基本カスタマイズガイド

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: *Basic Customization Guide, Sun Java™ Wireless Toolkit for CLDC*

目次

はじめに　v

1. 概要　1-1
   - 1.1　新しいエミュレータスキンの作成　1-1
   - 1.2　難読化ツールのプラグインの作成　1-2

2. エミュレータのスキニング　2-1
   - 2.1　スキンプロパティーファイル　2-1
   - 2.2　スキンの外観　2-2
     - 2.2.1　スキン画像　2-2
     - 2.2.2　画面の範囲と描画可能領域　2-4
     - 2.2.3　画面の特徴　2-6
       - 2.2.3.1　`isColor=`*[true|false]*　2-6
       - 2.2.3.2　`colorCount=`*number*　2-6
       - 2.2.3.3　`enableAlphaChannel=`*[true|false]*　2-6
       - 2.2.3.4　`gamma=`*number*　2-7
       - 2.2.3.5　`screenDoubleBuffer=`[*true*|*false*]　2-7
       - 2.2.3.6　`screenBorderColor=`*color*　2-7
       - 2.2.3.7　`screenBGColor=`*color*　2-7
     - 2.2.4　アイコン　2-7

2.2.5 フォント 2-9
2.2.6 ソフトウェアボタンのラベル 2-10
2.2.7 サウンド 2-11
2.3 ユーザー入力の割り当て 2-12
2.3.1 キーボードハンドラ 2-12
2.3.2 ボタン 2-12
2.3.3 ボタンへのデスクトップキーボードキーの割り当て 2-13
2.3.4 ゲームキーの割り当て 2-14
2.3.5 文字へのキーの割り当て 2-14
2.3.6 ソフトウェアボタンへのコマンドの割り当て 2-15
2.3.7 コマンドメニュー 2-16
2.3.8 一時停止と再開 2-16
2.3.9 ポインタイベント 2-17
2.4 ロケールと文字エンコーティング 2-17

3. 難読化ツールのプラグインの作成 3-1
3.1 プラグインの作成 3-1
3.2 ツールキットの設定 3-2

索引 索引-1

はじめに

『Sun Java™ Wireless Toolkit for CLDC 基本カスタマイズガイド』には、Sun Java Wireless Toolkit for CLDC のカスタマイズに関する詳細な技術情報が記載されています。

対象読者

このガイドは、新しいデバイスへの対応を可能にしたり、またはソースコードを修正してエミュレータ用に新しい API への対応を可能にしたりするなどの目的のために、Sun Java Wireless Toolkit for CLDC をカスタマイズする必要がある開発者を対象としています。

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、カスタマイズの可能性について概要を説明します。

第 2 章では、新しいエミュレータスキンの作成方法について説明します。

第 3 章では、難読化ツールのプラグインの作成方法について説明します。

パスとオペレーティングシステムのコマンド

このマニュアルでは、ターミナルウィンドウを開く、ディレクトリを変更する、環境変数を設定するなど、UNIX®、Linux、または Microsoft Windows の各オペレーティングシステムで使用する基本的なコマンドと操作手順については、説明を省略している場合があります。これらの情報については、使用するシステムに付属のソフトウェアマニュアルを参照してください。

このマニュアルの例は、Windows プラットフォーム用です。Linux システムでは、パス区切り文字の円記号 (¥) をスラッシュ (/) に置き換えてください。たとえば、次のようになります。

Windows	*toolkit*¥wtklib¥devices¥DefaultColorPhone¥netIndicatorOn.png
Linux	*toolkit*/wtklib/devices/DefaultColorPhone/netIndicatorOn.png

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常に表します。*workdir* は、ユーザーの作業用ディレクトリを表します。*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)

書体と記号について

表 P-1　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』
「 」	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'**

* 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。

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。

表 P-2　関連マニュアル

内容	マニュアル名
Sun Java Wireless Toolkit for CLDC	Sun Java Wireless Toolkit for CLDC ユーザーズガイド
カスタマイズ	Sun Java Wireless Toolkit for CLDC Advanced Customization Guide*
統合エミュレータインタフェース	Unified Emulator Interface Specification
PDAP Option Packages - 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
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
Mobile 3D Graphics - 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
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
MSA - JSR 248	*Mobile Service Architecture*

* 入手にはソースライセンスが必要です。

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。Mobility ページでこのほかの情報を確認してください。

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

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://developers.sun.com/
- http://java.sun.com/docs/

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 基本カスタマイズガイド』

第1章

概要

Sun Java Wireless Toolkit for CLDC は、MIDP アプリケーションを開発するためのエミュレーション環境を備えています。このマニュアルでは、次の 2 つの有用な方法でこのツールキットをカスタマイズする手順について説明します。

- 新しいエミュレータスキンの作成
- 難読化ツールのプラグインの作成

この章の残りの部分では、これらのカスタマイズについてそれぞれその概要を説明します。

1.1 新しいエミュレータスキンの作成

Sun Java Wireless Toolkit for CLDC では、次の方法でエミュレータをカスタマイズできます。

1. サードパーティ製のエミュレータをダウンロードして Sun Java Wireless Toolkit for CLDC にインストールします。
2. Sun Java Wireless Toolkit for CLDC のデフォルトのエミュレータに基づいて、新しいエミュレータスキンを作成します。この方法については、第 2 章を参照してください。
3. デフォルトエミュレータの実装をカスタマイズします。このためには、Sun Java Wireless Toolkit for CLDC のソースコードのライセンスを取得してエミュレータの実装をカスタマイズします。

1.2 難読化ツールのプラグインの作成

「難読化ツール」は、実行可能な MIDlet スイートのサイズを小さくするために使用するツールです。MIDlet スイートを小さくすると、ダウンロード時間が短くなります。つまり、現在の帯域幅不足に悩むワイヤレス環境での待ち時間が短くなり、ユーザーの通信料金が低くなる可能性があります。

Sun Java Wireless Toolkit for CLDC は、ProGuard 難読化ツール (`http://proguard.sourceforge.net/`) に対応していますが、あらゆるタイプの難読化ツールに対応可能な柔軟なアーキテクチャーを備えています。

第 3 章に詳細な技術情報が記載されています。

第2章

エミュレータのスキニング

この章では、エミュレータスキンの定義について説明します。エミュレータの既存のスキンを変更することも、新しいスキンを作成することもできます。この処理をエミュレータの「スキニング」と呼びます。

2.1 スキンプロパティーファイル

エミュレータスキンは、1 つのプロパティーファイルによって定義されます。各スキンプロパティーファイルは、*toolkit*¥wtklib¥devicest の独自のサブディレクトリに格納されます。ここで *toolkit* は Sun Java Wireless Toolkit for CLDC のインストールディレクトリです。プロパティーファイルの名前は、ディレクトリ名と一致します。

たとえば、DefaultColorPhone スキンは *toolkit*¥wtklib¥devices¥DefaultColorPhone ディレクトリの DefaultColorPhone.properties によって定義されます。

スキンプロパティーファイルは、エミュレータスキンの外観と動作を定義します。このファイルには、画像とサウンド (同じディレクトリにある場合もそうでない場合もある) へのポインタが含まれています。たとえば、DefaultColorPhone ディレクトリには電話そのものの画像が含まれますが、DefaultColorPhone のアイコンとサウンドは wtklib¥devices¥Share で定義されます。

この章の残りの部分では、スキンプロパティーファイルの内容について説明します。プロパティーファイルは、プレーンテキストファイルです。このファイルは、任意のテキストエディタを使って変更できます。通常、プロパティーファイルのエントリには後ろに値を付けたプロパティー名を指定します。名前と値は、コロンまたは等号で区切ります。ハッシュ記号 (#) で始まる行はコメントです。

新しいスキンを作成するもっとも簡単な方法は、既存のスキンをコピーして、それを変更することです。たとえば、次のようになります。

1. `DefaultColorPhone` ディレクトリをコピーします。
2. 新しいディレクトリに新しいスキンの名前を付けます。
3. プロパティーファイルの名前をディレクトリ名に合わせて変更します。

   ディレクトリ名を `NewSkin` にした場合は、プロパティーファイルの名前を `NewSkin.properties` に変更します。

2.2 スキンの外観

エミュレータスキンの全体の外観は、さまざまな要因によって決められます。この節では、それぞれの要因について説明します。

- スキン画像
- 画面の範囲と描画可能領域
- 画面の特徴
- アイコン
- フォント
- コマンド
- サウンド

2.2.1 スキン画像

スキンの外観の多くは、次の 3 つの画像によって決められます。

1. 「デフォルト」の画像には、ニュートラルな (標準の) 状態のデバイスが表示されます。
2. 「強調表示された」画像には、すべてのボタンが強調表示されている (ユーザーがボタンの上にマウスを移動したときのように) デバイスが表示されます。
3. 「押された」画像には、すべてのボタンが押されているデバイスが表示されます。

それぞれの画像には、デバイス全体が表示されます。このツールキットでは、これらの部分の画像を使用して、ボタンが強調表示された状態やボタンが押された状態を示します。

たとえば、図 2-1 に `DefaultColorPhone` の 3 つの画像を示します。

図 2-1　DefaultColorPhone の標準の状態、強調表示された状態、および押された状態の画像

図 2-2 に、3 つの画像の違いがわかるように、各キーパッドの一部を拡大して表示します。

図 2-2　標準の状態、強調表示された状態、および押された状態のエミュレータスキン画像の詳細

スキンプロパティーファイルでは、3 つの画像ファイルは次のプロパティーで指定されます。

- `default_image=`*image-file-name*
- `pressed_buttons_image=`*image-file-name*
- `highlighted_image=`*image-file-name*

画像ファイルは、PNG、GIF、または JPEG のいずれでもかまいません。画像ファイルはすべて同じ寸法にします。たとえば、`DefaultColorPhone.properties` には次のエントリが含まれます。

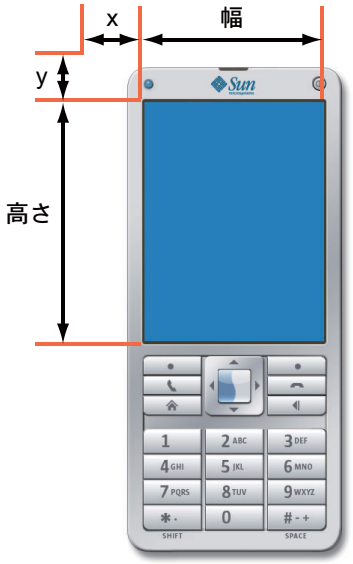
- `default_image=DefaultColorPhoneNormal.png`
- `pressed_buttons_image=DefaultColorPhonePressed.png`
- `highlighted_image=DefaultColorPhoneHiLite.png`

2.2.2 画面の範囲と描画可能領域

画面は、実際のデバイスのディスプレイを表します。画面は、画面全体の範囲、「描画可能」範囲、および表示色数のような要因を決めるその他のパラメータによって定義されます。

画面全体の範囲は、ディスプレイの全領域となります。左上の角 (0,0) にある画像ファイルの起点を基準にしたピクセル単位の測定値で定義されます。

図 2-3　画面の範囲



画面の範囲は、プロパティーファイル内で次のように指定されます。

- `screen.x=`*x-coordinate*
- `screen.y=`*y-coordinate*
- `screen.width=`*width*
- `screen.height=`*height*

たとえば、次のようになります。

- `screen.x=37`
- `screen.y=54`
- `screen.width=240`
- `screen.height=320`

ほとんどのデバイスでは、表示領域全体を MIDP アプリケーションが使用できるようにしていません。画面の残りの部分は通常、各種のアイコンやインジケータ用に予約されています。同様に、Sun Java Wireless Toolkit for CLDC エミュレータでは、MIDP アプリケーションに使用できる、「描画可能」領域と呼ばれる画面全体のサブセットを定義できます。描画可能領域の起点は、ディスプレイの左上の角を基準にした座標で表されます。たとえば、図 2-4 に示すように、`DefaultColorPhone` エミュレータスキンでは、いちばん上のバーをアイコンに使用し、いちばん下のバーをソフトウェアのラベルとその他のアイコンに使用しています。

図 2-4 `DefaultColorPhone` の描画可能な画面領域

エミュレータスキンプロパティーファイルでは、描画可能領域は次のように表されます。

- `screenPaintableRegion.x=`*x-coordinate*
- `screenPaintableRegion.y=`*y-coordinate*
- `screenPaintableRegion.width=`*width*
- `screenPaintableRegion.height=`*height*

x 座標と y 座標は、画面位置 (画面の左上の角) からの相対位置で指定します。また、screenPaintableRegion の幅と高さの値は、対応する screen.width と screen.height の値を超えてはいけません。

たとえば、次のようになります。

- screenPaintableRegion.x=0
- screenPaintableRegion.y=10
- screenPaintableRegion.width=240
- screenPaintableRegion.height=290

注 – 全画面モード (MIDP 2.0 以降) では、エミュレータは描画可能領域の起点から始まり、画面の右下の角まで広がる領域を使用します。DefaultColorPhone では、いちばん上のバーを除く画面領域全体になります。

2.2.3 画面の特徴

エミュレータスキンプロパティーファイルでは、画面によってサポートされる表示色数やピクセルの縦横比を指定します。iscolor では、エミュレータスキンでカラー表示とグレースケール表示のどちらを使用するかを指定します。

2.2.3.1 isColor=*[true|false]*

カラー表示の場合は *true*、グレースケール表示の場合は *false* を指定します。

2.2.3.2 colorCount=*number*

colorCount では、表示色数を指定します。グレースケール表示のデバイスでは、階調数を指定します。

たとえば、DefaultColorPhone には 4096 色のカラー表示画面があります。

```
isColor=true
colorCount=0x1000
```

2.2.3.3 enableAlphaChannel=*[true|false]*

このオプションでは、エミュレータのアルファ (透明度) の処理を有効にするかどうかを指定します。

2.2.3.4 gamma=*number*

この値では、ガンマ補正を有効にするかどうかを指定します。値 1 はエラー訂正がないことを意味します。

2.2.3.5 screenDoubleBuffer=[*true*|*false*]

`true` を指定するとダブルバッファリングが有効になり、`false` を指定すると無効になります。

2.2.3.6 screenBorderColor=*color*

`screenBorderColor` では、画面の描画不可能領域に使用される背景色を指定します。たとえば、`DefaultColorPhone` では次の色を使用します。

```
screenBorderColor=0xb6b6aa
```

2.2.3.7 screenBGColor=*color*

このプロパティーを使用して、グレースケール表示のデバイスでの画面の背景色を設定します。

2.2.4 アイコン

Sun Java Wireless Toolkit for CLDC エミュレータでは、アイコン (ユーザーに情報を伝達する小さい画像) を使用できます。通常、アイコンはディスプレイの描画可能領域の外側に配置されます。エミュレータは、表 2-1 で説明されている定義済みのアイコンを実装しています。

表 2-1 エミュレータのアイコン

名前	説明
battery	バッテリーの状態を示します
domain	実行中の MIDlet の保護ドメインを示します
down	スクロールが可能であることを示します
inmode	入力モード (小文字入力、大文字入力、数字入力) を示します
internet	インターネットの利用状況を示します
left	スクロールが可能であることを示します

表 2-1　エミュレータのアイコン

名前	説明
reception	ワイヤレス信号の強さを示します
right	スクロールが可能であることを示します
up	スクロールが可能であることを示します

アイコンは、画面の起点を基準にして測定した位置、デフォルト状態、および考えられる状態に対応する画像のリストによって定義されます。たとえば、DefaultColorPhone の down アイコンの定義は次のようになります。このアイコンは、表示可能な画面領域よりも長いリストまたはフォームが示されたときに表示される下向き矢印です。

```
icon.down: 113, 314, off
icon.down.off:
icon.down.on: ../Share/down.gif
```

最初の行では、アイコンが表示される位置を指定します。DefaultColorPhone の場合は、描画可能な画面領域の外側にあるいちばん下のバーの中央の位置になります。デフォルト状態は、off です。

off 状態に対応する画像ファイルはありませんが、on 状態では wtklib¥devices¥Share ディレクトリの画像 down.gif を使用します。

もう 1 つの興味深い例として、7 つの状態に 6 つの対応する画像ファイルを持つ inmode アイコンがあります。

```
icon.inmode: 113, 2, off
icon.inmode.off:
icon.inmode.ABC: ../Share/ABC.gif
icon.inmode.abc: ../Share/abc_lower.gif
icon.inmode.123: ../Share/123.gif
icon.inmode.kana: ../Share/kana.gif
icon.inmode.hira: ../Share/hira.gif
icon.inmode.sym: ../Share/sym.gif
```

アイコンに似ているエミュレータのもう 1 つの側面として、ネットワークインジケータがあります。ネットワークインジケータは、画面内に配置されるのではなく、エミュレータスキンに表示されます。DefaultColorPhone では、ネットワークインジケータはエミュレータスキンの左上に小さな緑色のライトとして表示されます。ネットワークインジケータは、次の 2 つのプロパティーを使用して定義されます。

- netindicator.image: **画像**
- netindicator.bounds: *x*, *y*, **幅**, **高さ**

たとえば、DefaultColorPhone では、ネットワークインジケータは次のように定義されます。

- netindicator.image: net_indicator.png
- netindicator.bounds: 53, 27, 30, 30

幅と高さは、ネットワークインジケータ画像の幅と高さに一致しているはすです。

2.2.5 フォント

エミュレータによって使用されるフォントは、スキンプロパティーファイルで定義します。フォントは、基本的に MIDP の Font クラスで利用できるフォントフェース、フォントスタイル、およびフォントサイズごとに定義できます。構文は次のとおりです。

font.**フェース**.**スタイル**.**サイズ**: **フォント指定子**

フェース、スタイル、サイズ の各パラメータは、MIDP の Font API から推測できますが、エミュレータスキンプロパティーファイルでは識別子は小文字になります。フォントフェースは system、monospace、または proportional、フォントスタイルは plain、bold、または italic、フォントサイズは small、medium、または large です。

フォント指定子 は、Java Platform, Standard Edition (Java SE) java.awt.Font クラスライブラリで設定された表記規則に従います。DefaultColorPhone の次の例では、3 つのすべてのサイズでプロポーショナルイタリックフォントを定義しています。

```
font.proportional.italic.small: SansSerif-italic-9
font.proportional.italic.medium: SansSerif-italic-11
font.proportional.italic.large: SansSerif-italic-14
```

使用できる定義がほかにない場合に使用されるデフォルトフォントを指定する必要があります。DefaultColorPhone では、10 ポイントの SansSerif フォントがデフォルトとして使用されます。

```
font.default=SansSerif-plain-10
```

フォントには下線を付けることもできます。デフォルトでは、下線付きフォントは MIDP の実装によってサポートされていますが、次のように特定のフォントに対してのみ下線の表示を無効にすることもできます。

font.*face*.*style*.*size*.underline.enabled=false

必要に応じて、次のようにすべてのフォントに対して下線の表示を無効にすることもできます。

```
font.all.underline.enabled=false
```

システムフォントを使用する代わりに、「ビットマップ」フォントを使用することもできます。ビットマップフォントは、フォントの文字形状を含む画像です。ビットマップフォントの画像は、いずれかの文字形状を含む 1 行のテキストです。ビットマップフォントを定義するには、次のプロパティーを使用します。

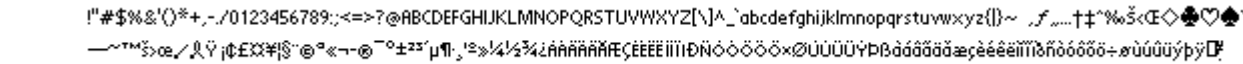
`font.`*`name=font-property-file`*

フォントプロパティーファイルには、次のプロパティー定義が含まれます。

- `font_image =` *image-file*
- `font_height =` *font-height*
- `font_ascent =` *font-ascent*
- `font_descent =` *font-descent*
- `font_leading =` *font-leading*

画像ファイルは、PNG 形式、GIF 形式、または JPEG 形式のいずれでもかまいません。このファイルでは、文字を 1 列に並べる必要があります。図 2-5 では、見やすさのために文字を 2 行で表示しています。

図 2-5 ビットマップフォントの画像



高さ、アセント、ディセント、およびレディングはすべてピクセル単位で指定されます。これらのフォント用語に詳しくない場合は、`java.awt.FontMetrics` の Java SE のマニュアルを参照してください。

フォントプロパティーファイルには、ASCII 文字コードと画像中のピクセル数で表した水平オフセットとのマッピングリストも含める必要があります。次の例では、ASCII コード 65 が水平オフセット 124 にマップされます。

```
ascii_x-65=124
```

ビットマップフォントを一度定義すると、その名前をフォント指定子として使用できます。

2.2.6 ソフトウェアボタンのラベル

ソフトウェアボタンとは、定義済みの機能を持たないボタンのことです。ソフトウェアボタンについては、この章で詳しく後述します。ソフトウェアボタンのラベルは、画面に表示されます。エミュレータスキンプロパティーファイルでは、ソフトウェアボタンのラベルの表示位置と表示方法を指定します。

ソフトウェアボタンのラベルのフォントは、フォントの別名 (フォントに割り当てる短い名前) を使用して定義されます。各ソフトウェアボタンのラベルは、次のプロパティーによって記述されます。

`softbutton.`*n*`=`*x*`,` *y*`,` *width*`,` *height*`,` *font-alias*`,` *alignment*

alignment の有効な値は、`left`、`right`、および `center` です。

たとえば、次のプロパティーを指定すると、ソフトウェアボタンのラベルには 12 ポイントの Courier フォントが使用されます。

- `font.softButton=Courier-plain-12`
- `softbutton.0=1,306,78,16, softButton, left`
- `softbutton.1=160,306,78,16, softButton, right`

まず、フォントの別名 `softButton` を定義します。最初のラベルは左詰めになり、2 番目のラベルは右詰めになります。

2.2.7 サウンド

MIDP の警告には、サウンドが関連付けられています。Sun Java Wireless Toolkit for CLDC エミュレータのサウンドは、MIDP の `AlertType` クラスに列挙されているタイプごとに 1 つずつ、ファイルを使用して定義されます。エミュレータでは、基本となる Java SE の実装によってサポートされているどのサウンドファイルタイプでも使用できます。Java SE Development Kit 1.5 では、AIFF、AU、WAV、MIDI、RMF などのタイプがあります。たとえば、`DefaultColorPhone` での定義を次に示します。

```
alert.alarm.sound:    ../Share/mid_alarm.wav
alert.info.sound:     ../Share/mid_info.wav
alert.warning.sound: ../Share/mid_warn.wav
alert.error.sound:    ../Share/mid_err.wav
alert.confirmation.sound: ../Share/mid_confirm.wav
```

デフォルトサウンドは、特定の警告タイプにサウンドが定義されていない場合に再生されます。

`alert.confirmation.sound:` **サウンドファイル**

また、電話の振動のように再生されるサウンドを定義することもできます。`DefaultColorPhone` では、次のように定義されます。

```
vibrator.sound: ../Share/vibrate.wav
```

2.3 ユーザー入力の割り当て

エミュレータスキンの説明は 2 つに分けられます。最初は外観についてで、すでに説明しました。2 つ目は、ユーザー入力がエミュレータでどのように割り当てられるかを定義します。

2.3.1 キーボードハンドラ

キーボードハンドラは、ボタンが押されたことを認識し、適切なアクションをエミュレータで実行します。たとえば、マウスを使ってソフトウェアボタンのいずれかを押した場合、エミュレータで適切なアクションが行われるようにするのはキーボードハンドラです。

キーボードハンドラでは 1 組の標準のボタン名を定義し、ユーザーはその名前を使ってボタンを定義します。エミュレータスキンのどこにボタンが配置されるかを指定するだけで、残りの作業はキーボードハンドラが行います。

Sun Java Wireless Toolkit for CLDC エミュレータにはキーボードハンドラが 2 つあります。1 つは、ITU-T キーパッド (`DefaultKeyboardHandler`) を備えた電話デバイス用で、もう 1 つは完全な Qwerty キーボードを備えたデバイス用です。たとえば、`DefaultColorPhone` には次のキーボードハンドラプロパティーが含まれます。

```
keyboard.handler = com.sun.kvem.midp.DefaultKeyboardHandler
```

`DefaultKeyboardHandler` では、次の標準のボタン名を認識します。`0` 〜 `9`、`POUND`、`ASTERISK`、`POWER`、`SEND`、`END`、`LEFT`、`RIGHT`、`UP`、`DOWN`、`SELECT`、`SOFT1`、`SOFT2`、`SOFT3`、`SOFT4`、`USER1` 〜 `USER10`。

`QwertyDevice` では、キーボードハンドラは次のように定義されます。

```
keyboard.handler = com.sun.kvem.midp.QwertyKeyboardHandler
```

`QwertyKeyboardHandler` では `DefaultKeyboardHandler` と同じボタンをサポートしますが、アルファベットキー、Shift キー、Alt キーなどの標準キーボードにあるボタンも含まれます。

2.3.2 ボタン

ボタンは、名前と 1 組の座標を使って定義します。2 組の座標を指定すると、四角形のボタンが定義されます。2 組を超える座標を指定すると、多角形の領域がボタンに使用されます。

ボタン領域は、デバイススキンの画像を基準にして定義されます。ユーザーが定義されたボタン領域の上にマウスを移動すると、強調表示されたスキン画像の対応する領域が表示されます。ユーザーがボタンを押すと、押された状態のスキン画像の対応する領域が表示されます。

ボタンは、それだけではあまり意味がありません。ボタン名を四角形または多角形の領域に関連付けるだけです。ボタン名をエミュレータの機能に割り当てるのは、キーボードハンドラの役目です。2-13 ページの 2.3.3 節「ボタンへのデスクトップキーボードキーの割り当て」では、デスクトップコンピュータのキーボードのキーをどのようにボタンに割り当てることができるかについて説明します。

次のプロパティーは、5 のボタンの四角形の領域を定義する方法を示しています。ボタンの起点は 140, 553 で、幅は 84、高さは 37 です。

```
button.5 = 140, 553, 84, 37
```

次に、アスタリスクボタンの多角形の定義の例を示します。

```
button.ASTERISK = 66, 605, 110, 606, 140, 636, 120, 647, 70, 637
```

この多角形は、次に示す点をつなぐ直線セグメントを使用して定義されます。

```
66, 605
110, 606
140, 636
120, 647
70, 637
```

2.3.3 ボタンへのデスクトップキーボードキーの割り当て

ボタンには、1 つ以上のデスクトップキーボードのキーを関連付けることができます。つまり、デバイススキンの上にマウスを移動してマウスボタンを押す代わりに、デスクトップキーボードを使用してエミュレータを制御できます。

たとえば DefaultColorPhone では、デスクトップキーボードの F1 キーを押すと、左ソフトウェアボタンと同じ操作を実行できます。左ソフトウェアボタンは、次のように、プロパティーファイルで SOFT1 として定義されています。

```
button.SOFT1 = 78, 417, 120, 423, 126, 465, 74, 440
```

デスクトップキーボードのショートカットは次のように定義されます。

```
key.SOFT1 = VK_F1
```

実際のキー定義は *virtual key codes* であり、これは Java SE の java.awt.event.KeyEvent クラスライブラリに定義されています。詳細については、Java SE のマニュアルを参照してください。

必要に応じて、複数のデスクトップキーボードキーを 1 つのボタンに割り当てることができます。次の DefaultColorPhone の例では、デスクトップキーボードの 5 キーまたは数値キーパッドの 5 キーはどちらもエミュレータスキンの 5 ボタンのショートカットとして定義されています。

```
key.5 = VK_5 VK_NUMPAD5
```

2.3.4 ゲームキーの割り当て

ゲームアクションは DefaultKeyboardHandler にすでに定義されていますが、QwertyKeyboardHandler を使用して独自のゲームアクションを指定できます。そのためには、次の形式を使用します。

game.*function* = *button-name*

機能 (function) には、LEFT、RIGHT、UP、DOWN、SELECT のいずれかを指定できます。標準のボタン名については、この章で前述しています。

デフォルト設定は次のとおりです。

- game.UP = UP
- game.DOWN = DOWN
- game.LEFT = LEFT
- game.RIGHT = RIGHT
- game.SELECT = SELECT

2.3.5 文字へのキーの割り当て

QwertyKeyboardHandler を使用すると、ボタンを単独で押したときや、Shift キーや Alt キーと同時に押したときに生成される文字を指定できます。

そのためには、次の形式を使用します。

keyboard.handler.qwerty.*button* = '*base-character*' '*shift-character*' '*alternate-character*'

ベース文字はボタンを単独で押したときに生成される文字、Shift 文字はボタンと Shift キーを同時に押したときに生成される文字、Alt 文字はボタンと Alt キーを同時に押したときに生成される文字です。

ボタンを Shift キーや Alt キーと同時に押したときの動作は、次の 2 とおりの方法でエミュレートします。

- 前述の節に従ってキーボード上のキーをボタンに割り当て、ボタンに割り当てたキーを Shift キーや Alt キーと同時に押します。

- Shift-Lock ボタンや Alt-Lock ボタンを押してからボタンを押します。Shift-Lock ボタンや Alt-Lock ボタンをもう一度押すと、Shift ロックや Alt ロックが解除されます。

たとえば、次のようになります。

```
keyboard.handler.qwerty.A = 'a' 'A' '?'
```

2.3.6 ソフトウェアボタンへのコマンドの割り当て

コマンドは、MIDP 仕様の一部です。コマンドは、ユーザーが利用できるようにすることが必要なアクションを指定する柔軟な方法です。特定のデバイスでどのように利用可能にするかを指定する必要はありません。

通常、MIDP デバイスではソフトウェアボタンを使用してコマンドを呼び出します。コマンドテキストは、ディスプレイのソフトウェアボタンに物理的に近い場所に表示されます。使用できるソフトウェアボタンよりも利用可能なコマンドの数が多い場合は、1 つのソフトウェアボタンのラベルがメニューとして表示されます。メニューのソフトウェアボタンを押すと、使用できるコマンドのメニューが表示されます。

Sun Java Wireless Toolkit for CLDC エミュレータでは、`javax.microedition.lcdui.Command` に指定されているコマンドの種類に基づいて、特定のタイプのコマンドの表示場所を指定できます。たとえば、2 つのソフトウェアボタンを持つエミュレータスキンでは、`BACK` コマンドおよび `EXIT` コマンドを常に左ソフトウェアボタンに表示し、`OK` コマンドを右ソフトウェアボタンに表示するようにできます。

このような種類の設定をエミュレータスキンプロパティーファイルに指定するには、次のような行を使用します。

`command.keys.`*`command-type=button`*

たとえば、`DefaultColorPhone` では次のようにコマンド設定を定義します。

```
command.keys.BACK = SOFT1
command.keys.EXIT = SOFT1
command.keys.CANCEL = SOFT1
command.keys.STOP = SOFT1

command.keys.OK = SOFT2
command.keys.SCREEN = SOFT2
command.keys.ITEM = SOFT2
command.keys.HELP = SOFT2
```

その他のボタン名を指定すると、特定のコマンドの種類にほかの優先ボタンを割り当てることができます。たとえば、次の行では、END が利用可能な場合は BACK コマンドを END に割り当て、そうでない場合は SOFT1 に割り当てるようにエミュレータに指示します。

```
command.keys.BACK = END SOFT1
```

最後に、必要に応じて、ソフトウェアボタンが特定のコマンドの種類にのみ使用されるように指定することもできます。次の定義では、コマンドの種類 BACK、EXIT、CANCEL、および STOP だけに SOFT1 キーを制限しています。

```
command.exclusive.SOFT1 = BACK EXIT CANCEL STOP
```

2.3.7 コマンドメニュー

使用できるソフトウェアボタンの数がコマンドの数よりも少ない場合、コマンドはメニューに配置されます。Sun Java Wireless Toolkit for CLDC エミュレータでは、コマンドメニューを制御できます。メニューの表示に使用するボタン、メニュー項目のスクロールに使用するボタン、およびメニューとして表示されるテキストラベルを選択できます。

DefaultColorPhone の次のプロパティーでは、メニューの表示または非表示に 2 番目のソフトウェアボタンを使用するようエミュレータスキンに指示します。

```
command.menu.activate = SOFT2
```

デフォルトでは、UP ボタンと DOWN ボタンがメニューのスクロールに使用され、SELECT ボタンがコマンドの選択に使用されます。これらの割り当ては、次のプロパティーを使用して変更できます。

- `command.menu.select =` *button*
- `command.menu.up =` *button*
- `command.menu.down =` *button*

2.3.8 一時停止と再開

MIDP 仕様では、着呼のようなほかの電話イベントに対応する場合など、アプリケーション (MIDlet) をいつでも一時停止できます。

エミュレータスキンプロパティーファイルを使用して、MIDlets を一時停止および再開するためのデスクトップキーボードのショートカットを定義できます。たとえば、DefaultColorPhone では、一時停止 (中断) に F6 キーを使用し、再開に F7 キーを使用します。

```
midlet.SUSPEND_ALL = VK_F6
midlet.RESUME_ALL = VK_F7
```

2.3.9 ポインタイベント

エミュレータスキンにタッチ画面があるかどうかは、次に示す 1 つのプロパティーによって決められます。

```
touch_screen=[true|false]
```

エミュレータスキンにタッチ画面がある場合は、ポインタイベントが Canvases に配信されます。

2.4 ロケールと文字エンコーティング

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。ソフトウェアでのロケールとは、ソフトウェアを特定の地理上の場所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を意味します。

一部の操作はロケールに依存するため、次のようなユーザーごとの情報に対してはロケールを特定する必要があります。

- ユーザーに表示されるメッセージ
- 日付や通貨形式などの文化情報

Sun Java Wireless Toolkit for CLDC エミュレータでは、プラットフォームのロケールによってデフォルトのロケールが決まります。

特定のロケールを定義するには、次の定義を使用します。

`microedition.locale:` **ロケール名**

ロケール名は、ハイフン (-) で区切られた 2 つの部分から構成されます。たとえば、en-US は、米国英語圏をロケールとして指定します。また、en-AU はオーストラリア英語圏を指定します。

最初の部分は指定可能な ISO 言語コードです。これらのコードは、ISO-639 で定義された 2 つの小文字のコードです。これらのコードの完全なリストについては、次のサイトを参照してください。
http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt

ロケール名の次の部分は指定可能な ISO 国コードです。これらのコードは、ISO-3166 で定義された 2 つの大文字のコードです。これらのコードの完全なリストについては、次のサイトを参照してください。
http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html

CLDC の入力 API と出力 API では、指定された文字エンコーディングを使用して、8 ビット文字と 16 ビット Unicode 文字との間で変換が行われます。特定の MIDP の実装では、使用できるエンコーディングのセットを指定することがあります。

エミュレータでは、使用するプラットフォームのエンコーディングがデフォルトになります。UTF-8 や UTF-16 など、Java SE プラットフォームがサポートするほかのエンコーディングを使用することもできます。

エミュレータスキンが使用する文字エンコーディングを定義するには、次の定義を使用します。

`microedition.encoding:` **エンコーディング**

使用可能なすべてのエンコーディングセットを定義するには、次の定義を使用します。

`microedition.encoding.supported:` **エンコーティングのリスト**

たとえば、次のようになります。

```
microedition.encoding: UTF-8
microedition.encoding.supported: UTF-8, UTF-16, ISO-8859-1,
ISO-8859-2, Shift_JIS
```

Java SE プラットフォームでサポートされているすべてのエンコーディングをサポートするには、次のように `microedition.encoding.supported` の定義を空白のままにします。

```
microedition.encoding.supported:
```

注 – ISO-8859-1 エンコーディングは、`microedition.encoding.supported` エントリで指定されているかどうかに関係なく、エミュレータで実行されるアプリケーションで常に使用可能になります。

第3章

難読化ツールのプラグインの作成

Sun Java Wireless Toolkit for CLDC では、バイトコードの難読化ツールを使用して MIDlet スイートの JAR ファイルのサイズを小さくできます。『Sun Java Wireless Toolkit for CLDC ユーザーズガイド』に説明されているように、このツールキットでは ProGuard がサポートされています。

別の難読化ツールを使用する場合は、Sun Java Wireless Toolkit for CLDC のプラグインを作成できます。

3.1 プラグインの作成

難読化ツールのプラグインによって、`com.sun.kvem.environment.Obfuscator` インタフェースが拡張されます。インタフェース自体は、*toolkit*¥wtklib¥kenv.zip に含まれています。

`Obfuscator` インタフェースには、実装する必要がある次の 2 つのメソッドが含まれています。

- `public void createScriptFile(File jadFilename File projectDir);`
- `public void run(File jarFileObfuscated, String wtkBinDir, String wtkLibDir, String jarFilename, String projectDir, String classPath, String emptyAPI) throws IOException;`

難読化ツールのプラグインをコンパイルするには、必ず `kenv.zip` を `CLASSPATH` に追加するようにしてください。

たとえば、ごく簡単なプラグインのソースコードを次に示します。このソースコードは、実際には難読化ツールを呼び出しませんが、`Obfuscator` インタフェースの実装方法を示しています。

```
import java.io.*;
public class NullObfuscator
    implements com.sun.kvem.environment.Obfuscator {
  public void createScriptFile(File jadFilename, File projectDir) {
    System.out.println("NullObfuscator:createScriptFile()");
   }

   public void run(File jarFileObfuscated, String wtkBinDir,
     String wtkLibDir, String jarFilename, String projectDir,
     String classPath, String emptyAPI) throws IOException {
       System.out.println("NullObfuscator:run()");
     }
}
```

このソースコードを、*toolkit*¥wtklib¥test¥NullObfuscator.java という名前で保存するとします。この場合、次のようにコマンド行でこのソースコードをコンパイルできます。

```
set classpath=%classpath%;toolkit¥wtklib¥kenv.zip
javac NullObfuscator.java
```

3.2 ツールキットの設定

難読化ツールのプラグインを作成したら、ツールキットにその場所を認識させる必要があります。このためには、*toolkit*¥wtklib¥Windows¥ktools.properties を編集します。難読化ツールのプラグインのクラス名を編集し、ツールキットにそのクラスの場所を認識させます。例にならっている場合は、次のようにプロパティーを編集します。

```
obfuscator.runner.class.name: NullObfuscator
obfuscator.runner.classpath: wtklib¥¥test
```

ツールキットを再起動して、プロジェクトを開きます。ここで、「プロジェクト」->「パッケージ」->「難読化パッケージを作成」を選択します。コンソールに、次のように NullObfuscator の出力が表示されます。

```
Project settings saved
Building "Tiny"
NullObfuscator: createScriptFile()
NullObfuscator: run()
Wrote C:¥WTK252¥apps¥Tiny¥bin¥Tiny.jar
Wrote C:¥WTK252¥apps¥Tiny¥bin¥Tiny.jad
Build complete
```

索引

D

DefaultKeyboardHandler, 2-12

Q

QwertyKeyboardHandler, 2-12
  割り当て, 2-14

W

workdir, vi

あ

アイコン, 2-7
  inmode の例, 2-8
  画像, 2-8
  場所, 2-8

い

一時停止と再開, 2-16

か

画面
  カラー表示またはグレースケール表示の指定, 2-6
  サイズと位置, 2-4
  全画面モード, 2-6
  範囲, 2-4
  描画可能領域, 2-5
  表示色数, 2-6
ガンマ補正, 2-6

き

キーボードキー
  ボタンへの割り当て, 2-13
キーボードハンドラ, 2-12

け

警告音, 2-11
ゲームキー, 2-14

こ

コマンド
  コマンドメニュー, 2-16
  ソフトウェアボタンの割り当て, 2-15

さ

サウンド, 2-11

す

スキニング, 2-1
スキン
  DefaultColorPhone, 2-1
  画像, 2-2
  作成, 2-1
  プロパティーファイル, 2-1

そ

ソフトウェアボタン
  コマンドの割り当て, 2-15
  排他的に使用, 2-16
  ラベル, 2-10

た

タッチ画面, 2-17

な

難読化ツール
  インタフェース, 3-1
  コード例, 3-2
  ツールキットの設定, 3-2

ひ

ビットマップフォント, 2-10

ふ

フォント, 2-9
  下線の表示, 2-9
  デフォルトフォント, 2-9
  ビットマップフォント, 2-10

ほ

ポインタイベント, 2-17
ボタン, 2-12
  エミュレータアクションへの割り当て, 2-12
  キーの割り当て, 2-13
  四角形, 2-12
  ソフトウェアボタンのラベル, 2-10
  多角形, 2-13

も

文字エンコーティング, 2-17
  プロパティー, 2-18
文字、キーの割り当て, 2-14

ろ

ロケール, 2-17