが点灯します。

図 A-22　ソケットクライアントからのメッセージの送信



ソケットサーバーを実行しているエミュレータでは、「Status」に「`Message received - Hello Server`」と表示されます。下矢印キーを使用して「Send」ボックスに移動し、返信を入力します。たとえば、「`Hello Client, I heard you`」と入力します。「Send」を選択します。図 A-23 を参照してください。

図 A-23　サーバーでの受信メッセージと送信メッセージの表示



ソケットクライアントでは、「Status」にサーバーから受信したメッセージが表示されます。新しいメッセージを送信するまで、「Send」ボックスには送信済みのメッセージがそのまま表示されます。

A.14.2 Datagram Demo

このデモは Socket Demo に似ています。

エミュレータインスタンスを 2 つ実行します。一方をデータグラムサーバーとして実行し、もう一方をデータグラムクライアントとして実行します。

1 つ目のエミュレータで、Datagram Demo を起動し、「Server peer」を選択します。「Start」を選択します。エミュレータから、デモでネットワークを通じてデータが送受信されることを示すメッセージが表示され、「Is it OK to use network?」という確認のメッセージが表示されます。「Yes」を選択します。起動後、データグラムサーバーのステータスは「`Waiting for connection`」になり、「Send」ボックスは空です。

2 つ目のエミュレータで、Datagram Demo を起動し、「Client peer」を選択します。次に、「Start」を選択します。エミュレータから、デモでネットワークを通じてデータが送受信されることを示すメッセージが表示され、「Is it OK to use network?」という確認のメッセージが表示されます。「Yes」を選択します。データ

グラムクライアントのステータスは、「Connected to server」になります。ナビゲーションの下矢印キーを使用して、「Send」ボックスを強調表示します。「Send」ボックスにメッセージを入力して、「Send」ソフトウェアボタンを選択します。たとえば、「Hello datagram server」と入力します。

データグラムサーバーを実行しているエミュレータでは、「Status」に「Message received - Hello datagram server」と表示されます。下矢印キーを使用して「Send」ボックスに移動し、クライアントへの返信を入力します。

データグラムクライアントでは、「Status」フィールドにサーバーから受信したメッセージが表示されます。「Send」ボックスには、最後に送信したメッセージがそのまま表示されます。

A.15 ObexDemo

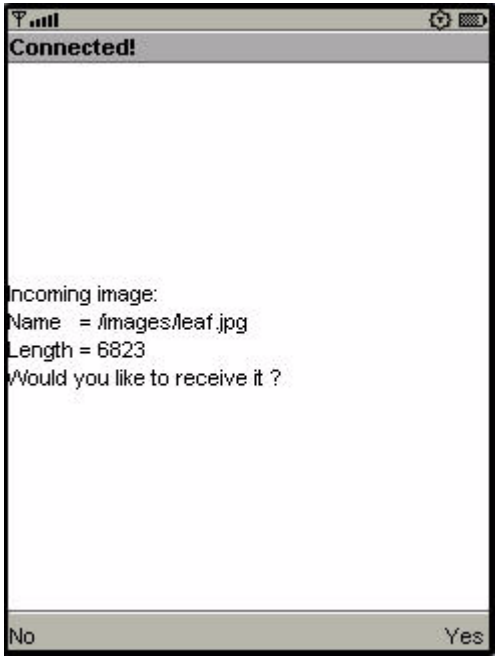
このアプリケーションは、OBEX API を使用してエミュレータインスタンス間で画像ファイルを転送する方法を示します。このデモアプリケーションは、シミュレートされた赤外線接続を介して OBEX を使用する方法を示します。

エミュレータインスタンスを 2 つ実行します。一方は接続の着信を待機し、他方は画像の送信を試みます。1 つ目のエミュレータでアプリケーションを起動して、「Obex Demo」を選択します。次に、「Receive Image」を選択します。エミュレータから、OBEX 接続がほかのデバイスに通信を許可することを示すメッセージが表示され、「Is it OK to make the connection?」という確認のメッセージが表示されます。「Yes」を選択します。待機側のエミュレータに、接続の着信を待機していることを示す画面が表示されます。

2 つ目のエミュレータ (送信側) で、Obex Demo を起動し、「Send Image」を選択します。画像の一覧が表示されます。1 つを選択し、「Send」をクリックします。エミュレータから、デモがクライアント接続を発信してよいかどうかを確認するメッセージが表示されます。「Yes」を選択します。「Send Image」ユーティリティーが画像をアップロードします。

待機側のエミュレータでは、ユーティリティーから画像の着信に関する情報が表示され、「Would you like to receive it?」というメッセージが表示されます。図 A-24 を参照してください。

図 A-24　接続の受け付けを確認中の待機側のエミュレータ



「Yes」を選択します。選択した画像が、シミュレートされた赤外線接続リンクを介して転送され、最初のエミュレータに表示されます。図 A-25 を参照してください。

図 A-25　正常に転送された画像

A.16　PDAPDemo

PDAPDemo は、PIM API と FileConnection API の使用方法を示します。これらの API は、JSR 75 仕様に含まれています。

A.16.1　ファイルの参照

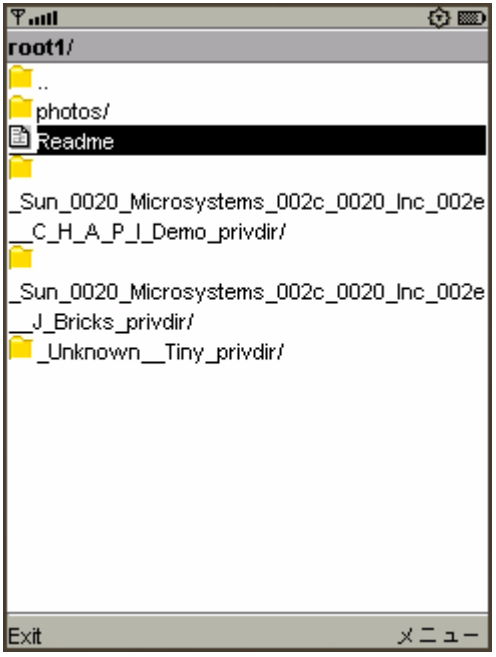
ファイルブラウザを実行するには、MIDlet に適切なセキュリティー承認が付与されていない場合、これを付与する必要があります。「編集」->「環境設定」を選択します。「セキュリティー」タブをクリックします。「セキュリティードメイン」を「maximum」に変更し、「了解」をクリックします。

PDAPDemo プロジェクトを開き、実行します。FileBrowser MIDlet を起動します。ディレクトリのリストが表示され、利用可能なディレクトリとファイルを参照できます。デフォルト設定では、root1 というディレクトリだけが存在します。

図 A-26　ファイルの参照

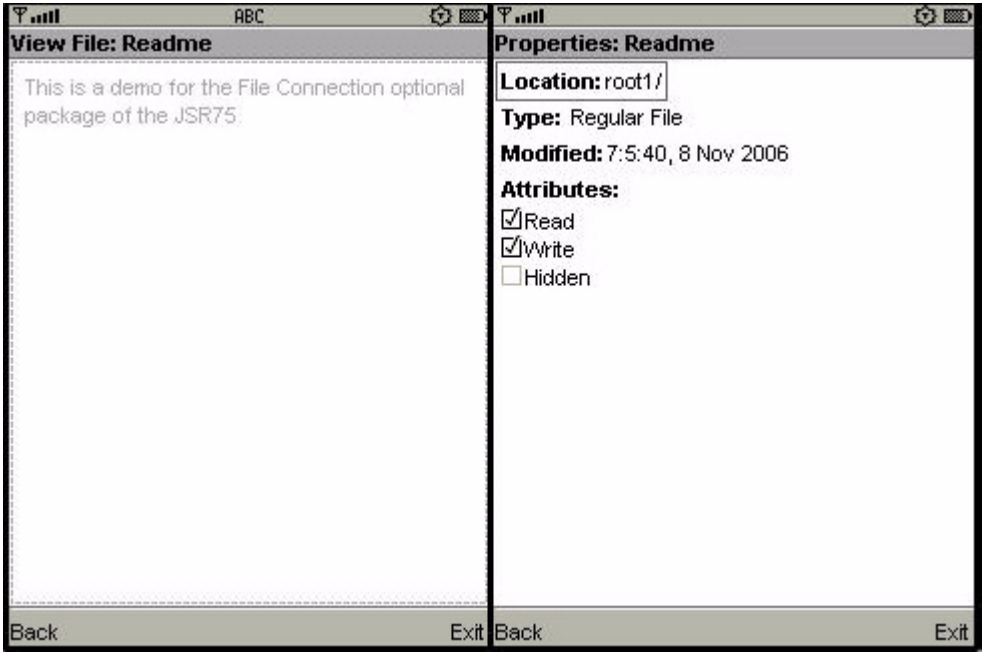
ディレクトリを選択し、「SELECT」ボタンをクリックしてそのディレクトリに入ります。

図 A-27　root1 ディレクトリの内容



デモアプリケーションのコマンドを使用して、ファイルやファイルのプロパティーを表示できます。ファイルを選択し、メニューから「Properties」または「View」を選択してみてください。

図 A-28　ファイルの内容とファイルのプロパティーの表示



DefaultColorPhone エミュレータスキンを使用している場合、実際のファイルは *workdir*¥appdb¥DefaultColorPhone¥filesystem にあります。必要に応じてファイルやルートディレクトリを追加できます。これらは、JSR 75 File API で可視になります。詳細は、第 10 章を参照してください。

A.16.2　PIM API

JSR75 PIM APIs サンプルは、連絡先リスト、カレンダ、予定表リストなどの個人情報にアクセスする方法をデモします。このサンプルを起動し、メインメニューからリストの種類を選択します。

このデモアプリケーションでは、どのリストの種類も同様に動作し、リストの種類ごとに 1 つのリストが含まれています。たとえば、「Contact Lists」を選択すると、そこには「Contacts」というリストだけが含まれています。「Event Lists」には「Events」というリストだけが含まれ、「To Do Lists」には「To Do」というリストだけが含まれています。

図 A-29　リストの種類の選択

リストの種類を選択し、特定のリストを選択すると、リスト内のすべてのアイテムを表示できます。このサンプルをはじめて実行するとき、このリストは空になっている可能性があります。

アイテムを追加するには、メニューから「New」を選択します。アイテムの「書式化された名前」を入力するように求めるメッセージが表示されます。このアイテムにデータフィールドを追加するには、メニューから「Add Field」を選択します。フィールド名の一覧が表示されます。1 つを選択し、この新しいフィールドの値を入力します。

図 A-30　連絡先のフィールドの追加

リストのアイテムを保存するには、メニューから「Commit」(オプション 3) を選択します。

リストに戻るには、「Back」コマンドを選択します。作成したアイテムがリストに表示されます。

作成したアイテムは、標準の vCard 形式または vCalendar 形式で *workdir*¥appdb¥*skin*¥pim ディレクトリに保存されます。詳細は、第 10 章を参照してください。

PIM API では、連絡先、カレンダ、および予定表のアイテムを標準形式でエクスポートできます。正確な形式は、リストの種類によって異なります。いずれかのリストでアイテムを表示しているとき、エクスポートされたアイテムを表示するコマンドがメニューに表示されます。

たとえば、連絡先リストのアイテムを表示しているときは、メニューに「Show vCard」コマンドが表示されます。このコマンドを選択すると、エクスポートされたアイテムが画面に表示されます。カレンダのアイテムと予定表のアイテムは、どちらも vCalendar 形式でエクスポートされます。

A.17 SATSADemos

SATSADemos には、SATSA (Security and Trust Services APIs) のデモが含まれています。デモの多くは、スマートカードとの通信方法を示します。エミュレータは、ソケットプロトコルを使用してシミュレートされたスマートカードと通信できます。ツールキットには、スマートカードシミュレータの cref が含まれています。詳細は、第 14 章を参照してください。

次の各節には、このデモのメニュー項目に対する指示が含まれています。それぞれのデモでは、エミュレータを起動する「前に」次の操作を実行する必要があります。

- コマンド行から cref のインスタンスを実行します。
- セキュリティードメインを maximum に設定する必要があります。

A.17.1 APDUMIDlet

この MIDlet は、データの小さいパケットである APDU (Application Protocol Data Unit) を使用した、スマートカードとの通信をデモします。APDUMIDlet では、2 つのシミュレートされたスマートカードを使用します。cref を使用してスマートカードシミュレータを実行できます。cref は、Java Card Development Kit の一部です。

Mohair アプリケーションには、cref で使用できるビルド済みのメモリーイメージが含まれています。メモリーイメージには、Mohair が対話する Java Card アプリケーションが含まれます。メモリーイメージは、Mohair プロジェクトのルートディレクトリに保存されます。

Windows では、次のように cref のインスタンスを 2 つ起動します。シミュレートされたカードスロットごとに 1 つです。ここでは、現在のディレクトリがツールキットのインストールディレクトリであると想定します。

```
start bin¥cref -p 9025 -i apps¥SATSADemos¥demo2.eeprom
start bin¥cref -p 9026 -i apps¥SATSADemos¥demo2.eeprom
```

Linux では、次のイメージを使用できます。

```
toolkit/bin/cref -p 9025 -i apps/SATSADemos/demo2.eeprom
toolkit/bin/cref -p 9026 -i apps/SATSADemos/demo2.eeprom
```

ポート番号 (この例では 9025 と 9026) は、第 14 章で説明したように、SATSA の環境設定で指定したポート番号に一致させる必要があります。また、demo2.eeprom の正しいパスを使用していることを確認してください。

2 つのスマートカードシミュレータを正しく実行したら、APDUMIDlet を実行できます。

A.17.2 SATMIDlet

SATMIDlet は、APDU 通信と少しだけ異なるスマートカード通信をデモします。

シミュレートされたスマートカードを設定するには、APDUMIDlet の場合と同様に cref を使用します。この場合は、ポート番号を指定する必要はありません。また、メモリーイメージが異なります。

Windows:	`start bin¥cref -i apps¥SATSADemos¥sat.eeprom`
Linux:	*toolkit*`/bin/cref -i apps/SATSADemos/sat.eeprom`

スマートカードシミュレータの動作中に、SATMIDlet を実行してカードアプリケーションと通信できます。

A.17.3 CryptoMIDlet

CryptoMIDlet は、SATSA の一般的な暗号化機能をデモします。スマートカードとは通信を行いません。

A.17.4 MohairMIDlet

MohairMIDlet には 2 つの機能があります。Find slots では、利用可能なすべてのカードスロットが表示されます。各スロットの番号には C または H が付加され、それぞれコールドスワップ可能またはホットスワップ可能を表します。スロットを確認したら、「Back」を選択して最初の画面に戻ります。

MohairMIDlet の 2 つ目の機能の SATSA-PKI Sign test では、スマートカードを使用してデジタル署名を生成します。前のデモと同様に、適切なメモリーイメージで cref を実行して、MohairMIDlet からの接続を準備する必要があります。インストールディレクトリで、次のように入力します。

Windows:	`start bin¥cref -i apps¥SATSADemos¥sat.eeprom`
Linux:	*workdir*`/bin/cref -i apps/SATSADemos/sat.eeprom`

エミュレータで、SATSA-PKI Sign test を強調表示し、「SELECT」を選択します。次の確認メッセージが表示されます。

```
This certificate will be used: MohairAuth
```

「OK」ソフトウェアボタンを選択します。

PIN 1 に 1234 と入力します。

「OK」ソフトウェアボタンを選択します。次の確認メッセージが表示されます。

```
This string will be signed: JSR 177 Approved
```

「OK」ソフトウェアボタンを選択します。次の確認メッセージが表示されます。

```
This certificate will be used: MohairAuth
```

「OK」ソフトウェアボタンを選択します。

否認防止キー 1 PIN に、2345 と入力します。

A.18 SATSAJCRMIDemo

このアプリケーションには、JCRMIMIDlet という MIDlet のみが含まれています。この MIDlet は、カードに対応したリモートオブジェクトプロトコルの、Java Card RMI を使用してカードアプリケーションと通信する方法を示します。SATSADemos の一部の MIDlet と同様に、適切なメモリーイメージで cref を起動する必要があります。

Windows:	`start bin¥cref -p 9025 -i apps¥SATSADemos¥demo2.eeprom`
Linux:	*workdir*`/bin/cref -i apps/SATSADemos/demo2.eeprom`

JCRMIMIDlet を実行して、アプリケーションがカードの分散オブジェクトと通信する方法を確認します。

A.19 SIPDemo

このアプリケーションは、SIP (JSR 180) を使用して 2 つのデバイス間で直接通信を行う簡単な例です。通常、デバイスはプロキシサーバーで SIP を使用して、直接通信を設定します。プロキシを含む詳細な例については、GoSip を参照してください。

SIPDemo の動作を確認するには、エミュレータのインスタンスを 2 つ実行します。1 つ目のエミュレータで、「Receive message」を選択します。デフォルトのポート 5070 を使用して、「Receive」を選択します。これで、1 つの目のエミュレータは着信メッセージの待機中になります。

2 つ目のエミュレータでは、「Send message」を選択します。受信側、ポート番号、件名、およびメッセージの値を入力するか、デフォルト設定をそのまま使用して、「Send」を選択します。1 つ目のエミュレータに、メッセージが表示されます。1 つ目のエミュレータの応答が、2 つ目のエミュレータに表示されます。

ネットワークモニターを有効にして、再度操作を実行します。ネットワークモニターの「SIP」タブで、エミュレータ間の通信を確認できます。

A.20 SVGContactList

このアプリケーションでは、同じ連絡先リストの情報とニュースバナーを、異なるスキンを使用して表示します。各スキンでは色とフォントが異なります。

「SVGContactlist(skin 1)」または「SVGContactlist(skin 2)」を選択して、「起動」をクリックします。

上下の矢印キーを使用して、連絡先のリストを移動します。強調表示された名前にマーク (> または中点) が付けられ、大きなフォントで表示されます。

図 A-31　スキン 2 で表示した連絡先リスト

「SELECT」ボタンを押して、強調表示された名前の詳細な情報を表示します。

図 A-32　連絡先リストの詳細

もう一度「SELECT」を押して、連絡先リストに戻ります。

A.21　SVGDemo

このスイートには、JSR 226 の Scalable 2D Vector Graphics API for J2ME のさまざまな使用方法をデモする MIDlet が含まれています。この API では、SVG コンテンツをロード、操作、描画、および再生する方法が提供されます。

Scalable Vector Graphics (SVG) 1.1 の仕様では、XML で 2 次元グラフィックスを描画するための言語が定義されています。完全な仕様は、`http://www.w3.org/TR/SVG11/` で入手できます。

SVG Tiny (SVGT) は、携帯電話などの小型デバイスに適した SVG のサブセットです。`http://www.w3.org/TR/SVGMobile/` を参照してください。SVG Tiny は、高度な対話型アニメーション 2D コンテンツを記述するための、コンパクトで強力な XML 形式です。グラフィカルな要素は、SVG マークアップで論理的にグループ化および識別できます。

A.21.1 SVG Browser

SVGBrowser MIDlet は、電話のファイルシステムにある SVG ファイルを表示します。このデモを実行する前に、SVG ファイルを *workdir*¥appdb¥DefaultColorPhone¥filesystem¥root1 ディレクトリに保存します。

デモを起動します。アプリケーションにより、root1 の内容が表示されます。SVG ファイルを選択し、「Open」ソフトウェアボタンを選択します。

A.21.2 Render SVG Image

Render SVG Image は、ファイルから SVG 画像をロードして描画します。デモコードを見ると、表示領域に合わせて画像のサイズが変更されていることがわかります。出力される画像は鮮明でシャープです。

A.21.3 Play SVG Animation

このアプリケーションは、ハロウィーンのグリーティングカードを描画する SVG アニメーションを再生します。8 を押すと再生します。また、5 で開始し、0 で停止します。8 を押した場合、5 を押すとアニメーションが再開されます。0 を押した場合、5 を押すとアニメーションが最初から開始されます。

SVG ファイルには、この短いアニメーションに関するさまざまな画像要素の時間変化が記述されます。

次のコード例では、SVG リソースをロードするために、JSR 226 の javax.microedition.m2g.SVGImage クラスが使用されています。続いて、javax.microedition.m2g.SVGAnimator クラスが SVG アニメーションのすべての構造を取得し、アニメーションを再生する java.awt.Component または javax.swing.JComponent を提供します。SVGAnimator クラスは、アニメーションを再生、一時停止、および停止するメソッドを提供します。

```
import javax.microedition.m2g.ScalableGraphics;
import javax.microedition.m2g.SVGImage;

...
String svgURI = ...;
SVGImage svgImage = (SVGImage) SVGImage.createImage(svgURI, null);
SVGAnimator svgAnimator = SVGAnimator.createAnimator(svgImage);

// If running a JSE applet, the target component is a JComponent.
```

```
JComponent svgAnimationComponent = (JComponent)
svgAnimator.getTargetComponent();
...

svgAnimator.play();
...
svgAnimator.pause();
...
svgAnimator.stop();
```

A.21.4 Create SVG Image from Scratch

このデモでは、API 呼び出しを使用して画像を作成します。空の SVGImage を作成してグラフィックのコンテンツを取り込み、そのコンテンツを表示します。

A.21.5 Bouncing Balls

Bouncing Balls デモでは、SVG アニメーションを再生します。8 を押すと再生します。また、5 で開始し、0 で停止します。8 を押した場合、5 を押すとアニメーションが再開されます。0 を押した場合、5 を押すとアニメーションが最初から開始されます。

A.21.6 Optimized Menu

このデモでは、選択したアイコンに黄色い枠が表示されます。別のアイコンに移動すると、そのアイコンが選択され、前のアイコンは非選択の状態に戻ります。アイコンのグリッドを超えて移動すると、選択はループして行われます。つまり、行の末尾のアイコンが選択されている場合、右に移動すると同じ行の先頭のアイコンが選択されます。

このデモは、豊富な機能 (グラフィックス、アニメーション、高度な 2D 描画) と、グラフィックの操作、事前描画、または再生に関する柔軟性という、UI マークアップと Java の組み合わせによって提供される柔軟性を示しています。

この例では、非選択状態から選択状態へのメニューアイコンの切り替え状態を定義する SVG アニメーションを、グラフィックアーティストが作成しています。プログラムにより、各アイコンのアニメーションシーケンスが、JSR 226 API を使用して (あとで高速に描画するために) オフスクリーンバッファーに個別に描画されます。

バッファリングにより、MIDlet はデバイスの画面解像度に適合することができ (グラフィックスが SVG 形式で定義されているため)、ビットマップの描画速度を維持できます。また、MIDlet は引き続き SVG のアニメーション機能を利用しています。

メニュー項目の外観とアニメーションの効果を定義する作業 (グラフィックアーティストおよびデザイナーの作業) は、メニューを表示しメニューの選択に基づいてアクションを開始する作業 (開発者の作業) と完全に切り離されています。アーティストと開発者の両方が SVG ドキュメント構造の規則に従う限り、これらの 2 つの作業は独立して変更することができます。

A.21.7 Picture Decorator

このデモでは、電話のキーを使用して、写真に装飾を追加します。各キーの機能は次のとおりです。

1	縮小
2	次の画像
3	拡大
4	ヘルプ
5	水平方向に反転
6	垂直方向に反転
7	反時計回りに回転
8	前の画像
9	時計回りに回転
#	ピッカーオプションを表示

このデモでは、装飾を行う 16 の画像が用意されています。

2 と 8 のキーを使用して、次の写真または前の写真に移動します。

装飾するには、# を押してピッカーを表示します。矢印キーを使用して、グラフィックオブジェクトを強調表示します。強調表示されたオブジェクトは拡大されます。「SELECT」を押して現在のグラフィックを選択するか、矢印キーを押して別のグラフィックを強調表示します。もう一度「SELECT」を押して、写真にグラフィックを追加します。装飾を追加すると、グラフィックに赤い + が表示されます。これは装飾が選択され、移動、サイズ変更、および操作が可能であることを示します。

図 A-33　吹き出しが選択されている装飾された画像

ナビゲーションの矢印キーを使用して、グラフィックを移動します。1 キーを使用するとグラフィックが縮小され、3 キーを使用するとグラフィックが拡大されます。5 または 6 を使用するとグラフィックが反転し、7 または 9 で回転します。位置が決まったら、「SELECT」を押します。緑色の三角形が表示されます。これはカーソルです。ナビゲーションキーを使用して、緑色の三角形を画像上で移動します。カーソルがオブジェクトに重なると、オブジェクトが赤いボックスで強調表示されます。「SELECT」を押します。赤色の + はオブジェクトが選択されていることを示します。

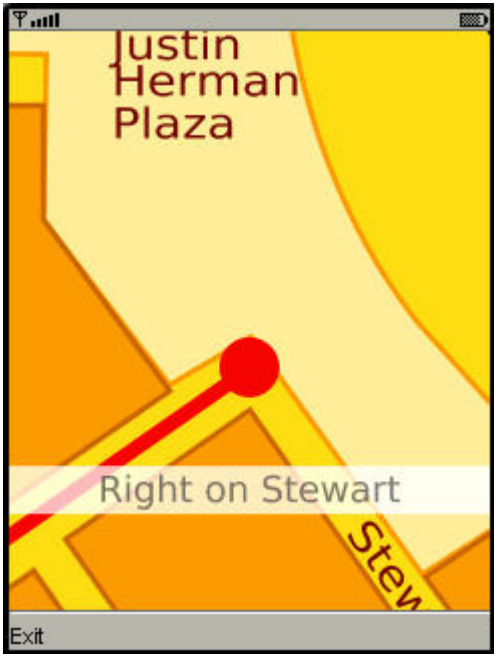
図 A-34　強調表示されている口ひげ

装飾 (プロパティー) を削除するには、オブジェクトを選択し、「メニュー」ソフトウェアボタンをクリックします。2 を押して、プロパティーを削除します。

A.21.8 Location Based Service

アプリケーションを起動します。スプラッシュ画面が表示されます。この画面はヘルプとしても使用されます。移動ガイドの初期表示は、サンフランシスコの市街図です。港 (青色) が画面の右側に表示されます。1 を押すと移動ガイドを開始します。アプリケーションにより、地図上でのユーザーの所在位置が拡大表示されます。進路を変更するたびに、方向が横長の白いボックスの中に表示されます。移動ガイドの実行中に 7 を押すと、マップが反時計回りに回転します。マップを回転させると、テキストは縦長の向きで表示されます。もう一度 7 を押すと、デフォルトの向きに戻ります。4 を押すと、ヘルプ画面が表示されます。

図 A-35



A.22 WMADemo

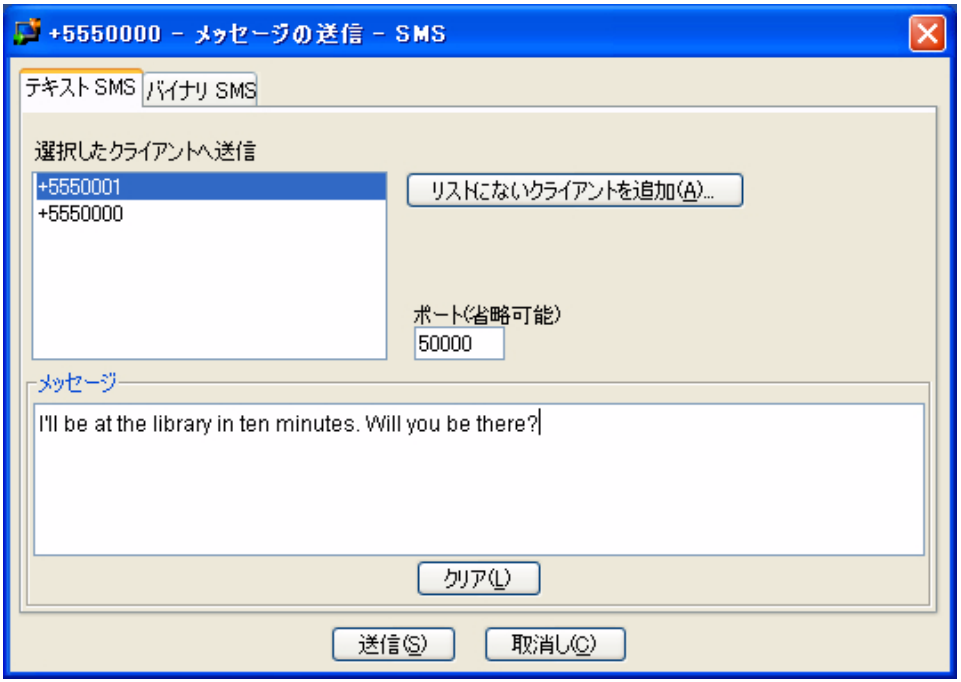
このアプリケーションは、SMS、CBS、および MMS の各メッセージを送受信する方法を示します。Sun Java™ Wireless Toolkit for CLDC により、メッセージングをサポートするための柔軟なエミュレーション環境が提供されます。エミュレータインスタンス間でのメッセージの交換、WMA コンソールユーティリティーを使ったメッセージの生成や受信が可能です。

サンプルでは転送レジストリを利用するため、「実行」ボタンを使用した場合は、その機能のすべてを確認することはできません。実際のデバイスにアプリケーションをインストールするときと同様の手順で、「OTA 経由で実行」機能を使ってアプリケーションをエミュレータにインストールします。この手順がわからない場合は、第 2 章を参照してください。

転送レジストリを理解するには、WMA コンソールを使用してエミュレータにメッセージを送信します。「ファイル」->「ユーティリティー」を選択して、コンソールを起動します。「WMA」ボックスの「コンソールを開く」ボタンをクリックして、WMA コンソールを起動します。

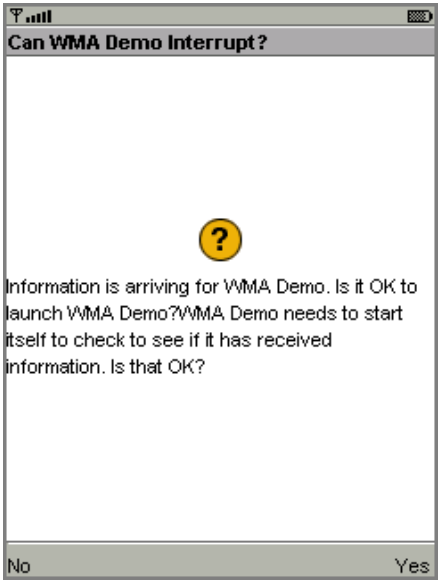
WMA コンソールウィンドウの「SMS の送信」ボタンをクリックします。エミュレータに対応する番号 (おそらく +5550000) を選択します。エミュレータで使用されている番号がわからない場合は、タイトルバーの表示で確認してください。SMS メッセージのウィンドウでこの番号を選択し、ポート番号 50000 を入力します。「メッセージ」フィールドにテキストメッセージを入力し、「送信」をクリックします。

図 A-36　テキストメッセージの送信



WMADemo アプリケーションを起動してもよいかどうかを尋ねるメッセージが表示されます。

図 A-37　転送レジストリのメッセージ



「はい」を選択します。`SMSReceive` MIDlet が起動され、着信した SMS メッセージがすぐに表示されます。

図 A-38　着信したテキストメッセージ

WMA コンソールを使用して、CBS メッセージや MMS メッセージの送受信を行うこともできます。詳細は、第 7 章を参照してください。

注 – WMA コンソールを使用してテキストメッセージを `WMADemo` に送信するときは、ポート番号として必ず 50000 を指定してください。CBS メッセージの場合は、ポート番号 50001 を使用します。MMS メッセージの場合は、アプリケーション ID として `example.mms.MMSDemo` を使用します。

たとえば、WMA コンソールからエミュレータに MMS メッセージを送信するには、前述のように「OTA 経由で実行」を使用して `WMADemo` をインストールしておきます。デモを起動して、「MMS Receive」を選択します。

WMA コンソールの「MMS の送信...」をクリックして、MMS 作成ウィンドウを開きます。メッセージの件名、アプリケーション ID `example.mms.MMSDemo`、および実行中のエミュレータの電話番号を入力します。

図 A-39　MMS メッセージの送信先の指定

次に「部分」タブをクリックします。WMA コンソールでは、MMS メッセージの一部として送信するファイルをハードディスクから選択できます。メッセージにファイルを追加するには、「追加」をクリックします。送信するファイルをファイルブラウザで探し、「了解」をクリックします。

図 A-40　MMS メッセージへの部分の追加

「送信」をクリックしてメッセージを送信します。

WMADemo を起動してもよいかどうかを尋ねるメッセージが表示されます。「はい」をクリックします。画像とその情報が表示されます。

図 A-41　WMADemo での画像の受信

付録B

コマンド行リファレンス

この付録では、Sun Java™ Wireless Toolkit for CLDC をコマンド行から実行する方法と、アプリケーションのビルドおよび実行の詳細な手順を説明します。また、`MEKeyTool` と呼ばれる Sun Java™ Wireless Toolkit for CLDC の証明書マネージャーユーティリティーと、`JadTool` (Java Application Descriptor Tool) と呼ばれる MIDlet 署名ユーティリティーについても説明します。

B.1 前提条件

コマンド行からアプリケーションをビルドして実行する前に、Java SE ソフトウェア開発キットのバージョンが 1.4.2 以降であることを確認してください。`jar` コマンドが、パスに含まれていることを確認してください。開発キットのバージョンを確認するには、`jar` コマンドを実行したあと、コマンド行で `java -version` を実行します。

関連する例については、デモアプリケーションの `bin` ディレクトリにあるファイル `build.bat` と `run.bat` について見てみます。これらのファイルは、次の場所にあります。

Windows:	*toolkit*¥apps¥*demo*¥bin
Linux:	*toolkit*/apps/*demo*/bin

toolkit は、Sun Java™ Wireless Toolkit for CLDC のインストールディレクトリで、*demo* はデモアプリケーションの名前を表します。

B.2 開発サイクル

MIDP アプリケーションの開発に関する詳細については、第 2 章を参照してください。この節では、開発サイクルの各手順をコマンド行から実施する方法について説明します。

B.2.1 ビルド

ユーザーインタフェースで、1 つの手順でプロジェクトをビルドできます。ただし、実際には、バックグラウンドで 2 つの手順が実行されます。まず、Java ソースファイルをコンパイルして、Java クラスファイルを生成します。次に、クラスファイルの「事前検証」を行なって、CLDC KVM で使用できるようにします。

Java ソースファイルをコンパイルするには、Java SE 開発キットの `javac` コンパイラを使用します。既存の Sun Java™ Wireless Toolkit for CLDC プロジェクトのディレクトリ構造を使用できます。MIDP API を使用するようにコンパイラに指示するには、`-bootclasspath` オプションを指定する必要があります。コンパイル済みクラスファイルの保存先をコンパイラに指示するには、`-d` オプションを指定します。

次の例は、ソースファイルを `src` ディレクトリから取り出し、クラスファイルを `tmpclasses` ディレクトリに出力して、MIDP 2.0 アプリケーションをコンパイルする方法を示しています。わかりやすくするために改行が追加されています。

Windows

```
javac
  -bootclasspath ..¥..¥lib¥cldcapi10.jar;..¥..¥lib¥midpapi20.jar
  -d tmpclasses
  src¥*.java
```

Linux

```
javac
  -bootclasspath ../../lib/cldcapi10.jar;../../lib/midpapi20.jar
  -d tmpclasses
  src/*.java
```

ツールキットでサポートされているオプションの API を使用する場合は、その JAR ファイルを `-bootclasspath` オプションに追加します。

`javac` の詳細は、Java SE のマニュアルを参照してください。

次の手順では、クラスファイルの事前検証を行います。Sun Java™ Wireless Toolkit for CLDC の bin ディレクトリに、preverify という便利なユーティリティーがあります。preverify コマンドの構文は次のとおりです。

```
preverify [options] files | directories
```

次のようなオプションがあります。

`-classpath` *classpath*

クラスを読み込む場合の基になるディレクトリや JAR ファイルを、セミコロン (;) で区切ったリストの形式で指定します。

`-d` *output-directory*

出力クラスのターゲットディレクトリを指定します。このディレクトリは、実行前検証を行う前に作成しておく必要があります。このオプションを指定しなかった場合は、クラスは `output` というディレクトリに出力されます。

コンパイルの例に続いて、次のコマンドを使ってコンパイル済みクラスファイルを検証します。前の例と同様に、わかりやすくするために改行を追加しています。

Windows

```
preverify
  -classpath ..¥..¥lib¥cldcapi10.jar;..¥..¥lib¥midpapi20.jar
  -d classes
  tmpclasses
```

Linux

```
preverify
  -classpath ../../lib/cldcapi10.jar;../../lib/midpapi20.jar
  -d classes
  tmpclasses
```

このコマンドを実行すると、事前検証済みのクラスファイルが `classes` ディレクトリに出力されます。アプリケーションが WMA または MMAPI を使用している場合や、ほかのバージョンの CLDC または MIDP を使用している場合には、関連する `.jar` ファイルを必ず classpath に含めてください。

B.2.2 パッケージ化

MIDlet スイートをパッケージ化するには、マニフェストファイル、アプリケーション JAR ファイル、最後に MIDlet スイート記述子を作成します。

MIDP 仕様の説明に従って、適切な属性が記述されているマニフェストファイルを作成します。マニフェストファイルは、任意のテキストエディタで作成できます。たとえば、マニフェストファイルに次の情報を記述します。

```
MIDlet-1: My MIDlet, MyMIDlet.png, MyMIDlet
MIDlet-Name: MyMIDlet
MIDlet-Vendor: My Organization
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC -1.0
MicroEdition-Profile: MIDP -2.0
```

マニフェストファイル、および MIDlet スイートのクラスファイルとリソースファイルを組み込んだ JAR ファイルを作成します。この JAR ファイルを作成するには、Java SE ソフトウェア開発キットに付属している `jar` ツールを使用します。構文は次のとおりです。

`jar cfm` *file manifest* `-C` *class-directory* `.-C` *resource-directory* `.`

引数は次のとおりです。

- *file* - 作成する JAR ファイルを指定します。
- *manifest* - MIDlet のマニフェストファイルを指定します。
- *class-directory* - アプリケーションのクラスを含むディレクトリを指定します。
- *resource-directory* – アプリケーションのリソースを含むディレクトリを指定します。

たとえば、`classes` ディレクトリのクラスファイルと `res` ディレクトリのリソースファイルから `MyApp.jar` という JAR ファイルを作成するには、次のコマンドを使用します。

```
jar cfm MyApp.jar MANIFEST.MF -C classes .-C res .
```

MIDP 仕様の説明に従って、適切な属性が記述されている JAD ファイルを作成します。JAD ファイルは、任意のテキストエディタで作成できます。このファイルには拡張子 `.jad` を付ける必要があります。

注 – `MIDlet-Jar-Size` エントリを、前の手順で作成した JAR ファイルのサイズに設定する必要があります。

たとえば、JAD ファイルに次の情報を記述します。

```
MIDlet-Name: MyMIDlet
MIDlet-Vendor: My Organization
MIDlet-Version: 1.0
MIDlet-Jar-URL: MyApp.jar
MIDlet-Jar-Size: 24601
```

B.2.3 実行

コマンド行からエミュレータを実行できます。Sun Java™ Wireless Toolkit for CLDC の `bin` ディレクトリに、`emulator` というコマンドがあります。`emulator` コマンドの構文は次のとおりです。

`emulator` *options*

一般的なオプションは次のとおりです。

- `-help` - 指定可能なオプションのリストを表示します。
- `-version` - エミュレータのバージョン情報を表示します。
- `-Xquery` - エミュレータスキンの情報を標準出力ストリームに出力し、ただちに終了します。この情報には、スキン名、画面サイズ、その他の機能などが含まれています。

MIDlet スイートの実行に関連するオプションは次のとおりです。

- `-Xdevice:`*skin-name* - 指定したスキン名を使用して、アプリケーションをエミュレータ上で実行します。スキン名のリストについては、4-1 ページの 4.1 節「エミュレータのスキン」を参照してください。
- `-Xdescriptor:`*jad-file* - アプリケーションのローカル実行時に使用する JAD ファイルを指定します。
- `-classpath` *classpath* - アプリケーションの実行に必要なライブラリのクラスパスを指定します。アプリケーションのローカル実行時にはこのオプションを使用します。
- `-Dcom.sun.midp.io.http.proxy` - 実行時の HTTP および HTTPS プロキシサーバーを設定します。たとえば、次のようになります。

  `-Dcom.sun.midp.io.http.proxy=`*proxy-host*`:`*proxy-port*
- `-Dcom.sun.midp.midlet.platformRequestCommand` - アプリケーションが URL を呼び出すときに使用するブラウザを指定します。たとえば、次のようになります。

  `-Dcom.sun.midp.midlet.platformRequestCommand=`*browser*

  呼び出し例は次のようになります。

  `emulator -Dcom.sun.midp.midlet.platformRequestCommand=firefox -Xjam:install=`*URL-to-app-using-platformRequest-method*

  常に同じブラウザを使用する場合は、*toolkit*`/lib/system.config` に次の行を追加できます。

  `com.sun.midp.midlet.platformRequestCommand:` *browser*

  Windows: Windows の場合、このパラメータが指定されていないと、デフォルトのブラウザが使用されます。

Linux: Linux では、通常、Linux システムにデフォルトのブラウザがないため、このパラメータが必要です。パラメータが指定されていない場合、アプリケーションが URL を開こうとしても何も起こりません。

- `-Xjam:`*command=application* - OTA 配信を使って実行するには、アプリケーション管理ソフトウェア (AMS) を使用してアプリケーションをリモートで実行します。引数を付けないでアプリケーションを指定した場合は、グラフィック形式の AMS が起動します。コマンドは次のとおりです。

  `install=`*jad-file-url* | `force` | `list` | `storageNames`|

  指定した JAD ファイルを使用して、アプリケーションをデバイス上にインストールします。

  または次のように指定します。

  `run=[`*storage-name* | *storage-number*`]`

  インストール済みのアプリケーションを実行します。対象のアプリケーションは、有効なストレージ名またはストレージ番号を使用して指定します。

  `remove=[`*storage-name* | *storage-number* | `all]`

  インストール済みのアプリケーションを削除します。対象のアプリケーションは、有効なストレージ名またはストレージ番号を使用して指定します。`all` を指定すると、インストール済みのアプリケーションがすべて削除されます。

- `transient=`*jad-file-url* - 指定した JAD ファイルを使用して、アプリケーションがインストール、実行、および削除されます。`transient` を指定すると、アプリケーションのインストール、実行、および削除が 3 回行われます。

B.2.4 デバッグ

デバッグやトレースを行うには、エミュレータで次のオプションを使用します。

- `-Xverbose:`*trace-options* - 次のようにオプションのコンマ区切りリストで指定されているとおりに、トレース出力を表示します。
  - `gc` - ガベージコレクションをトレースします。
  - `class` - クラスのロードをトレースします。
  - `all` - すべてのトレースオプションを使用します。
- `-Xdebug` - 実行時のデバッグを有効にします。`-Xrunjdwp` オプションとともに使用する必要があります。

- -Xrunjdwp:*debug-settings* - コンマで区切って指定したデバッグ設定を使用し、Java デバッグワイヤープロトコルセッションを開始します。このオプションは、-Xdebug オプションとともに使用する必要があります。次のデバッグ設定を指定できます。
  - transport=*transport-mechanism* - デバッガとの通信に使用する転送方式。dt_socket だけを指定できます。
  - address=*host:port* - デバッガに接続するための転送アドレス。*host* は省略できます。*host* を省略した場合は、ホストマシンが *localhost* とみなされます。
  - server=*{y|n}* - デバッグエージェントをサーバーとして起動。デバッガは、指定したポートに接続する必要があります。指定可能な値は、y と n です。現時点では、y だけを指定できます。すなわち、エミュレータはサーバーとして動作する必要があります。

B.3 ツールキットの GUI コンポーネントの起動

Sun Java™ Wireless Toolkit for CLDC のコンポーネントは、すべてコマンド行から起動できます。各コンポーネントは、ツールキットの bin ディレクトリにあります。

表 B-1 ツールキットコンポーネントのコマンド

コマンド	説明
DefaultDevice	デフォルトのエミュレータスキンを選択するためのダイアログを表示します
ktoolbar	ユーザーインタフェースを起動します。
prefs	ツールキットの環境設定を起動します
utils	ツールキットのユーティリティーウィンドウを起動します

B.4 エミュレータの環境設定

エミュレータの環境設定をコマンド行から変更するには、-Xprefs オプションを指定して emulator コマンドを実行します。構文は次のとおりです。

-Xprefs:*filename*

filename には、プロパティーファイルのフルパス名を指定します。このファイルの値によって、「環境設定」ダイアログボックスの値が上書きされます。プロパティーファイルには、次の表に示すプロパティーを記述できます。

表 B-2 エミュレータの環境設定プロパティー一覧

プロパティー名	プロパティーの説明と有効値
`http.version`	「ネットワーク構成」->「HTTP バージョン」 値: HTTP/1.1、HTTP/1.0
`http.proxyHost`	「ネットワーク構成」->「HTTP アドレス」 値: ホスト名
`http.proxyPort`	「ネットワーク構成」->「HTTP ポート」 値: 整数
`https.proxyHost`	「ネットワーク構成」->「HTTPS アドレス」 値: ホスト名
`https.proxyPort`	「ネットワーク構成」->「HTTPS ポート」 値: 整数
`kvem.memory.monitor.enable`	「モニター」->「メモリーモニターを有効にする」 値: true、false
`kvem.netmon.comm.enable`	「モニター」->「Comm モニターを有効にする」 値: true、false
`kvem.netmon.datagram.enable`	「モニター」->「Datagram モニターを有効にする」 値: true、false
`kvem.netmon.http.enable`	「モニター」->「HTTP モニターを有効にする」 値: true、false
`kvem.netmon.https.enable`	「モニター」->「HTTPS モニターを有効にする」 値: true、false
`kvem.netmon.socket.enable`	「モニター」->「Socket モニターを有効にする」 値: true、false
`kvem.netmon.ssl.enable`	「モニター」->「SSL モニターを有効にする」 値: true、false
`kvem.profiler.enable`	「モニター」->「プロファイラを有効にする」 値: true、false
`netspeed.bitpersecond`	「パフォーマンス」->「(ビット/秒)」コンボボックス 値: 整数
`netspeed.enableSpeedEmulation`	「パフォーマンス」->「ネットワークスループットエミュレーションを有効」 値: true、false

表 B-2　エミュレータの環境設定プロパティー一覧 (続き)

プロパティー名	プロパティーの説明と有効値
screen.graphicsLatency	「パフォーマンス」->「グラフィックプリミティブの応答時間」 値: 整数
screen.refresh.mode	「パフォーマンス」->「表示の更新」(ラジオボタン) 値: default、immediate、periodic
screen.refresh.rate	「パフォーマンス」->「表示の更新」(スライダ) 値: 整数
vmspeed.bytecodespermilli	「パフォーマンス」->「VM スピードエミュレーションを有効」(チェックボックス) 値: 整数
vmspeed.enableEmulation	「パフォーマンス」->「VM スピードエミュレーションを有効」(スライダ) 値: true、false
storage.root	「ストレージ」->「ストレージルートディレクトリ」 値: 文字列 (*appdb* への相対パス)
storage.size	「ストレージ」->「ストレージサイズ」 値: 整数
mm.control.capture	「MMedia」->「サウンドの取り込み」 値: true、false
mm.control.midi	「MMedia」->「MIDI 音質」 値: true、false
mm.control.mixing	「MMedia」->「サウンドのミキシング」 値: true、false
mm.control.record	「MMedia」->「サウンドの録音」 値: true、false
mm.control.volume	値: true、false
mm.format.midi	「MMedia」->「MIDI 形式」 値: true、false
mm.format.video	「MMedia」->「ビデオ形式」 値: true、false
mm.format.wav	「MMedia」->「WAV オーディオ形式」 値: true、false
wma.client.phoneNumber	「WMA」->「次のエミュレータの電話番号」 値: 整数

表 B-2　エミュレータの環境設定プロパティー一覧 (続き)

プロパティー名	プロパティーの説明と有効値
wma.server.firstAssignedPhoneNumber	「WMA」->「最初に割り当てられた電話番号」 値: 整数
wma.server.percentFragmentLoss	「WMA」->「% のランダムメッセージ断片の消失」 値: 整数
wma.server.deliveryDelayMS	「WMA」->「メッセージ断片の配信遅延」 値: 整数

B.5　セキュリティー機能の使用

Sun Java™ Wireless Toolkit for CLDC のセキュリティー機能も、すべてコマンド行から起動できます。エミュレータのデフォルト保護ドメインの変更、MIDlet スイートへの署名、および証明書の管理を行うことができます。

B.5.1　エミュレータのデフォルトの保護ドメインの変更

エミュレータのデフォルトの保護ドメインを変更するには、次のオプションを指定して emulator コマンドを実行します。

-Xdomain *domain-type*

MIDlet スイートにセキュリティードメインを割り当てます。ドメインタイプには、untrusted、trusted、minimum、および maximum があります。

B.5.2　MIDlet スイートへの署名

JadTool は MIDlet スイートに署名するためのコマンド行インタフェースで、MIDP 2.0 仕様に準拠した公開鍵暗号方式を使用します。MIDlet スイートの署名とは、JAR ファイルの署名者の証明書とデジタル署名を JAD ファイルに追加することです。JadTool は、支払い更新 (JPP) ファイルにも署名できます。

JadTool は、Java SE プラットフォームキーストアに保管されている証明書とキーだけを使用します。Java SE ソフトウェアには、Java SE プラットフォームキーストアを管理するためのコマンド行ツール keytool が用意されています。

JadTool ユーティリティーは JAR ファイルにパッケージ化されています。このユーティリティーを起動するには、コマンドプロンプトを開き、カレントディレクトリを *toolkit*¥bin に変更し、次のコマンドを入力します。

```
java -jar JadTool.jar command
```

コマンドは次のとおりです。

- -help

  JADTool の使用法を出力します。

- -addcert -alias *alias* [-keystore *keystore*] [-storepass *password*] [-certnum *number*] [-chainnum *number*] [-encoding *encoding*] -inputjad | inputjpp *input-file* -outputjad | outputjpp *output-file*

  指定したキーストアのキーペアの証明書を JAD ファイルまたは JPP ファイルに追加します。

- -addjarsig -jarfile *jarfile* -keystore *keystore* -alias *alias* -storepass *password* -keypass *password* -inputjad *input-jadfile* -outputjad *output-jadfile*

  指定した JAR ファイルのデジタル署名を、指定した JAD ファイルに追加します。-jarfile のデフォルト値は、JAD ファイル内の MIDlet-Jar-URL プロパティーです。

- -showcert [([-certnum *number*] [-chainnum *number*]) |-all [-encoding *encoding*] -inputjad *filename*

  指定した JAD ファイル内の証明書のリストを表示します。

- -addjppsig -alias *alias* -keypass *password* [-keystore *keystore*] [-storepass *password*] [-encoding *encoding*] -inputjpp *filename* -outputjpp *filename*

  入力 JPP ファイルのデジタル署名を、指定した出力 JPP ファイルに追加します。

デフォルト値は次のとおりです。

- -encoding - UTF-8
- -jarfile - JAD ファイル内の MIDlet-Jar-URL プロパティー
- -keystore - %HOMEPATH%¥.keystore
- -certnum - 1
- -chainnum - 1

B.5.3 証明書の管理

MEKeyTool は、Java SE SDK に付属している keytool ユーティリティーと同様に、認証局 (CA) の公開鍵を管理します。この公開鍵を使用すると、SSL を使用し、セキュリティー保護された HTTP 通信 (HTTPS) が簡単になります。

MEKeyTool を使用する前に、Java Cryptography Extension キーストアにアクセスできるようにしておく必要があります。これは、Java SE keytool ユーティリティーを使用して作成できます。

Windows

```
http://java.sun.com/javase/6/docs/technotes/tools
/windows/keytool.html
```

Linux

```
http://java.sun.com/javase/6/docs/technotes/tools
/solaris/keytool.html
```

MEKeyTool を起動するには、コマンドプロンプトを開き、カレントディレクトリを *toolkit*¥bin に変更し、次のコマンドを入力します。

Windows:	*toolkit*¥bin¥mekeytool.exe *-command*
Linux:	*toolkit*/bin/mekeytool *-command*

コマンドのキーワードは次のようになります。MEKeyTool はインストールディレクトリから起動しますが、デフォルトのキーとユーザーが作成するキーは各ユーザーの個人用ディレクトリ (*workdir*¥appdb) に存在します。

- -help

  MEKeyTool の使用法を出力します。

- -import -alias *alias* [-keystore *JCEkeystore*] [-storepass *storepass*] -domain *domain-name*

  指定された JCE キーストアパスワードを使用して、指定された Java Cryptography Extension キーストアから ME キーストアに公開鍵をインポートします。デフォルトの ME キーストアは *workdir*¥appdb¥_main.mks、デフォルトの Java Cryptography Extension キーストアは *user.home*¥.keystore です。

- -list

  ME キーストアに含まれているキーと、これらのキーの所有者と有効期限を一覧表示します。ME キーストアは *workdir*¥appdb¥_main.mks です。

- -delete (-owner *owner* | -number *key-number*)

  ME キーストアから、指定された所有者が所有しているキーを削除します。ME キーストアは *workdir*¥appdb¥_main.mks です。

注 – Sun Java™ Wireless Toolkit for CLDC には、`_main.ks` という ME キーストアが含まれています。このキーストアは `appdb` サブディレクトリにあります。このキーストアには、デフォルトの Java SE プラットフォームキーストア内のすべての証明書が含まれています。Java SE プラットフォームキーストアは Java SE SDK で提供されます。

B.6 スタブジェネレータの使用

スタブジェネレータを使用して、モバイルクライアントから Web サービスにアクセスすることができます。`wscompile` ツールによって、Java API for XML (JAX) RPC のクライアントとサービスで使用されるスタブ、タイ、シリアライザ、および WSDL ファイルが生成されます。このツールを実行するときには、設定ファイルを読み込みます。このファイルには、WSDL ファイル、モデルファイル、またはコンパイル済みのサービスエンドポイントインタフェースが指定されています。スタブジェネレータコマンドの構文は次のとおりです。

`wscompile [`*options*`]` *configuration-files*

B.6.1 オプション

表 B-3 `wscompile` コマンドのオプション

オプション	説明
`-d` *output directory*	生成される出力ファイルの配置場所を指定します
`-f:`*features*	指定された機能を有効にします
`-features:`*features*	`-f:`*features* と同じです
`-g`	デバッグ情報を生成します
`-gen`	`-gen:client` と同じです
`-gen:client`	クライアントアーティファクト (スタブなど) を生成します
`-httpproxy:`*host:port*	HTTP プロキシサーバー (デフォルトのポートは 8080) を指定します
`-import`	インタフェースと値型のみを生成します
`-model` *file*	指定されたファイルに内部モデルを書き込みます
`-O`	生成されたコードを最適化します

表 B-3　wscompile コマンドのオプション (続き)

オプション	説明
-s *directory*	生成されるソースファイルの配置場所を指定します
-verbose	コンパイラが実行している処理に関するメッセージを出力します
-version	バージョン情報を出力します
-cldc1.0	CLDC のバージョンを 1.0 (デフォルト) に設定します。float および double は string になります。
-cldc1.1	CLDC バージョンを 1.1 に設定します (float および double が許可されます)
-cldc1.0info	CLDC 1.0 のすべての情報メッセージと警告メッセージを表示します。

注 – -gen オプションは、1 つだけ指定してください。-f オプションに複数の機能を指定する場合は、コンマ (,) で区切る必要があります。

表 B-4 は、-f オプションに指定できる機能 (コンマで区切る) の一覧です。wscompile ツールは、WSDL ファイル、コンパイル済みのサービスエンドポイントインタフェース (SEI)、またはモデルファイルを入力として読み込みます。ファイルの種類の欄は、その機能で使用できるファイルを示しています。

表 B-4　wscompile のコマンドがサポートしている機能 (-f)

オプション	説明	ファイルの種類
explicitcontext	明示的なサービスコンテキストマッピングを有効にします	WSDL
nodatabinding	リテラルエンコーディングのデータバインディングを無効にします	WSDL
noencodedtypes	エンコーディングタイプ情報を無効にします	WSDL、SEI、モデル
nomultirefs	複数参照のサポートを無効にします	WSDL、SEI、モデル
novalidation	インポートされた WSDL ドキュメントの完全な検査を無効にします	WSDL
searchschema	スキーマからサブタイプを積極的に検索します	WSDL
serializeinterfaces	インタフェースタイプの直接直列化を有効にします	WSDL、SEI、モデル

表 B-4　wscompile のコマンドがサポートしている機能 (-f) (続き)

オプション	説明	ファイルの種類
wsi	WSI-Basic Profile 機能 (デフォルト) を有効にします	
resolveidref	xsd:IDREF を解決します	
nounwrap	ラップを解除しません	

例

```
wscompile -gen -d generated config.xml
wscompile -gen -f:nounwrap -O -cldc1.1 -d generated config.xml
```

付録C

各国語対応

この付録では、Sun Java™ Wireless Toolkit for CLDC で表示される言語の設定、およびエミュレーション環境の各国語対応の設定について説明します。

C.1 ロケール設定

ロケールとは、同一の言語、習慣、あるいは文化的慣習などを共有する、特定の地理的領域や政治的領域、またはコミュニティーを指します。ソフトウェアにおけるロケールは、ソフトウェアを特定の地域に適合させるのに必要となる各種情報を含んだ、ファイル、データ、およびコードの集合体で表されます。

一部のソフトウェアでは、次のようなユーザー向けの情報をカスタマイズするために、ロケールが使用されます。

- ユーザーに表示されるメッセージ
- 使用するフォントなど、書法に関連する情報

デフォルトでは、ユーザーインタフェースのすべての文字列は、サポートされているプラットフォームのロケールの言語で表示されます。

たとえば、適切にローカライズされた Sun Java™ Wireless Toolkit for CLDC がダウンロードされてインストールされている場合は、日本語の Microsoft Windows マシンで動作しているツールキットに日本語の文字を表示することができます。

`wtk.locale` プロパティーを設定して、指定したロケールの言語でユーザーインタフェースを表示できます。3-14 ページの 3.8 節「Wireless Toolkit の設定」で説明したように、`ktools.properties` をインストールディレクトリから作業用ディレクトリ (*workdir*`/wtklib/ktools.properties`) にコピーし、コピーしたファイルを編集します。たとえば、ロケールプロパティーを `en-US` に設定し、適切な補助ソフトウェアをダウンロードして Sun Java™ Wireless Toolkit for CLDC 上にインストールすると、Wireless Toolkit を日本語マシンで動作させたままユーザーインタフェースを英語で表示することができます。

C.2 エミュレート中のロケール

デバイスのロケールは、システムプロパティー `microedition.locale` に格納されます。エミュレータのロケールを変更するには、「編集」->「環境設定」を選択し、「国際化」を選択します。コンボボックスからロケールを選択するか、直接入力します。

`microedition.locale` の詳細については、MIDP の仕様を参照してください。

C.3 文字エンコーディング

CLDC のシステムプロパティーである `microedition.encoding` は、MIDP 環境のデフォルトの文字エンコーディングを定義します。Sun Java™ Wireless Toolkit for CLDC のエミュレータでは、使用しているウィンドウシステムに応じてこのプロパティーが設定されます。プロパティーの値には、同じウィンドウシステムで動作している Java SE プラットフォームのデフォルトのエンコーディングが設定されます。たとえば、英語版のウィンドウシステムでは、エンコーディングは次のように設定されます。

```
microedition.encoding=ISO8859_1
```

デフォルトの値を変更するには、*workdir*¥`wtklib¥ktools.properties` ファイルに `microedition.encoding` プロパティーを追加します。たとえば、Microsoft Windows でデフォルトの設定として UTF-8 を使用する場合は、*workdir*¥`wtklib¥ktools.properties` ファイル内でプロパティーを次のように設定します。

```
microedition.encoding=UTF-8
```

文字エンコーディングの詳細については、CLDC の仕様を参照してください。

注 – エミュレートした環境では、すべての Java SE プラットフォームエンコーダを使用できます。特定のデバイスで使用できるエンコーダの一覧を制限する方法については、『Sun Java™ Wireless Toolkit for CLDC 基本カスタマイズガイド』を参照してください。

C.4 Java テクノロジコンパイラのエンコーディング設定

javac.encoding プロパティーは、javac コンパイラでソースファイルをコンパイルする際に使用するエンコーディングを定義します。プロパティーの値には、同じウィンドウシステムで動作している Java SE プラットフォームのデフォルトのエンコーディングが設定されます。

デフォルトの値を変更するには、ktools.properties ファイルに javac.encoding プロパティーを追加します。たとえば、英語のシステムを使用していて、日本語のリソースバンドルをコンパイルする必要がある場合は、次のように日本語の文字セットを指定できます。

```
javac.encoding=EUCJIS
```

C.5 デフォルトエミュレータのフォントサポート

エミュレートした環境で使用されるデフォルトフォントは、基になるウィンドウシステムのロケールに応じて設定されます。デフォルト設定では、MIDP 環境のフォントはデフォルトの Java SE プラットフォーム Java テクノロジフォントにマップされます。通常、これらのフォントは、対象のウィンドウのロケールで必要なすべての文字をサポートしています。

これらのフォントを変更して、デフォルトのフォントでサポートしていないほかの文字をサポートすることができます。設定方法については、『Sun Java Wireless Toolkit for CLDC 基本カスタマイズガイド』を参照してください。

索引

K

L

M

O

P

R

S

T