Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

iアプリコンテンツ開発ガイド　for　DoJa-5.x/5.x LE

～　詳細編　～

第3.00版

平成20年5月27日
株式会社ＮＴＴドコモ

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.
本製品または文書は著作権法により保護されており、その使用、複製、再頒布および逆コンパイルを制限するライセンスのもとにおいて頒布されます。NTT ドコモ（その他に許諾者がある場合は当該許諾者も含めて）の書面による事前の許可なく、本製品および関連する文書のいかなる部分も、いかなる方法によっても複製することが禁じられます。フォントを含む第三者のソフトウェアは、著作権法により保護されており、その提供者からライセンスを受けているものです。

Sun、Sun Microsystems、Java、J2ME および J2SE は、米国およびその他の国における米国 Sun Microsystems,Inc.の商標または登録商標です。サンのロゴマークは、米国 Sun Microsystems, Inc.の登録商標です。

「i モード」、「i アプリ／アイアプリ」、「i-αppli」ロゴ、「DoJa」、「i メロディ／アイメロディ」、「トルカ」、「ToruCa」ロゴは NTT ドコモの商標または登録商標です。

その他、掲載されている会社名、製品名、サービス名は各社の商標または登録商標です。

なお、本書では、コピーライト及び商標・登録商標表記はしておりません。

文書は現状有姿で提供されており、その商品性、特定目的への適合性、第三者の権利の非侵害を含み、明示的または黙示のいかなる条件、表明および保証も行われず、また提供されません。但し、かかる保証の否定が、適用される法律の下で無効である場合は、この限りではありません。

Copyright 2000-2008 NTT DOCOMO, Inc. 2-11-1 Nagata-cho, Chiyoda-ku, Tokyo,100-6150, Japan and Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto,California, 94303, U.S.A. All rights reserved.

本書は、DoJa-5.x プロファイルに対応する Java アプリケーション実行環境（本書中では i アプリ実行環境と呼びます）をターゲットとして記述されています。 i アプリの仕様はプロファイルバージョンで識別できます。現在までのプロファイルバージョンの流れと主要な搭載機種は以下の通りです。

DoJa-1.0 プロファイル：	デジタル・ムーバ 503i シリーズおよび初期の FOMA 携帯電話モデル
DoJa-2.0 プロファイル：	ムーバ 504i シリーズ
DoJa-2.1 プロファイル：	2002 年冬以降発売の FOMA 携帯電話モデル
DoJa-2.2 プロファイル：	2003 年春以降発売の FOMA 携帯電話モデル
DoJa-3.0 プロファイル：	ムーバ 505i シリーズ
DoJa-3.5 プロファイル：	FOMA 900i シリーズ
DoJa-4.0 プロファイル：	FOMA 901i シリーズ
DoJa-4.1 プロファイル：	FOMA 902i シリーズ
DoJa-5.0 プロファイル：	FOMA 903i シリーズ、FOMA 904i シリーズ
DoJa-5.1 プロファイル：	FOMA 905i シリーズ、FOMA 906i シリーズ

また DoJa-5.x プロファイルでは、エントリー向けモデルのために一部機能を省略した DoJa-5.x LE（Limited Edition）プロファイルが派生仕様として存在しています。DoJa-5.x プロファイルと DoJa-5.x LE プロファイルの仕様差分については本書の付録にて解説します。

各機種に搭載されている i アプリ実行環境のプロファイルバージョンは、別途 NTT ドコモよりアナウンスされます。

本書の内容は、以前のプロファイルから存在している機能であっても、以前のプロファイルにのみ対応した携帯電話には適用されませんのでご注意ください。本書で解説するプロファイルには、既存機能に関する未定義の仕様の明確化も含まれています。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

目　次

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

本書の目的

『i アプリコンテンツ開発ガイド』は、i モード携帯電話向けに Java ベースの新しいサービスを提供する i アプリの設計や開発に必要な情報を、i モード向けコンテンツプロバイダに提供するものです。このガイドの目的は、i アプリの作成や開発に役立つシンプルなモデルを提供することにあります。さらに、このガイドには、この技術に関する説明と、使い易い i アプリを作成する上で開発者が準拠すべきプロセスの詳細が述べられています。この巻頭部分では、さらに次の項目について説明しています。

- 対象読者
- 本書の構成
- 表記上の規則
- 関連ドキュメント

対象読者

『i アプリコンテンツ開発ガイド』には、i アプリを利用したサービスの開発者向けの情報が記載されています。したがって、サービスの設計者や、サーバーおよびクライアントプログラムのプログラマ、サービスの導入担当者は、このガイドを読む必要があります。このガイドでは、次の知識を前提としています。

- Java プログラミング言語を使ったネットワークアプリケーションの基礎知識
- 開発した i アプリが実行される i モード携帯電話の製品基本仕様

本書の構成

次の表に、各主題について記載している章を示します。

内容	参照先
i アプリ実行環境の紹介と概要	第 1 章、第 2 章
設計上の考慮事項	第 3 章
仕様の解説	第 4 章～第 14 章
開発プロセス	第 15 章、第 16 章

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

表記上の規則

このガイドで使用されている表記上の規則を次に示します。

表記	意味
タイプ文字	タイプ文字は、そのテキストがコード例、クラス、インタフェース、メソッド、またはデータ型であることを示します。
ボールドテキスト	ボールドは、そのテキストがメニューやソフトキーのオプションであることを示します。
[ ]	書式などの説明において、[ ] で囲まれた内容はオプションであることを示します。実際の指定では、必要に応じてこれらを設定したり省略することができます。
< >	書式などの説明において、< > で囲まれた内容は、その部分を適切な値に置き換えて設定することを示します。

このガイドでは、DoJa-3.0 プロファイルや DoJa-4.0 プロファイルなどプロファイル毎の特記事項を【DoJa-3.0】【DoJa-4.0】といった表記で示しています。特段の注記がない限り、より古いプロファイルの特記事項は新しいプロファイルにおいても適用されます。

関連ドキュメント

このガイドに関連するドキュメントを次に示します。

- 『i アプリコンテンツ開発ガイド for DoJa-5.x/5.x LE　API リファレンス編』
- 『i アプリコンテンツ開発ガイド for DoJa-5.x/5.x LE　i アプリオプション・i アプリ拡張編』
- 『i アプリコンテンツ開発ガイド for DoJa-5.x/5.x LE　オプション・拡張 API リファレンス編』
- 『DoJa5.0 API 向け iαppli Development Kit　ユーザーズガイド』
- 『Java 2 Platform, Micro Edition (J2ME) Connected Limited Device Configuration (CLDC) specification』（ http://java.sun.com/products/cldc/ ）

DoJa-4.0 プロファイル以降では、CLDC のバージョンは 1.1 となります。DoJa-3.5 プロファイル以前では CLDC のバージョンは 1.0 です。

なお、上記に含まれる URL は予告なく変更される場合があります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

第１章
はじめに

ｉアプリ対応携帯電話では、電話やインターネットアクセスの他に、i モードコンテンツプロバイダの Web サイトから Java アプリケーション（ｉアプリ）をダウンロード、実行する機能がサポートされます。このガイドは、ｉアプリを開発するソフトウェアエンジニア向けのプログラミングガイドです。

ｉアプリ対応携帯電話では、Java 2 Platform Micro Edition（J2ME）の Connected, Limited Device Configuration（CLDC）がサポートされます。J2ME CLDC では、小型のネットワーク接続デバイス向けの Java 仮想マシン仕様およびクラスライブラリに関して必要最小限の機能を定義します。CLDC は、K Virtual Machine（KVM）をベースに作成されています。KVM は、携帯電話（本書では特段の注記がない限り、「携帯電話」はｉアプリ対応携帯電話のことを指します）のようにメモリや CPU、消費電力に制約があるデバイス向けに設計されたコンパクトで移植性の高い Java Virtual Machine です。KVM は、Java 2 ファミリにおける上位エディションの Java 実行環境に含まれる機能の多くを共有していますが、小型デバイスの要件を満たすために、小型デバイスの制約に合わせて最適化されています。J2ME/CLDC プラットフォームはコンパクトであるとはいえ、必要な Java Virtual Machine やクラスライブラリを備えていますので、ｉモード向けコンテンツプロバイダは、i モードユーザーが必要に応じてインストールできる内容豊富でエキサイティングなｉアプリを開発できます。

Java には write-once, run-anywhere という特性があり、コンテンツプロバイダの作成したｉアプリは、CPU やオペレーティングシステムに関係なく、どのメーカー・どの機種の携帯電話でも容易に動作させることができます。Java にはアプリケーションをインターネット経由で安全に流通させる手段が備わっているため、ｉモードユーザーは、i モードコンテンツプロバイダの Web サイトに接続するだけでｉアプリをいつでもダウンロードし、インストールすることができます。Java を使えば、ｉモードコンテンツの開発者は、ゲームや強固な電子商取引サービスといった広範囲のサービスを開発することができます。ｉモード携帯電話への Java の搭載により、コンテンツプロバイダはより新しく、よりダイナミックなサービスを展開することが可能になります。

この章では、ｉアプリ対応携帯電話におけるアプリケーション実行環境（ｉアプリ実行環境）の全般的な特徴と機能について説明します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

1.1 iアプリ実行環境の利点

Java コンテンツプロバイダは、iアプリを Java プログラミング言語を使って作成します。iアプリ実行環境では、携帯電話のユーザーインタフェース、通信、テキスト変換、グラフィックス、マルチメディア、携帯電話上へのデータの保管といった処理を制御する API（iアプリ API）が提供されます。iアプリ API は以下の 3 つのカテゴリの API 群から構成されます。

- iアプリ基本 API

  共通仕様として API および動作が規定されており、全ての機種に共通に搭載される API です。

- iアプリオプション API

  共通仕様として API および動作が規定されていますが、搭載有無の判断はメーカーに委ねられる API です。このカテゴリの API の使用にあたっては、API が搭載されていない機種があることを前提にしなければなりません。

- iアプリ拡張 API

  同一の機能を実現するものであっても、API および動作がメーカー毎に異なる可能性のある API です。このカテゴリの API の使用にあたっては、API が搭載されていない機種があることと、機種により API の使用方法などが異なる（同機能を持つ機種間であってもソースコードレベルでの互換性が保持されない）ことを前提にしなければなりません。

【DoJa-2.0】

iアプリ実行環境の API カテゴリ分類は、DoJa-1.0 プロファイルでのカテゴリ分類から変更されています。また、DoJa-2.0 プロファイル以降においては、上記の 3 カテゴリ全ての API について i アプリコンテンツ開発ガイドにて公開されます。

なお、本書「iアプリコンテンツ開発ガイド　詳細編」では主に J2ME/CLDC API および i アプリ基本 API を解説の対象とします。iアプリオプション API および i アプリ拡張 API の詳細については、「iアプリコンテンツ開発ガイド　iアプリオプション・iアプリ拡張編」および「iアプリコンテンツ開発ガイド　オプション・拡張 API リファレンス編」を参照してください。

iアプリ実行環境には、この他に次の機能があります。

- iアプリの動的な配布
- Java Application Manager （JAM）
- シンプルなサンドボックスモデルなどのセキュリティ機能。このモデルでは、`java.lang` パッケージや `java.util` パッケージのクラスなど、安全な Java API セットだけが開発者に提供されます。

iアプリ実行環境は、次の 4 つの API レイヤから構成されています。

- J2ME/CLDC で規定された API
- iアプリ API（iアプリ基本 API）
- iアプリ API（iアプリオプション API）
- iアプリ API（iアプリ拡張 API）

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

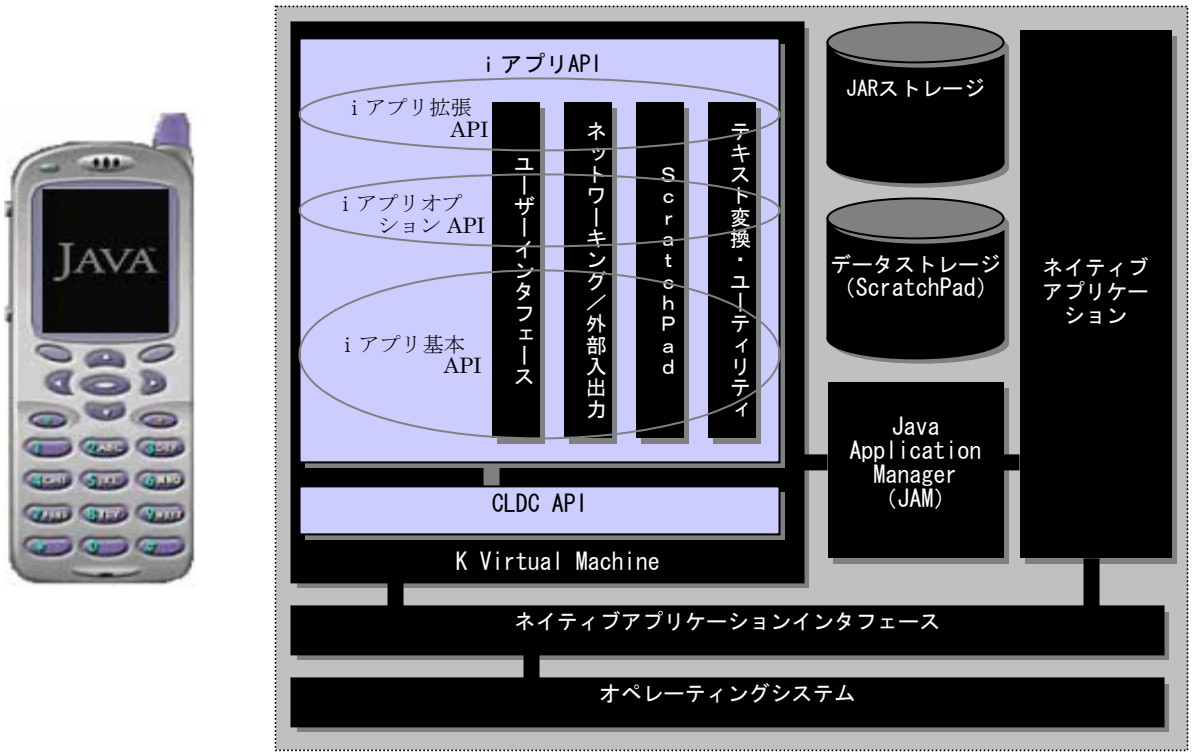


図 1. i アプリ対応携帯電話における i アプリ実行環境

注意事項：

- 多くの i アプリでは、J2ME/CLDC API と i アプリ基本 API だけを使用します。しかし特殊な i アプリの中には、ユーザーへの表現力を増すために i アプリオプション API や i アプリ拡張 API を使用するものがあります。これらの API を使用する i アプリは、メーカーにより動作しない可能性があるため注意が必要です。
- i アプリオプション API、 i アプリ拡張 API は、将来のプロファイルにおいて i アプリ基本 API に取り入れられる可能性のあるものが含まれています。また逆に、将来のプロファイルで削除される可能性のあるものも含まれています。

J2ME/CLDC は、資源に制約がある小型のネットワーク接続デバイス向けの標準的な Java プラットフォームを定義したものです。このようなデバイスには、次の特徴があります。

- メモリの全体量が少ない。
- 消費電力が限られている。一般には電池で動作する。
- 何らかのネットワークに接続される。途切れやすい無線ネットワークに接続されていることも多く、帯域幅も限られている。
- デバイスの提供するサービスに最適化されたユーザーインタフェースが必要となる。

CLDC 仕様でサポートされる一般的なデバイスには、携帯電話、双方向ポケットベル、PDA などの携帯情報端末、家電機器、POS 端末などがあります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

J2ME/CLDC は、 i アプリ実行環境に導入されている安定したアプリケーションプラットフォームです。上位エディションの Java 実行環境と同じように、J2ME でも、Java 技術の利点として知られている次のような特性があります。

- 時、場所、デバイスを選ばない、製品間での実装の一貫性
- 多くの開発者および開発ツールに支えられた高レベルなオブジェクト指向プログラミング言語環境
- コードの移植性
- ネットワークによる安全なアプリケーションの配布
- J2SE や J2EE への上位拡張性

CLDC 上に構築される i アプリ実行環境には、次のような機能を持つ API が含まれています。

- HTTP や HTTPS によるネットワーク接続
- ユーザーインタフェースを定義するコンポーネント
- 低レベルのグラフィックス制御
- データの保存に使用する ScratchPad
- 待ち受け状態の携帯電話に i アプリを常駐させる待ち受けアプリケーション制御
- ｉアプリと他のアプリケーション（ブラウザ、メーラ、 i アプリなど）との連携動作
- 赤外線ポートを利用しての外部接続（IrOBEX、赤外線リモコン）

【DoJa-2.0】

待ち受けアプリケーション、IrOBEX 外部接続、アプリケーション連携は DoJa-2.0 プロファイルにおいて新規に導入された機能です。待ち受けアプリケーションは第 9 章、IrOBEX 外部接続は第 10 章、アプリケーション連携は第 11 章でそれぞれ詳細に解説しています。

【DoJa-3.0】

赤外線リモコン機能および i アプリ間のアプリケーション連携は DoJa-3.0 プロファイルにおいて新規に導入された機能です。赤外線リモコン機能については第 12 章で詳細に解説しています。

1.1.1 スタンドアロン型およびクライアント/サーバー型の最適化されたサポート

ｉアプリサービスでは、スタンドアロン型とクライアント/サーバー型の両方の形態のプログラミングをサポートしています。

スタンドアロン型 i アプリでは、アプリケーションファイルとデータの両方が携帯電話上に存在し、サーバーと対話動作することはありません。ゲームや計算機、ユーティリティの多くはスタンドアロン型 i アプリとして作成することができます。

スタンドアロン型 i アプリでは、その i アプリの本来の目的を達するためにサーバーに接続することはありませんが、i アプリを携帯電話にダウンロードする際には、サーバーとの接続が必要です。ｉアプリを、インターネットを介して携帯電話にダウンロードできるようにするための方法などについては、第 16 章を参照してください。

スタンドアロン型 i アプリにも価値があるとはいえ、魅力ある i アプリの多くはクライアント/サーバー型の形態を取るでしょう。クライアント/サーバー型 i アプリでは、携帯電話の制約された処理能力が、サーバーのはるかに大きな処理能力や強力なデータ検索能力によって拡張されます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.
iアプリ実行環境には、iアプリとコンテンツプロバイダのサーバーをインターネットを介して接続するための、HTTP および HTTPS に基づくネットワーク接続を行う API が含まれています。また、赤外線ポートを介して近距離にある 2 台の携帯電話をクライアントおよびサーバーとして動作させるための IrOBEX API も含まれています。

HTTP(S)接続

HTTP(S)接続を利用するiアプリでは、クライアントコンピュータ（この場合は携帯電話）の処理能力に、インターネット上にあるリモートサーバーの処理能力やデータアクセス機能を結合することができます。携帯電話は処理能力や記憶装置容量が限られていますが、強力な記憶装置とプロセッサを持つサーバーのフロントエンド・プロセッサとして機能することができます。携帯電話にはデータ入力や表示サイズに制限があるものの、適切な設計やプログラミングに基づいてiアプリを作成することにより、iアプリ対応携帯電話のユーザーは、快適に情報にアクセスしたり、電子商取引のようなリモートサービスを受けることができるようになります。

クライアント/サーバー型iアプリにおけるサーバーの代表的な役割は、要求されたデータをデータベースから検索し、それを要求元のクライアントに返すというものです。しかし場合によっては、クライアントの限られた処理能力とメモリ容量では不適切または不可能な計算処理をサーバーで代行することもあります。

HTTP(S)接続を利用したクライアント/サーバー型iアプリの作成については、第 5 章を参照してください。

IrOBEX 外部接続

IrOBEX（以降、単に OBEX と呼びます）外部接続は、携帯電話に装備された赤外線ポートを使用して、近距離の外部機器との間でデータ送受信を行うためのシンプルなクライアント/サーバーモデルを実現します。 iアプリ実行環境は OBEX 外部接続のクライアント API とサーバーAPI の両方を持っており、携帯電話間でのデータの送受信もサポートしています。この機能を利用することで、名刺交換アプリケーションやスケジュール交換アプリケーションのような、携帯電話間で小さなデータをやりとりするiアプリを実現することができます。

OBEX 外部接続では、接続先の機器のハードウェア条件や周囲の環境などによっては通信を行えない場合があります。

なお、以降は特に注記のない場合クライアント、サーバーはそれぞれ HTTP(S)通信を行う場合のクライアント、サーバーを指すものとします。OBEX においてはそれぞれ OBEX クライアント、OBEX サーバーと記載します。

1.1.2 永続データのローカル記憶装置サポート

iアプリ実行環境では、ScratchPad と呼ぶローカル記憶装置のメカニズムがサポートされます。このメカニズムでは、永続性を持つデータ（iアプリの実行が終了しても保持され続けるデータ　永続データとも言います）の保管領域が携帯電話内の記憶装置に確保されます。永続データを格納する必要があるスタンドアロン型iアプリは ScratchPad にデータを格納しますが、クライアント/サーバー型iアプリは、通常は一部の情報を ScratchPad に格納するだけで、残りのデータはサーバー側に格納します。

サーバー側の永続データは、ほとんどの場合、データベースに格納されます。クライアントデータは HTTP リクエストでサーバーに送信され、HTTP リクエストがサーブレットや CGI スクリプトと対話します。そして、サーブレットや CGI スクリプトが HTTP リクエストとデータベースアクセスプロトコルとの仲介として働きます。このため、サーバー側には常に HTTP サーバーが必要です。

永続データの格納場所を決定するには、データの性質を考慮する必要があります。永続データを失っても影響が小さい場合は、ScratchPad に格納し、そうでない場合は、サーバー側に格納します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

ScratchPad に保存されたデータは、携帯電話のハードウェア的な故障などの原因によって失われてしまう可能性があるためです。

ｉアプリ対応携帯電話のセキュリティ機能の一つに、Java Application Manager（JAM：次項参照）によって個々のｉアプリに個別に割り当てられた ScratchPad メモリの領域は、他のｉアプリからはアクセスできないという点があります。また、携帯電話上で同時に動作可能なｉアプリは 1 つだけであり、複数のｉアプリがアプリケーション実行メモリ上でデータを共有することもできません。複数のｉアプリ間でデータを共有する場合は、サーバーを各ｉアプリ共通のデータ格納領域として使用する必要があります。

1.2 Java Application Manager

ｉアプリ実行環境が搭載されている携帯電話には、携帯電話にインストールされているｉアプリを管理するコンポーネントが含まれています。このコンポーネントは Java Application Manager（JAM）と呼ばれ、次のような機能を持っています。

- 携帯電話に格納されているｉアプリをリスト表示する。
- ｉアプリの実行管理(起動や強制終了、ｉアプリ実行環境と他のアプリケーションの間の仲介など)を行う。
- ｉアプリをインストールまたは更新（バージョンアップ）する。
- 携帯電話に格納されているｉアプリを削除する。いったん携帯電話上に保存されたｉアプリの削除は、ユーザーの明示的な指示に基づいてのみ行われます。

JAM は KVM から独立したネイティブコンポーネントであり、これをｉアプリのプログラムロジック（以降、単にアプリケーションプログラムと呼びます）から制御することはできません。

ｉアプリを終了させようとしても終了しない場合は、携帯電話の終話キーを押すことによって、JAM にｉアプリの強制終了を指示することができます。

【DoJa-2.0】

DoJa-2.0 プロファイル以降では、ｉアプリの強制終了は終話キーに統一されます。なお、DoJa-1.0 プロファイルでは強制終了のキーアサインはメーカーにより異なります。

1.3 ｉアプリ対応携帯電話の Java セキュリティモデル

CLDC に用意されたセキュリティ機能の他に Java Application Manager（JAM）では、ｉアプリ実行環境特有の機能として、次のセキュリティ機能が提供されます。

- 携帯電話上では、同時に 1 つのｉアプリしか動作できません。そのため、あるｉアプリが別のｉアプリに干渉したり、別のｉアプリのデータにアクセスしたりすることはできません。ただし KVM はマルチスレッドをサポートしており、1 つのｉアプリの中で通信とユーザーインタフェース処理を並行して行うといったマルチスレッドプログラミングは可能です。
- JAM は KVM から独立したネイティブコンポーネントであり、ｉアプリからこれを制御することはできません。ｉアプリのダウンロードや管理は JAM が行います。そのためユーザーにとっては、ダウンロードしたｉアプリの不正な行為やバグによる誤動作などから携帯電話が保護されることになります。.
- ｉアプリは、携帯電話の記憶装置（ScratchPad）に、制限されたアクセスしかできません。あるｉアプリに割り当てられた ScratchPad の領域に、他のｉアプリからアクセスすることは許可されません。したがってダウンロードされたｉアプリが、他のｉアプリのために ScratchPad に保存された個人情報などをユーザーの知らないうちに読み込んで別のサーバーに送信したり書き換えたりするなどの不正な行為を行うことはできません。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- ｉアプリが電話帳などの個人情報を含んだネイティブデータを参照することはできません。また、ブラウザやダイヤラーなど携帯電話上の他のアプリケーションにアクセスする機能では、ユーザーの意思に反して Web ブラウジングや通話発信を行うことのないよう、API 呼び出し時に必ずユーザーに動作への同意を求めます。そのためユーザーは、ｉアプリを安心してダウンロードおよび実行することができます。

  【DoJa-2.0】

  DoJa-1.0 プロファイルでは、セキュリティ上の問題からｉアプリと他のアプリケーションの連携動作をサポートしませんでした。DoJa-2.0 プロファイル以降では、ｉアプリと他のアプリケーション間の連携動作が行われる際に必ずユーザーの同意を得る機構を取り入れ、ユーザーにとって安全なアプリケーション連携をサポートします。

- ｉアプリ実行環境の通信機能では、ｉアプリ対応携帯電話でサポートされる HTTP 通信および HTTPS 通信に対する簡単で使いやすいインタフェースが提供されます。ｉアプリは、そのｉアプリ自身のダウンロード元であるサーバーとしか通信できません。このネットワークセキュリティ機能により、ｉアプリが、ユーザーの知らない別のサーバーに対して情報を送信するようなことはありません。

【DoJa-3.0】

DoJa-3.0 プロファイル以降では、認定されたｉアプリに対してのみこれらのセキュリティ機能を緩和するトラステッドｉアプリ（サービス名"ｉアプリ DX"）の機構が取り入れられています。トラステッドｉアプリは、電話帳などのネイティブデータへのアクセスやｉアプリダウンロード元以外のサーバーとの通信など、緩和されたセキュリティ条件下で動作することができます。ただしトラステッドｉアプリといえども、電話帳などの特定個人に結びつくデータについては、携帯電話の外に流出することを防止するため一定の制約の元でのみアクセスが可能となります。

これらの機能（トラステッド API）は、ｉモードサーバーに登録された、ｉモードメニュー上のサービスでのみ使用することができます。それ以外のｉアプリでは、トラステッド API の機能を使用することはできません。

本書では、トラステッドｉアプリに関する詳細は記載しません。

1.4 ｉアプリのインストールとライフサイクル

携帯電話上でｉアプリを実行するためには、コンテンツプロバイダのｉモード対応 Web サイトからｉアプリをダウンロード（インストール）しなければなりません。ｉアプリがダウンロードされると、ｉアプリの保存のために必要な各メモリを JAM が割り当てます。ｉアプリの JAR ファイルと ScratchPad は、どちらも携帯電話の不揮発性メモリに格納されます。

次の各項では、ｉアプリのダウンロード、実行、削除といったライフサイクルについて説明します。

1.4.1 ｉアプリのダウンロード

ｉアプリのダウンロードは一般的に以下のような流れで行われます。なお、携帯電話の具体的な動作の詳細はメーカーにより一部異なる場合があります。

1. ｉモードブラウザを使って、必要なｉアプリのダウンロードリンク先を決めます。

ｉアプリのダウンロードは、ｉモードブラウザの操作を契機として開始されます。ダウンロードを行うための Web ページ（HTML コンテンツ）には、通常、ｉアプリの説明文や、ダウンロードを開始するためにクリックするリンクが表示されています。リンクには、アプリケーションディスクリプタファイル（".jam"拡張子をもつファイル　ADF、JAM ファイルとも呼ばれます　以降本書では ADF とします）への参照が含まれています。ADF は SJIS コードのテキストファイルであり、その ADF に対応付けられているｉアプリの情報を表すキーと値のペアが 1 行ずつ記述されています。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

このファイルの目的の 1 つは、ダウンロードの開始前に、選択された i アプリを携帯電話に正しくダウンロードできるかどうかを JAM が判定できるようにすることです。事前に正しくダウンロードできないと判定できれば、i アプリを実際に携帯電話にダウンロードするコストが節約されます。数 K～数 10K バイト以上の i アプリに対して ADF は 100～300 バイト程度と小さく、 i アプリ全体をダウンロードする前に ADF をダウンロードして保存に必要な記憶容量やその機種に対する互換性をチェックすることは、通信コストの点から有利となります。

なお ADF の内容に関する詳細は 15.5.1 項を、 i アプリを携帯電話に配布するための Web サーバーの準備については第 16 章を参照してください。

2. リンクをクリックして、インストールプロセスを開始します。

このユーザー操作により、ブラウザは、JAM に ADF の位置を示す URL を渡します。以降、 i アプリのダウンロードとインストールは JAM により行われます。

3. ADF の内容を検査し、 i アプリのインストール可否を確認します。

JAM は、指定された ADF にすべての必須キーが指定されているかチェックします。そして ADF の記述内容（JAR ファイルサイズや ScratchPad サイズ、ターゲットとする機種情報など）から、その i アプリをその携帯電話にインストール可能かどうかチェックし、可能であれば次のステップに進みます。

なお以下の場合には、i アプリはすでに携帯電話にインストールされているものと見なされ新規にインストールすることはできません（1.4.3 項に示す i アプリの更新は可能です）。

- インストールしようとしている i アプリと同じ URL の ADF を持つ i アプリがすでに携帯電話にインストールされている場合（DoJa-3.0 プロファイル以降では、このような i アプリのダウンロードが指示されると携帯電話はその i アプリを更新可能か調べ、可能であれば 1.4.3 項に示す i アプリの更新処理に移ります）
- インストールしようとしている i アプリと異なる URL の ADF に対応付けられている、同じ名前（ADF の `AppName` キー）および JAR ファイル URL（ADF の `PackageURL` キー）を持つ i アプリがすでに携帯電話にインストールされている場合

4. i アプリのダウンロードが行われます。

i アプリが携帯電話上にインストール可能であると判断されたら、JAM は ADF の `PackageURL` キーを使って JAR ファイルの URL を入手し（`PackageURL` キーが相対 URL で記述されている場合は、ADF の位置をベース URL として使用します）、HTTP ダウンロードプロセスを開始します。もしダウンロードプロセス中にユーザーの中断指示や通信エラーが発生したら、JAM はダウンロード中の i アプリを破棄して携帯電話をダウンロードが行われる前の状態に戻します。

5. インストール済み i アプリのリストを更新します。

JAM は、インストール済み i アプリのリストに前項でインストールされた i アプリを追加します。また ADF の情報と JAR ファイルを不揮発メモリ上に保存します。

ADF で i アプリが HTTP 通信などユーザーの許可が必要な機能を使用するように設定されている場合は、ここで i アプリにそれらの機能の使用を許可するかをユーザーに問い合わせます。また ADF で i アプリが待ち受けアプリケーションとして動作可能となるよう設定されている場合は、その i アプリを待ち受けアプリケーションとして登録するか否かをユーザーに問い合わせます。

1.～5.の過程を経て i アプリは携帯電話上にインストールされ、利用可能（実行可能）な状態になります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

【DoJa-3.0】

DoJa-3.0 プロファイル以降では、i アプリのダウンロードから起動までを 1 クリックで行うためのダウンロード即起動機能がサポートされます（そのような i アプリをダウンロード即起動 i アプリと呼びます）。ダウンロード即起動 i アプリは一切のユーザー確認なしにダウンロードから起動までが行われるかわりに、本来ユーザー確認が必要とされている機能の利用には制限があります。

ダウンロード即起動 i アプリの詳細については第 16 章を参照してください。

【DoJa-5.0】

DoJa-5.0 プロファイル以降、トルカのバージョン 2.0 フォーマットに対応している携帯電話では、トルカ（詳細）の HTML コンテンツ部分に i アプリダウンロード用のタグ（16 章を参照）を記述することでトルカビューア経由で i アプリをダウンロードすることもできます。

1.4.2 i アプリの起動

i アプリの通常の起動手順やユーザーインタフェースはメーカーによって異なりますが、一般には次のようにして行われます。

1. ユーザーの指示によりインストール済み i アプリのリストを表示します。
2. ユーザーがリストから起動する i アプリを選択します。
3. JAM が ADF の `AppClass` キーで示される i アプリのメインクラス名を指定して Java Virtual Machine を起動します。
4. 携帯電話のディスプレイに i アプリの画面が表示されます。

また、上記のユーザー操作による i アプリの起動だけではなく、i アプリ実行環境は以下のような起動形態のバリエーションを持っています。

- タイマー起動
- アプリケーション連携による、ブラウザ、メーラ、外部機器（赤外線ポートなど）、i アプリからの i アプリ起動
- 待ち受けアプリケーションの登録による、待ち受け状態復帰時の自動起動
- ダウンロード即起動

タイマー起動については、15.5.1 項の `LaunchAt` キーの項を参照してください。またアプリケーション連携については第 11 章を、待ち受けアプリケーションについては第 9 章を参照してください。また、ダウンロード即起動 i アプリについては第 16 章を参照してください。

【DoJa-2.0】【DoJa-3.0】

DoJa-1.0 プロファイルでは、ユーザー操作による起動とタイマー起動のみをサポートしていました。DoJa-2.0 プロファイル以降では、アプリケーション連携によるブラウザ、メーラ、外部機器からの i アプリ起動および待ち受けアプリケーションの待ち受け時起動が追加されています。また、i アプリからの i アプリ起動、およびダウンロード即起動は DoJa-3.0 プロファイルで追加されました。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

1.4.3 ｉアプリの更新

ｉアプリの更新は、ユーザーがインストール済みｉアプリのリストから、メニューなどによりｉアプリ更新を指示することにより行われます（詳細な操作方法はメーカーにより異なります）。

ｉアプリをインストールした際に、JAM は ADF の URL を保存しています。ユーザーがｉアプリの更新を指示すると、JAM は再度この URL にアクセスし、新たな ADF を取得します。そしてその内容に従ってｉアプリを更新する必要があるかどうかを判定し、必要と判断された場合のみ実際の更新処理を行います。

バグの修正や新しい機能の追加を行うなどの理由でｉアプリを更新する場合には、コンテンツプロバイダは次のことを行う必要があります。

- ADF の `LastModified` キーに前のバージョンより後の日時を設定します。その他のキーは必要に応じて適切な値に設定します。ただし ScratchPad サイズ（`SPsize` キー）の値は、前のバージョンより小さい値は指定しないようにしてください（ScratchPad を分割管理している場合、分割された個々のサイズが以前のものより小さくならないようにします）。更新しようとしているｉアプリの ScratchPad サイズが以前のものと同じか大きい場合に限り、以前の ScratchPad の内容が更新後のｉアプリに引き継がれます。
- コンテンツプロバイダの Web サイトに更新済み JAR ファイルと ADF ファイルを登録します。新しい ADF の URL は以前のものと同じにします。ADF の URL を変更すると、すでに携帯電話にｉアプリをダウンロードしているユーザーがｉアプリの更新を行えなくなるため注意が必要です。

ｉアプリの更新を指示されても、ADF の `LastModified` キーが更新されていなければ、JAM はｉアプリの更新は行いません。

なお、DoJa-3.0 プロファイルではユーザー操作によるｉアプリ更新の他に、API を使用してｉアプリからユーザーにｉアプリ更新を促す機能が追加されています。この機能は、ｉアプリ更新のためのネイティブコンポーネントをｉアプリから呼び出すアプリケーション連携の 1 つとして実現されています。ｉアプリ更新 API については第 11 章を参照してください。

【DoJa-3.0】

ｉアプリ更新 API を持たない DoJa-2.0 プロファイル以前の機種に対しては、ユーザーにｉアプリの更新を促すためには Web サイトやメールなどを使用して告知を行う必要があります。

1.4.4 ｉアプリの削除

ユーザーの要求によりｉアプリを削除する際の手順は、メーカーによって異なります。JAM はｉアプリの削除をユーザーに指示されると、携帯電話上の JAR ファイルおよびそのｉアプリに割り当てられた ScratchPad を削除します。

1.4.5 携帯電話本体と外部メモリ間のｉアプリの移動

FOMA 902iS シリーズ以降では、オプション機能として、携帯電話本体と外部メモリの間でのｉアプリ（ｉアプリや ScratchPad、ADF などの付随情報一式）の移動をサポートする機種があります。ユーザーはこの機能を利用することで、携帯電話本体にあるｉアプリを外部メモリ上に退避したり、外部メモリ上にあるｉアプリを携帯電話本体内（移動元とは別の機体でも可）に戻したりすることができるようになります。

ただしコンテンツ制作者の権利保護のため、ｉアプリの移動は以下のような制約下で行われます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- 携帯電話本体から外部メモリに i アプリを移動する際、携帯電話本体内にあった i アプリは削除されます。また、外部メモリ上では i アプリは暗号化された形で保存されており、i アプリのファイルの内容を PC などで読み出すことはできません。
- i アプリを外部メモリから携帯電話本体（当初その i アプリがダウンロードされた機体以外の機体も含みます）に戻す際、当初その i アプリがダウンロードされる際に使用された UIM カードが携帯電話に挿入されている必要があります。またその後、その i アプリを移動先の携帯電話上で実行する際も、ダウンロード時に使用された UIM カードが携帯電話に挿入されている必要があります。

なお本機能をサポートする携帯電話では、i アプリが自分自身について、外部メモリを経由して移動されたことがあるかどうかを調べるための API が提供されます。これら API の詳細は、API リファレンスの `IApplication` クラス（`isMoved()`、`isMovedFromOtherTerminal()`、`clearMoved()`の各メソッド）を参照してください。

なお、外部メモリから携帯電話本体に i アプリが移動される際には、移動先の携帯電話上で新たにその i アプリをダウンロードする場合と同様のチェックが行われます（同じ i アプリが存在していないか、 i アプリや ScratchPad、FeliCa チップ上に必要なリソースは確保可能か　など）。これらの条件に該当した場合は、その i アプリを携帯電話上に移動することはできません。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

第 2 章
i アプリ実行環境

この章では、 i アプリ対応携帯電話に搭載される Java アプリケーション実行環境（ i アプリ実行環境）の機能構成、API 構成の概要を解説します。

2.1 i アプリ実行環境

i アプリ実行環境は、携帯電話のネイティブアプリケーション、Java Application Manager、 i アプリ API、K Virtual Machine、および CLDC API から構成されています。次に i アプリ実行環境の構成を示します。

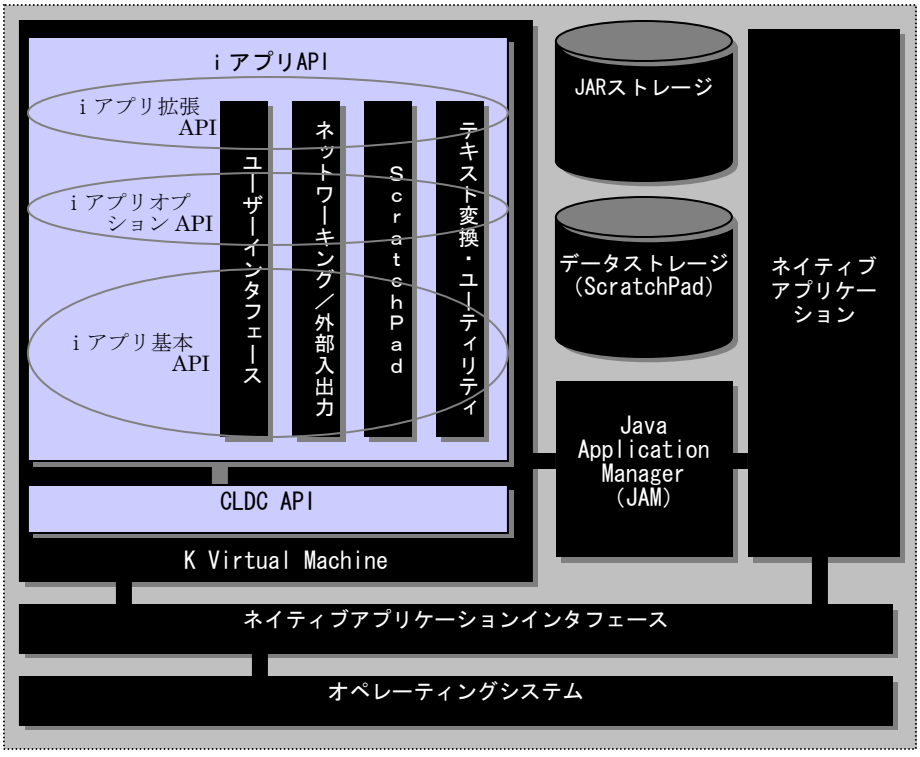


図 2: i アプリ実行環境

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

2.1.1 ネイティブアプリケーションとオペレーティングシステム

通常の電話や従来のブラウザベースのｉモードサービスは、携帯電話上のオペレーティングシステムおよびネイティブアプリケーション（ブラウザ、ネットワーキングコンポーネント、ダイヤラーなど）を使って行われます。ｉアプリ実行環境には、JAM を通じてこれらのネイティブアプリケーションと連携動作するための機能も含まれています。

2.1.2 Java Application Manager

Java Application Manager（JAM）により、JAR ファイルとして格納されている個々のｉアプリの管理や K Virtual Machine との通信が行われます。

Java Application Manager の主な機能については 1.2 項を参照してください。

2.1.3 K Virtual Machine

K Virtual Machine は Java Virtual Machine を小型組み込みデバイス向けに再設計したもので、メモリや CPU 能力、消費電力に制約があるデバイスで使用するように設計されています。

2.1.4 CLDC API

CLDC API には、Java 2 Standard Edition（J2SE）から継承されたサブセットクラスが含まれています。CLDC API は上位エディションの Java 実行環境に含まれる機能の多くを共有していますが、小型デバイスの要件を満たすために小型デバイスの制約に合わせて最適化されています。

CLDC API については、2.2.2 項を参照してください。

【DoJa-4.0】

DoJa-4.0 プロファイル以降のｉアプリ実行環境では、CLDC のバージョンが 1.1 となります（DoJa－3.x プロファイル以前のｉアプリ実行環境では 1.0）。CLDC-1.1 では浮動小数点数サポートなどの機能追加が行われており、関連する API の追加も行われています。

2.1.5 ｉアプリ API

ｉアプリ API は、ｉアプリ対応携帯電話向けに設計されたものです。CLDC が小型組み込みデバイスで共通的に使用可能なアプリケーション基盤を提供しているのに対して、ｉアプリ API では CLDC を土台として、ｉアプリサービスに特化したアプリケーション基盤を提供しています。ｉアプリ API は、次のような要素から構成されています。

- ユーザーインタフェース API
- ネットワーキング API
- SJIS テキスト処理
- ScratchPad ローカル記憶装置
- アプリケーション連携起動 API およびネイティブ機能呼び出し API
- 赤外線ポート制御（OBEX による外部接続および赤外線リモコン）

ｉアプリ API に含まれる API については、2.2.3 項を参照してください。また各機能の具体的な利用方法は第 4 章以降で詳細に解説されています。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

なおｉアプリ API は、以下の 3 つのカテゴリに分類されます。

- 共通仕様として API および動作が規定されており、全ての機種に共通的に搭載されるｉアプリ基本 API
- 共通仕様として API および動作が規定されているが、搭載有無の判断がメーカーに委ねられるｉアプリオプション API
- 同一の機能を実現するものであっても、API および動作がメーカー毎に異なる可能性のあるｉアプリ拡張 API

全機種で動作することが求められるｉアプリでは、ｉアプリオプション API およびｉアプリ拡張 API は使用しないようにしてください。

2.2 ｉアプリ実行環境における API の概要

次の各項では、ｉアプリ実行環境でサポートされる API について説明します。この節は、次の各項に分割されています。

- 2.2.1 ネットワークアーキテクチャ
- 2.2.2 CLDC API
- 2.2.3 ｉアプリ API

CLDC API には標準 J2SE ライブラリのサブセットクラス、および機能的には J2SE にマッピング可能ですが小型デバイス向けに再設計された CLDC 独自のクラスが含まれています。これらは Java Community Process（JCP）を経て策定されました。

一方、ｉアプリ API はｉモードサービスに特化した API 群であり、NTT ドコモにより規定されています。

なお、本項ではｉアプリ API のうちｉアプリ基本 API についてのみ記載します。ｉアプリオプション API およびｉアプリ拡張 API の詳細については、「ｉアプリコンテンツ開発ガイド　ｉアプリオプション・ｉアプリ拡張編」および「ｉアプリコンテンツ開発ガイド　オプション・拡張 API リファレンス編」を参照してください。

2.2.1 ネットワークアーキテクチャ

ｉアプリサービスのネットワークアーキテクチャは、従来のｉモードサービスにおけるネットワークアーキテクチャとほぼ同様です。携帯電話とコンテンツプロバイダサイトの間のアプリケーションレベルの通信は、HTTP を使用して行われます。以下にネットワークアーキテクチャの概念図を示します。

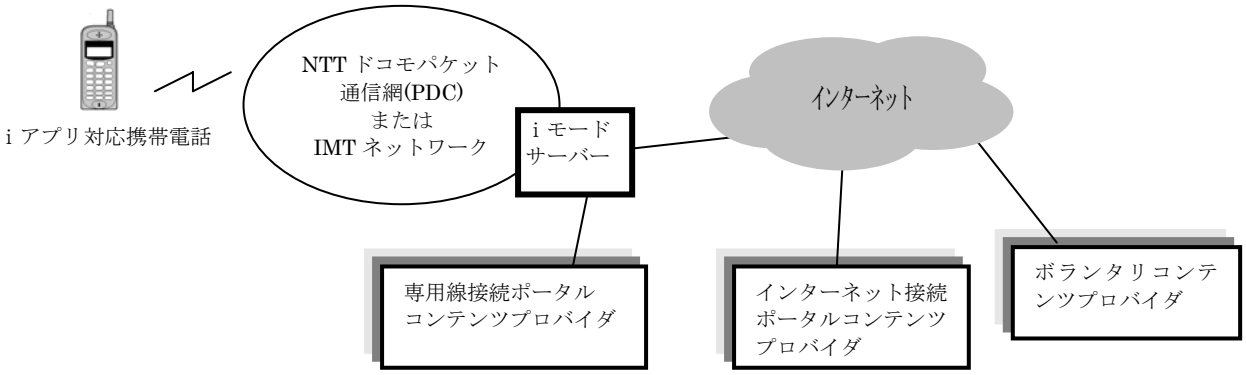


図 3: ネットワークアーキテクチャ概念図

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

ｉアプリサービスに関するネットワークオペレーションは以下の２種類に分類されます。

- ｉアプリのダウンロードとインストール
- 実行中のｉアプリからのサーバーアクセス

ｉアプリサービスのネットワークオペレーションは、従来のｉモードサービスと同様にｉモードサーバーを介して行われます。携帯電話上でもアプリケーションレベルの通信プロトコルでは HTTP を使用しており、ｉモードサーバーではプロトコル変換に類する処理は行いません。ダウンロードされるｉアプリも、コンテンツプロバイダ側 Web サイトに搭載されているファイルがそのまま携帯電話まで送信されます。

またｉアプリ対応携帯電話では、携帯電話側にて SSL（Secure Sockets Layer）によるサーバー認証をサポートしています。ｉアプリ対応携帯電話の通信機能と SSL サポートを利用することで、セキュア HTTP（HTTPS）通信を行うこともできます。この場合のセキュア区間は、携帯電話からコンテンツプロバイダ側 Web サーバーまでの間となります。

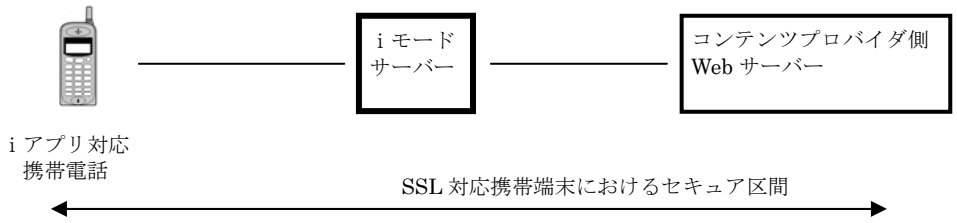


図 4: ｉモードサービスの SSL サポート

初期の機種を除く FOMA 携帯電話では、ユーザー毎に発行されたクライアント証明書を UIM カードに格納し、インターネットアクセスの際に SSL クライアント認証機能を提供している FirstPass を利用することができます。FirstPass に対応した FOMA 携帯電話上では、ｉアプリも FirstPass のクライアント証明書を使用したクライアント認証を利用することができます。その場合は、Web サーバー側で FirstPass に対応するためにドコモルート CA 証明書をインストールする必要があります。 FirstPass については、以下の URL を参照してください。

http://www.nttdocomo.co.jp/service/anshin/firstpass/index.html

FirstPass に対応していない携帯電話（全ての PDC 携帯電話機種を含みます）では、クライアント認証には対応しておりません。

2.2.2 CLDC API

CLDC に含まれているクラスライブラリの大半は、Java アプリケーションの上位互換性と移植性を確保するため、上位エディションの Java 実行環境（J2SE）のサブセットになっています。上位互換性は非常に望ましい目標ですが、J2SE ライブラリには、セキュリティ、入出力、ユーザーインタフェース、ネットワーキング、記憶装置といった重要な分野でサブセットを作成することを非常に困難にする内部的な強い依存関係があります。これらの依存関係は、Java クラスライブラリの発展とともに必要になった設計の革新と再利用の必然的な結果です。残念ながら、この依存関係があるために、ライブラリの一部だけを使用し、他の部分を使用しないということは非常に困難です。そのため、ライブラリの一部、特にネットワーキングと入出力の部分は再設計されています。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

CLDC でサポートされる J2SE クラス

CLDC には、Java 2 Standard Edition（J2SE）から継承されたクラスが多数あります。これらのクラスの機能や、継承されたメソッドの使用方法に変更はありません。これらのクラスには、対応する J2SE クラスにない public や protected のメソッドやフィールドは追加されていません。これは、J2ME コンフィグレーションの規則により、J2SE と同じパッケージ名とクラス名をもつクラスは、対応する J2SE クラスの厳密なサブセットでなければならず、独自のメソッドやフィールドを追加することは許されていないためです。

システムクラス

標準の Java クラスライブラリには、Java Virtual Machine と密接に結びついたクラスが含まれています。またいくつかの標準的な Java ツールも、特定のクラスがシステムに存在するものと仮定しています。たとえば標準的な Java コンパイラは、クラス `String` と `StringBuffer` の特定のメソッドを使用するコードを生成します。

```
java.lang.Object
java.lang.Class
java.lang.Runtime
java.lang.System
java.lang.Thread
java.lang.Runnable （インタフェース）
java.lang.String
java.lang.StringBuffer
java.lang.Throwable
```

データ型クラス

CLDC では、`java.lang` パッケージの次の基本データ型クラスがサポートされます。これらのクラスはそれぞれ、Java 2 Standard Edition の対応するクラスの厳密なサブセットです。

```
java.lang.Boolean
java.lang.Byte
java.lang.Short
java.lang.Integer
java.lang.Long
java.lang.Character
java.lang.Double
java.lang.Float
```

【DoJa-4.0】

DoJa-4.0 プロファイル以降の i アプリ実行環境に搭載されている CLDC -1.1 では浮動小数点数がサポートされており、それに伴いデータ型クラスに `Double` クラスと `Float` クラスが追加されています。

参照オブジェクトクラス

CLDC では、`java.lang.ref` パッケージの次の参照オブジェクトクラスがサポートされます。CLDC-1.1 では弱参照オブジェクトのサブセットがサポートされます。

```
java.lang.ref.Reference
java.lang.ref.WeakReference
```

【DoJa-4.0】

DoJa-4.0 プロファイル以降の i アプリ実行環境に搭載されている CLDC -1.1 では J2SE の参照オブジェクトクラスのサブセットがサポートされており、それに伴い上記パッケージとクラスが追加されています。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

Collection クラス

CLDC では、java.util パッケージの次の Collection クラスがサポートされます。

java.util.Vector
java.util.Stack
java.util.Hashtable
java.util.Enumeration （インタフェース）

入出力クラス

CLDC では、java.io パッケージの次のクラスがサポートされます。国際化のサポートには、Reader、Writer、InputStreamReader、InputStreamWriter の各クラスが必要です。

java.io.InputStream
java.io.OutputStream
java.io.PrintStream
java.io.ByteArrayInputStream
java.io.ByteArrayOutputStream
java.io.DataInput （インタフェース）
java.io.DataOutput （インタフェース）
java.io.DataInputStream
java.io.DataOutputStream
java.io.Reader
java.io.Writer
java.io.InputStreamReader
java.io.OutputStreamWriter

Micro Edition クラス

CLDC では、javax.microedition.io パッケージの次のクラスがサポートされます。

javax.microedition.io.Connector
javax.microedition.io.ConnectionNotFoundException
javax.microedition.io.Connection （インタフェース）
javax.microedition.io.ContentConnection （インタフェース）
javax.microedition.io.Datagram （インタフェース）
javax.microedition.io.DatagramConnection （インタフェース）
javax.microedition.io.InputConnection （インタフェース）
javax.microedition.io.OutputConnection （インタフェース）
javax.microedition.io.StreamConnection （インタフェース）
javax.microedition.io.StreamConnectionNotifier （インタフェース）

カレンダと時間のクラス

CLDC には、J2SE の標準クラスである java.util.Calendar、java.util.Date、java.util.TimeZone の小さなサブセットが含まれています。CLDC では、時間帯として GMT がサポートされます。他の時間帯については実際の実装で提供できます。 i アプリ実行環境では追加の時間帯として JST をサポートしています。また、国際ローミングに対応する携帯電話の中には、日本以外の国・地域に対する時間帯をサポートする場合があります。

java.util.Calendar
java.util.Date
java.util.TimeZone

追加のユーティリティクラス

CLDC には、追加のユーティリティクラスが 2 つあります。java.util.Random クラスは、ゲームなどの Java アプリケーションを実装するときに使用する簡単な擬似乱数ジェネレータです。java.lang.Math クラスには、数々の数値演算メソッドが用意されています。

java.lang.Math
java.util.Random

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

エラーと例外クラス

CLDC のクラスライブラリは一般には J2SE のクラスライブラリと高度な互換性をもつことが定められているため、CLDC に含まれているクラスは、対応する J2SE クラスと全く同じ例外をスローします。このため CLDC のクラスライブラリには、数多くの例外クラスが組み込まれています。

エラークラス

```
java.lang.Error
java.lang.VirtualMachineError
java.lang.OutOfMemoryError
java.lang.NoClassDefFoundError
```

【DoJa-4.0】

`java.lang.NoClassDefFoundError` は、CLDC -1.1 にて追加されました。

例外クラス

```
java.lang.Exception
java.lang.ClassNotFoundException
java.lang.IllegalAccessException
java.lang.InstantiationException
java.lang.InterruptedException
java.lang.RuntimeException
java.lang.ArithmeticException
java.lang.ArrayStoreException
java.lang.ClassCastException
java.lang.IllegalArgumentException
java.lang.IllegalThreadStateException
java.lang.NumberFormatException
java.lang.IllegalMonitorStateException
java.lang.IndexOutOfBoundsException
java.lang.ArrayIndexOutOfBoundsException
java.lang.StringIndexOutOfBoundsException
java.lang.NegativeArraySizeException
java.lang.NullPointerException
java.lang.SecurityException
java.util.EmptyStackException
java.util.NoSuchElementException
java.io.EOFException
java.io.IOException
java.io.InterruptedIOException
java.io.UnsupportedEncodingException
java.io.UTFDataFormatException
```

CLDC プラットフォームではサポートされない JVM 機能

仮想マシンのサイズを小さくするために、CLDC の仮想マシン（KVM）では次の機能が標準の Java Virtual Machine から削除されています。

JNI

KVM では Java Native Interface（JNI）はサポートされません。

スレッドグループ

KVM にはマルチスレッド機能が実装されていますが、スレッドグループはサポートされません。したがって、スレッドの開始や停止などのスレッド操作は、個々のスレッドオブジェクトにしか適用できません。開発者がス

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

レッドグループに対してスレッド操作を行いたい場合は、アプリケーションレベルでコレクションオブジェクトを明示的に使用してスレッドオブジェクトをグループ管理する必要があります。

ファイナライズ

CLDC では `Object` クラスに `finalize()`メソッドが含まれておらず、KVM ではオブジェクトのファイナライズはサポートされません。また CLDC では、`finalize()`メソッドを定義することはできません。

【DoJa-4.0】

DoJa-4.0 以降の i アプリ実行環境では、CLDC-1.1 の浮動小数点サポートにより浮動小数点数を扱うことができます。これに対し、CLDC-1.0 をプラットフォームとする DoJa-3.x 以前の i アプリ実行環境では、浮動小数点数を取り扱うことはできません。

Generic Connection フレームワーク（javax.microedition.io パッケージ）

J2ME の実装サイズを小さくするために、J2SE のネットワーククラスと入出力クラスを汎用化する必要がありました。この新しいフレームワークの目標は、これが J2SE におけるネットワーク機能や入出力機能の厳密なサブセットとして機能するようにすることです。このサブセットは、一般的な低レベルハードウェアや任意の J2SE 実装に容易にマップ可能であると同時に、新しいデバイスやプロトコルのサポートでは、よりよい拡張性、柔軟性、一貫性を備えています。さまざまな種類の通信先に対し個々の概念に基づくアクセス手段を提供するのではなく、次に示すような統合された概念に基づいたアクセス手段がアプリケーションプログラミングレベルで使用されます。

すべての接続は、`Connector` クラスのクラスメソッド `open()`を使って行われます。接続が正常に行われると、`open()`メソッドは Generic Connection インタフェース群（`Connection` インタフェースをルートとするサブインタフェース群）の中の 1 つを実装したオブジェクトを返します。このインタフェース群は、図 5 に示されるように多数のインタフェースから構成されています。

`open()`メソッドは、URL パラメータを次の一般的な形式で受け取ります。

```
Connector.open("<protocol>:[target][parameters]");
```

この設計の目的は、アクセス対象毎の初期化の違いをできる限り、アクセス対象のタイプを特徴付ける文字列（この場合 URL 文字列）に集約することです。`Connector.open()`へのパラメータにはこの URL 文字列を指定します。このアプローチの主な利点は、使用される接続の種類が何であれ、アプリケーションプログラムの大部分は変更の必要がないことです。これは J2SE や他のアプリケーション実行環境における従来の実装とは異なります。従来の実装では、アクセス対象を変更すると（例えばファイル入出力をソケット入出力に変更すると）、アプリケーションプログラムに適用する概念が大幅に変更になることがよくあります。

プロトコルの特定の初期化に伴う違いが `Connector.open()`のパラメータで識別でき、その違いをプラットフォーム側で吸収することができれば、プロトコル固有の処理をアプリケーションプログラム側で行う必要はありません。そのため、このような機構を採用することによりアプリケーションプログラムの移植性を向上させることができます。パラメータ文字列を実行時環境から取得するアプリケーションプログラムは、プラットフォームの点でもプロトコルの点でも高度な移植性が備わります。したがってこのようなアプリケーションプログラムでは、接続の形式が変わっても、コードの書き直しは必要ないか、最小限に抑えることができます。

プロトコルとアプリケーションプログラムとのバインドは実行時に行われます。実装レベルでは、`Connector.open()`の URL パラメータのプロトコル部により、サポートされているプロトコル実装のうちどのプロトコル実装を使用するかが決定されます。アプリケーションプログラムが、実行時に異なるプロトコルに動的に適応できるのは、この実行時のバインドメカニズムによるもので

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

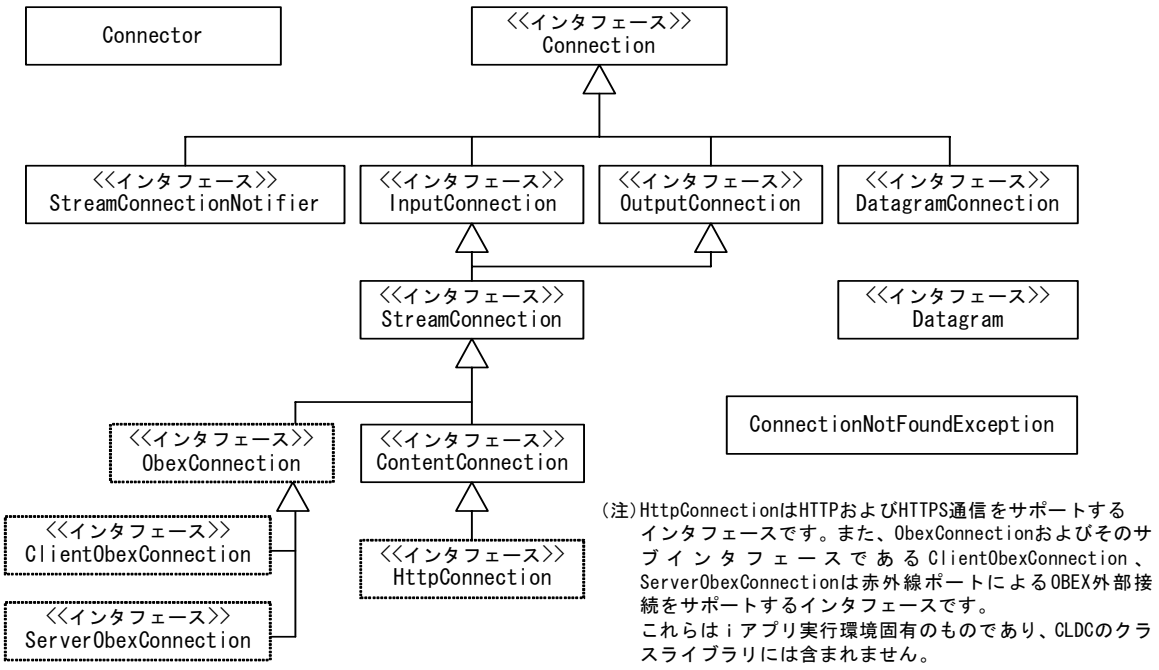
す。概念的にいえば、これは、PC やワークステーションのアプリケーションプログラムとデバイスドライバの関係に似ています。

タイプが異なるデバイスへの接続では、それぞれのデバイスに特有の操作を行えるようにしなければならない場合があります。例えばファイルはその名前を変更することができますが、TCP/IP にはこれに似た操作はありません。Generic Connection フレームワークにはこのような個別の機能が反映されており、論理的に同じ操作は同じ API を共有します。

この新しいフレームワークは、同じセマンティクスをもつプロトコルクラスをグループ化した `Connection` インタフェースから継承されるサブインタフェース階層として実装されています。Generic Connection フレームワークで扱う基本的なインタフェースタイプには、全てのインタフェースのスーパーインタフェースである `Connection` インタフェースを除き、次の 6 つがあります。

- 基本的なシリアル入力 : `InputConnection`
- 基本的なシリアル出力 : `OutputConnection`
- データグラム指向の通信 : `DatagramConnection`
- ストリーム指向の通信（`InputConnection` および `OutputConnection` のサブインタフェース）
  : `StreamConnection`
- クライアント/サーバー接続をサーバーに知らせる通知メカニズム
  : `StreamConnectionNotifier`
- コンテンツ送受信用通信（`StreamConnection` のサブインタフェース）
  : `ContentConnection`

`Connection` インタフェース群は 1 つの階層を形成し、階層がルート `Connection` インタフェースから継承されるに従って階層の機能が向上します。この編成により、開発者は、作成するアプリケーションプログラムに最も適した、プロトコル間の移植性レベルを選択できます。次に、Generic Connection クラス階層の主なクラス間の関係を示します。



Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

図 5: Generic Connection クラス階層

i アプリ実行環境では、i アプリが外部にアクセスする際に使用する以下のインタフェースをサポートしています。

- HTTP(S)通信　: `HttpConnection`
  （i アプリ API に含まれる、`ContentConnection` のサブインタフェース）
- ScratchPad　: `StreamConnection`
- リソース　: `InputConnection`
- OBEX 外部接続 : `ObexConnection`、`ClientObexConnection`、`ServerObexConnection`
  （i アプリ `API` に含まれる、`StreamConnection` のサブインタフェース）

なお、`Connector.open()`メソッドにはオプショナルのパラメータとして以下の 2 つが用意されています。

- アクセスモード（`READ`、`WRITE`、`READ_WRITE`）
- タイムアウト例外要求フラグ

アクセスモードは、アクセス対象に行う操作の内容に応じて適切な値を設定します。また、タイムアウト例外要求フラグは、HTTP(S)通信および OBEX 外部接続では true を、ScratchPad およびリソースでは false を設定します。

【DoJa-2.0】

DoJa-2.0 プロファイル以降では赤外線ポートを介した OBEX 外部接続をサポートしますが、このための API も Generic Connection フレームワークの枠組みの中で規定されています。

2.2.3 i アプリ API

次の各項では、`com.nttdocomo` パッケージ中の `io`、`net`、`util`、`lang`、`ui`、`device`、`system`、`security` の各サブパッケージに含まれる、 i アプリ基本 API に属するクラスについて説明します(一部のクラスには、i アプリオプション API に属するメソッドを含んでいるものがあります)。`ui` パッケージには、さらに 3D グラフィックス、3D サウンドの機能を提供するためのサブパッケージ `graphics3d`、`graphics3d.collision`、`sound3d`、`util3d` が存在しています。

【DoJa-3.0】

`com.nttdocomo.device` パッケージおよび `com.nttdocomo.system` パッケージは DoJa-3.0 プロファイルで新設されました。

【DoJa-4.0】

`com.nttdocomo.ui.graphics3d` パッケージ、`com.nttdocomo.ui.sound3d` パッケージおよび `com.nttdocomo.ui.util3d` パッケージは DoJa-4.0 プロファイルで新設されました。

【DoJa-4.1】

`com.nttdocomo.security` パッケージは DoJa-4.1 プロファイルで新設されました。

【DoJa-5.0】

`com.nttdocomo.graphics3d.collision` パッケージは DoJa-5.0 プロファイルで新設されました。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

com.nttdocomo.io パッケージ

CLDC の Generic Connection フレームワークは、 i アプリ実行環境の入出力パッケージの基礎をなすものです。`com.nttdocomo.io` パッケージでは、URL の形式で指定されたネットワーク資源に接続して入出力を行うためのインタフェースおよび例外クラスが提供されます。このインタフェースは、HTTP/HTTPS プロトコルによるネットワーク接続、および赤外線ポートを介した OBEX 外部接続をサポートします。またこのパッケージには、テキストのストリーム処理を効率よく行うために、CLDC の入出力クラスを補完するクラスも含まれています。

なお、Generic Connection フレームワークについては 2.2.2 項を参照してください。

このパッケージに含まれるインタフェース：

名前	説明
`ClientObexConnection`	OBEX クライアントとしての赤外線ポートへのアクセスを定義するインタフェースです。`javax.microedition.io.Connector.open()`メソッドに URL `"obex:/irclient"`を指定することで、本インタフェースの実装インスタンスが得られます。 `ClientObexConnection`は、i アプリ実行環境においてOBEXサーバー、OBEXクライアントの共通インタフェースを定義する `ObexConnection` のサブインタフェースです。 【DoJa-2.0】本インタフェースは DoJa-2.0 プロファイルで新設されました。
`HttpConnection`	HTTP 接続へのアクセスを定義するインタフェースです。`javax.microedition.io.Connector.open()`メソッドに HTTP または HTTPS の URL を指定することで、本インタフェースの実装インスタンスが得られます。 `HttpConnection` は Generic Connection フレームワークにおける `javax.microedition.io.ContentConnection` インタフェースのサブインタフェースです。
`ObexConnection`	OBEX サーバー、OBEX クライアントの共通インタフェースを定義します。このインタフェースを直接実装したインスタンスをアプリケーションプログラムで使用することはありません。 このインタフェースには、レスポンスステータスやオペレーションコード定数、OBEXサーバーとOBEXクライアントで共通のアクセッサメソッドが定義されています。 `ObexConnection` は Generic Connection フレームワークにおける `javax.microedition.io.StreamConnection` インタフェースのサブインタフェースです。 【DoJa-2.0】本インタフェースは DoJa-2.0 プロファイルで新設されました。
`ServerObexConnection`	OBEX サーバーとしての赤外線ポートへのアクセスを定義するインタフェースです。`javax.microedition.io.Connector.open()`メソッドに URL `"obex:/irserver"`を指定することで、本インタフェースの実装インスタンスが得られます。 `ServerObexConnection`は、i アプリ実行環境においてOBEXサーバー、OBEXクライアントの共通インタフェースを定義する `ObexConnection` のサブインタフェースです。 【DoJa-2.0】本インタフェースは DoJa-2.0 プロファイルで新設されました。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

このパッケージに含まれるクラス：

名前	説明
BufferedReader	バッファリングされたテキスト入力ストリームをサポートします。このクラスは、J2SE における java.io.BufferedReader クラスのサブセットに相当する機能を提供します。 【DoJa-3.0】本クラスは DoJa-3.0 プロファイルで新設されました。
ConnectionException	このクラスは、com.nttdocomo.io パッケージにおける入出力処理で異常が検出されたときにスローされます。例外の詳細は getStatus()メソッドで取得します。 【DoJa-4.0】本クラスで定義されている例外ステータスについて、DoJa-4.0 にて他の例外クラスと統一するため一部名称の変更がありました。 旧：RESOURCE_BUSY → 新：BUSY_RESOURCE 旧：NO_RESOURCE → 新：NO_RESOURCES 互換性保持のため DoJa-4.0 では旧名称も定義されていますが、その使用は推奨されません。
PrintWriter	フォーマットされたオブジェクト表現を出力するためのテキスト出力ストリームをサポートします。このクラスは、J2SE における java.io.PrintWriter クラスのサブセットに相当する機能を提供します。 【DoJa-3.0】本クラスは DoJa-3.0 プロファイルで新設されました。

HttpConnection を使用したクライアント/サーバープログラミングについては第 5 章で解説しています。また OBEX による外部接続プログラミングについては第 10 章で解説しています。

com.nttdocomo.net パッケージ

このパッケージに含まれるクラス：

名前	説明
URLEncoder	このクラスには、String を URL エンコード形式（MIME タイプ x-www-form-urlencoded で使用されるエンコード形式）にエンコードするユーティリティメソッドが含まれています。エンコードの際、String は SJIS コードに変換されます。
URLDecoder	このクラスには、URL エンコード形式を String にデコードするユーティリティメソッドが含まれています。入力データは SJIS コードをベースとしているものと見なされます。.

これらのクラスは、それぞれ Java 2 Standard Edition の java.net パッケージに含まれる同名のクラスと同じ機能を持ちます。

com.nttdocomo.util パッケージ

このパッケージには、次のクラスおよびインタフェースが含まれています。

名前	説明
Base64	バイト列の Base64 形式文字列へのエンコード機能、および Base64 形式文字列のバイト列へのデコード機能を提供するユーティリティクラスです。 【DoJa-3.5】本クラスは DoJa-3.5 プロファイルで新設されました。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

EventListener	iアプリ API で取り扱うイベントのリスナーインタフェースは、全てこのインタフェースを継承しています。
JarFormatException	Jar ファイル形式のエラーが発生したことを示す例外クラスです。この例外は、JarInflater クラスにおいて不正な形式のデータを処理しようとした場合に発生します。 【DoJa-3.0】本クラスは DoJa-3.0 プロファイルで新設されました。
JarInflater	Jar 形式のファイルイメージから、それに含まれるエントリを伸張して取り出す機能を提供します。Jar ファイルイメージは、バイト配列またはバイトストリームのいずれかの形式のものを使用することができます。 【DoJa-3.0】本クラスは DoJa-3.0 プロファイルで新設されました。
MessageDigest	任意のバイトデータからメッセージダイジェスト（ハッシュ値）を求めるための機能を提供します。DoJa-3.0 プロファイルでは MD5 および SHA-1 アルゴリズムがサポートされています。 【DoJa-3.0】本クラスは DoJa-3.0 プロファイルで新設されました。
Phone	Phone クラスはダイヤラーを表し、携帯電話のネイティブの通話機能にアクセスする手段を提供します。 iアプリから通話機能を呼び出すと、JAM はユーザーに通話発信を行ってもよいか確認を求めます。この確認を省略することはできません。 また、このクラスではダイヤラーの個体識別情報（製造番号など、機体毎にユニークな情報）を取得するためのメソッドを備えています。 【DoJa-2.0】本クラスは DoJa-2.0 プロファイルで新設されました。 【DoJa-2.1】個体識別情報取得メソッドは DoJa-2.1 プロファイルで追加されました。
ScheduleDate	スケジュール時刻を表現するためのクラスです。このクラスは i アプリからネイティブアプリケーションにスケジュール時刻を設定する際に使用します。このクラスを使用して、スケジューラやアラーム設定などのネイティブアプリケーションにスケジュールを登録することができます。 【DoJa-3.0】本クラスは DoJa-3.0 プロファイルで新設されました。
TimeKeeper	iアプリ API では、一定時間が経過したことをイベントとして i アプリに通知するタイマー機能をサポートしています。タイマークラス（com.nttdocomo.util.Timer および com.nttdocomo.ui.ShortTimer）はこのインタフェースを実装しています。 なお、タイマーの解像度（処理可能な最小のタイマー間隔）はメーカーにより異なります。 【DoJa-4.0】DoJa-4.0 プロファイルにてタイマーの解像度に関する仕様が明確化されました。新設された getMinTimeInterval()メソッドにより、その機種でタイマーイベントを発生させることのできる最小間隔を調べることができます。また getResolution()メソッドでは、getMinTimeInterval()の値から何 msec 刻みでタイマーイベントを発生させることができるかを調べることができます。つまり getMinTimeInterval()の値を x、getResolution()の値を y とすると、その機種では x+ny msec（n は整数）の間隔でタイマーイベントを発生させることができます。アプリケーションプログラムがタイマーの時間間隔としてこの式に合致しない値を指定した場合、この式に合致する最小の値が指定されたものとして切り上げられます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

Timer	タイマークラスです。このクラスは、ワンショットタイマーとインタバルタイマーをサポートします。TimerListener オブジェクトを登録することでタイマーイベントを受け取ることができます。タイマーイベントが発生すると、リスナーオブジェクトの timerExpired()メソッドが呼び出されます。タイマーの有効期間は i アプリの起動から終了までの全体に渡り、画面が切り替わってもタイマー動作は継続されます。ただし、i アプリの実行が中断された場合や、待ち受けアプリケーションが休眠状態に入るとき、タイマーは停止します。
TimerListener	このインタフェースは com.nttdocomo.util.Timer の通知するタイマーイベントを取得するリスナーを実装するためのインタフェースです。なお com.nttdocomo.ui.ShortTimer のイベント処理は Canvas の processEvent()メソッドを使用して行うため、本インタフェースは使用しません。

com.nttdocomo.lang パッケージ

このパッケージには、次のクラスが含まれています。

名前	説明
IllegalStateException	操作やメソッド呼び出しが行われてはならない状態の時にそのような操作やメソッド呼び出しが行われると、実行時例外として IllegalStateException がスローされます。 【DoJa-2.0】本クラスは DoJa-2.0 プロファイルで新設されました。
IterationAborted Exception	配列などに対する繰り返し処理が、何らかの例外発生によって中断したことを示す例外クラスです。このクラスのメソッドを使用して、繰り返し処理失敗の直接的な原因となった例外の内容を知ることもできます。 【DoJa-5.0】本クラスは DoJa-5.0 プロファイルで新設されました。
MemoryManager	アプリケーション実行メモリ（ヒープメモリ）の使用状況をモニタリングするためのユーティリティ機能を提供するクラスです。 【DoJa-5.0】本クラスは DoJa-5.0 プロファイルで新設されました。
UnsupportedOperation Exception	サポートされない操作やメソッドを実行すると、実行時例外として UnsupportedOperationException がスローされます。

com.nttdocomo.ui パッケージ

このパッケージに含まれているインタフェースは以下の通りです。

名前	説明
Audio3DListener	3D サウンド制御に関するイベントをアプリケーションプログラムに通知するためのイベントリスナを定義するインタフェースです。本プロファイルでは、SoundMotion（移動する定位）における移動完了イベントが定義されています。 【DoJa-4.0】本インタフェースは DoJa-4.0 プロファイルで新設されました。
Audio3DLocalization	定位を表すインタフェースです。3D サウンド制御機能では定位を表現する方法（クラス）が複数ありますが、どのクラスもこのインタフェースを実装しています。 【DoJa-4.0】本インタフェースは DoJa-4.0 プロファイルで新設されました。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

ComponentListener	コンポーネントイベントを受け取るリスナーインタフェースです。コンポーネントイベントの処理を行うクラスはこのインタフェースを実装する必要があります。リスナーオブジェクトは Panel.setComponentListener()メソッドを使用して Panel に登録します。コンポーネントイベントが発生するとリスナーオブジェクトの componentAction()メソッドが呼び出され、イベントのタイプ、イベントが発生したコンポーネント、イベントパラメータが引数として渡されます。
FocusManager	フォーカスマネージャとは、Panel 上に配置されたコンポーネント間のフォーカス移動を管理するオブジェクトです。このインタフェースはフォーカスマネージャクラスが実装するインタフェースです。フォーカスマネージャに関する指定が特に行われていない場合、デフォルトのフォーカスマネージャクラスが使用されます。現在までのプロファイルでは、デフォルトフォーカスマネージャ以外のフォーカスマネージャを規定していません。
Interactable	ユーザーが操作できるコンポーネント（Button、ListBox、TextBox など）が実装するインタフェースです。このインタフェースでは、コンポーネントへのフォーカス設定や操作可否設定を行うためのメソッドが定義されています。
KeyListener	高レベル API において、個別のキー操作を待ち受けるためのリスナーインタフェースです。本インタフェースの実装オブジェクトは Panel に登録して使用します。このインタフェースには、キープレスイベントとキーリリースイベントに対応する keyPressed()と keyReleased()の 2 つのメソッドが定義されています。
LayoutManager	レイアウトマネージャとは、Panel 上へのコンポーネントの配置を管理するオブジェクトです。このインタフェースはレイアウトマネージャクラスが実装するインタフェースです。 i アプリ実行環境では、レイアウトマネージャに関する指定が特に行われていない場合に使用されるデフォルトレイアウトマネージャ、および HTML 風のレイアウトを行う HTML レイアウトマネージャが定義されています。 【DoJa-2.0】HTML レイアウトマネージャは DoJa-2.0 プロファイルで新設されました。
MediaData	MediaData の実装オブジェクトは MediaManager.getData()メソッドを使って取得します。実際に返されるオブジェクトは実装によって異なります。MediaData はメーカーが独自のメディアデータを新たに実装する際のために用意されたインタフェースです。
MediaImage	MediaImage は静止画像または動画像を表現します。MediaImage の実装オブジェクトは MediaManager.getImage()メソッドを使って取得します。MediaImage は VisualPresenter オブジェクトを使って再生します。使用可能なデータについては 4.4 項を参照してください
MediaListener	このインタフェースは、マルチメディアデータを再生するメディアプレゼンターからイベントを受け取るためのメディアリスナーが実装するインタフェースです。このインタフェースで定義されている mediaAction()メソッドでは、メディアプレゼンターの再生開始・終了・停止などのイベントを受け取ることができます。
MediaPresenter	メディアデータの再生オブジェクトに実装されるインタフェースを定義します。プレゼンタークラスは、メディアデータを実際の内容に従って再生します。サポートされるメディアデータのタイプは端末に依存します。端末がサポートしないメディアデータが設定されると、UIException がスローされます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

MediaResource	メディアデータインタフェース（MediaData、MediaImage、MediaSound）が継承しなければならないインタフェースを定義します。MediaManager を使用して取得した MediaResource インスタンスを実際に使用する前には use()メソッドを、使用した後には unuse()メソッドをそれぞれ呼び出す必要があります。MediaResource インスタンスをそれ以上使用しない場合は、dispose()メソッドを呼び出してインスタンスを破棄すべきです。 【DoJa-5.0】DoJa-5.0 プロファイルにて、MediaResource やサブインタフェース（メディアデータインタフェース）について、メモリ管理方法をコントロールするためのメソッドが追加されました。
MediaSound	MediaSound はサウンドを表現します。MediaSound の実装オブジェクトは MediaManager.getSound()メソッドを使って取得します。MediaSound は AudioPresenter オブジェクトを使って再生します。サウンドデータとして使用可能な形式については、4.4 項を参照してください。
SoftKeyListener	高レベル API において、携帯電話に 2 つ用意されているソフトキー操作を待ち受けるためのリスナーインタフェースです。本インタフェースの実装オブジェクトは Panel に登録して使用します。このインタフェースには、ソフトキープレスイベントとソフトキーリリースイベントに対応する softKeyPressed()と softKeyReleased()の 2 つのメソッドが定義されています。

このパッケージに含まれているクラスは以下の通りです。

名前	説明
AnchorButton	AnchorButton コンポーネントは、HTML のアンカー風のボタン（ボタンとしての操作が可能な下線付きテキスト）を表現します。テキストの前に見出し用のイメージ画像を指定することもできます。また、テキストは複数行に渡ることができます。アンカーボタン押下により処理を行う場合は、ComponentListener を実装したオブジェクトを、Panel の setComponentListener()メソッドによりリスナー登録します。アンカーボタンが操作されると、リスナーは BUTTON_PRESSED イベントを受け取ります。 【DoJa-2.0】本クラスは DoJa-2.0 プロファイルで新設されました。
Audio3D	各オーディオプレゼンタ毎に 3D サウンド制御を行うためのコントローラです。Audio3D オブジェクトを使用して 3D サウンド制御の有効/無効を切り替えたり、定位の制御を行います。 i アプリの中でいくつの仮想音源を使用できるか（3D サウンド制御リソース数）はメーカーにより異なります。詳細は 13.2 項を参照してください。 【DoJa-4.0】本クラスは DoJa-4.0 プロファイルで新設されました。
AudioPresenter	このクラスは、サウンドメディアデータ（MediaSound）を再生するときに使用します。AudioPresenter は高レベル API 使用時、低レベル API 使用時のいずれの場合も使用することができます。
Button	Button コンポーネントはラベル付きボタンを表現します。ボタン押下により処理を行う場合は、ComponentListener を実装したオブジェクトを、Panel の setComponentListener()メソッドによりリスナー登録します。ボタンが操作されると、リスナーは BUTTON_PRESSED イベントを受け取ります。
Canvas	Canvas は画面上の空白の矩形域です。 i アプリはこの領域に描画したり、この領域でユーザーの入力イベントをトラップしたりすることができます。開発者は Canvas を継承するサブクラスを定義して使用します。 i アプリから Canvas に描画を行うには、Canvas クラスの paint()メソッドをオーバーライドします。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

Component	Component はグラフィックスで表現されたユーザーインタフェース部品であり、画面に表示したり、これを使ってユーザーと対話したりすることができます。Component のサブクラスには Button、ListBox、TextBox などがあります。Component は高レベル API に属し、Panel と組み合わせて使用します。
Dialog	Dialog は、タイトル、テキストメッセージ、1 つ以上のボタンを備えたトップレベルウィンドウです。Panel および Canvas のいずれの上にも Dialog を表示することができます。
Display	Display は画面領域を管理するクラスです。Display のすべてのメソッドは static であり、そのインスタンスが作成されることはありません。Display クラスは、画面（Panel または Canvas）を切り替えたり、画面上に表示されている表示可能オブジェクトに入力イベントを渡したりします。このクラスは、低レベルイベントを受け取り、それを適切な表示可能オブジェクトに渡します。そして、このオブジェクトがイベントの処理方法を決定します。さらに、Display には、表示領域の幅と高さや、キーやイベント処理で使用するさまざまな定数といったデバイス情報が含まれています。
EncodedImage	Canvas や Image の描画内容を、ImageEncoder クラスによりバイトイメージ（JPEG 形式など）にエンコードした結果を保持するクラスです。 i アプリは本クラスを使用して、描画内容のバイトイメージを取り出したり、メディアイメージに再変換して取り出したりすることができます。 【DoJa-3.0】本クラスは DoJa-3.0 プロファイルで新設されました。
Font	Graphics コンテキストや Component、Dialog に文字を表示する際のフォントを表します。どのようなフォントがサポートされるかは、メーカーにより異なります。 【DoJa-3.0】Component および Dialog にフォントを設定する機能は DoJa-3.0 プロファイルで追加されました。
Frame	Frame は、画面を表現する abstract クラスです。携帯電話のディスプレイに一度に表示できる内容は、1 つの Frame インスタンスに対応します。Frame のサブクラスには Canvas、Dialog、Panel があります。画面の切り替えは、Display.setCurrent()メソッドを、切り替えたい画面に対応する Frame オブジェクトを引数として呼び出すことで行います（ただし Dialog のみ、Dialog.show()メソッドにより表示します）。
Graphics	Canvas および Image への描画とそれに関連する操作をカプセル化します。なお Image への描画は、その Image が Image.createImage()メソッドにより新規に作成された Image インスタンスである場合に限り可能です。 【DoJa-2.0】Image インスタンスからの Graphics インスタンス取得と描画の機能は、DoJa-2.0 プロファイルで追加されました。
HTMLLayout	HTML 風のレイアウトを行うためのレイアウトマネージャです。Panel に追加された順にレイアウトしていく点ではデフォルトレイアウトマネージャと同様ですが、HTML レイアウトマネージャでは、用意されたメソッドにより HTML の改行、段落およびアライメントに相当する指定を行うことができます。 【DoJa-2.0】本クラスは DoJa-2.0 プロファイルで新設されました。
IApplication	IApplication クラスは、 i アプリの雛型クラスです。 i アプリのメインクラス（ i アプリの起動時に最初に呼び出されるクラス）は IApplication のサブクラスでなければなりません。IApplication クラスは、 i アプリの起動から終了までのライフサイクルを管理します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

Image	Image クラスは静止画像を表現する abstract クラスです。静止画像から MediaImage インスタンスを作成し、getImage()メソッドを呼び出すことで Image の実装インスタンスを取得することができます。また Image.createImage()メソッドにより、新規に Image インスタンスを作成することもできます。静止画像として使用可能なデータ形式については、4.4 項を参照してください。 【DoJa-2.0】createImage()メソッドは DoJa-2.0 プロファイルで追加されました。
ImageButton	ImageButton コンポーネントは、静止画像を貼り付けることのできるボタンを表現します。イメージボタン押下により処理を行う場合は、ComponentListener を実装したオブジェクトを、Panel の setComponentListener()メソッドによりリスナー登録します。イメージボタンが操作されると、リスナーは BUTTON_PRESSED イベントを受け取ります。 【DoJa-2.0】本クラスは DoJa-2.0 プロファイルで新設されました。
ImageEncoder	Canvas や Image の描画内容を、JPEG 形式などのバイトイメージにエンコードする機能を提供します。エンコードされた結果は EncodedImage インスタンスとして返されます。各機種で共通的に使用可能なエンコード形式は JPEG 形式です。 【DoJa-3.0】本クラスは DoJa-3.0 プロファイルで新設されました。
ImageLabel	ImageLabel コンポーネントは、Panel に静止画像を表示するときに使用します。Image インスタンスを ImageLabel に設定し、それを Panel 上に配置することができます。
ImageMap	ImageMap は、何種類かの小さなイメージを縦横に並べて見かけ上大きなイメージを作成する機能を提供します。小容量の画像データで大きな背景画像などを構成するのに適しています。 【DoJa-3.5】本クラスは DoJa-3.5 プロファイルで新設されました（i アプリオプション API からの移行）。
Label	Label コンポーネントは、Panel にテキストを表示するときに使用します。Label は、1 行の読み取り専用テキストを表現します。テキストはアプリケーションプログラムからは変更することはできますが、ユーザーが直接編集することはできません。
ListBox	ListBox コンポーネントは、複数のテキスト項目からなるスクロールリストを表示するときに使用します。インスタンス化時の指定によって、開発者は ListBox の外観や機能を選択することができます。
MApplication	MApplication クラスは、待ち受けアプリケーションの雛型クラスです。MApplication クラスは IApplication クラスを継承しており、待ち受けアプリケーションのメインクラス（待ち受けアプリケーションの起動時に最初に呼び出されるクラス）はこのクラスのサブクラスでなければなりません。MApplication クラスは、待ち受けアプリケーションの起動・終了・実行状態管理などのライフサイクルを管理します。 【DoJa-2.0】本クラスは DoJa-2.0 プロファイルで新設されました。
MediaManager	MediaResource を管理するためのクラスです。MediaManager はメディアデータ（MediaData、MediaImage、MediaSound）のファクトリクラスです。URL などのロケーション文字列、ストリーム、バイト配列のいずれかを指定してメディアデータを生成することができます。 【DoJa-3.0】ストリームおよびバイト配列を指定してのメディアデータ生成は DoJa-3.0 プロファイルで追加されました。
Palette	PalettedImage と組み合わせて使用する、色パレットを表すクラスです。 【DoJa-3.5】本クラスは DoJa-3.5 プロファイルで新設されました（i アプリオプション API からの移行）。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

PalettedImage	GIF データを元に作成される、色パレットを指定することができるイメージです。 【DoJa-3.5】本クラスは DoJa-3.5 プロファイルで新設されました（i アプリオプション API からの移行）。
Panel	Panel は、高レベル API 使用時において、コンポーネントを画面に表示するためのコンテナクラスです。Panel が表すスペースには任意のコンポーネントを付加することができます。Panel は、同時に 1 つしか画面に表示できません。
PhoneSystem	このクラスは、プラットフォームのネイティブ資源にアクセスするときに使用します。プラットフォームのネイティブ資源には、ディスプレイ上に表示されるアイコン情報（メールアイコンなど）を含みます。 【DoJa-2.0】アイコン情報の参照は DoJa-2.0 プロファイルで追加されました。
ShortTimer	低レベル API で使用することを想定した、比較的オーバーヘッドの低いタイマーです。このクラスは、ワンショットタイマーとインタバルタイマーをサポートします。ShortTimer は、本クラスの static メソッド getShortTimer()を使用して取得します。getShortTimer()メソッドの引数には Canvas を指定しますが、タイマーイベントはこの Canvas の processEvent()メソッドに通知されます。ShortTimer は対応する Canvas がディスプレイに表示されているときにだけ動作します。ShortTimer が動作している状態でフレームの切り替えが発生すると、タイマーは自動的に停止します。
Sprite	スプライト機能において、画面上を移動する個々のスプライトを表現します。Sprite はそれ単独では表示させることはできず、SpriteSet と組み合わせて使用します。 【DoJa-3.5】本クラスは DoJa-3.5 プロファイルで新設されました（i アプリオプション API からの移行）。
SpriteSet	スプライト機能において、複数のスプライトを束ねる役割を持ちます。アプリケーションプログラムは必要な数のスプライトをスプライトセットに属させ、スプライトセットを描画します。 【DoJa-3.5】本クラスは DoJa-3.5 プロファイルで新設されました（i アプリオプション API からの移行）。
TextBox	TextBox コンポーネントは、テキストの入力や表示を行うときに使用します。TextBox は、ユーザーが任意の文字列を入力するための唯一のコンポーネントです。
Ticker	Ticker コンポーネントは、左方向に自動スクロールするテキストを表示するために使用するコンポーネントです。このコンポーネントは、テキストをティッカーテープ形式（i モード HTML の MARQUEE タグ相当）で表示します。
UIException	UIException は、UI パッケージや UI パッケージの使用するネイティブ機能などが動作しているときに異常を検出した際にスローされる実行時例外クラスです。
VisualPresenter	VisualPresenter コンポーネントは、画像メディアデータ（MediaImage）を高レベル API で再生するときに使用するプレゼンタークラスです。

com.nttdocomo.ui.graphics3d パッケージ

このパッケージには、3D グラフィックス描画機能を提供するための次のクラスおよびインタフェースが含まれています。

名前	説明
ActionTable	3D モデルのアクション（アニメーションデータ）を保持するクラスです。アニメーションデータは市販の 3D オーサリングツールなどを利用して作成します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

DrawableObject3D	レンダリング可能な 3D オブジェクトの基底となるクラスです。このクラスを継承する Figure クラス、Group クラス、Primitive クラスの各オブジェクトは、Graphics3D.renderObject3D()メソッドによるレンダリング対象とすることができます。また、3D 空間内で 2 つのオブジェクト同士が衝突しているかどうかを簡易に判定するためのメソッドや、ブレンド機能、テクスチャ歪み補正機能などのメソッドを備えています。
Figure	3D モデルのデータを保持するクラスです。このオブジェクトはレンダリング可能であり、Graphics3D.renderObject3D()メソッドにレンダリング対象として指定することができます。
Fog	フォグ効果を与えるためのデータを保持するクラスです。線形フォグモードと指数フォグモードをサポートします。
Graphics3D	3D グラフィックス描画機能をサポートするグラフィックスコンテキストが備えるべきメソッドを定義するインタフェースです。
Group	3D オブジェクトの集合体であるグループを表現するクラスです。このオブジェクトはレンダリング可能であり、Graphics3D.renderObject3D()メソッドにレンダリング対象として指定することができます。複数の Figure オブジェクトや Primitive オブジェクトを属させて一度にレンダリングを行わせたり、Fog オブジェクトや Light オブジェクトを属させてグループ内のレンダリングに効果を与えたりすることができます。
Light	光源データを保持するクラスです。環境光源、平行光源、点光源、スポット光源をサポートします。
Object3D	全ての 3D オブジェクトの基底となるクラスです。3D オブジェクトとは、3D グラフィックスのレンダリング結果に影響を与えるオブジェクト全般を指します。3D オブジェクトには、フィギュアやプリミティブのように直接レンダリングの対象物とすることのできるオブジェクトと、ライトやフォグのようにレンダリング結果に効果を与える性質を持つオブジェクトに大別されます。
Primitive	プリミティブ（点や線、面などの図形）を描画する際に用いる頂点情報などを格納するための、プリミティブ配列を保持するクラスです。このオブジェクトはレンダリング可能であり、Graphics3D.renderObject3D()メソッドにレンダリング対象として指定することができます。
Texture	テクスチャのデータを保持するクラスです。テクスチャオブジェクトを使用することで、モデルマッピングに使用するテクスチャと環境マッピングに使用するテクスチャの両方を取り扱うことができます。

com.nttdocomo.ui.graphics3d.collision パッケージ

このパッケージには、3D グラフィックス描画機能と併用することで 3D 空間上での物体の衝突判定機能を提供するための次のクラスおよびインタフェースが含まれています。

名前	説明
AABBox	ワールド座標軸に平行な Box を表すクラスです。
AABCapsule	ワールド座標の Y 軸に中心軸が平行となる Capsule を表すクラスです。
AABCylinder	ワールド座標の Y 軸に中心軸が平行となる Cylinder を表すクラスです。
AbstractBV	形状のうち、立体を表す抽象クラスです。このクラスは BoundingVolume インタフェースを実装しています。立体形状は、このクラスを継承した具象クラスとして提供されます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

AbstractShape	全ての形状の基底となる抽象クラスであり、Shape インタフェースを実装しています。非立体形状は、このクラスを継承した具象クラスとして提供されます。また立体形状は、このクラスを継承した抽象クラス AbstractBV を継承した具象クラスとして提供されます。
AxisAlignedBV	比較判定で使用される立体の中でも、ワールド座標に平行に配置される特性を持つ立体が実装するインタフェースです。
BoundingVolume	比較判定で使用される形状のうち、立体が実装するインタフェースです。本インタフェースは Shape インタフェースを継承しています。本パッケージでは、具体的な立体形状として直方体、円柱、球形、カプセル形をサポートしています。
Box	比較判定で使用される立体のうち、直方体を表すクラスです。
BVBuilder	フィギュアから、BVFigure オブジェクトや、BVFigure オブジェクトに貼り付けることのできる BoundingVolume オブジェクト（BoundingVolume インタフェースを実装した各種立体のオブジェクト）を生成するためのクラスです。
BVFigure	ボーン構造を持ち、複数の BoundingVolume で表現されたフィギュアです。通常のフィギュア（Figure クラス）ではポリゴンを貼り付けることによって形状を表現しますが、BVFigure は BoundingVolume を貼り付けることによって形状を表現します。
Capsule	比較判定で使用される立体のうち、カプセル形を表すクラスです。
Collision	衝突判定に関する機能を提供するクラスです。衝突判定、交差判定、距離算出などの機能が本クラスで提供されます。
CollisionObserver	Collision クラスでの衝突判定の結果、衝突したと判定された場合に、衝突情報とともに通知を受けるためのオブザーバクラスが実装すべきインタフェースです。通知を受けたいアプリケーションは、このインタフェースを実装したオブザーバクラスのオブジェクトを作成して Collision オブジェクトにセットします。
Cylinder	比較判定で使用される立体のうち、円柱を表すクラスです。
IntersectionAttribute	Figure と Ray の交差判定において、交差したと判断された場合に CollisionObserver の通知メソッドに渡される交点情報を表すクラスです。
Line	比較判定で使用される非立体形状のうち、線分を表すクラスです。
Plane	比較判定で使用される非立体形状のうち、無限平面を表すクラスです。
Point	比較判定で使用される非立体形状のうち、点を表すクラスです。
Ray	比較判定で使用される非立体形状のうち、半直線を表すクラスです。
Shape	衝突判定に使用される全ての形状が実装するインタフェースです。このパッケージで提供される衝突判定は Figure 同士を直接比較判定するのではなく、Figure の位置に近似的に配置されたシンプルな形状同士を比較判定することにより行われます。Shape インタフェースは、この比較判定で使用される全ての形状で実装されています。
Sphere	比較判定で使用される立体のうち、球を表すクラスです。
Triangle	比較判定で使用される非立体形状のうち、三角形を表すクラスです。
ViewVolume	視錐台（カメラから見た視界）における、BoundingVolume の可視判定を行う機能を提供するクラスです。このクラスを使用して非可視と判断されたものは配置上カメラの視界に入らないため、多くの物体の中から真に描画すべき物体を絞り込むことができます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

com.nttdocomo.ui.sound3d パッケージ

このパッケージには、3D サウンド機能を提供するための次のクラスおよびインタフェースが含まれています。これらのクラスやインタフェースは、3D サウンドを制御するためのコントローラである com.nttdocomo.ui.Audio3D クラスと組み合わせて使用します。

名前	説明
CartesianListener	CartesianPosition クラスで定位を表現する際に、デカルト座標系上に置く聴者を表現するクラスです。デカルト座標系で定位を表すために必要となる聴者の向きの設定は、このクラスを使用して行います。
CartesianPosition	3D サウンドの制御において、x 軸、y 軸、z 軸の 3 軸から構成されるデカルト座標系を使用して聴者と仮想音源の位置関係（定位）を表します。
PolarPosition	極座標系を使用して聴者と仮想音源の位置関係を表します。極座標系ではその原点にある決められた方向を向いた聴者がいるものとし、そこからの方位角、仰角、距離を指定して仮想音源の位置を表現します。
SoundMotion	SoundMotion クラスは「移動する定位」を表現するためのクラスです。移動開始からの経過時間とその時点での定位を経路情報として設定することで、その経路に沿って仮想音源が移動していくようにユーザーに感じさせることができます。
SoundPosition	聴者の位置と仮想音源の位置によって表される、固定された定位を表すインタフェースです。このインタフェースを実装したクラスでは、聴者の位置と仮想音源の位置を指定することによって両者の位置関係を表します。

com.nttdocomo.ui.util3d パッケージ

このパッケージでは、3D グラフィックス機能や 3D サウンド機能を使用する上で必要となる数値演算やベクトル演算などを行うためのユーティリティ的な機能が提供されます。

名前	説明
FastMath	3D 関連機能での使用を前提とした高速数値演算ユーティリティです。本プロファイルに対応する携帯電話では CLDC-1.1 が搭載されており、このバージョンの KVM がサポートする浮動小数点数演算が使用できますが、このクラスでは浮動小数点数演算を内部的に整数演算に置き換えます。このため、通常の浮動小数点数演算より大きな誤差を含む可能性はありますが高速に演算を行うことができます。
Transform	3 次元アフィン変換用の行列を扱うクラスです。このクラスでは、4 行 4 列の行列でアフィン変換行列を表現します。
Vector3D	3 次元ベクトルを表すクラスです。内積・外積計算、正規化といったユーティリティメソッドを備えています。座標系上の位置や向きなどを表現するために使用されます。

com.nttdocomo.device パッケージ

このパッケージには、携帯電話に搭載されている周辺デバイスを制御するための次のクラスが含まれています。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

名前	説明
Camera	携帯電話に搭載ざれているカメラ機能を呼び出して、静止画または動画を撮影する機能にアクセスする手段を提供します。実際の撮影は、iアプリが起動したカメラ機能に対してユーザーがシャッター操作を行うことにより行われます。動画撮影をサポートするか否かはメーカーにより異なります。
CodeReader	携帯電話に搭載されたカメラデバイスを使用して、バーコード（JAN コードや 2 次元バーコード）を読み取るコード認識機能を提供します。 【DoJa-3.5】コード認識機能は DoJa-3.5 プロファイルで新設されました（i アプリオプション API からの移行）。
DeviceException	DeviceException は、主に device パッケージや device パッケージの使用するネイティブ機能などが動作しているときに異常を検出した際にスローされる実行時例外クラスです。
IrRemoteControl	赤外線リモコン機能を提供します。IrRemoteControl クラスは、IrRemoteControlFrame クラスで表現されるリモコン信号を赤外線ポートから送り出すための、送出器としての役割を持ちます。
IrRemoteControlFrame	赤外線リモコン機能から送出される、外部機器の制御内容を示すデータフレームを表現します。一般的に、どのようなデータフレームを受信した際にどのような制御を行うかは、その外部機器のメーカーが独自に決定しています。開発者は、対象とする外部機器の仕様に合致したデータフレームを送出するようアプリケーションプログラムを設計する必要があります。

com.nttdocomo.system パッケージ

このパッケージには、電話帳機能やブックマーク、スケジューラなど携帯電話のネイティブ機能をiアプリから利用するための次のクラスおよびインタフェースが含まれています。なお、このパッケージにはトラステッドiアプリにしか使用が許可されないクラスやメソッドが多く含まれています。ここではそれらの機能については解説しません。

名前	説明
ApplicationStore	携帯電話ネイティブのiアプリ管理機能にアクセスする機能を提供します。このクラスを使用することで、iアプリから携帯電話にダウンロードされている他のiアプリを起動（ランチャーモード）する際に必要となるエントリ ID を取得することができます。
Bookmark	携帯電話ネイティブのブックマーク管理機能にアクセスする機能を提供します。このクラスを使用することで、携帯電話にブックマーク項目を新規登録することができます。
ImageStore	携帯電話ネイティブの画像データ管理機能にアクセスする機能を提供します。このクラスを使用することで、携帯電話の画像データ保存領域からの画像の読み込みと、画像の新規登録を行うことができます。
InterruptedOperationException	iアプリが呼び出したネイティブアプリケーション機能をユーザーが操作している際に、通話着信などの外部イベントの発生によりその操作が中断されたことを示す例外です。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

MovieStore	携帯電話ネイティブの映像データ（i モーション）管理機能にアクセスする機能を提供します。このクラスを使用することで、携帯電話の映像データ保存領域への映像データの新規登録を行うことができます。なお、映像データ保存領域からの映像データの取得は i アプリオプション API に該当し、メーカーによりサポート状況が異なります。 【DoJa-4.0】【DoJa-4.1】本クラスは DoJa-4.0 プロファイルで新設されましたが、DoJa-4.0 プロファイルでは本機能はトラステッド i アプリのみ使用可能です。DoJa-4.1 プロファイル以降、一般 i アプリも本クラスを使用できるようになりました。
PhoneBook	携帯電話ネイティブの電話帳管理機能にアクセスする機能を提供します。このクラスを使用することで、携帯電話に電話帳エントリを新規登録することができます。
PhoneBookConstants	電話帳管理機能に関連する定数をまとめたインタフェースです。このインタフェースは PhoneBook クラスおよび PhoneBookParam クラスに実装されています。 開発者がこのインタフェースの実装クラスを作成する必要はありません。
PhoneBookGroup	携帯電話ネイティブの電話帳管理機能のうち、電話帳グループ管理機能にアクセスする機能を提供します。このクラスを使用することで、携帯電話に電話帳グループを新規登録することができます。このクラスで作成した電話帳グループに対して、PhoneBook クラスを使用して電話帳エントリを新規登録することもできます。
PhoneBookParam	新規登録用の電話帳エントリを表現するクラスです。このオブジェクトに名前や電話番号などの電話帳エントリデータを設定し、PhoneBook クラスで携帯電話に登録します。
Schedule	携帯電話ネイティブのスケジューラにアクセスする機能を提供します。このクラスを使用することで、携帯電話にスケジュールを新規登録することができます。
StoreException	携帯電話ネイティブのアプリケーションが管理するデータ領域にアクセスしようとしたときに、空きエリア不足などのアクセスエラーが発生したことを示す例外です。

注意事項：

- com.nttdocomo.system パッケージのクラスには、携帯電話のネイティブ機能を呼び出してユーザーデータを新規登録する機能が多く含まれています。携帯電話ユーザーは、携帯電話の設定によりこれらの機能を無効化することができます。またこれらの機能を呼び出した場合、 i アプリ実行環境はそれらのデータを登録することについてユーザーに確認を求めます。これらの機構によりユーザーの許可が得られなかった場合、そのデータを登録することはできません。

com.nttdocomo.security パッケージ

このパッケージには、証明書やデジタル署名など、セキュリティに関する機能を提供するクラスが含まれます。

名前	説明
CertificateException	証明書に関するエラーが発生したことを示す例外です。証明書の検証に必要となるルート認証局の証明書が端末にない場合や、ルート認証局の証明書は存在するが有効期限切れなどの理由で失効している場合などに発生します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

PKCS7SignedData	デジタル署名付きデータを表現するクラスです。PKCS7Signer によりデジタル署名されたデータは、このクラスのオブジェクトとして表現されます。
PKCS7Signer	デジタル署名付きデータを生成する機能を提供するクラスです。任意のデータをこのクラスに与えることで、そのデータにデジタル署名を施すことができます。
PKCS7SignerInfo	デジタル署名に含まれる署名者情報を表現するクラスです。 【DoJa-5.1】このクラスは DoJa-5.1 プロファイルにて追加されました。
SignatureException	デジタル署名に関するエラーが発生したことを示す例外です。デジタル署名処理に失敗した場合や、デジタル署名の検証に失敗した場合などに発生します。
X509Certificate	X509 証明書を表現するクラスです。 i アプリの内部では、証明書はこのクラスのオブジェクトとして表現されます。

IApplication と MApplication

com.nttdocomo.ui.IApplication クラスは、全ての i アプリの雛型となるクラスです。通常、i アプリのメインクラス（ADF の AppClass キーで指定される、 i アプリ起動時に最初に呼び出されるクラス）はこのクラスを継承していなければなりません。これは、PC のブラウザ上で動作する Java アプレットが java.applet.Applet クラスを継承しなければならないことと似ています。

IApplication クラスには、主に i アプリの起動から終了までのライフサイクルに関わるメソッドが定義されています。

- i アプリの実行は、IApplication.start()メソッドから開始されます。開発者はこのメソッドをオーバーライドして、i アプリの処理をクラスに実装します。この際、IApplication.getArgs()メソッドを呼び出して i アプリの起動パラメータ（ADF の AppParam キーで指定された値）を取得することができます。
- i アプリダウンロード後の初回起動時、およびアプリケーション連携機能により i アプリの起動が指示された場合には、HTML やメール本文などから i アプリにパラメータを与えることができます。
- i アプリの実行を終了するには、IApplication.terminate()メソッドを呼び出します。
- i アプリの実行中に、電話着信などの優先度の高い外部イベントが発生した場合、 i アプリの実行は中断されて外部イベントに対応した処理が行われます。また、その後外部イベントの処理が終了すると、 i アプリの実行は自動的に再開されます。 i アプリの実行が再開される時には IApplication.resume()メソッドが呼び出されます。開発者はこのメソッドをオーバーライドすることで、 i アプリ実行再開時に必要な処理を実装することができます。また、DoJa-3.0 プロファイルで追加された IApplication.getSuspendInfo()メソッドを使用すると、前回 i アプリが中断するきっかけとなった理由と、中断中に発生した事象の情報を取得することができます（メーカーによっては中断中に発生した事象の情報は、その中断のきっかけとなった 1 つのみ取得可能です）。なお、 i アプリ実行再開時のスレッド動作順序、およびスレッドスイッチ制御の詳細は Java 仮想マシンのスレッド制御に委ねられるため、resume()メソッドの呼び出しは i アプリ実行の再開において必ず最初に行われるとは限りません。
- IApplication.launch()メソッドを使用することで、 i アプリから他のネイティブアプリケーションや i アプリを起動することができます。これらの起動に成功すると、原則的には元の i アプリの実行は終了します。

また、com.nttdocomo.ui.MApplication クラスは、待ち受けアプリケーション特有の複雑なライフサイクルに対応するために IApplication クラスを継承、拡張したものです。待ち受けアプリケーションのメインクラスは MApplication クラスを継承していなければなりません。

MApplication クラスには、IApplication の持つ機能の他に以下のような機能が含まれています。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

待ち受けアプリケーションの状態制御：

待ち受けアプリケーションには、通常の i アプリが実行されている状態と同等の状態である「活性化状態」の他に、

- i アプリは動作しているが、キーイベントおよびソフトキーイベントは携帯電話のネイティブの待ち受け機能が使用するためフックされ、i アプリに通知されない「非活性化状態」
- 電池の消費を抑えるため、画面表示は保持したまま i アプリの動作が停止する「休眠状態」

という 3 つの状態を持ちます。`MApplication` クラスは、これらの状態の遷移を管理します。

システムイベントのハンドリング：

待ち受けアプリケーションでは、常駐している i アプリに外部からのイベントがシステムイベントとして通知されます。開発者は、`MApplication` クラスの持つシステムイベントハンドラをオーバーライドすることで、上記に示した 3 つのいずれの状態にあってもこれらのイベントの通知を受け、処理することができます。

待ち受けアプリケーションの詳細については、第 9 章を参照してください。

注意事項：

- i アプリの中で、`IApplication` およびそのサブクラスを直接インスタンス化してはいけません。これらは i アプリ実行環境がインスタンス化します。
- 明示的に i アプリメインクラスのコンストラクタを定義する場合、必ずアクセス指定子 public を指定してください。また i アプリ実行環境は、メインクラスのインスタンス化時に引数のあるコンストラクタは使用しません。明示的にメインクラスのコンストラクタを定義する際は、必ず引数なしのコンストラクタを設けるようにしてください。
- CLDC には、`java.lang.Runtime.exit()`メソッドおよび `java.lang.Sysytem.exit()`メソッドが定義されています。しかし i アプリ実行環境では、i アプリの実行を終了するためにこれらのメソッドを使用することはできません（これらを呼び出した場合には、`SecurityException` がスローされます）。i アプリの実行を終了するためには、`IApplication.terminate()`メソッドを呼び出すようにしてください。

テキスト変換

i アプリでは、テキスト処理に日本語文字セットと ASCII 文字セットを使用します。`java.io` パッケージには、Unicode 文字ストリームと、非 Unicode テキストのバイトストリームとの間で変換を行うクラスが含まれています。`byte` ストリームを `character` ストリームに変換するには `InputStreamReader` クラスを、`character` ストリームを `byte` ストリームに変換するには `OutputStreamWriter` クラスをそれぞれ使用します。テキスト変換に関する詳細は、第 6 章を参照してください。

ScratchPad

携帯電話の i アプリ向け記憶装置モデルを ScratchPad と呼びます。ScratchPad に接続するには、`Connector` クラスの `open()`メソッドを使用します。このメソッドに ScratchPad の URL を渡すと ScratchPad への接続がオープンされ、CLDC で定義されている `javax.microedition.io.StreamConnection` インタフェースの実装インスタンスが返されます。ScratchPad 領域の読み取りや書き込みには、`StreamConnection` インタフェースを使用します。ScratchPad の利用に関する詳細は、第 7 章を参照してください。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

第 3 章
設計上の考慮事項

この章では、利用可能なシステム資源を効率的に使用し、無線通信環境に適合した使いやすいｉアプリを設計するためのガイドラインを提供します。

3.1 ｉアプリ対応携帯電話の特性

携帯電話の表示画面やアプリケーション実行メモリ、アプリケーション記憶メモリは、携帯電話の物理的なサイズをできるだけ小さくし、バッテリ寿命をできるだけ長くするため、パーソナルコンピュータに比べ非常に小さくなっています。さらに、プロセッサもパーソナルコンピュータより遅い速度で動作します。液晶（LCD）画面の電力消費量は非常に少ないのですが、リフレッシュ速度は他のタイプのディスプレイに劣ります。テキスト入力は携帯電話のキーパッドを使って行い、しかも各キーに複数の文字が割り当てられているため、テキストの入力はユーザーにとって煩雑な操作です。したがって、テキスト入力は最小限に留めるべきです。

ｉアプリを活用したサービスを設計する場合、このようなプラットフォーム固有の特性を考慮してｉアプリの設計を行う必要があります。次の各節では、これらの問題を設計の観点から検討し、使いやすいｉアプリを設計するためのガイドラインを提供します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

3.2 メモリの問題

次に、3 つの i アプリに対して JAM が記憶領域（JAR ファイルと ScratchPad の保存領域）を割り当てる例を示します。

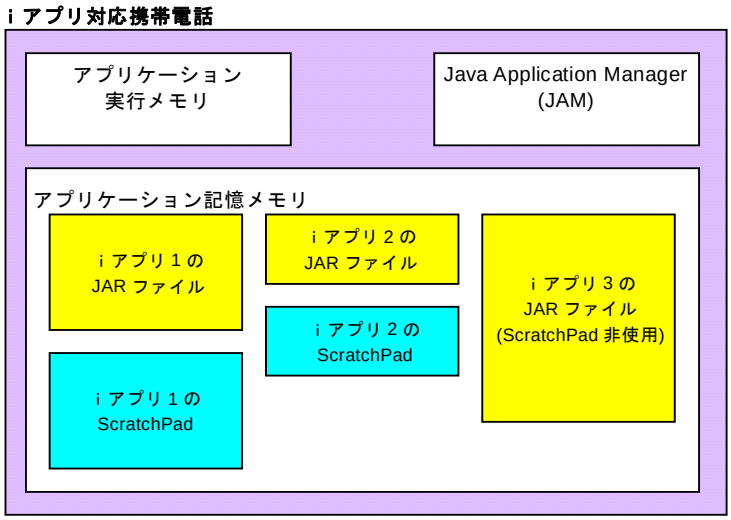


図 6: i アプリ対応携帯電話のメモリ

開発者にとって、メモリには次の 2 つの制約があります。

- 限られたアプリケーション実行メモリ。i アプリはこの領域で実行されます。実行時に生成されるオブジェクトもここに格納されます。
- 限られたアプリケーション記憶メモリ。ここには i アプリの JAR ファイルと、その i アプリに対応する ScratchPad が格納されます。

アプリケーション実行時サイズに対する絶対的な制約は、i アプリが実行される携帯電話に搭載されているアプリケーション実行メモリの量によります。搭載されるアプリケーション実行メモリの量、および i アプリ実行時にアプリケーション実行メモリがどのように使われるかはメーカーにより異なります。

またアプリケーション記憶サイズは、プロファイルの世代により以下のように制限されています（以降、1K バイト=1024 バイトとします）。これらのサイズの制限を越える i アプリは携帯電話に正しくダウンロードできないのでご注意ください。

プロファイル	PDC 携帯電話		FOMA 携帯電話	
	JAR ファイル	ScratchPad	JAR ファイル	ScratchPad
DoJa-1.x	10K バイト	5K バイト	30K バイト	10K バイト
DoJa-2.x	30K バイト	100K バイト	30K バイト	200K バイト
DoJa-3.x	30K バイト	200K バイト	100K バイト	400K バイト
DoJa-4.x	————	————	100K バイト	400K バイト
DoJa-5.x	————	————	JAR ファイルと ScratchPad の合計で 1024K バイト (*1)	

(*1)メーカーによっては、ダウンロード即起動 i アプリでは DoJa-4.x プロファイルと同様に、JAR ファイルサイズが 100K バイトまで、ScratchPad サイズが 400K バイトまでに制限される場合があります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

注意事項：

- ScratchPad サイズは、メーカーによっては規定のサイズより大きいサイズをサポートする場合があります。またメーカーの採用する領域管理方式によっては、新たにインストールしようとしている i アプリの ScratchPad サイズが、インストール済み i アプリが使用している ScratchPad サイズの合計サイズにより制限を受ける場合もあります。いずれの場合にも、i アプリが ADF の `SPsize` キーで宣言した容量の ScratchPad を確保できない場合、その i アプリは携帯電話にダウンロードすることはできません。

【DoJa-3.0】

DoJa-3.0 プロファイル以降では、1 つの i アプリに割り当てられた ScratchPad を内部的に最大 16 個まで分割管理することができます。分割管理された ScratchPad へのアクセス方法は 7.1 項を、ScratchPad を分割管理するための ADF の設定方法は 15.5.1 項を参照してください。

【DoJa-5.0】

DoJa-5.0 プロファイル以降では、上記の規定により 100K バイトを超える JAR ファイルをダウンロードすることができます。ただし携帯電話に 100K バイトを超える JAR ファイルをダウンロードできるようにするためには、 i アプリの配布に使用する Web サーバーがレンジリクエストによるバイト単位の部分的 GET に対応していなければなりません。レンジリクエストに対して部分的 GET の応答（206 Partial content）を返さない Web サーバーからは、100K バイトを超える JAR ファイルをダウンロードすることはできないため注意してください。

特に、JAR ファイルのダウンロードにおいて CGI を利用しているような場合には注意が必要です。

また上記で解説したレンジリクエストでは、If-Unmodified-Since ヘッダを使用した条件付きリクエストが行われます。これに起因して、i アプリのダウンロード元を複数サーバでロードバランス処理させている場合などでは、個々のサーバのシステム時刻、およびそれらに配置する JAR ファイルのタイムスタンプを合わせていない場合、 i アプリのダウンロードが正しく行えないことがあるため注意してください。

3.2.1 メモリ使用量の最小化

ここでは、大きなカテゴリとして、アプリケーション実行メモリとアプリケーション記憶メモリに分けて説明します。

アプリケーション実行メモリ（ヒープメモリ）

通常、アプリケーション実行メモリを占める最大のコンポーネントは実行可能プログラム（ i アプリ実行環境および i アプリを構成するクラス群）そのものです。 i アプリとそれに必要なすべてのオブジェクトが携帯電話のアプリケーション実行メモリに入りきらない場合は、実行可能プログラムのサイズか生成されるオブジェクトセットのサイズ、またはその両方を削減する以外に方法はありません。
実行可能プログラムのサイズを削減する方法については、3.2.2 項を参照してください。

生成されるオブジェクトのサイズを削減する場合は、次に示す内容を検討します。

- ガベージコレクションを頻繁に呼び出す。特に搭載メモリ量の少ない i アプリ実行環境では、アプリケーション実行メモリ上に発生したフラグメンテーションが原因で、一度にまとまったサイズのメモリを確保できなくなるケースが発生します（`OutOfMemoryError`、`UIException`、`ConnectionException` などの例外が発生したり、 i アプリの動作が不安定になるなどの問題として現れます）。この場合、できるだけ早め早めにガベージコレクションを呼び出すようにすることで問題が解決することがあります。
- メモリにはできるだけ少ないオブジェクトを保持する。または生成するオブジェクトの数およびサイズを最小限に抑える。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- アプリケーションプログラムの分割を検討する（例えば複数のレベルをもつゲームの設計では、レベル毎にアプリケーションプログラムを分割し複数の i アプリとするなど）。
- 可能な限り static を使用する。
- 過度のメソッド呼び出しのネストや再帰を避けてスタックの深さを最小限に抑える。
- メソッドに渡すパラメータの数を最小限に抑える。
- クラスの数、および個々のクラスのサイズを最小限に抑える。
- スレッドの多用を避ける。
- サイズがあらかじめ決められたクラスを作成する。`Hashtable`、`ByteArrayInputStream`、`ByteArrayOutputStream`、`Vector` など、サイズが動的に大きくなるクラスを使用すると、メモリのフラグメンテーションを引き起こすことがあります。
- 必要がなくなったメディアデータやイメージは速やかに破棄（`dispose()`）する。

生成されるオブジェクトサイズを可能な限り小さくし、アプリケーション実行メモリを効率的に使用しているにもかかわらず、まだメモリが不足する場合は、次の内容を検討してください。

- 変数に保存しているデータを ScratchPad に移す。
- メモリを消費しやすい処理をクライアント側からサーバー側に移す。
- サーバー側からデータをダウンロードする場合、一括してダウンロードするのではなく、必要最小限のものに絞り部分的にダウンロードする。

【DoJa-5.0】

DoJa-5.0 プロファイルにて、アプリケーション実行メモリの使用状況をモニタリングするためのユーティリティである `MemoryManager` クラスが `com.nttdocomo.lang` パッケージに追加されました。

CLDC の `java.lang.Runtime` クラスにも `freeMemory()`や`totalMemory()`といった類似の機能がありますが、`MemoryManager` クラスではそれら機能に加え、アプリケーション実行メモリ上の最大連続空き容量（フラグメントによって分断されていない連続空きエリアのうち、最大のエリアの容量）を知ることができます。また、i アプリ実行環境が複数のアプリケーション実行メモリ区画を持っている（Java オブジェクト用のヒープの他にメディアデータ専用のヒープを持っているなど）場合には、区画ごとにこれらの情報をモニタリングすることができます。

アプリケーション記憶メモリ

i アプリを保存するために、アプリケーション記憶メモリの中に以下の 2 つのブロックが確保されます。

- JAR ファイルの格納に割り当てられるブロック
- ScratchPad に割り当てられるブロック

JAR ファイル

JAR ファイルには、できるだけコンパクトにした必須のファイルだけを入れる必要があります。i アプリオプション API や i アプリ拡張 API を使用するなどの理由で携帯電話の機種ごとに異なるモジュールを i アプリに入れる必要がある場合は、機種ごとに別の i アプリ（別の JAR ファイル）を作成することを検討してください（機種別にダウンロードを制御する方法については第 16 章を参照）。このようにすれば、携帯電話にダウンロードされる JAR ファイルに、不要なエントリが含まれることはありません。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

また JAR ファイルには画像やサウンドなどのリソース（第 7 章参照）を含めることができますが、これは JAR ファイルサイズに大きな影響を与えます。使用頻度の低いリソースは、 i アプリ実行時に通信機能を使用して入手することを検討してください。

なお、 i モードサービスでは通信料金にパケット課金システムが採用されています。JAR ファイルを小さくすることは、 i アプリダウンロード時のユーザーの通信コストを削減することにもつながります。

ScratchPad

必要以上に大きな ScratchPad は指定しないでください。さらに、どの位の ScratchPad 容量を割り当てるにしろ、できる限り効率的に使用してください。ScratchPad サイズは、 i アプリの ADF に指定します（詳細は 15.5.1 項を参照）。なお、 i アプリが使用可能な ScratchPad サイズには、3.2 項で解説した制限があります。

なお、ScratchPad にどのような物理デバイス（ハードウェア）を使用するかはメーカーにより異なります。必要以上に大きな ScratchPad を割り当てると、機種によっては ScratchPad へのアクセス速度が低下して i アプリの実行パフォーマンスにも影響を与えることがありますので注意してください。

3.2.2 クラスファイルサイズ

コンパイルした Java クラスファイルにはデバッグのための情報が含まれることがあり、この情報によりクラスファイルのサイズが増大することがあります。定義するクラスの数を最小限に抑えるとともに、コードを`-g:none` オプションでコンパイルすることでデバッグ情報を取り除き、クラスファイルのサイズを小さくするようにしてください。

なお、NTT ドコモの提供する i アプリ開発環境のビルド機能を利用して i アプリをビルドする場合、コンパイル時には`-g:none` オプションが設定されています。 i アプリ開発環境の詳細については、第 15 章を参照してください。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

3.3 使い易さと GUI の設計

この節では、簡単で直感的に使用できる i アプリを作成するためのガイドラインを示します。携帯電話のようなプラットフォームでは、一般的にはたくさんのキー操作を必要とする複雑な i アプリは敬遠され、よりキー操作が少ない i アプリが求められることが想定されます。注意すべき点を次に列挙します。

- i アプリの全体を通して一貫した Look&Feel を維持します。
- 物理的な画面領域が小さいため、1 つの画面に表示する UI コンポーネントの数を最小限に抑えます。特にユーザー操作可能な UI コンポーネント（ボタンなど）を 1 つの画面に多数表示した場合、フォーカス移動操作などの手間により、ユーザーにとって操作が煩わしく感じられることがあります。
- UI レイアウト管理はメーカーによって異なりますが、ほとんどの場合、デフォルトのレイアウト管理は簡単なフローレイアウトです。UI レイアウトをシンプルにし、 i アプリ実行画面の外観が、メーカー間で大きく変わらないようにします。
- スクロールを最小限に抑えます。なお、`Panel` は垂直方向のスクロールをサポートしますが、`Canvas` を使用する場合は画面のスクロールはアプリケーションプログラム側で制御する必要があります。
- ソフトキーは、ユーザー操作を簡単にする便利な方法です。複雑なユーザー操作が発生するような画面では、ソフトキーをショートカットキーのように使用することを検討してください。たとえば多くの入力用コンポーネント部品が配置された画面において、ソフトキーはどのコンポーネントにフォーカスがある状態でも確定動作を行わせるためのキーとして使用することができます。
- 携帯電話でのテキスト入力は煩雑なため、入力する文字数を最小限に抑えます。可能な限り選択肢を表示し、その中からユーザーが選択する形の入力方式にします。
- よく行われる操作は、多数のメニュー階層を移動したり、多数のキー操作をしなくても使用できるように画面構成を設計します。

3.3.1 ターゲットユーザー向けの設計

電話はだれでも使いますが、コンピュータはだれもが使うわけではありません。したがって、UI コンポーネント間のフォーカス移動やスクロールという考え方は、全てのユーザーにとって直感的にわかるわけではありません。広範なユーザーに使用してもらうためには、ユーザーインタフェースはできるだけ直感的に操作できなければなりません。i アプリのユーザーインタフェースを設計する際は、それを使用すると見込まれているターゲットユーザー層を見極める必要があります。

3.3.2 i アプリの操作方法の設計指針

ユーザー操作は、それがどのくらい頻繁に行われるかや、ユーザーにとってどの程度重要なものであるかによって、いくつかのカテゴリに分類する必要があります。一般には、頻繁に行われる操作はより簡単に使えるようにすべきです。たとえば、銀行残高の確認は口座間の振り替えよりも頻繁に行われます。したがって理想的には、残高の確認は振り替えよりも少ないキー操作でできるようにすべきです。

一方、個人識別 ID の入力など、いくつかのキー操作が必要な操作は、それが重要な機能である限り（個人識別 ID の例ではユーザーの銀行口座などへの不正アクセスの防止）、ユーザーに受け入れられます。さらに i アプリを携帯電話にインストールして最初に使い始める際に、多少煩雑ではあってもその後の操作を簡単にする目的で、一定の情報入力をユーザーが行うようにするということも検討の余地があります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.
操作には一貫性がなければなりません。項目の選択方法は、どの画面でも一貫しているべきです。ソフトキーは、誤操作や使いづらさを避けるため、i アプリ全体を通して一貫した考え方に基づいて使用できなければなりません。また最も重要かつ頻繁に使用される操作は、最も少ないキー操作でできるようにすべきです。

3.3.3 スクロール

理論的には、物理画面よりも大きな仮想画面を設計することができます。しかしそのような画面では、画面全体を見たり UI コンポーネント間をフォーカス移動するには、多くのスクロールが必要になります。スクロールは必ずしもレスポンスのよい画面操作ではないため、ユーザーはスクロールの多い画面を使いづらいと感じる可能性があります。スクロールは最小限に抑えることを目標にすべきです。なお、`Panel` は垂直方向のスクロールのみサポートします。

【DoJa-2.0】

DoJa-1.0 プロファイルにおいては、`Panel` のスクロールサポート有無はメーカーの実装に委ねられていました。DoJa-2.0 プロファイル以降では、デフォルトレイアウトマネージャまたは HTML レイアウトマネージャが設定されている `Panel` は縦方向のスクロールをサポートします。

3.3.4 キーおよびソフトキー

i アプリでは以下の標準的なキーを使用することができます。

- 上下左右キーおよび選択（決定）キー
- 0～9 までの数字キーおよび#キー、*キー
- 待ち受けアプリケーション切り替えキー

メーカーによっては、i アプリオプションのカテゴリに属する機能としてこれら以外の特別なキーをサポートする場合があります。

また、i アプリ対応携帯電話には上記以外に 2 つのソフトキーが設けられます。ソフトキーの外観や場所はメーカーによって異なります。前述の通り、ソフトキーを使用するとユーザーインタフェースの使用感が大幅に改善されることがあります。i アプリの設計ではソフトキーの使用を考慮すべきですが、特定の外観や場所に依存することは避けるようにしてください。

【DoJa-2.0】

待ち受けアプリケーション切り替えキーは、DoJa-2.0 プロファイルの待ち受けアプリケーション機能のために新設されました.。待ち受けアプリケーションが休眠状態や非活性化状態である場合を除き、i アプリは直接このキーを使用（キーイベントの取得）することができます。待ち受けアプリケーションが休眠状態や非活性化状態である場合は、待ち受けアプリケーション切り替えキーのイベントはシステムが使用します。

なお、待ち受けアプリケーションの詳細については第 9 章を参照してください。

【DoJa-5.0】

DoJa-5.0 プロファイル以降、メーカーによっては ADF の `DrawArea` キーを使用することで、デフォルト（`DrawArea` キー未使用時）の表示領域よりも広い表示領域を使用できる場合があります。そのような機種には、`Canvas.setSoftLabelVisible()`メソッドを使用することで、`Canvas` 表示時のソフトキーラベルの表示のオン、オフを切り替えることができるものがあります。なお、`setSoftLabelVisible()`メソッドは `Frame` クラスで定義されたメソッドですが、現在のプロファイルでは `Canvas` 以外のフレームオブジェクト（`Panel` や `Dialog`）に対してこのメソッドを呼び出すことはできません。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

3.3.5 データ入力

ユーザーがデータを入力する手段は、原則的に携帯電話のキーパッドだけです。したがって、テキストデータを入力するよりも数字データを入力する方がより簡単に行えます。これはテキストデータの場合、1 文字を入力するために複数回のキー操作が必要になるためです。テキスト入力は最小限に抑えるべきです。

3.3.6 パスワード入力

パスワードの入力は次のいずれかの方法で行います。

- パスワードを表示しない場合は、`TextBox` のコンストラクタで表示モードパラメータに `TextBox.DISPLAY_PASSWORD` を渡します。すべての入力文字は「*」で表示されます。
- パスワードを表示する場合は、`TextBox` のコンストラクタで表示モードパラメータに `TextBox.DISPLAY_ANY` を渡します。この場合、入力された文字はすべて画面上に表示されます。

1 つ目の方法を使用する場合、一般的にはパスワードには数字のみを使用します。携帯電話では PC と異なり、数字以外の文字をパスワード入力することはユーザーにとって非常に困難な作業となります。パスワードに使用する文字を数字のみとする場合、`TextBox.setInputMode()`メソッドを使用して `TextBox` の初期入力モードを数字入力モードとしておくことで、ユーザーの誤操作を防ぐことができます。

パスワードに数字以外の文字（テキスト文字）を許可する場合は、2 つ目の方法を使用します。

3.3.7 スレッドの使用

スレッドは、ネットワーク入出力や UI での対話など、長時間を要するブロック操作を行うときには非常に便利です。しかし、どのようなマルチスレッドシステムでも、アクティブスレッド間のコンテキスト切り替えにはオーバーヘッドが必要です。携帯電話の処理能力は限られているため、ユーザー操作をストレスなく可能にするためにスレッドの数は最小限にすべきです。

また、アプリケーションプログラム側でマルチスレッドプログラミングを行っていなくても、システムがイベント処理などのためにスレッドを生成することがあります。どの機種でも動作する i アプリを作成するには、スレッドセーフなプログラム設計が必要です。

注意事項：

- i アプリ実行環境では、以下のような状況でダイアログやポップアップが表示されます。
  - アプリケーションプログラム自身が `Dialog` クラスを使用してダイアログを表示させる場合
  - HTTP(S)通信において、Web サーバーから認証（BASIC 認証）を求められた場合
  - HTTPS 通信においてサーバー証明書の内容に問題があり、ユーザーに警告を示す場合
  - OBEX 外部接続機能により赤外線ポートを使用する場合
  - アプリケーション連携機能により、他のアプリケーションの起動や呼び出しをユーザーに確認する場合

  これらのダイアログやポップアップは排他的にのみ表示可能であり、複数のスレッドからこれらを同時に表示させることはできません。これらを複数のスレッドから同時に表示させようとした際の振る舞いはメーカーにより異なります。開発者は、これらの状況が複数スレッドで同時に発生することのないようアプリケーションプログラムを設計する必要があります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

3.4 セキュリティ

クライアントとサーバー間で機密情報を送受信するｉアプリを設計する場合には、HTTPS プロトコルを使用することを検討します。ただし、クライアント側でこのメカニズムを利用する際には、CPU 負荷が高くなることに注意してください。これは、各 SSL セッションの始めにクライアント側（この場合ｉアプリ）でマスターキーを生成する必要があるのに加え、データが処理されてサーバーとのデータ送受信が行われるたびに、これらのキーに基づくデータの暗号化や暗号化解除が必要になるためです。

HTTPS プロトコルを利用した通信については、5.3 項を参照してください。

なお、ｉアプリがサーバーとの通信を行う場合、その接続先はｉアプリのダウンロード元とプロトコル、サーバーアドレス、ポート番号が同じでなければなりません。したがって、HTTPS プロトコルを使用するｉアプリは HTTPS プロトコルを使用してダウンロードされている必要があります。

3.5 無線ネットワークでの操作

ｉアプリは、通信接続の中断がごく普通に発生する環境でも使用されることに留意してください（ユーザーは携帯電話の電波が届く区域を時に頻繁に出入りします）。またｉモードサービスではパケット課金システムが採用されているため、ｉアプリが通信を行う場合は、必要なデータだけを送受信するようにしてください。

クライアント/サーバー型のｉアプリの設計においては、小さいデータパケットトランザクションをいくつも送受信する（切断によるデータの喪失を最小限にする）か、1 つの大きなデータパケットを送受信する（通信オーバーヘッドを最小限にする）かについてバランスをとる必要があります。

接続の確立、テスト、終了に関する問題や、トランザクションやセッションの管理に関する問題の詳しい説明は、第 5 章を参照してください。さらに、次項「処理の中断と再開」を参照してください。

注意事項：

- 通信を行うｉアプリでは、ｉアプリ開発者は通信エラーや通信の結果得られたデータのエラーをリカバリーするために通信処理のリトライを行う処理を組み込むことがあります。

  通信に関するエラーをｉアプリが検出して自動的にリトライを行うことは、通信品質（電波状況）が頻繁に変化する携帯電話上でｉアプリを快適に使用できるようにする観点で有用な手段です。しかしｉアプリが無限に通信のリトライを行わないよう、そのような処理を組み込む場合は必ず繰り返し回数に上限を設けるようにしてください。

  ｉモードサービスではパケット課金システムが採用されており、通信料金は実際に行われた通信量に比例して課金されます。1 回の通信サイズが小さくても、上限なく通信のリトライが行われると短時間のうちに非常に高額の課金が行われる可能性があります。

  通信処理、または通信処理を含むアプリケーションロジックのリトライは多くても数回程度までとし、その回数で目的の処理を完遂することができなかった場合はダイアログ表示などによりユーザーにエラーを通知するような仕組みとすることを強く推奨します。

  （待ち受けアプリケーションの場合には、エラーを検出した際にｉアプリを自動的に終了させるような仕組みとすることも推奨されません。待ち受けアプリケーションはシステムにより自動的に再起動され、結果として同じ処理を繰り返してしまう可能性があるためです。）

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

3.6 処理の中断と再開

JAM はｉアプリ実行中に通話の着信があると、動作しているｉアプリを自動的に中断します。そしてユーザーが通話を終了すると、Resume（再開）イベントをｉアプリに送信して実行を再開します。このように、ｉアプリを実行中の携帯電話上で、より優先度の高い別の機能が動作する必要が生じた場合、ｉアプリ実行の中断や再開が行われる場合があります。

ｉアプリの実行が中断される条件は、通話の着信時以外にもいくつか存在します。以下に、ｉアプリの中断、再開が行われるケースを示します。

- ｉアプリ実行中に通話着信があった場合（ｉアプリが通信実行中の場合、携帯電話のｉモード着信設定により処理が中断されない場合もあります）。および、待ち受けアプリケーションが非活性化状態または休眠状態で実行されている際にメール・メッセージ着信があった場合。なお待ち受けアプリケーションが活性化状態で実行されている場合は、通常のｉアプリの実行時と同様に、メール・メッセージの着信を受けてもその実行は継続されます。
- ｉアプリ実行中に、他のアプリケーションが起動された場合。これには、ｉアプリ実行中に時刻アラームなどのネイティブアプリケーションが割り込んで自動起動した場合や、アプリケーション連携機能によりｉアプリから他のネイティブアプリケーションやｉアプリを起動した場合を含みます。なお、起動されたアプリケーションがパケット通信を行うタイプのもの（ブラウザやメーラ、ｉアプリなど）である場合、呼び出し元のｉアプリは中断から再開されることなく終了します。
- ｉアプリから、ユーザー操作を伴うネイティブ機能を呼び出した場合。これには、アプリケーション連携機能を中心に、システムによりユーザー確認が求められる多くの機能が含まれます。
- ユーザーがｉアプリの強制終了操作を行い、終了確認ダイアログでユーザーが強制終了をキャンセルした場合。
- 携帯電話が上記以外にｉアプリを中断、再開するための機構（中断・再開キーなど）を持っており、その操作が行われた場合。

【DoJa-2.0】

ｉアプリの中断・再開条件については DoJa-2.0 プロファイルにおいて明確化されました。DoJa-1.0 プロファイルでは必ずしも上記の動作とはならない場合がありますので注意してください。

【DoJa-3.0】

ｉアプリの中断・再開は DoJa-3.0 プロファイルにて標準機能に取り入れられました。DoJa-2.0 プロファイル以前のプロファイルに対応した携帯電話では、ｉアプリの中断・再開をサポートしない場合があります。そのような携帯電話では、ｉアプリの実行が中断されるべき状況になるとそのｉアプリは終了します。

【DoJa-3.5】

DoJa-3.5 プロファイル以降に対応した FOMA 携帯電話では PDC 携帯電話と異なり、ｉアプリが通常実行中の場合、および待ち受け活性化状態でメール・メッセージの着信を受けると、ｉアプリをサスペンドすることなくバックグラウンドでそれらを取得します。なお、DoJa-3.5 プロファイル以降に対応した FOMA 携帯電話でも、ｉアプリが待ち受け非活性化状態または休眠状態の場合は、従来同様ｉアプリをサスペンドしてからメール・メッセージを取得します。

ｉアプリの実行が再開されると、ｉアプリメインクラス（`com.nttdocomo.ui.IApplication` を継承したクラス）の `resume()`メソッドが呼び出されます。開発者は `resume()`メソッドをオーバーライドすることによりｉアプリの再開時処理を実装することができます。

スタンドアロンの電卓アプリケーションなど、ある種のｉアプリでは、開発者が中断/再開のために特別なステップを行う必要はありません。これは、ｉアプリが中断されても内部的には何の変化

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.
も起らないことと、アプリケーションプログラムの処理進行が完全にユーザー生成のイベント（キーの操作）によって駆動されるからです。しかし特定の種類のｉアプリでは、実行を再開する際に特別なステップを行う必要があります。一般に、次のような状況では、ｉアプリの実行を再開する際に何らかのアクションをとる必要があります。

- クライアント/サーバー型のｉアプリでは、ｉアプリの実行が再開されたら通信結果をチェックし、必要に応じて通信のリトライを行います。これは、ｉアプリが通信処理を行っている際に中断・再開が発生すると、その通信処理は失敗するためです。
- ｉアプリが中断、再開する際には、そのｉアプリが中断する前に動作していたタイマー（`Timer` および `ShortTimer`）は停止しています。これらのタイマーは、ｉアプリ再開時に明示的に再スタートを行う必要があります。

  なお DoJa-1.0 プロファイルでは、タイマーを使用中にｉアプリの実行が中断された場合、ｉアプリは再開後にタイマーのインスタンスを破棄した上で再作成する必要があります。
- ｉアプリ中断時に `AudioPresenter` がサウンドを再生していた場合、`AudioPresenter` は動作を停止します。ｉアプリ再開後も自動的に再生を開始することはありません。この場合、ｉアプリから明示的に再生指示（`play()`）を行うことにより、サウンドデータの再生はデータの先頭から行われます。なお、`VisualPresenter` が動画を再生しているときにｉアプリの実行が中断された場合は、動画再生は一時的に停止しますが、ｉアプリの再開とともに中断時のフレームから動画再生も再開されます。
- DoJa-2.0 プロファイルでは、ｉアプリの中断・再開により `Canvas` が再表示される際、ｉアプリ実行環境からその `Canvas` の `paint()`メソッドが 1 回呼び出されることが保証されます。しかしその `paint()`メソッド呼び出しで全画面が完全には復元されないようなケース（アプリケーションプログラムがイベントに応じて差分描画のみ行うようになっているなど）では、どの程度システムが画面を復元できるかは機種により異なります。

  なお DoJa-3.0 プロファイル以降では、ｉアプリの再開時にシステムが `Canvas` の描画内容をすべて復元できた場合は、`paint()`メソッドは呼び出されません。
- ｉアプリの中断・再開が発生した場合、再開したｉアプリが適切なプロンプトメッセージを表示して一旦停止することが望ましい場合があります。たとえば、多くのアクションゲームのようにユーザーとの即座対話が必要なケースでは、ｉアプリが再開されてからユーザーの操作の準備が整うまでの間に若干の時間的な猶予を持たせることで、再開時のｉアプリの操作性を向上させることができます。

【DoJa-2.0】

中断・再開がｉアプリに与える影響については DoJa-2.0 プロファイルにおいて詳細に規定されました。DoJa-1.0 プロファイルでは必ずしも上記の動作とはならない場合がありますので注意してください。

【DoJa-3.0】

DoJa-3.0 プロファイルでは、ｉアプリの中断・再開が発生した際に、中断の理由と中断中に発生した事象を取得するための `IApplication.getSuspendInfo()`メソッドが追加されています。

注意事項：

- ｉアプリが通信を行っている場合、通話の着信でｉアプリが中断するか否かは、携帯電話のｉモード着信設定の内容に従います。ｉモード通信中に通話の着信を許可しない設定になっている場合は、ｉアプリが通信を行っている間は通話の着信を受けてもｉアプリの実行は中断されません。
- ｉアプリの実行が中断された際に未処理のイベントがイベントキューにキューイングされていた場合、ｉアプリ再開時にはそれらは全て破棄されます。
- ｉアプリ実行再開時のスレッド動作順序、およびスレッドスイッチ制御の詳細は Java 仮想マシンのスレッド制御に委ねられるため、`resume()`メソッドの呼び出しはｉアプリ実行の再開において必ず最初に行われるとは限りません。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

3.7 ハードウェアのアクセス

iアプリ実行環境では、携帯電話のハードウェア（プラットフォームリソース）にアクセスする手段として `PhoneSystem` クラスを提供しています。このクラスを使用することで、ディスプレイのバックライトやバイブレータの制御などを行うことができます。iアプリからプラットフォームリソースに頻繁にアクセスすると、携帯電話のバッテリーを大量に消費する場合がありますので注意してください。iアプリでこれらの設定を変更した場合でも、iアプリ終了時には自動的に元の設定に戻されます。

【DoJa-3.0】

DoJa-2.0 プロファイル以前では、プラットフォームリソースアクセス機能のほとんどは i アプリオプション API のカテゴリに含まれており、メーカーによっては一部の機能が搭載されないことがありました。DoJa-3.0 プロファイルでは、これらの多くが i アプリ基本 API に取り入れられています。

3.8 エラー処理

iアプリ実行環境の提供する API では、処理中に異常を検出した場合に、一般的な Java API と同じくエラーや例外をスローします。アプリケーションプログラムでは発生した例外をキャッチして、必要に応じて適切な処理（例外メッセージをユーザーに提示するなど）を行うようにしてください。

なお、実行時例外（`java.lang.RuntimeException` およびサブクラス）やエラー（`java.lang.Error` およびサブクラス）がアプリケーションプログラムでキャッチされなかった場合、そのiアプリは異常終了します。

3.9 待ち受けアプリケーション

DoJa-2.0 プロファイル以降では、時計アプリケーションをはじめとする常駐タイプのiアプリを使い勝手よく実装するための機構として、待ち受けアプリケーションをサポートしています。

待ち受けアプリケーションは、実行中のiアプリの画面を、待ち受け画像のように待ち受け画面に表示することのできる常駐型アプリケーションです。もちろん、アプリケーションプログラムが画面の内容を変更すると、それは直ちに待ち受け画面に反映されます。

DoJa-1.0 プロファイルの上でも、iアプリが常に実行されている状態を保つことで待ち受け画面にiアプリが常駐しているようにユーザーに見せることは可能でした。しかしこの方法では、以下のような点において問題がありました。

- 電話をかけたり Web ブラウジングを行おうとした場合、ユーザーは明示的にiアプリを終了させなければならない。また、それらの作業が済んだあとはユーザーが明示的にiアプリを再起動しなければならない。
- iアプリ実行中はメールの着信を受けることができない。このため、ユーザーはメールが来ていることに気が付かないことがある。
- iアプリの実行時間がそのまま電力消費量に結びつき、長時間駆動し続けることができない。

待ち受けアプリケーションでは上記のような問題を解消し、ユーザーにとって使い勝手のよい常駐型アプリケーションを実現するための機構が取り入れられています。常に実行され続けていること

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.
を前提としたiアプリを開発する場合、そのiアプリを待ち受けアプリケーションとして実装することを検討してください。

なお、待ち受けアプリケーションの詳細は、第9章を参照してください。

3.10 携帯電話の機種の判別

iアプリが現在どのような機種で実行されているかを知るための手段として、以下の2つが用意されています。

- iアプリ実行環境におけるシステムプロパティ `microedition.platform`

  アプリケーションプログラムは、`java.lang.System.getProperty()`メソッドを使用してシステムプロパティ（システムの情報を取得するためのプロパティセット）を参照することができます。`microedition.platform`プロパティは、CLDCでホストプラットフォームを示すシステムプロパティとして規定されていますが、iアプリ実行環境ではこのプロパティの値に携帯電話の機種名が格納されます。iアプリは、この値を調べることで自身が動作している携帯電話の機種を知ることができます。

  このプロパティに格納される機種名は、ユーザーエージェントに含まれている名前と同等のものとなります。

- HTTP通信時にサーバーに送出される`User-Agent`リクエストヘッダ

  ブラウザからのWebアクセスと同様、iアプリからHTTP通信を実行した際にはリクエストヘッダにユーザーエージェント（`User-Agent`ヘッダ）が含まれます。iアプリと通信を行うサーバーアプリケーションは、`User-Agent`ヘッダの内容を調べることで現在対話を行っている携帯電話の機種を知ることができます。

【DoJa-2.0】

DoJa-1.0プロファイルでは、`microedition.platform`にどのような値を設定するかはメーカーの判断に委ねられているため、メーカーによっては機種名が取得できないことがあります。

3.11 FOMA携帯電話における複数アプリケーション同時起動

FOMA携帯電話の中には、iモードのサイト閲覧と通話を同時に行うなどのために、複数の端末アプリケーション（例えばブラウザと通話機能など）を同時に起動することができるものがあります。ここでは、そのような携帯電話上でiアプリのライフサイクルが受ける影響について解説します。

(1) iアプリ実行中の、ユーザー操作による他アプリケーションへの切り替え

iアプリ実行中にユーザー操作（タスクキー操作等）により他のアプリケーションへの切り替えが指示された場合、原則としてそのiアプリの実行は中断します。ユーザー操作により再度そのiアプリへの切り替えが行なわれた時点でiアプリの実行は再開されます。なお、どのようなアプリケーションがiアプリと同時に起動した状態にできるかはメーカーにより異なります。iアプリ起動中に、ユーザーがその機種でiアプリと同時に起動できないアプリケーション（例えばブラウザなど）の起動を指示すると、iアプリが終了してそのアプリケーションが起動されます。

アプリケーションプログラムは`IApplication.getSuspendInfo()`メソッドを使用してiアプリ実行の中断理由と中断中に発生した事象を取得することができますが、DoJa-3.5プロファイル以降では、本項で説明したアプリケーション切り替えによる中断を判別するために、中断中の発生事象`IApplication.SUSPEND_MULTITASK_APPLICATION`が追加されています。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

(2) i アプリ実行中のメール着信動作

PDC 携帯電話では i アプリ実行中（ i アプリ通常起動中および i アプリ待ち受け実行における活性化状態中）にメールの着信を受けても、メールの取得は行なわずアイコン等により i モードセンターに未取得メールがある旨をユーザーに通知する仕様となっています。これに対し FOMA 携帯電話では、このような状態でメールの着信を受けると i アプリを中断することなくバックグラウンドでメールの取得を行ないます。バックグラウンドでメールの取得が行われている状態で、パケット通信と同時に実行することのできない機能（赤外線リモコン機能など）を i アプリが使用すると例外が発生します。

なお FOMA 携帯電話であっても、 i アプリ待ち受け実行における非活性化状態中および休眠状態中では PDC 携帯電話と同様に i アプリをサスペンドしてメールの取得が行なわれます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

第 4 章
ユーザーインタフェースの設計

ユーザーインタフェース（本書では UI と記すことがあります）機能には、次のような基本的な要件があります。

- ボタンやラベルといった高レベルなユーザーインタフェース部品（コンポーネント）の作成
- イベント処理
- レイアウト処理
- 画像ファイルの表示、サウンドファイルの再生機能

i アプリ実行環境のユーザーインタフェース API には、高レベル API と低レベル API の 2 種類があります。次の各節では、個々の基本的な機能分野について説明し、そのコード例を示します。

高レベル API では、さまざまなコンポーネントが提供されます。開発者は、これらを組み合わせて i アプリのユーザーインタフェースを作成します。高レベル API を使用した i アプリはハードウェア特性を考慮する必要性が低く、メーカーの異なる携帯電話でも容易に動作させることができます。高レベル API では開発者の自由度は限定されますが、少ないコード量で高機能な i アプリを作成することができます。高レベル API のユーザーインタフェースを構成する部品をコンポーネントと呼びます。開発者は、これらのコンポーネントを組み合わせて i アプリを作成します。高レベル API を使用した画面は、おおむね i モード HTML と同程度の表現を行うことができます。

低レベル API では、携帯電話の画面サイズやキーパッドなどハードウェアの特性を考慮しながら i アプリを作成します。低レベル API を使用すれば、携帯電話の各機種の特性を引き出した i アプリを作成することができます。この場合、開発者の自由度は増しますが、グラフィックスの描画や画面サイズと論理座標値といった細かい点を考慮する必要があります。

1 枚の画面は、`Panel` または `Canvas` のインスタンスとして表現されます。`Panel` は高レベル API を使用する画面（コンポーネントベースで構築される画面）で、`Canvas` は低レベル API を使用する画面（描画ベースで構築される画面）で利用されます。ディスプレイには、`Panel` または `Canvas` のいずれかを 1 面だけ表示することが可能です。`Canvas` によって構成される画面でコンポーネントを使用することはできません。ただし `Panel` によって構成される画面には、イメージを取り扱うためのコンポーネントを使用することで低レベル描画を取り入れることができます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.
1 つの i アプリは複数の Panel や Canvas を持つことができます。アプリケーションプログラムは Display.setCurrent()メソッドを使用して、ディスプレイに表示する Panel や Canvas を切り替えることができます。

ボタンを押したり、タイマーが切れるといったイベントの処理は、高レベル API と低レベル API では異なります。低レベル API では、キーの押下や解放を含め、すべてのイベントを i アプリで受け取ることができます。一方高レベル API では、コンポーネントによっては、いくつかの低レベルのイベントを内部的に処理したり、より高レベルのイベントに変換して i アプリに通知したりします。

低レベル API の場合は、キーの押下や解放などによって発生した低レベルイベントは Canvas オブジェクトに通知されます。低レベル API では、開発者は Canvas クラスを継承したサブクラスを作成して使用しますが、このとき Canvas クラスの processEvent()メソッドをオーバーライドすることで低レベルイベントを受け取ることができるようになります。イベントが発生するとこのメソッドが呼び出されるので、その中でイベントに対する処理を行います。processEvent()メソッドには、発生したイベントのタイプと、イベントに特有の引数が渡されます。

高レベル API の場合は、ボタン押下などの高レベルイベントは Panel オブジェクトに通知されます。 i アプリの中でこのような高レベルイベントに対する処理を行う場合は、Panel オブジェクトにリスナーオブジェクトを登録する必要があります。Panel オブジェクトは、イベントが発生すると Panel オブジェクトに登録されているリスナーオブジェクトメソッドを呼び出し、イベントの発生を通知します。高レベルイベントのリスナーインタフェースは、イベント発生源ごとに異なるものが用意されています。イベント発生時にリスナーメソッドに通知される情報も、リスナーの種類により異なります。

イベントリスナーの詳細は、4.3 項を参照してください。

次に、ユーザーインタフェースのためのクラス構成を示します。なお、このクラス構成図では 13 章にて別途解説する 3D グラフィックス描画機能および 3D サウンド制御機能に関するクラス、インタフェースは省略しています。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

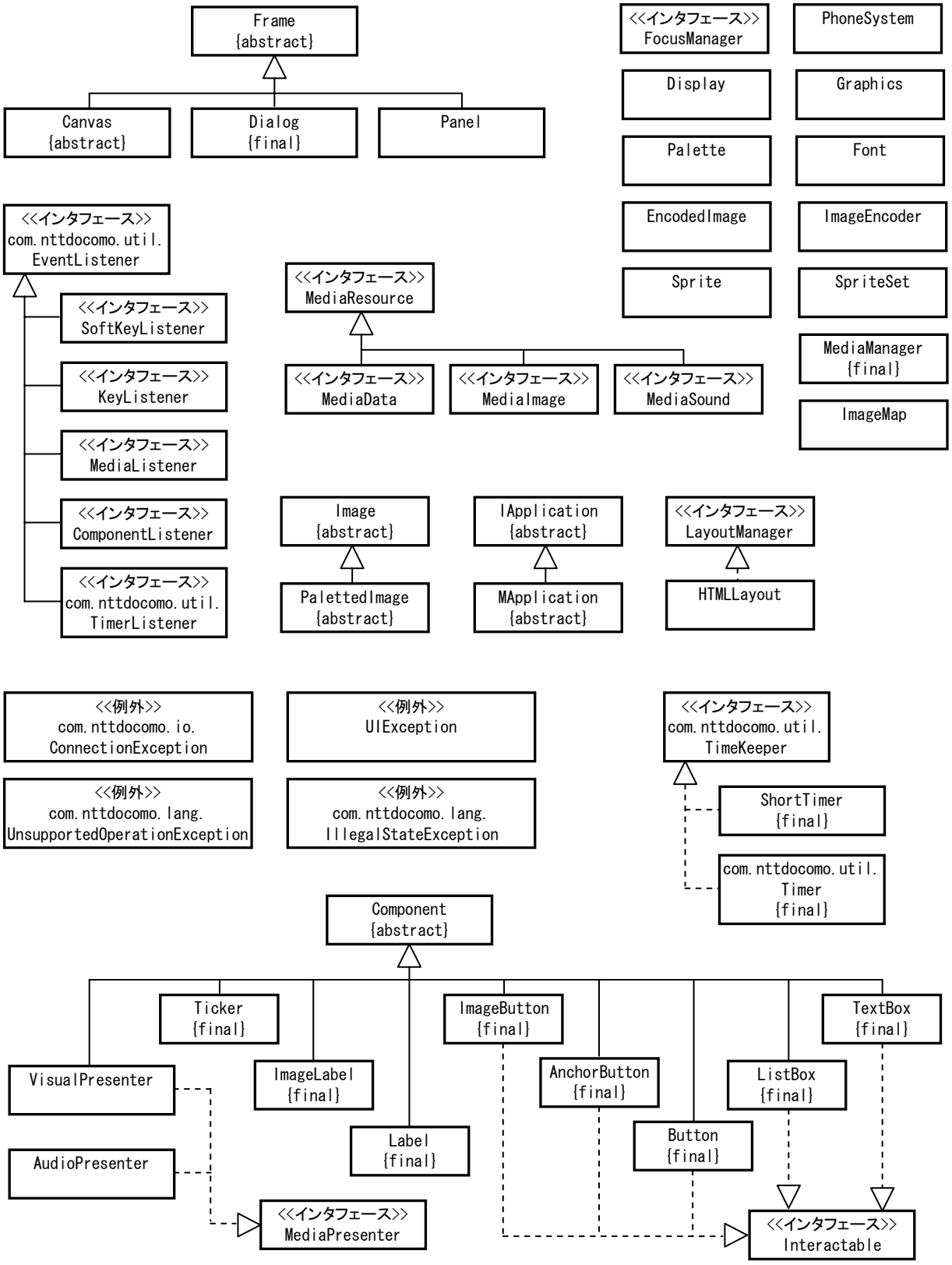


図 7: UI クラス階層

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

4.1 高レベル UI コンポーネントによるプログラミング

i アプリ実行環境では、9 つの基本的な UI コンポーネントが提供されます。

- `Label`
- `ImageLabel`
- `Button`
- `AnchorButton`
- `ImageButton`
- `ListBox`
- `TextBox`
- `Ticker`
- `VisualPresenter`

この他の基本 UI 部品として `Dialog` があります。上の各要素の動作は携帯電話の各メーカー間でほぼ同様ですが、Look&Feel は異なる可能性があります。これは、Windows、Macintosh、UNIX といったコンピュータプラットフォーム向けに開発された標準的な GUI アプリケーションの場合と同じです。これらのアプリケーションはどのプラットフォームでも同じように動作しますが、Look&Feel は異なっています。たとえば、ボタンの形状は、プラットフォームによって丸みがかっていたり四角だったり、枠が太かったり細かったりとそれぞれ異なります。これは携帯電話の場合も同様です。低レベルのグラフィックルーチンは、メーカーによって異なる外観を生成することがあります。デフォルトのレイアウト、スクロール、フォーカスの管理に関する方針は、どのメーカーでもほぼ同様です。ただしメーカーは個別に、自社の携帯電話向けにレイアウト、スクロール、フォーカスの管理を拡張している場合があります。次の各項の要素やレイアウトマネージャの説明は一般的なものであり、携帯電話各機種の詳細な振舞いは各機種毎に異なる場合があります。

なお、i アプリ実行環境には UI コンポーネントの共通的な特性を定義する `Component` クラスがありますが、このクラスはメーカーが上記の基本 UI コンポーネントを提供するために用意されているものです。開発者は `Component` クラスを継承した独自のコンポーネントクラスを定義、使用することはできません。

【DoJa-2.0】

`AnchorButton`、`ImageButton` は DoJa-2.0 プロファイルで新設されたコンポーネントです。

4.1.1 Panel の使用

`Panel` クラスは、UI 部品のシンプルなコンテナとして機能します。画面には、同時に 1 つのパネルしか表示できません。ただし、これは、1 つの i アプリが 1 つのパネルしか定義できないことを意味するわけではありません。多くの i アプリは複数のパネルから構成されています。ユーザーは、通常、これらのパネル間をナビゲーションしながら i アプリを操作します。

`Panel` には、その機能や目的を表示するためのタイトルを設定することができます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

4.1.2 Dialog の使用

ダイアログは、テキストメッセージを表示して警告や情報をユーザーに示したり、ユーザーに決定を促がすときに使用します。Dialog は Panel または Canvas の上に重なるように表示される画面であり、Frame クラスのサブクラスです。ダイアログには、タイトルとテキストメッセージ、それに最低 1 つのボタンがあります。ダイアログは同時に 1 つしか表示できません。

ダイアログには次のような種類があります。

- 情報
- 警告
- エラー
- Yes/No
- Yes/No/Cancel

注意事項：

- 同時に 2 つ以上のダイアログを表示しようとすると、例外がスローされます。
- ダイアログはモーダルです。
- バックグラウンドのスレッドがダイアログのモーダル操作に干渉しないようにアプリケーションプログラム側では注意が必要です。例えばダイアログの表示中にバックグラウンドスレッドが背後の Panel や Canvas を切り替えようとした際の動作は、機種により異なりますので注意してください。
- Dialog は、Panel や Canvas と同様に Frame クラスのサブクラスですが、Dialog を表示するために Display.setCurrent()メソッド（画面に表示されているフレームを切り替えるメソッド）を使用することはできません。Dialog を画面に表示するには、Dialog.show()メソッドを使用します。

4.1.3 コンポーネントの使用

UI コンポーネント（com.nttdocomo.ui.Component のサブクラス）には、Label、ImageLabel、Button、AnchorButton、ImageButton、TextBox、ListBox、Ticker、VisualPresenter があります。AnchorButton を除き、コンポーネントのサイズが画面サイズ（Panel にタイトルが設定されている場合はタイトル領域を除いた画面サイズ）を超えることはできません。

コンポーネントのオブジェクトは、1 つにつき 1 回だけ Panel に追加（Panel.add()メソッド）することができます。

次の各項では、各 UI コンポーネントについて詳しく説明します。

【DoJa-2.0】

DoJa-1.0 プロファイルでは、コンポーネントに画面サイズを超えるサイズを指定した場合の動作はメーカーにより異なります。

Label

Label クラスは Panel 上に 1 行のテキストを表示するためのコンポーネントです。Label コンポーネントでは、枠のない文字列が表示されます。Label コンポーネント内のテキストの表示位置は、左、中央、または右に位置合わせすることが可能です。次のコードは、Label コンポーネントの例です。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

例: Label コンポーネント

```
package uidemo;

import com.nttdocomo.ui.*;

/**
 * ラベルウィジェットとそれに対して呼び出された setSize メソッドの
 * 機能を示すクラス。
 * このクラスは 1 つのボタンと 2 つのラベルウィジェットを作成する。
 * ラベルウィジェットの可視/不可視の機能を示すため、
 * ボタンウィジェットを使用する。
 */
public class LabelDemo extends UIDemoPanel {
       Label lbl1, lbl2;
       Button btn1;
       int count = 0;

       LabelDemo() {
              lbl1 = new Label("Label1");
              lbl2 = new Label("This is a Label2");
              btn1= new Button("Button1");
              lbl1.setSize(25,15);   // ラベルのサイズを設定する。

              this.add(btn1);
              this.add(lbl1);
              this.add(lbl2);
              ListenerClass lclass = new ListenerClass();
              this.setComponentListener(lclass);
       }
}
```

注意事項：

- Label コンポーネントがフォーカスを得ることはありません。

ImageLabel

ImageLabel クラスは Panel に画像を表示するためのコンポーネントです。ImageLabel コンポーネントでは、静的なイメージ（静止画像）のみ表示可能です。イメージが設定されていない ImageLabel は、そのコンポーネントの背景色で塗りつぶされます。Java GUI の標準 Swing クラスでは Label コンポーネントにイメージを設定してアイコンを作成可能ですが、 i アプリ実行環境では別個のコンポーネントが定義されています。次のコードは、ImageLabel コンポーネントの例です。

例: ImageLabel コンポーネント

```
import com.nttdocomo.ui.*;
import com.nttdocomo.io.*;

public class ImageLabelDemo extends UIDemoPanel {
       ImageLabelDemo() {
              ImageLabel l1;

              // メディアイメージを取得する。
              MediaImage mi =
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
                MediaManager.getImage("resource:///UIDemo/img/cup.gif");
            try {
                mi.use();
            } catch (ConnectionException ce) {
                // リソース入出力に関する例外処理を記述
            }
            Image im = mi.getImage();

            // イメージオブジェクトをもつイメージラベルを作成する。
            l1 = new ImageLabel(im);
            // イメージラベルをパネルに追加する。
            add(l1);
    }
    …
}
```

注意事項：

- `ImageLabel` コンポーネントがフォーカスを得ることはありません。
- DoJa-2.0 プロファイルでは、`Image` から取り出した `Graphics` オブジェクトを使用してイメージへの描画を行うことができます。この機能を使用して、すでに画面に表示されている `ImageLabel` のイメージに描画を行った場合、メーカーによってはその内容が即時には表示に反映されない場合があります。このような場合、`ImageLabel.setImage()`メソッドによりイメージを再設定するなどによりコンポーネント全体を再描画することで描画内容が画面に反映されます。

Button

`Button` クラスはボタンを表現するコンポーネントで、ユーザーの操作を受け付けるためのインタフェース `Interactable` を実装しています。`Button` コンポーネントには、テキストを設定することができます。このコンポーネントは有効または無効な状態に設定できます。次のコードは、`Button` コンポーネントの例です。

例: `Button` コンポーネント

```
import com.nttdocomo.ui.*;

public class ButtonDemo extends UIDemoPanel implements ComponentListener {
    Button b1, b2, b3, b4;
    ButtonDemo() {
            // 4 つのボタンを作成し、それをパネルに追加する。
            b1 = new Button("B4 enabled");
            b2 = new Button("B4 disabled");
            b3 = new Button("ChangeB4");
            b4= new Button("B4");

            this.add(b1);
            this.add(b2);
            this.add(b3);
            this.add(b4);
            …
      }

      …
}
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

AnchorButton

AnchorButton クラスはHTMLのアンカー風ボタン(下線付きテキストとして表示されるボタン)を表現するコンポーネントで、ユーザーの操作を受け付けるためのインタフェース Interactable を実装しています。AnchorButton コンポーネントには、テキストおよびその見出しとなるイメージを設定することができます。コンポーネントの幅より長いテキストが指定された場合、テキストは自動的に折り返され複数行に渡って表示されます。

AnchorButton の縦方向のサイズは、画面の縦方向のサイズに制限されることはありません。ただし横方向のサイズは画面の横方向のサイズに制限されます。

このコンポーネントは有効または無効な状態に設定できます。無効化された AnchorButton のテキストは下線付きテキストではなく通常のテキストのような Look&Feel で表示されます。このため無効化された AnchorButton は、Panel においてユーザーに長い文章などを提示するためのコンポーネントとしても使用することができます。

次のコードは、AnchorButton コンポーネントの例です。

例: AnchorButton コンポーネント

```
import com.nttdocomo.ui.*;

public class AnchorButtonDemo extends UIDemoPanel implements ComponentListener {
       AnchorButton anchorButton1, anchorButton2;
       AnchorButtonDemo() {
                 // 操作可能なアンカーボタンを作成する。
                 anchorButton1 = new AnchorButton("次画面へジャンプ");
                 this.add(anchorButton1);
                 // 操作不能なアンカーボタンを作成し、ユーザーに文章を提示するために使用する。
                 anchorButton2 = new AnchorButton(
                                  "操作不能に設定されたアンカーボタンは、ユーザーに" +
                                  "長いテキストを提示するために使用することもできます。");
                 anchorButton2.setEnable(false);
                 this.add(anchorButton2);
                 …
         }

         …
}
```

注意事項：

- DoJa-2.0 プロファイルでは、Image から取り出した Graphics オブジェクトを使用してイメージへの描画を行うことができます。この機能を使用して、すでに画面に表示されている AnchorButton のイメージに描画を行った場合、メーカーによってはその内容が即時には表示に反映されない場合があります。このような場合、AnchorButton.setImage()メソッドによりイメージを再設定するなどによりコンポーネント全体を再描画することで描画内容が画面に反映されます。

ImageButton

ImageButton クラスはイメージを貼り付けることのできるボタンコンポーネントで、ユーザーの操作を受け付けるためのインタフェース Interactable を実装しています。有効なイメージが設定されていない ImageButton は、そのコンポーネントの背景色で塗りつぶされています。

このコンポーネントは有効または無効な状態に設定できます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

次のコードは、ImageButton コンポーネントの例です。

例: ImageButton コンポーネント

```
import com.nttdocomo.ui.*;
import com.nttdocomo.io.*;

public class ImageButtonDemo extends UIDemoPanel implements ComponentListener {
       MediaImage mImage;
       Image image;
       ImageButton imageButton;
       ImageButtonDemo() {
                 // イメージボタンを作成し、それをパネルに追加する。
                 try {
                       mImage = MediaManager.getImage("resource:///label.gif");
                       mImage.use();
                       image = mImage.getImage();
                 } catch (ConnectionException ce) {
                       …
                 }
                 imageButton = new ImageButton(image);
                 this.add(imageButton);
                 …
         }

         …
}
```

注意事項：

- DoJa-2.0 プロファイルでは、Image から取り出した Graphics オブジェクトを使用してイメージへの描画を行うことができます。この機能を使用して、すでに画面に表示されている ImageButton のイメージに描画を行った場合、メーカーによってはその内容が即時には表示に反映されない場合があります。このような場合、ImageButton.setImage()メソッドによりイメージを再設定するなどによりコンポーネント全体を再描画することで描画内容が画面に反映されます。

ListBox

ListBox クラスは各種の選択リストを表現するためのコンポーネントで、ユーザーの操作を受け付けるためのインタフェース Interactable を実装しています。ListBox コンポーネントは矩形の枠をもっており、その中で項目をスクロール表示することが可能です。ListBox に表示する項目全体が画面に入りきらないと、上下のスクロールを示すインジケータが表示されます。ListBox には次のような種類があります。

- 単一選択リスト
- 複数選択リスト
- ラジオボタン
- チェックボックス
- 番号付きリスト
- オプションメニュー（ポップアップ）

枠の内部に含まれる項目の表示と選択方法は、使用する ListBox のタイプによって異なります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

タイプ	選択	
	単一選択	複数選択
リスト	単一選択リスト（SINGLE_SELECT）	複数選択リスト（MULTIPLE_SELECT）
番号付きリスト	番号付きリスト（NUMBERED_LIST）	なし
チェックボックス	なし	チェックボックスリスト（CHECK_BOX）
ラジオボタン	ラジオボタンリスト（RADIO_BUTTON）	なし
ポップアップ	オプションメニュー（CHOICE）	なし

このコンポーネントは有効または無効な状態に設定できます。次のコードは、ListBox コンポーネントの例です。

例: ListBox コンポーネント

```
package uidemo;
import com.nttdocomo.ui.*;

/**
 * リストボックスウィジェットの機能を示すクラス。
 * ここでは、リストタイプとしてラジオボタン、番号付きリスト、チェックボックス、
 * オプションメニュー、単一選択リスト、複数選択リストを示す。
*/
public class ListBoxDemo extends UIDemoPanel implements SoftKeyListener {
       ListBox l1, l2,l3,l4,l5,l6;
       int count = 0;
       int newItemCnt=0;
       ListBoxDemo() {
              Label radioLabel, checkLabel, numberLabel;
              Label choiceLabel, singleLabel, multipleLabel;

              radioLabel = new Label("Radio Button List");
              // RADIO_BUTTON タイプのリストボックスを作成し、いくつかの項目を追加する。
              l1 = new ListBox(ListBox.RADIO_BUTTON);
              l1.append("radio1");
              l1.append("long radio label");
              l1.append("radio3");
              l1.append("radio4");

              checkLabel = new Label("Checkbox List");
              // CHECK_BOX タイプのリストボックスを作成し、いくつかの項目を追加する。
              l2 = new ListBox(ListBox.CHECK_BOX);
              l2.append("checkbox1");
              l2.append("checkbox2");
              l2.append("checkbox3");
              l2.append("checkbox4");
              l2.append("checkbox5");

              numberLabel = new Label("Number List");
              // NUMBERED_LIST タイプのリストボックスを作成し、いくつかの項目を追加する。
              l3 = new ListBox(ListBox.NUMBERED_LIST);
              l3.append("number1");
              l3.append("number2");
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
            l3.append("number3");
            l3.append("number4");
            l3.append("number5");

            singleLabel = new Label("Single List");
            // SINGLE_SELECT タイプのリストボックスを作成し、いくつかの項目を追加する。
            l4 = new ListBox(ListBox.SINGLE_SELECT);
            l4.append("single1");
            l4.append("single2");
            l4.append("single3");
            l4.append("single4");
            l4.append("single5");

            multipleLabel = new Label("Multiple List");
            // MULTIPLE_SELECT タイプのリストボックスを作成し、いくつかの項目を追加する
            l5 = new ListBox(ListBox.MULTIPLE_SELECT);
            l5.append("multiple1");
            l5.append("multiple2");
            l5.append("multiple3");
            l5.append("multiple4");
            l5.append("multiple5");

            choiceLabel = new Label("Choice List");
            // CHOICE タイプのリストボックスを作成し、いくつかの項目を追加する。
            l6 = new ListBox(ListBox.CHOICE);
            l6.append("choice1");
            l6.append("choice2");
            l6.append("choice3");
            l6.append("choice4");
            l6.append("choice5");

            this.add(radioLabel);
            this.add(l1);
            this.add(checkLabel);
            this.add(l2);
            this.add(numberLabel);
            this.add(l3);
            this.add(singleLabel);
            this.add(l4);
            this.add(multipleLabel);
            this.add(l5);
            this.add(choiceLabel);
            this.add(l6);

            // このパネルに softkeylistener を設定する。
            this.setSoftLabel(Frame.SOFT_KEY_2, "List");
            setSoftKeyListener(this);
        }
        …
}
```

注意事項：

- 番号付きリストは単一選択リストの特別な形式です。ユーザーは、番号付きリストの先頭から 9 個目までの項目を、携帯電話の数字キーを使用して選択することができます。最初の項目を選択するには数字 1 のキー、2 つめの項目を選択するには数字 2 のキーというように使用します。10 個目以後の項目は単一選択リストと同じ操作により選択します。
- オプションメニュー（ポップアップ）も単一選択リストの特別な形式です。オプションメニューにフォーカスがある状態で携帯電話の「選択」ボタンを押すと、ポップアップが表示されます。ユーザーはこのポップ

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

アップから目的の項目を選択します。なお、他の種類の ListBox と異なり、オプションメニューは初期状態では先頭の項目が選択された状態になっています。

- カレントフレームに追加済みの ListBox へのアプリケーションプログラムからの項目設定処理（setItems()や append()など）と、その ListBox に対するユーザー操作が同時に行われた場合の振る舞いはメーカーにより異なります。カレントフレーム上にある ListBox に、アプリケーションプログラムから項目の設定を行う際は、setEnabled()メソッドによりユーザー操作を無効化するようにしてください。

TextBox

TextBox クラスは入力フィールドを表現するコンポーネントで、ユーザーの操作を受け付けるためのインタフェース Interactable を実装しています。TextBox コンポーネントでは、単一行または複数行のテキストが表示されます。テキスト入力が可能なコンポーネントは TextBox だけです。アプリケーションプログラムで TextBox の編集モードを編集可能に設定すると、ユーザーが IME（Input Method Editor）を起動して、テキストを入力したり変更したりすることが可能になります。編集モードを編集不可に設定すると、TextBox は読み取り専用になります。

TextBox のコンストラクタには、表示モードパラメータとして DISPLAY_PASSWORD または DISPLAY_ANY を指定します。入力テキストを非表示とする必要のある i アプリでは、TextBox コンポーネントを DISPLAY_PASSWORD モードで使用すると、すべての文字が"*"で表示されます。表示モードパラメータを DISPLAY_ANY に設定すると、TextBox コンポーネントは入力テキストをそのまま表示します。

初期入力モード（かな漢字入力、アルファベット入力などの入力モード）を変更するには、setInputMode()メソッドに初期入力モードパラメータとして KANA（かな漢字入力モード）、ALPHA（アルファベット入力モード）、または NUMBER（数字入力モード）を指定します。表示モードが DISPLAY_ANY の TextBox では、setInputMode()メソッドを呼び出していない状態では初期入力モードは KANA になっています。

このコンポーネントは、有効または無効な状態に設定することができます。次のコードは、TextBox コンポーネントの例です。

例: TextBox コンポーネント

```
import com.nttdocomo.ui.*;

public class TextBoxDemo extends UIDemoPanel implements ComponentListener {
       TextBox t;               // 単一行テキストボックス（テキストフィールド）
       TextBox t2;              // 複数行テキストボックス（テキスト領域）
       Button b;
       TextBoxDemo() {
                 t2 = new TextBox(
                          "This is a test for textbox",20,3,TextBox.DISPLAY_ANY);
                 t = new TextBox(
                          "This is a test for textbox",20,1,TextBox.DISPLAY_ANY);
                 b = new Button("TestText");

                 this.add(b);
                 this.add(t2);
                 this.add(t);
                 …
       }
       …
}
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

注意事項：

- ユーザーが TextBox でテキストの編集を行っている状態（IME が起動している状態）では、バックグラウンドのスレッドが IME のモーダル操作に干渉しないようにアプリケーションプログラム側では注意が必要です。例えば IME が起動されているときにバックグラウンドスレッドが背後の Panel を他のフレームに切り替えようとした際の動作は、機種により異なりますので注意してください。

【DoJa-4.1】

DoJa-4.1 プロファイル以降、TextBox に設定するテキストの長さ（setText()メソッドなどで設定するテキストや IME で入力するテキストの長さ）を制限するための setInputSize()メソッドが追加されました。ただし setInputSize()メソッドのサポートはオプショナルであり、メーカーによってはサポートされない場合があります。

Ticker

Ticker クラスはスクロールする文字列を Panel に表示させるためのコンポーネントです。Ticker コンポーネントでは、テキストがティッカーテープ表示形式で表示されます。文字列は、画面を右から左にスクロールします。

次のコードは、Ticker コンポーネントの例です。

例: Ticker コンポーネント

```
package uidemo;
import com.nttdocomo.ui.*;

// ティッカーウィジェットの機能を示すクラス。
// 最初にパネルが表示されたときには、ティッカーのデフォルト文字列が動いている。
// ユーザーは、テキストボタンを押してテキストを変更できる。
// ティッカーは可視または不可視に設定可能

public class TickerDemo extends UIDemoPanel implements ComponentListener {
        Ticker ticker;         // ティッカーウィジェット
        Button textButton;     // テキストの変更

        static int    whichText   = 1;
        static String textString1 = "This is a test text 1 for the ticker panel";
        static String textString2 = "Now you see test text 2";

        TickerDemo() {
                ticker    = new Ticker(textString1);
                textButton = new Button("change text");

                this.add(ticker);
                this.add(textButton);
                this.setComponentListener(this);
        }



        // Component c でイベントが発生したことを示している。
        //
        // @param c Component。このイベントが発生したコンポーネント
        // @param type イベントタイプ
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
        // @param param イベントのパラメータ
        public void componentAction(Component c, int type, int param) {

                // コンポーネントアクションが発生した。textButton がクリックされると、
                // ティッカーの文字列が他のものに変更される。
                if (c == textButton) {
                        if (whichText==1) {
                                whichText=2; ticker.setText(textString2);
                        } else if (whichText==2) {
                                whichText=1; ticker.setText(textString1);
                        }
                }
        }
}
```

注意事項：

- コンポーネント内のテキストのスクロール動作をアプリケーションプログラムから制御することはできません。
- Ticker コンポーネントがフォーカスを得ることはありません。
- テキストのスクロール速度はメーカーにより異なります。

VisualPresenter

VisualPresenter クラスは動画像を再生するためのコンポーネントで、MediaPresenter を実装しています。VisualPresenter コンポーネントは、画面に表示されるメディアデータ（GIF アニメーションや FOMA の i モーション形式データなど）の再生を行う高レベル API のコンポーネントです。このコンポーネントでは、メディアデータの再生に関するイベント処理のために MediaListener を使用することができます。イベント処理の詳細は、4.3 項を参照してください。メディアデータを再生するコード例は、4.4 項で説明します。

注意事項：

- VisualPresenter はコンポーネントであり、高レベル API でのみ使用することができます。
- VisualPresenter コンポーネントがフォーカスを得ることはありません。

4.1.4 コンポーネントの配置

開発者は、コンポーネントの配置をレイアウトマネージャに任せるか、レイアウトマネージャを無効にして Component クラスの setLocation()メソッドでコンポーネントごとに配置位置を指定するかを選択することができます。

レイアウトマネージャは Panel に設定して使用します。Panel にレイアウトマネージャを設定するには、Panel.setLayoutManager()メソッドを使用します。

i アプリ実行環境で用意されているレイアウトマネージャには、Panel を生成した初期状態で設定されているデフォルトレイアウトマネージャと、HTML 風の指定方法によりレイアウトを行うための HTML レイアウトマネージャの 2 つがあります。

デフォルトレイアウトマネージャの基本レイアウト方針は、シンプルなフローレイアウトです。パネルに追加されたコンポーネントは画面の左から右に向けて順次横並びに配置されていき、画面の

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.
横幅で収まりきらなくなった時点で次のコンポーネント配置が可能な位置（1 段下の列）へと配置されます。

また HTML レイアウトマネージャでは、コンポーネントを Panel に追加していく過程で改行（BR タグに相当）、段落（P タグに相当）、アライメント（CENTER や DIV タグなどに相当）を指定することで、開発者がより明示的にレイアウトを指示することができます。

どのメーカーの携帯電話にも適用できる汎用的な i アプリを作成するためには、コンポーネントのレイアウト構成をシンプルに保つことを推奨します。なお、配置可能なコンポーネントの数は、メモリ使用状況や各機種固有の制限などの影響を受けます。

注意事項：

- LayoutManager インタフェースはメーカーがレイアウトマネージャを実装するために用意されているものであり、開発者が本インタフェースを利用して独自のレイアウトマネージャを作成することはできません。Panel オブジェクトの setLayoutManager()メソッドを呼び出すことで、その Panel が使用するレイアウトマネージャを変更することができます。
- レイアウトマネージャを無効にするには、Panel オブジェクトに対して、パラメータに null を指定して setLayoutManager()メソッドを呼び出します。レイアウトマネージャを無効にした場合、アプリケーションプログラムは各コンポーネントの setLocation()メソッドを使用して、コンポーネント毎にそれを配置する位置を指定します。なお、レイアウトマネージャを無効にした場合、機種によってはスクロール機能に制限が発生します。
- 1 つの HTML レイアウトマネージャオブジェクトは、2 つ以上の Panel に同時に設定することはできません。1 つの i アプリの中に HTML レイアウトを適用したい Panel が複数ある場合には、それぞれの Panel に対し別々の HTML レイアウトマネージャオブジェクトを割り当てるようにしてください。
- HTML レイアウトマネージャは、setLayoutManager()メソッドで Panel に設定される度に初期状態（アライメントが左寄せで改行、段落指定のない状態）に戻ります。

次のコードは、HTML レイアウトマネージャを使用してコンポーネントの配置を行う例を示したものです。

例: HTML レイアウトマネージャによるコンポーネントの配置

```
import com.nttdocomo.ui.*;

public class HTMLLayoutDemo extends UIDemoPanel {
         Label           label1, label2, label3, label4;
         HTMLLayout      lm;

         HTMLLayoutDemo() {
                 label1 = new Label("Label 1");
                 label2 = new Label("Label 2");
                 label3 = new Label("CENTER");
                 label4 = new Label("RIGHT");

                 // HTML レイアウトマネージャを生成し、Panel に設定する。
                 lm = new HTMLLayout();
                 setLayoutManager(lm);

                 // レイアウトを指定しつつ Panel に Label を追加する。
                 add(label1);
                 //以降のコンポーネントは次行に配置する（BR タグ相当）。
                 lm.br();
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
                add(label2);
                // 以降のコンポーネントは次行に配置する（BR タグ相当）。
                lm.br();

                // begin()から対応する end()までの間に追加されたコンポーネントを
                // 中央寄せで表示する。
                lm.begin(HTMLLayout.CENTER);
                add(label3);
                lm.end();

                // begin()から対応する end()までの間に追加されたコンポーネントを
                // 右寄せで表示する。
                lm.begin(HTMLLayout.RIGHT);
                add(label4);
                lm.end();
        }
}
```

また次のコードは、レイアウトマネージャを無効化して、アプリケーションプログラムが独自にコンポーネントの配置を行う例を示したものです。

例: `setLocation()`によるコンポーネントの配置

```
import com.nttdocomo.ui.*;

public class NullLayoutDemo extends UIDemoPanel {
        Button button1, button2;
        Button offScreen;
        TextBox text;
        Label title, button1Label, button2Label;

        NullLayoutDemo() {
                // ここですべてのコンポーネントを使用する。
                title           = new Label("Null Layout ");
                button1         = new Button("Button1");
                button2         = new Button("Button2");
                button1Label    = new Label("ONE:");
                button2Label    = new Label("TWO:");
                text            = new TextBox("A simple sentence in a textbox", 40,
                                                2, TextBox.DISPLAY_ANY);

                /*
                * 画面外に配置するコンポーネントの例としてこのボタンを作成する。
                */
                offScreen = new Button("This button is never seen");

                /*
                 * ここでレイアウトマネージャに null を設定する。したがって、個々のコンポー
                 * ネントの場所を設定する必要がある。注: 物理的な画面の座標外に配置される
                 * コンポーネントは全く表示されない。null レイアウトにより、スクロールは無効に
                 * される場合がある。
                 */
                setLayoutManager(null);

                /* 注: この場合でも、コンポーネントを追加する順序には意味がある。コンポーネントは、
                 * この順序に従ってフォーカスを与えられる。
                 */
                add(title);
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
                add(button1Label);
                add(button1);
                add(button2Label);
                add(button2);
                add(text);
                // このコンポーネントも追加する。
                add(offScreen);

                /*
                 * ここで、各コンポーネントの場所を設定する必要がある。つまり、パネルのレイアウト
                 * を独自に指定する。コンポーネントのサイズは自動的に設定することも、setSize を
                 * 直接呼び出してサイズを指定することもできる。
                 */

                title.setLocation(12, 0);
                button1Label.setLocation(10, 18);
                button1.setLocation(45, 18);
                button2Label.setLocation(10, 40);
                button2.setLocation(45, 40);
                text.setLocation(2, 60);

                /*
                 * 注: このボタンは画面外に配置される。したがって、表示もされないし、フォーカスを
                 * 得ることもない。
                 */
                offScreen.setLocation(1000, 1000);
        }
}
```

【DoJa-2.0】

HTML レイアウトマネージャ（HTMLLayout クラス）は DoJa-2.0 プロファイルにて新設されました。

4.1.5 フォーカスとスクロール

フォーカス管理とスクロール管理の詳細な振る舞いは、メーカーにより異なる場合があります。i アプリ対応携帯電話にはシンプルなデフォルトフォーカスマネージャが搭載されています。典型的なデフォルトフォーカスマネージャでは、コンポーネントが Panel に追加された順序がフォーカスを得る順序になります。i アプリが起動されて画面が表示されると、有効な状態に設定されている最初の対話可能なコンポーネントにフォーカスが設定されます。ただし、フォーカスが設定されているコンポーネントがディスプレイの領域外に配置されている状態では、そのコンポーネントを操作することはできません。

デフォルトのスクロール動作は、フォーカス管理と緊密な関係があります。これは、画面のスクロールがフォーカスの移動に伴って行われるためです。

注意事項：

- FocusManager インタフェースはメーカーがフォーカスマネージャを実装するために用意されているものであり、開発者が本インタフェースを利用して独自のフォーカスマネージャを作成することはできません。Panel オブジェクトの setFocusManager()メソッドを呼び出すことで、その Panel が使用するフォーカスマネージャを変更することができます。ただし現在までのプロファイルでは、デフォルトフォーカスマネージャ以外のフォーカスマネージャは規定されていません。

  なおレイアウトマネージャと異なり、フォーカスマネージャを無効化することはできません。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

4.1.6 高レベルイベントの処理

Panel により構成された画面では、コンポーネントによって生成されたイベントは、ComponentListener を実装したオブジェクトの componentAction()メソッドに渡されます。またソフトキーイベントは、SoftKeyListener を実装したオブジェクトの softKeyPressed()または softKeyReleased()メソッドに渡されます。このように、高レベルイベント処理にはリスナーモデルが採用されています。

ComponentListener も SoftKeyListener も Panel に登録して使用します。高レベルイベント処理におけるイベントリスナーとコード例の詳細は、4.3 項を参照してください。

4.1.7 コンポーネントのフォントサポート

DoJa-3.0 プロファイルでは Component クラスおよび Dialog クラスに setFont()メソッドが追加され、これらに表示される文字のフォントをアプリケーションプログラムから指定することができます。使用したいフォントを表す Font オブジェクトを生成して setFont()メソッドの引数に指定することで、デフォルトフォント以外のフォントを表示することができます。

なおコンポーネントにフォントを設定する場合、このメソッドは 1 つのコンポーネント上に表示される文字列全体に作用します。同一コンポーネント上に異なる複数のフォントで構成される文字列を表示することはできません。また、文字を表示する機能を持たないコンポーネントに対しては、このメソッドを呼び出しても何も行われません。

また、ダイアログにフォントを設定する場合、このメソッドはダイアログの本文テキスト全体にのみ作用します。ダイアログタイトルやダイアログ上のボタンのフォントを変更することはできません。

注意事項：

- パネルタイトルやソフトキーラベルの表示文字のフォントを変更することはできません。

4.2 低レベル API によるプログラミング

ゲームアプリケーションなどの分野では、コンポーネントを使用したユーザーインタフェースよりもグラフィックス描画機能を多用する場合があります。この節では、主にグラフィックス描画を行うための低レベル API の使い方とイベントの処理方法について説明します。

4.2.1 Canvas の使用

Canvas は、グラフィックスを描画するための画面領域です。開発者は Canvas クラスのサブクラスを作成し、paint()メソッドをオーバーライドすることによって、Canvas に描画することができます。paint()メソッドは、システムが Canvas の描画が必要と判断したときやアプリケーションプログラムが明示的に再描画を要求したとき（repaint()メソッドを呼び出したとき）にシステムからコールバックされます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

Panel と同様に、1 つの Canvas オブジェクトは 1 つの画面に相当します。したがって、1 つの画面に複数の Canvas を表示したり、Canvas と Panel を同時に 1 つの画面に表示することはできません。ただし複数の Canvas や Panel を、1 つの画面上に順次切り替えて表示することは可能です。

次に Canvas に簡単な描画を行う例を示します。

例: Canvas への描画

```
import com.nttdocomo.ui.*;
public class CanvasDemo extends Canvas {
         int x = 20;
         int y = 10;

         CanvasDemo() {
                 // ソフトキーラベル「back」を作成する。
                 setSoftLabel(Frame.SOFT_KEY_1, "Back");
         }

         /**
          * キャンバス領域に文字列と矩形を描画する。
          */
         public void paint(Graphics g) {
                 g.lock();
                 g.clearRect(0, 0, 1000, 1000);
                 g.drawString("Hello",  x, y);
                 g.drawString("Canvas", x, y+15);
                 g.drawRect(x+10, y+30, 20, 5);
                 g.unlock(true);
         }

         …
}
```

注意事項：

- repaint()メソッドはシステムに対し描画の必要があることを指示するためのものであり、1 回の repaint()メソッド呼び出しが 1 回の paint()メソッド呼び出しに対応することを保証するものではありません。このためアプリケーションプログラムから見た場合に、複数の連続した repaint()メソッド呼び出しが 1 回の paint()メソッド呼び出しにまとめられることがあります。開発者は、paint()メソッド呼び出しが 1 回にまとめられたような場合にも画面の描画が正しく行われるよう paint()メソッドを記述する必要があります。
- DoJa-3.0 プロファイルでは、カレントフレームに設定されていない Canvas に対する repaint()メソッドの呼び出しは無視されます。なお、DoJa-2.x プロファイル以前のプロファイルでは、カレントフレームに設定されていない Canvas に対して repaint()メソッドを呼び出した場合の振る舞いはメーカーにより異なります。

【DoJa-5.0】

DoJa-5.0 プロファイル以降、メーカーによっては ADF の DrawArea キーを使用することで、デフォルト（DrawArea キー未使用時）の表示領域よりも広い表示領域を使用できる場合があります。そのような機種には、Canvas.setSoftLabelVisible()メソッドを使用することで、Canvas 表示時のソフトキーラベルの表示のオン、オフを切り替えることができるものがあります。なお、setSoftLabelVisible()メソッドは Frame クラスで定義されたメソッドですが、現在のプロファイルでは Canvas 以外のフレームオブジェクト（Panel や Dialog）に対してこのメソッドを呼び出すことはできません。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

4.2.2 グラフィックスの描画

線や図形、テキストなどの描画には、グラフィックスオブジェクト（Graphics クラスのオブジェクト）を使用します。このオブジェクトには描画可能な領域、フォント、色などの情報が含まれているため、グラフィックスコンテキストとも呼ばれます。前項の例ではグラフィックスコンテキスト g を使い、clearRect()メソッドによる画面のクリアの後に drawString()メソッドを呼び出してテキストを描画し、続いて drawRect()メソッドを呼び出して矩形を描画しています。

上の例で使用している lock()メソッドと unlock()メソッドは、グラフィックス描画のダブルバッファリングを管理するために Graphics クラスに用意されているメソッドです。ダブルバッファリング中は、指示された描画はオフスクリーンバッファに対して行われ、ダブルバッファリング終了とともにオンスクリーンにコピーされます。このため、ダブルバッファリングを使用するとフリッカーが発生しません。

ダブルバッファリングは、メーカーによりサポートされない場合があります。ダブルバッファリングをサポートしない機種では、この 2 つのメソッドは何も行いません。ダブルバッファリングをサポートするデバイスでは lock()メソッドが呼び出されると（1 回または複数回）、unlock()メソッドが引数に true が指定されて呼び出されるか、引数に false が指定されて lock()メソッド呼び出しと同じ回数だけ呼び出されるまで、画面への描画は一時的に保留されます。

注意事項：

- ダブルバッファリングの実装は携帯電話によって異なります。アプリケーションプログラムの正しい動作を保証するために、ダブルバッファリングを使用する際は次のガイドラインに従うようにしてください。
  - 異なる Graphics オブジェクトの lock()メソッドと unlock()メソッドの呼び出しを混在して行わない。
  - paint()メソッドを終了する前には必ず Graphics オブジェクトのロックを解除する。

次のコードに、ロックおよびロック解除の正しい使い方を示します。

例: 正しい実装（上記ガイドラインに従っている）

```
import com.nttdocomo.ui.*;
public void paint(Graphics g) {
     Graphics g2 = g.copy();
     g.lock();
     // 何かを描画する .....
     g.unlock(false);
     g2.lock();
     // さらに描画する.....
     g2.unlock(false);
}
```

次のコードに、ロックおよびロック解除の正しくない使い方を示します。

例: 正しくない実装

```
import com.nttdocomo.ui.*;
public void paint(Graphics g) {
     Graphics g2 = g.copy();
     g.lock();
     // 何かを描画する.....
     g2.lock();              // g と g2 のロック区間が重複している
     // さらに描画する .....
     g.unlock(false);        // g2 のロック解除が行われていない
}
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- システムからコールバックされた paint()メソッドの引数で渡されるグラフィックスオブジェクトに対し、Graphics.dispose()メソッドは発行しないでください。この場合のグラフィックスオブジェクトの破棄は、必要に応じて paint()メソッドの呼び出し元である i アプリ実行環境により行われます。また、アプリケーションプログラムが Canvas オブジェクトや Image オブジェクトから getGraphics()メソッドを使用してグラフィックスオブジェクトを取得した場合、そのグラフィックスオブジェクトが不要になった時点で dispose()メソッドを呼び出すようにしてください。

- i アプリ実行環境では、repaint()メソッドの引数に矩形領域を設定することで指定された矩形領域のみの再描画を行うことができます。DoJa-1.0 プロファイルでは、描画デバイスの持つクリッピング描画能力によりこの機能をサポートしない携帯電話があることを想定していますが、DoJa-2.0 プロファイル以降に対応した携帯電話は全てこの機能をサポートします。

- i アプリ実行環境では透過 GIF がサポートされており、透過 GIF のメディアイメージから生成したイメージオブジェクトを描画すると透過部分は描画対象外となります。ただしこの効果は透過 GIF から生成されたイメージオブジェクトを描画する際のみであり、その描画結果からは透過情報は失われます。

- DoJa-2.0 プロファイルでは、i アプリの中断・再開により Canvas が再表示される際、i アプリ実行環境からその Canvas の paint()メソッドが 1 回呼び出されることが保証されます。しかしその paint()メソッド呼び出しで全画面が完全には復元されないようなケース（アプリケーションプログラムがイベントに応じて差分描画のみ行うようになっているなど）では、どの程度システムが画面を復元できるかは機種により異なります。

  なお DoJa-3.0 プロファイル以降では、i アプリの再開時にシステムが Canvas の描画内容をすべて復元できた場合は、paint()メソッドは呼び出されません。

- DoJa-2.0 プロファイルでは､低レベル文字列描画に使用されるデフォルトフォントは 12 ドットフォントです。これに対し DoJa-1.0 プロファイルでは、デフォルトフォントサイズはメーカーにより異なります。なお、いずれのプロファイルにおいても、デフォルトフォント以外にどのようなフォントをサポートするかについてはメーカーにより異なります。一般には、携帯電話の限られたメモリにフォント情報を保持する必要があるため、実際に搭載されるフォントの数は数種類に限られます。DoJa-1.0 プロファイルおよび DoJa-2.0 プロファイルでは、デフォルトフォントのサイズは Font.SIZE_MEDIUM に対応しています。

  なお DoJa-3.0 プロファイル以降でも、プロファイル間におけるアプリケーションプログラムの互換性の観点からデフォルトフォントは 12 ドットフォントとしています。ただし、携帯電話に搭載される LCD デバイスの高精細化に伴い､デフォルトフォントのサイズは DoJa-3.0 プロファイルにて新設された Font.SIZE_TINY に対応するよう変更されています｡ミディアムフォントは DoJa-3.0 プロファイル以降では 24 ドットフォントとなります（タイニー、ミディアム以外のフォントについては、メーカーによりサイズが異なる場合があります）。

  高精細化の進んだ LCD デバイス上では同一ドット数のフォントでも視覚上のサイズは小さくなるため､画面設計においては状況に応じてより大きなフォントを使用することも検討してください。

  【DoJa-5.0】

  DoJa-5.0 プロファイルにて、既定のフォントサイズ（SIZE_TINY、SIZE_SMALL、SIZE_MEDIUM、SIZE_LARGE）以外のサイズのフォントをサポートする機種が現れる可能性があります。そのような機種では、Font.getSupportedFontSizes()メソッドを使用してその機種でサポートしているフォントサイズのリストを取得することができます（既定のフォントサイズのみサポートする機種ではこのメソッドを使用して情報を得ることはできません）。

  既定のフォントサイズ以外のフォントについては、DoJa-5.0 プロファイルで新設された Font.getFont(int,int)メソッドを使用して、フォントサイズを数値で直接指定することで、そのサイズに対応するフォントオブジェクトを取得することができます。

- DoJa-2.0 以降の各プロファイルでは、グラフィックス描画に関して i アプリオプションおよび i アプリ拡張のカテゴリに属する機能（スプライトや 3D グラフィックスなど）が追加で規定されています。これらの詳細については、「i アプリコンテンツ開発ガイド　i アプリオプション・i アプリ拡張編」および「i アプリコンテンツ開発ガイド　オプション・拡張 API リファレンス編」を参照してください。なお、それらの中には、プロファイルのバージョンアップの過程で i アプリ基本 API に取り入れられたものもあります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- DoJa-3.0 プロファイル以降では、Canvas に対応付けられているグラフィックスオブジェクトへの描画はその Canvas がカレントフレームに設定されている場合のみ行われます。対応付けられている Canvas がカレントフレームに設定されていない場合、そのグラフィックスオブジェクトに対する描画メソッドの呼び出しは無視されます。

4.2.3 Image オブジェクトへの描画

DoJa-2.0 プロファイル以降では、Image オブジェクトへの描画をサポートします。

Image オブジェクトからの Graphics オブジェクトの取得は、Image.getGraphics()メソッドを呼び出すことにより行います。Image オブジェクトから取得した Graphics オブジェクトへの描画指示は、元となる Image オブジェクトの表示内容に反映されます。このようにして描画された Image オブジェクトを ImageLabel や ImageButton に設定することで、アプリケーションプログラムが行った描画処理の結果を Panel 使用時にも表示することができます。

Image オブジェクトから Image.getGraphics()メソッドにより Graphics オブジェクトを取得するには、その Image オブジェクトが Image.createImage()メソッドにより生成された新規の Image オブジェクトでなければなりません。MediaImage.getImage()メソッドで取得した Image オブジェクトに対して getGraphics()メソッドを呼び出した場合、UnsupportedOperationException がスローされます。

なお、Image.createImage()で生成されたイメージは、初期状態ではその機種における Canvas のデフォルト背景色と同じ色で塗りつぶされています。ただし Image 自身には背景色の概念はありません。Image から取得した Graphics に対して描画内容のクリアを指示するには、Graphics.clearRect()メソッドではなく、適切な色の設定を行った上で Graphics.fillRect()メソッドを呼び出します。

次のコードは、アプリケーションプログラムからの指示により描画された Image オブジェクトを ImageLabel で使用する例です。

例: Image オブジェクトへの描画

```
import com.nttdocomo.ui.*;

public class ImageDrawDemo extends UIDemoPanel {
        ImageLabel      iLabel;
        Image           img;
        Graphics        g;

        ImageDrawDemo() {
                // Image オブジェクトを新規に生成し、Graphics オブジェクトを取得する。
                int width  = 100;
                int height = 50;
                img = Image.createImage(width, height);
                g = img.getGraphics();

                // 取得した Graphics オブジェクトに描画する。
                g.lock();
                g.drawRect(0, 0, width-1, height-1);
                g.drawString("Graphics による", 5, 16);
                g.drawString("Image への", 5, 32);
                g.drawString("描画のサンプル", 5, 48);
                g.unlock(true);

                // Image オブジェクト ImageLabel に設定し、Panel 上に表示する。
                iLabel = new ImageLabel(img);
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
                add(iLabel);
        }
}
```

注意事項：

- `Image.createImage()`メソッドでは、少なくともその機種におけるネイティブ描画領域の長辺を 1 辺とする正方形のサイズまでの `Image` を生成できることが全機種で保証されます。これより大きなサイズの `Image` を生成可能かどうかはメーカーにより異なりますので注意してください。

  【DoJa-2.0】【DoJa-3.0】

  DoJa-2.x および DoJa-3.x プロファイルでは、`Image.createImage()`メソッドでは少なくともその機種における i アプリの描画領域サイズまでの `Image` を生成できることが全機種で保証されます。これより大きなサイズの `Image` を生成可能かどうかはメーカーにより異なります。

4.2.4 カラーサポート

`Graphics` オブジェクトには、オブジェクトの色情報が含まれています。`Graphics` クラスには次のメソッドが含まれており、それぞれが色を表す整数値を返します。

`getColorOfName(int name)`	: 色名（`Graphics` 定数）に対応する整数値を返します。
`getColorOfRGB(int r,int g,int b)`	: RGB 値に対応する整数値を返します。
`getColorOfRGB(int r,int g,int b,int a)`	: RGB 値およびアルファ値に対応する整数値を返します。

アルファ値を含む色指定を使用することで、半透明の描画（描画先のピクセル色と描画元のピクセル色のブレンド）が可能となります。アルファ値 a には 0～255 までの値を指定し、255 が完全に不透明であることを意味します。ただし、色のアルファ値要素が有効となる描画メソッドは以下のように限定されています。

DoJa-4.x プロファイル	: `fillRect()`のみ
DoJa-5.x プロファイル以降	: `fillRect()`、`fillArc()`、`fillPolygon()`のみ

他の描画メソッドや高レベル UI の色指定メソッドではアルファ値要素の設定内容は無効となり、アルファ値として 255 が指定されたものとして処理されます。

色の管理方法はメーカーに依存するため、i アプリ実行環境内部で色に割り当てられる整数値（これらのメソッドの返り値）はメーカーによって異なる可能性があります。開発者はこの整数値にいかなる仮定も置くことはできません。たとえば、あるメーカーの白黒画面では黒の値が 1 であったとしても、別のメーカーのカラー画面では黒の値は 255 かもしれません。アプリケーションプログラムでは上記のメソッドを使用して、機種間で共通の色の定義（名前または RGB 値）を各機種固有の色の定義に変換する必要があります。

以下に、正しくないコードの例を示します。

例: グラフィックスキャンバス

```
import com.nttdocomo.ui.*;
public class MyCanvas extends Canvas {
       …
       public void paint(Graphics g) {
              …
              if(g.getColorOfName(Graphics.BLACK) == 1) {//何かをする。
                  // あるメーカーの機種では結果が真になり、ここのコードが実行されたとしても、
                  // 他のメーカーの機種では偽になり、コードはスキップされるかもしれない。
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
            }
            …
        }
}
```

【DoJa-2.0】

iアプリではカラー対応絵文字（iモードサービス向けにNTTドコモが定義した外字セット）を使用することができます。カラー対応絵文字では、1つ1つの絵文字に対しデフォルトの表示色が割り当てられています。

DoJa-2.0プロファイル以降の`Graphics`クラスには、`Graphics`オブジェクトに描画色が明示的に指定されている場合に、絵文字をデフォルトの表示色で描画するか`Graphics`オブジェクトに指定された描画色で描画するかを選択するためのメソッド`setPictoColorEnabled()`が追加されています。

DoJa-1.0プロファイルでは、`Graphics`オブジェクトに描画色が明示的に指定された状態で絵文字を描画した際に、その絵文字をデフォルトの表示色で描画するか`Graphics`オブジェクトに指定された描画色で描画するかはメーカーにより異なります。

【DoJa-4.0】

アルファ値を使用した色指定（`getColorOfRGB(int,int,int,int)`メソッド）は、DoJa-4.0プロファイルにて新設されました。

4.2.5 低レベルイベントの処理

`Canvas`使用時に適用される低レベルイベント処理の方法は、`Panel`使用時における高レベルイベント処理の場合とは異なります。`Canvas`オブジェクトは低レベルイベントだけを処理します。開発者は`Canvas`クラスの継承クラスを作成する際に、`processEvent()`メソッドをオーバーライドすることができます。このメソッドには低レベルイベント発生時に、イベント種別とそれに伴うパラメータが渡されます。次の表は、`Canvas`に通知される有効なイベントタイプを示したものです。これらのイベントタイプは`Display`クラスに定義されています。

イベントタイプ	イベントタイプ値	パラメータ
`KEY_PRESSED_EVENT`	0	押されたキーの値
`KEY_RELEASED_EVENT`	1	離されたキーの値
`RESUME_VM_EVENT`	4	使用されない
`UPDATE_VM_EVENT`	6	使用されない
`TIMER_EXPIRED_EVENT`	7	タイマーイベントの発生した`ShortTimer`のタイマーID

次のコードは、低レベルイベントを処理する`processEvent()`メソッドの実装例です。

例: イベント処理の実装

```
package spaceinv;

import com.nttdocomo.ui.*;
import java.util.Random;
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
/**
 * このクラスは、特定のハイスコアサーバーに設定されている歴代の殿堂入りハイスコアを表示するときに
 * 使用する。スコアとその名前がアニメーション風に表示される。このクラスは、ユーザー入力を受け取るか、
 * このクラスが維持するアイドルタイマーが切れると、次の画面に移る。
 */
public class HighScoreScreen extends Canvas implements Runnable {
        private boolean done;
        private ShortTimer timer;

        /** このクラスの新しいインスタンスを作成する。 */
        public HighScoreScreen() {
                …
        }
        /** このインスタンスを初期状態にリセットする。 */
        public void reset(String[] list) {
                …
        }
        /** Runnableインタフェースに対して必要。*/
        public void run() {
                …
        }
        /** この画面を表示する。  */
        public void paint(Graphics g) {
                …
        }

        /** イベント処理を行うためにCanvasのprocessEvent()をオーバーライドする。 */
        public void processEvent(int type, int param) {
                if (type == Display.KEY_PRESSED_EVENT ||
                        type == Display.TIMER_EXPIRED_EVENT) {
                        done = true;
                        timer.stop();
                        timer.dispose();
                        SpaceInvaders.WELCOME.reset();
                        Display.setCurrent(SpaceInvaders.WELCOME);
                }
        }
}
```

【DoJa-2.0】

DoJa-2.0 プロファイル以降では、 i アプリ再開時の低レベルイベント発生の流れを以下のように規定しています。

- i アプリ再開時は、 i アプリの実行が再開されたことを示す RESUME_VM_EVENT の通知を行った後、 i アプリメインクラスの resume()メソッドが呼び出されます。
- i アプリ再開時に、描画データの破壊を意味する UPDATE_VM_EVENT が発生する場合は、 RESUME_VM_EVENT が発生した後で UPDATE_VM_EVENT が発生します。

【DoJa-5.0】

DoJa-5.0 プロファイル以降では本項で解説した基本的な低レベルイベントの他に、メーカー独自のハードウェア機構に関連した、オプショナルのイベントを定義するクラスとして com.nttdocomo.opt.ui.Display2 クラスが用意されています。

用意されているイベントの詳細については、i アプリコンテンツ開発ガイド　オプション・拡張 API リファレンス編の Display2 クラスの項を参照してください。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

注意事項：

- processEvent()メソッドの呼び出しは、ユーザーの操作やタイマーなど非同期なイベント発生を契機として行われます。processEvent()メソッドの呼び出し時に、それがどのようにスレッドに割り当てられるかはメーカーにより異なりますので注意してください。例えばあるメーカーの機種では、同時に発生した複数のイベントに対して 1 つのスレッドで処理を行うかもしれません。processEvent()メソッド内で行われる処理は極力短時間で完結するようにしてください。

4.2.6 Canvas における IME の利用

Panel を用いた画面では TextBox のユーザー操作により IME を起動することができ、ユーザーが入力した文字列を i アプリに与えることができます。これに対し Canvas を用いた画面では、DoJa-3.0 プロファイルにて新設された API を使用してアプリケーションプログラムから IME を起動することができます。これらの API を利用することで、Canvas 使用時も Panel 使用時と同様にユーザーが入力した文字列を i アプリに与えることが可能となります。

アプリケーションプログラムが Canvas.imeOn()メソッドを呼び出すと IME が起動されます。このメソッドは非ブロック型であり、IME の起動が完了するとアプリケーションプログラムに復帰します。

ユーザーが IME の操作を完了（確定またはキャンセル）すると、アプリケーションプログラムは IME イベントの通知を受けます。開発者は Canvas.processIMEEvent()メソッドをオーバーライドすることにより、IME の処理結果を受け取ることができます。

次のコードは、IME を利用する Canvas の例です。

例: Canvas からの IME の利用

```
import com.nttdocomo.ui.*;

class MainCanvas extends Canvas {
      String inputString = "";

      public void processEvent(int type, int param) {
            // ソフトキー1 のリリースで IME を起動する。
            if(type==Display.KEY_RELEASED_EVENT&&param==Display.KEY_SOFT1)
                  imeOn("initial string",TextBox.DISPLAY_ANY, TextBox.KANA);
            // IME の起動が完了するとアプリケーションプログラムに制御が戻る。
      }

      public void paint(Graphics g) {
            g.lock();
            g.clearRect(0,0,200,200);
            g.drawString("InputString: '" + inputString + "'", 0, 20);
            g.unlock(true);
      }

      public void processIMEEvent(int type, String text) {
            // IME のユーザー操作が完了するとこのメソッドが呼び出される。
            // IME イベントのパラメータには、イベントタイプ（確定またはキャンセル）と
            // 入力文字列がある。
            if (type==Canvas.IME_CANCELED) inputString = "[CANCELED]";
            else if (type==Canvas.IME_COMMITTED) inputString = text;
            repaint();
      }
}
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

注意事項：

- IME が起動している状態では、バックグラウンドのスレッドが IME のモーダル操作に干渉しないようにアプリケーションプログラム側では注意が必要です。例えば IME が起動されているときにバックグラウンドスレッドが背後の `Canvas` を他のフレームに切り替えようとした際の動作は、機種により異なりますので注意してください。
- `imeOn()`メソッドの呼び出しは、その `Canvas` がカレントフレームに設定されていない状態や `Dialog` が表示されている状態では無視されます。
- `processIMEEvent()`メソッドの呼び出しは、ユーザーの IME 操作完了による非同期なイベント発生を契機として行われます。`processIMEEvent()`メソッドの呼び出し時に、それがどのようにスレッドに割り当てられるかはメーカーにより異なりますので注意してください。例えばあるメーカーの機種では、IME イベントとその他のイベントに対して 1 つのスレッドで処理を行うかもしれません。`processIMEEvent()`メソッド内で行われる処理は極力短時間で完結するようにしてください。
- `imeOn()`メソッドは、非活性化状態の待ち受けアプリケーションから呼び出すことはできません。

【DoJa-4.1】

DoJa-4.1 プロファイル以降、`imeOn()`メソッドのバリエーションに、IME で入力可能なテキストの長さを制限するための引数を持つものが追加されました。ただしこのメソッドのサポートはオプショナルであり、メーカーによってはサポートされない場合があります。

4.3 イベントリスナー

イベントリスナーとは、イベントを通知するオブジェクト（`Panel` や `MediaPresenter` など）に登録され、特定タイプのイベントを待ち受けるオブジェクトです。イベントが発生すると、オブジェクトは適切なイベントタイプを構築し、そのイベントを対応するイベントリスナーに通知します。イベントリスナーには、次の 5 つのタイプがあります。

- `ComponentListener`
- `SoftKeyListener`
- `KeyListener`
- `MediaListener`
- `TimerListener`

ダイアログで使用されるイベントリスナーはありません。ダイアログを画面に表示するには、`Dialog.show()`メソッドを使用します。`Dialog.show()`メソッドは、ユーザーがダイアログとの対話を終了したときにアプリケーションプログラム側に制御を戻します。`Dialog.show()`メソッドの戻り値は、ユーザーがダイアログ内のどのボタンを押したかを表しています。

注意事項：

- リスナーメソッドの呼び出しは、ユーザーの操作やタイマーなど非同期なイベント発生を契機として行われます。リスナーメソッドの呼び出し時に、それがどのようにスレッドに割り当てられるかはメーカーにより異なりますので注意してください。例えばあるメーカーの機種では、同時に発生した複数のイベントに対して 1 つのスレッドで処理を行うかもしれません。リスナーメソッド内で行われる処理は極力短時間で完結するようにしてください。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

4.3.1 ComponentListener

ComponentListener インタフェースは、ディスプレイに表示されている Panel 上の UI コンポーネントに対して起動されたイベントを待ち受けるときに使用します。このインタフェースのメソッドは componentAction()だけです。コンポーネントリスナーの定義と登録は、以下のように行います。

- リスナーとして使用するクラスに、ComponentListener インタフェースおよび componentAction()メソッドを実装します。コンポーネントイベントが発生すると、componentAction()メソッドが呼び出され、イベントタイプ、イベントが発生したコンポーネント、およびその他の必要なパラメータがイベントリスナーに渡されます。
- Panel オブジェクトに対して setComponentListener()メソッドを呼び出し、上記で実装したリスナークラスのインスタンスを登録します。

ComponentListener には、次の 3 つのイベントタイプが定義されています。

- BUTTON_PRESSED:
  Button が押されたときに発生するイベント。イベントパラメータは使用しません。
- SELECTION_CHANGED:
  ListBox の選択内容が確定されたときに発生するイベント。イベントパラメータとして、選択または状態変更された項目のインデックスが渡されます(複数選択リストでは選択された項目の中で最も小さいもののインデックスが渡されます)。
- TEXT_CHANGED:
  TextBox の文字入力が終了したときに発生するイベント。イベントパラメータは使用しません。

次に、ComponentListener を内部クラスとして定義し、使用する例を示します。

例: ComponentListener を実装する内部クラス

```
package uidemo;
import com.nttdocomo.ui.*;

/**
 * ラベルウィジェットとそれに対して呼び出された setSize メソッドの機能を示すクラス。
 * このクラスは 1 つのボタンと 2 つのラベルウィジェットを作成する。
 * ラベルウィジェットの可視/不可視の機能を示すため、ボタンウィジェットを使用する。
 */
public class LabelDemo extends UIDemoPanel {
       Label lbl1, lbl2;
       Button btn1;
       int count = 0;

       LabelDemo() {
              lbl1 = new Label("Label1");
              lbl2 = new Label("This is a Label2");
              btn1= new Button("Button1");
              lbl1.setSize(25,15);   // ラベルのサイズを設定する。

              this.add(btn1);
              this.add(lbl1);
              this.add(lbl2);
              ListenerClass lclass = new ListenerClass();
              this.setComponentListener(lclass);
       }

       /**
         * ボタンイベントを処理する内部クラス
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
        */
       class ListenerClass implements ComponentListener {
       /**
         * イベントが Component c で発生したことを示す。
         *
         * @param c このイベントが発生した Component
         * @param type イベントのタイプ
         * @param param イベントのパラメータ
         */
              public void componentAction(Component c, int type, int param) {
              /*
              * コンポーネントアクションが行われた。コンポーネントがボタンの場合、カウントが 0 なら、
              * このパネルの最初のラベルが不可視に、カウントが 1 なら、同じラベルを可視にそれぞれ
              * 設定する。
              */
                     if (c == btn1) {
                            if (count == 0) {
                                   lbl1.setVisible(false);
                                   count++;
                            }
                            else {
                                   lbl1.setVisible(true);
                                   count=0;
                            }
                     }
              }
       }
}
```

注意事項：

- TextBox および ListBox は、それらのクラスが備えるメソッドで TextBox に設定されているテキストや ListBox の選択状態を変更することができます。このように、ユーザーの操作ではなくメソッドを使用して状態の変更を行った場合もコンポーネントイベントが通知されます。

4.3.2 SoftKeyListener

SoftKeyListener インタフェースは、2 つのソフトキーが操作されるのを待ち受けるときに使用します。このインタフェースで定義されているメソッドは、softKeyPressed()と softKeyReleased()の 2 つです。ソフトキーリスナーの定義と登録は、以下のように行います。

- リスナーとして使用するクラスに、SoftKeyListener インタフェースおよび softKeyPressed()、softKeyReleased()の 2 つのメソッドを実装します。ソフトキーが押されると softKeyPressed()メソッドが、放されると softKeyReleased()メソッドが呼び出されます。
- Panel オブジェクトに対して setSoftKeyListener()メソッドを呼び出し、上記で実装したリスナークラスのインスタンスを登録します。

注意事項：

- ソフトキーは、Canvas からも使用することができます。その場合は、低レベルイベントの処理方法に従ってソフトキーリスナーの代わりに Canvas.processEvent()を使用します（通常のキー操作と同様に、KEY_PRESSED_EVENT または KEY_RELEASED_EVENT が通知されます）。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

4.3.3 KeyListener

KeyListener インタフェースは、ソフトキー以外のいずれかのキーが操作されるのを待ち受けるときに使用します。このインタフェースで定義されているメソッドは、keyPressed()と keyReleased()の 2 つです。キーリスナーの定義と登録は、以下のように行います。

- リスナーとして使用するクラスに、KeyListener インタフェースおよび keyPressed()、keyReleased()の 2 つのメソッドを実装します。キーが押されると keyPressed()メソッドが、放されると keyReleased()メソッドが呼び出されます。
- Panel オブジェクトに対して setKeyListener()メソッドを呼び出し、上記で実装したリスナークラスのインスタンスを登録します。

【DoJa-2.0】

Panel では、画面のスクロールやコンポーネントの操作などのために方向キーおよび選択キーを使用します。DoJa-2.0 プロファイル以降では、高レベル API 利用時におけるこれらのキーのキーイベントについて以下のように規定されています。

- 方向（上下左右）キーはキーイベントを発生しません。これらのキーは、スクロールおよびフォーカス制御のために常に Panel で処理されます。
- 選択（決定）キーは、フォーカスを得ているコンポーネントがディスプレイ上に表示されている場合に限りキーイベントを発生しません。フォーカスを得ているコンポーネントがディスプレイ上に表示されている状態では、選択キーはそのコンポーネントの操作のために Panel で処理されます。

なお、DoJa-1.0 プロファイルでは、どのような状況でどのようなキーイベントが Panel で処理されるかについてはメーカーにより異なります。

4.3.4 MediaListener

MediaListener インタフェースは、メディアデータ再生においてメディアイベント（AudioPresenter または VisualPresenter が通知するイベント）を待ち受けるときに使用します。このインタフェースで定義されているメソッドは mediaAction()だけです。メディアリスナーの定義と登録は、以下のように行います。

- リスナーとして使用するクラスに、MediaListener インタフェースおよび mediaAction()メソッドを実装します。メディアイベントが発生すると、このメソッドが呼び出されます。
- MediaPresenter オブジェクトに対して setMediaListener()メソッドを呼び出し、上記で実装したリスナークラスのインスタンスを登録します。

AudioPresenter には、次の 6 つのイベントタイプが定義されています。

- AUDIO_COMPLETE : サウンドメディアデータの再生完了イベント
- AUDIO_PLAYING : サウンドメディアデータの再生開始イベント
- AUDIO_STOPPED : サウンドメディアデータの再生停止イベント
- AUDIO_PAUSED : サウンドメディアデータの再生一時停止イベント
- AUDIO_RESTARTED : サウンドメディアデータの再生再開イベント
- AUDIO_SYNC : サウンドメディアデータの同期イベント

VisualPresenter には、次の 3 つのイベントタイプが定義されています。

- VISUAL_COMPLETE : 画像メディアデータの再生完了イベント
- VISUAL_PLAYING : 画像メディアデータの再生開始イベント
- VISUAL_STOPPED : 画像メディアデータの再生停止イベント

なお、メディアデータの詳細については 4.4 項を参照してください。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

【DoJa-2.0】【DoJa-3.0】

- DoJa-1.0 プロファイルで発生が保証されていたメディアイベントは AUDIO_COMPLETE のみです。他のメディアイベントをサポートするかどうかは、DoJa-1.0 プロファイルではメーカーにより異なります。また、メディア開始・停止・完了イベント以外のメディアイベントは、DoJa-1.0 プロファイルではサポートされません。
- DoJa-2.0 プロファイルでは、AudioPresenter、VisualPresenter ともにメディア再生開始・停止・完了イベントが全ての機種でサポートされます。また、それら以外のメディアイベントは DoJa-2.0 プロファイルでは i アプリオプション API のカテゴリに含まれます。
- DoJa-3.0 プロファイルでは、本項に挙げた全てのメディアイベントが i アプリ基本 API に取り入れられています。

4.3.5 TimerListener

TimerListener インタフェースは、タイマー終了イベントを待ち受けるときに使用します。このインタフェースで定義されているメソッドは timerExpired()だけです。タイマーリスナーの定義と登録は、以下のように行います。

- リスナーとして使用するクラスに、TimerListener インタフェースおよび timerExpired()メソッドを実装します。タイマーが終了すると、このメソッドが呼び出されます。
- Timer オブジェクトに対して setListener()メソッドを呼び出し、上記で実装したリスナークラスのインスタンスを登録します。

注意事項：

- ShortTimer には、これに対応するインタフェースはありません。ShortTimer では、Canvas の processEvent()メソッドを通じて TIMER_EXPIRED_EVENT を受け取ります。

4.4 マルチメディアデータの使用

4.4.1 i アプリが利用可能なマルチメディアデータ

i アプリ実行環境には、マルチメディアデータとして画像（静止画像および動画像）やサウンドを取り扱うための機能が備えられています。以下に、i アプリ実行環境で標準的に取り扱うことのできるマルチメディアデータのフォーマットを記載します。

- GIF 形式画像データ： DoJa-1.0 以降

  GIF87a、GIF89a をサポートします。通常の画像の他にインターレース GIF、透過 GIF、アニメーション GIF を取り扱うことができます。アニメーション GIF は、動画像として VisualPresenter クラスで再生します（4.4.2 項参照）。

- JPEG 形式画像データ： DoJa-2.0 以降（FOMA 携帯電話では DoJa-1.0 で先行サポート）

  各機種で共通的に使用可能な形式は JFIF Baseline エンコーディングです。メーカーによっては EXIF を取り扱えるものもあります。拡張 DCT 方式など非対応のエンコーディングを用いたデータを使用すると例外が発生する場合があります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- BMP 形式画像データ： DoJa-5.0 以降

  以下の条件を満たす BMP 形式データを使用することができます。

  - Windows BMP 形式で、bottom-up DIB であること
  - ピクセルあたりのビット数は 1, 4, 8, 16, 24 のいずれかであること
  - 無圧縮であること（圧縮形式が BI_RGB であること ただし 1 ピクセル 16 ビットの場合は BI_BITFIELDS（RGB 5-5-5 および RGB 5-6-5）にも対応）

  なお、BMP 形式のデータは i アプリ内でのみ使用可能であり、それらをネイティブの画像保存領域（マイピクチャ）に保存することはできません。

- MFi 形式サウンドデータ： DoJa-1.0 以降

  i モードにおける標準的なサウンドデータである、MFi（Melody For i-mode）形式のサウンドデータをサポートします。サウンドの基本的な再生方法については 4.4.2 項を参照してください。MFi 形式サウンドデータでは、ADPCM や 3D サウンドなど多彩な拡張機能を利用することもできます。

- SMF 形式サウンドデータ： DoJa-1.0 以降（ただし FOMA 携帯電話のみ）

  FOMA 携帯電話では、サウンドデータとして MFi 形式の他に SMF（Standard MIDI File）形式のサウンドデータを使用することもできます。サウンドの基本的な再生方法については 4.4.2 項を参照してください。

- i モーション形式動画データ： DoJa-1.0 以降（ただし FOMA 携帯電話のみ）

  FOMA 携帯電話には、動画像コンテンツとして i モーションをサポートしているものがあります。そのような FOMA 携帯電話では、 i アプリから i モーション形式動画データを取り扱うことができます（4.4.2 項、4.4.5 項参照）。なお、 i モーション形式動画データには複数のバージョンがあり、各機種についてそれぞれに適したものを使用する必要があります。どの機種がどのバージョンのものをサポートしているかについては、別途 NTT ドコモより公開されます。

4.4.2 プレゼンタの利用

i アプリ実行環境では､アニメーションを含む画像の再生やサウンドの再生を行う機能が提供されます。画像を再生するクラスとサウンドを再生するクラスとしてそれぞれ、`VisualPresenter` と `AudioPresenter` があります。

注意事項：

- `VisualPresenter` はコンポーネントであり、高レベル API にて `Panel` オブジェクトに追加して使用します。`VisualPresenter` では、標準的に再生可能な画像として GIF アニメーションをサポートしています。また FOMA 携帯電話では､映像データとして i モーション形式データをサポートしています(4.4.5 項参照)。
- `AudioPresenter` はコンポーネントではありません。`AudioPresenter` は `Panel` 表示時、`Canvas` 表示時のいずれの場合も使用することができます。
- NTT ドコモの提供する DoJa-2.0 プロファイル以降対応の i アプリ開発環境では、`AudioPresenter` は MFi 形式データと MIDI 形式データをサポートしています。 i アプリ開発環境を使用した実際の開発手順などについては第 15 章を参照してください。

  【DoJa-2.0】

  NTT ドコモの提供する DoJa-1.0 プロファイル対応 i アプリ開発環境では、サウンドデータとして MFi 形式データを使用することはできません。DoJa-1.0 対応 i アプリ開発環境で使用可能なサウンドデータは MIDI 形式データのみとなります。

- FOMA 携帯電話の中には、 i アプリやブラウザ、通話などを、タスクを切り替えながら並列に起動して利用できるものがあります。このような携帯電話において、別のタスクでサウンドや音声を取り扱う機能が起動されている状態では、 i アプリからサウンドや音声を取り扱う機能（`AudioPresenter` によるサウンド再生や `VisualPresenter` による i モーション再生など）を使用すると例外（`UIException`）が発生します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

iアプリのサウンド・音声関連機能と同時に使用できない機能には、通話機能や音楽プレーヤーなどがあります（機種によっては音楽プレーヤーを実行中にiアプリがサウンド再生を行っても例外を発生させず、音楽プレーヤーとiアプリのサウンドの切り替え制御などを行うものもあります）。

次に、VisualPresenterクラスを使って画像を再生する例と、AudioPresenterクラスを使ってサウンドを再生する例を示します。

この例では、画像ファイルやサウンドファイルとして、iアプリのJARファイルに同梱されたリソースを使用しています（"resource:///"で始まるURLの使用）。リソースについては、第7章を参照してください。なお、このURLにはリソースだけではなく、Webサーバー上のファイルを示すHTTPのURLや、ScratchPad上の位置を示すURLを指定することもできます。

例：　画像ファイルの表示

```
package uidemo;
import java.util.*;
import com.nttdocomo.ui.*;
import com.nttdocomo.io.ConnectionException;

public class VisualPresenterDemo extends UIDemoPanel {
    int           px, py;         // 現在の画像の表示位置（VisualPresenter内）
    VisualPresenter theVP;        // プレゼンター
    Button        startButton;    // GIFアニメーション再生ボタン
    Button        stopButton;     // GIFアニメーション停止ボタン
    ListBox       repeatChoice;   // 繰り返し再生指示用チェックボックス
    Ticker        ticker;         // キーマップ説明用ティッカーテープ

    static String instr = "Control ImagePos: Key 2-Down 8-Up 4-Left 6-Right";

    /**
     * VisualPresenterDemo コンストラクタ
     * 必要な画像データを取得して、VisualPresenterに引き渡す。
     */
    VisualPresenterDemo() {

        // 画像データを取得する。
        MediaImage mi =
            MediaManager.getImage("resource:///uidemo/img/singleloop.gif");

        // use()メソッドを呼び出すまで画像データを利用することはできない。
        try {
            mi.use();
        } catch (ConnectionException ce) {
            // ここで入出力の例外を処理する。
        } catch (UIException uie) {
            // ここでUIの例外を処理する。
        }

        px    = 0;
        py    = 0;

        theVP             = new VisualPresenter();
        theVP.setImage(mi);

        ticker            = new Ticker(instr);

        startButton       = new Button("start");
        stopButton        = new Button("stop ");

        repeatChoice      = new ListBox(ListBox.CHECK_BOX);
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
    repeatChoice.append("Repeat");

    add(theVP);
    add(ticker);
    add(startButton);
    add(stopButton);
    add(repeatChoice);

    // メディアリスナーを設定する。
    theVP.setMediaListener(new MediaListenerClass());

    setTitle("Status Line");
    this.setComponentListener(new ListenerClass());
    this.setKeyListener(new KeyListenerClass());
}


class KeyListenerClass implements KeyListener {

    /**
     * キーリリースイベントは無視する。
     */
    public void keyReleased(Panel p, int key) {}

    /**
     * キープレスイベントにより画像表示位置の調整を行う
     *
     * @param p      イベントの発生した Panel
     * @param key    押されたキーの値
     */
    public void keyPressed(Panel p, int param1) {

        if (VisualPresenterDemo.this == p) {

            if (param1 == Display.KEY_8) {             // 画像の位置（下）
                py--;
            } else if (param1 == Display.KEY_2) {      // 画像の位置（上）
                py++;
            } else if (param1 == Display.KEY_4) {      // 画像の位置（左）
                px++;
            } else if (param1 == Display.KEY_6) {      // 画像の位置（右）
                px--;
            }

            // VisualPresenter の setAttribute()メソッド呼び出しにより
            // 画像表示位置を設定する。
            theVP.setAttribute(VisualPresenter.IMAGE_XPOS, px);
            theVP.setAttribute(VisualPresenter.IMAGE_YPOS, py);
        }
    }
}

class ListenerClass implements ComponentListener {

    /**
     * コンポーネントイベントにより再生の開始、停止を制御する
     *
     * @param c        イベントの発生したコンポーネント
     * @param type     イベントタイプ
     * @param param    イベントパラメータ
     */
    public void componentAction(Component c, int type, int param) {
        if (c == startButton) {                // 再生ボタンを押すと再生開始
            theVP.play();
        } else if (c == stopButton) {          // 停止ボタンを押すと再生停止
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
                theVP.stop();
            }
        }
    }

    class MediaListenerClass implements MediaListener {

        /**
         * VisualPresenter からのイベントを受け取り、必要に応じて再度再生を行うための
         * メディアリスナー。ループ再生機能はここで実現される。
         */
        public void mediaAction(MediaPresenter source, int type, int param) {

            if (source == theVP) {
                switch (type) {

                // 画像の再生完了イベントはここで処理される。Panel タイトルに
                // 再生を完了したことを表示し、必要に応じて再度再生を繰り返す。
                // この処理は非ループ再生型 GIF アニメーションでのみ有用。
                case VisualPresenter.VISUAL_COMPLETE :
                    setTitle("Complete");

                    if ((repeatChoice.isIndexSelected(0))) {
                        ((VisualPresenter) source).play();
                        setTitle("[Comp]Play again");
                    }
                    break;

                // 画像の再生開始イベントはここで処理される。
                // Panel タイトルに再生を開始したことを表示する。
                case VisualPresenter.VISUAL_PLAYING :
                    setTitle("Playing");
                    break;

                // 画像の再生停止イベントはここで処理される。
                // Panel タイトルに再生を停止したことを表示する。
                case VisualPresenter.VISUAL_STOPPED :
                    setTitle("Stopped");
                    break;
                }
            }
        }
    }

    /**
     * MediaImage は不要になったら破棄（unuse()および dispose()）すべきである。
     * 破棄することにより資源の解放が行われる。ここでは MediaImage を再利用する可能性
     * があるため破棄は行わない。
     */
    public void leave() {
    }
}
```

例: サウンドファイルの再生

```
package uidemo;

import com.nttdocomo.ui.*;
import com.nttdocomo.io.ConnectionException;

public class AudioPresenterDemo extends UIDemoPanel {
       ListBox           loopChoice;     // ループ再生フラグ
       Button            playButton;     // サウンド再生ボタン
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
Button            stopButton;     // サウンド停止ボタン
Label             statusLabel;    // ステータス表示ラベル
AudioPresenter    theAP;          // AudioPresenter
MediaSound        theMediaSound;  // MediaSound

/**
* AudioPresenterDemo コンストラクタ
* 再生/停止用の 2 つのボタンと、ループ再生を選択するためのチェックボックスを作成する。
*/
AudioPresenterDemo() {
       loopChoice = new ListBox(ListBox.CHECK_BOX);
       loopChoice.append("loop");
       playButton = new Button("play");
       stopButton = new Button("stop");
       statusLabel = new Label("status line");

       theMediaSound =
              MediaManager.getSound("resource:///uidemo/img/music.mld");

       // use()メソッドを呼び出すまでサウンドデータを利用することはできない。
       try {
              theMediaSound.use();
       } catch (ConnectionException ce) {
              // ここで入出力の例外を処理する。
       } catch (UIException uie) {
              // ここで UI の例外を処理する。
       }

       theAP = AudioPresenter.getAudioPresenter();
       theAP.setSound(theMediaSound);

       this.add(playButton);
       this.add(stopButton);
       this.add(loopChoice);
       this.add(statusLabel);
       this.setComponentListener(new ListenerClass());
       theAP.setMediaListener(new MediaListenerClass());
}

class ListenerClass implements ComponentListener {
       /**
       * サウンドデータの再生/停止を制御するためのボタンのイベントを受け取る
       * リスナークラス。
       */
       public void componentAction(Component c, int type, int param) {
              if (c == playButton) {
                     theAP.play();
              } else if (c == stopButton) {
                     theAP.stop();
              }
       }
}

class MediaListenerClass implements MediaListener {
       /**
       * AudioPresenter からのイベントを受け取り、必要に応じて再度再生を
       * 行うためのメディアリスナー。ループ再生機能はここで実現される。
       */
       public void mediaAction(MediaPresenter source,int type,int param) {
              if (source == theAP) {
                     switch(type) {
                     // サウンドの再生完了イベントはここで処理される。ステータス表示ラベル
                     // に再生を完了したことを表示し、必要に応じて再度再生を繰り返す。
                            case AudioPresenter.AUDIO_COMPLETE:
                                   statusLabel.setText("complete");
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
                                        if (loopChoice.isIndexSelected(0))
                                                ((AudioPresenter)source).play();
                                        break;

                                // サウンドの再生開始イベントはここで処理される。
                                // ステータス表示ラベルに再生を開始したことを表示する。
                                        case AudioPresenter.AUDIO_PLAYING:
                                                statusLabel.setText("playing ");
                                                break;

                                // サウンドの再生停止イベントはここで処理される。
                                // ステータス表示ラベルに再生を停止したことを表示する。
                                        case AudioPresenter.AUDIO_STOPPED:
                                                statusLabel.setText("stopped ");
                                                break;
                                }
                        }
                }
        }

        /**
        * この画面が非表示になる際に、サウンドの再生を停止するため、AudioPresenter.stop()メソッド
        * の呼び出しを行う。MediaSound をこれ以上使用しない場合は、MediaSound. unuse()メソッドお
        * よび MediaSound.dispose()メソッドも呼び出すべきである（各種資源の解放が行われる）。ここで
        * は MediaSound を再度利用する可能性があるため、これらのメソッドは呼び出さない。
        */
        public void leave() {
                theAP.stop();
        }
}
```

【DoJa-2.0】

- DoJa-1.0 プロファイルでは、メディアデータがセットされていないメディアプレゼンタに対し stop()メソッドを呼び出した際の振る舞いはメーカーに依存していました。これに対し DoJa-2.0 プロファイル以降では、メディアデータがセットされていないメディアプレゼンタに対し stop()メソッドを呼び出すと例外（UIException）が発生します。メディアプレゼンタにメディアデータがセットされているかどうかは MediaPresenter.getMediaResource()メソッドを使用して判定してください。
- DoJa-2.0 プロファイル以降では、VisualPresenter によるアニメーション GIF 再生について以下の規定が追加されています。
  - 再生可能なフレーム数は、最低限 8 フレームが保証されます。
  - ループ型アニメーション GIF を再生した場合、1 回のループが終了する毎に再生完了イベントが発生します。
  - ループ型アニメーション GIF の最大ループ回数は 16 回です。これを超えたループ回数が指定されている場合でも、16 回ループした時点で再生は停止します。

【DoJa-3.0】

DoJa-3.0 プロファイル以降では、メディアデータの取得元として URL だけでなくデータファイルのバイトイメージを指定することができます。バイトイメージは、byte 配列または InputStream の形式で与えることができます。

また DoJa-3.0 プロファイル以降では、AudioPresenter に関して以下の項目を調整するための属性が追加されています。

- テンポ（AudioPresenter.CHANGE_TEMPO）
- ボリューム（AudioPresenter.SET_VOLUME）
- キー（AudioPresenter.TRANSPOSE_KEY）

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

類似の機能が DoJa-2.0 プロファイルの i アプリオプション API（AudioPresenter2 クラス）に含まれていますが、それらとはプログラミング上の互換はありませんのでご注意ください。

【DoJa-4.0】

DoJa-4.0 プロファイル以降では、メディアデータの共通インタフェースである MediaResource に以下の機能が追加されました。

- isRedistributable()： このメディアデータが再配布可能に設定されているかどうかを取得します。再配布可非が設定できるかどうか、およびその設定方法は、メディアデータの種類および携帯電話プラットフォーム（PDC または FOMA）により異なります。再配布不可としてネイティブの保存領域に保存されたデータは、メール添付や赤外線機能などを使用して携帯電話外部に持ち出すことはできません。
- setRedistributable()： このメディアデータに、再配布可/不可の設定を行います。
- getProperty()： メディアデータのプロパティ値を取得します。サポートされるプロパティの種別は、メディアデータの種別ごとに定められています。DoJa-4.0 プロファイルでは、以下のプロパティがサポートされています。
  - MediaSound.AUDIO_3D_RESOURCES

    このメディアサウンドのデータに 3D サウンド制御情報（定位情報）が埋め込まれている場合、それを使用するためには 3D サウンド制御リソースをいくつ必要とするかを取得します。3D サウンド制御および 3D サウンド制御リソースについては 13 章を参照してください。
  - MediaImage.MP4_AUDIOTRACK

    このメディアイメージのデータが i モーション形式データである場合、データに含まれるオーディオトラックの数（数を表す数字文字列）を取得します。
  - MediaImage.MP4_TEXTTRACK

    このメディアイメージのデータが i モーション形式データである場合、データに含まれるテキストトラックの数（数を表す数字文字列）を取得します。
  - MediaImage.MP4_VIDEOTRACK

    このメディアイメージのデータが i モーション形式データである場合、データに含まれるビデオトラックの数（数を表す数字文字列）を取得します。

【DoJa-4.1】

DoJa-4.1 プロファイル以降、メーカーによっては MediaManager.getImage()メソッドの引数に Flash コンテンツのデータを指定して呼び出すことで、MediaImage オブジェクトを取得することをサポートすることがあります。ただし、Flash コンテンツから生成された MediaImage オブジェクトは、プレゼンタで再生することはできません。Flash コンテンツから生成された MediaImage オブジェクトは、ImageStore.addEntry()メソッドにより Flash コンテンツをネイティブの画像保存領域に保存する目的でのみ使用することができます。

【DoJa-5.0】

DoJa-5.0 プロファイル以降の MediaManager クラスでは、複数の MediaImage オブジェクト、または複数の MediaSound オブジェクトを一括して利用可能状態にするための static メソッド use()が設けられています。この use()メソッドでは、メディアデータオブジェクトの利用を一度きりとするかどうかを指定することができます。メディアデータオブジェクトにおける「一度きり」の利用指定については、4.4.7 項も参照してください。

なお、DoJa-5.0 プロファイルにて AudioPresenter クラスに以下の機能が追加されています。ただしこれらはオプショナルであり、サポートの有無はメーカーにより異なります。

- ループ演奏指定およびループ再生イベント（LOOP_COUNT 属性および AUDIO_LOOPED イベント）
- i モーションのオーディオトラックの再生（getAudioTrackPresenter()メソッド）
- 曲全体の演奏時間の取得（getTotalTime()メソッド）

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

4.4.3 複数の AudioPresenter の同時再生

DoJa-2.0 プロファイル以降においては、アプリケーションプログラムが複数の AudioPresenter を取得し、それらに対して同時に再生指示を与えることができます。この機能は、DoJa 各プロファイル間で若干動作が異なります。以下に、それぞれのプロファイルにおける動作の概要を解説します。

【DoJa-2.0】

DoJa-2.0 プロファイルでは、各機種で共通に保証される AudioPresenter の同時再生可能数は最低 1 つです。つまり、メーカーによっては複数 AudioPresenter の同時再生を行うことができない場合があります。

アプリケーションプログラムから同時再生可能なプレゼンタの数を超えて再生指示を行った場合、どのプレゼンタが有効なものとして再生されるかは、プレゼンタに設定された優先順位属性（AudioPresenter.PRIORITY）の値により決定されます。同時に 1 つのプレゼンタだけ再生可能な携帯電話において、優先順位の低いプレゼンタを再生中に優先順位の高いプレゼンタの再生が指示された場合、優先順位の高い方のプレゼンタが低い方のプレゼンタに割り込んで再生されます。優先順位の高い方のプレゼンタの再生が終了すると、低い方のプレゼンタの再生が再開されます。

優先順位属性をサポートしていない携帯電話では、後に再生したプレゼンタの方が再生中のプレゼンタより優先度が高いものとして扱われます。また、携帯電話の能力（同時発音可能数や使用可能なチャンネル数）を超えた再生指示が行われた場合の振る舞いはメーカーにより異なります。

【DoJa-3.0】

DoJa-3.0 プロファイルでは、各機種で共通に保証される AudioPresenter の同時再生可能数は最低 2 つです。

アプリケーションプログラムから同時再生可能なプレゼンタの数を越えて再生指示を行った場合、どのプレゼンタが有効なものとして再生されるかは、プレゼンタに設定された優先順位属性（AudioPresenter.PRIORITY）の値により決定されます。同時に 2 つのプレゼンタが再生可能な携帯電話において、中優先度、低優先度の 2 つのプレゼンタを再生中に高優先度のプレゼンタの再生が指示された場合、低優先度のプレゼンタは停止して新たに指示されたプレゼンタの再生が開始されます（中優先度のプレゼンタは継続して再生されつづけます）。なお、停止すべき優先度のプレゼンタが複数存在する場合、それらのうちどのプレゼンタが停止するかはメーカーにより異なります。

優先順位属性をサポートしていない携帯電話では、同時再生可能なプレゼンタの数を越えて再生指示を行った場合、後に再生したプレゼンタの方が再生中のプレゼンタより優先度が高いものとして扱われます。

携帯電話の能力の範囲（同時発音可能数や使用可能なチャンネル数、プレゼンタ数）であれば、同時に再生した音はプレゼンタの優先順位に関係なく全て聞こえます。携帯電話の能力を超えた再生指示が行われた場合には、同時再生可能なプレゼンタの数の範囲内であっても全ての音が聞こえるとは限りません。

なお DoJa-3.0 プロファイルでは、AudioPresenter の同時再生に関してポートの概念が導入されています。ポートとは、同時再生可能な AudioPresenter の数分用意された仮想的な再生器です。アプリケーションプログラムは、AudioPresenter.getAudioPresenter()メソッドのオプショナルの引数により、個々の AudioPresenter をどのポートに割り付けるかを明示的に宣言することができます。ポート番号に割り付けられた AudioPresenter と割り付けられていない AudioPresenter を同時に再生することはできません。

【DoJa-4.0】

DoJa-4.0 プロファイルでは、AudioPresenter にポート指定を行う場合については、各機種で共通に保証される AudioPresenter の同時再生可能数は最低 4 つに拡張されています。その他については DoJa-3.0 プロファイルでの動作に準じます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

注意事項：

- FOMA 携帯電話ではサウンドデータとして MFi 形式フォーマットおよび SMF 形式フォーマットをサポートしていますが、両者を混在させて同時に再生できるかどうかはメーカーにより異なります。複数のサウンドを同時再生する場合は、サウンドデータのフォーマットをいずれか一方に統一するようにしてください。

4.4.4 サウンド再生の同期イベント

DoJa-3.0 プロファイル以降の `AudioPresenter` では、サウンド再生における同期イベント発生機能をサポートしています。同期イベントとは、サウンドの再生進行に同期してメディアリスナーに通知されるメディアイベントです。`AudioPresenter.setAttribute()`メソッドを使用して、同期イベント発生可否属性（`AudioPresenter.SYNC_MODE`）に対しオン（`AudioPresenter.ATTR_SYNC_ON`）またはオフ（`AudioPresenter.ATTR_SYNC_OFF`）を指定することでイベントの発生有無を制御することができます。

同期イベント機能を利用する場合、サウンドデータを構成するうちの任意の 1 チャンネルをイベント発生源として指定します。指定されたチャンネルの指定されたノートメッセージに連動して、メディアリスナーに同期イベントが通知されます。

なお、イベント発生源として指定されたチャンネルも発音の対象となります。発音が不要な場合、サウンドデータ作成時においてそのチャンネルに対し消音指定を行うようにしてください。

【DoJa-2.0】【DoJa-3.0】

サウンド再生の同期イベントは DoJa-1.0 プロファイルではサポートされておらず、また DoJa-2.0 プロファイルでは i アプリオプション API の位置付けとなっています。この機能は DoJa-3.0 プロファイルから i アプリ基本 API に取り入れられました。

4.4.5 FOMA 携帯電話による VisualPresenter での i モーションの再生

PDC 携帯電話においては、`VisualPresenter` で再生可能な動画メディアはアニメーション GIF のみです。これに対し FOMA 携帯電話では、アニメーション GIF に加え i モーション形式データを `VisualPresenter` で再生することができます。 i モーション形式データは、アニメーション GIF と同様の方法により `MediaImage` オブジェクトを生成し、`VisualPresenter` を使用することで再生することができます。

i モーション形式データから生成した `MediaImage` オブジェクトから `getImage()`メソッドを使用して `Image` オブジェクトを取り出そうとしたり、`ImageStore` クラスでその `MediaImage` オブジェクトの保存や再取得を行うことはできません（ i モーション形式データの保存や取得には、i アプリオプション API である `MovieStore` クラスを使用します）。

DoJa-3.5 プロファイル以降の i アプリ実行環境では、`VisualPresenter` クラスと `MediaImage` クラスについて、 i モーションを取り扱うための以下の拡張が行われています。

・プレーヤーモードの指定

i モーションの再生には、再生指示（`VisualPresenter.play()`）により i アプリが一時中断し、ネイティブの i モーションプレーヤーが起動するネイティブプレーヤーモードと、`Panel` に配置された `VisualPresenter` コンポーネントが i アプリを中断することなく i モーション再生を行うインラインプレーヤーモードがあります。また DoJa-5.0 プロファイル以降では、ネイティブプレーヤーがディスプレイサイズを活かして拡大再生を行うための全画面プレーヤーモードが追加されています。全画面プレーヤーモードはネイティブプレーヤーモードのバリエーションの `1` つであり、再生指示によって i アプリは一時中断しま

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

す。また、iアプリから全画面プレーヤーモードを使用した場合のプレーヤーの表示（見映え）は、ネイティブプレーヤーで全画面プレーヤーモードを使用した場合のプレーヤーの表示に準じます。

どの機種でも使用可能なプレーヤーモードはネイティブプレーヤーモードです。インラインプレーヤーモードと全画面プレーヤーモードはオプショナルであり、使用できるかどうかはメーカーにより異なります。また、インラインプレーヤーモードや全画面プレーヤーモードでテキストトラックを含むiモーションデータを再生できるかどうかはメーカーにより異なります。

プレーヤーモードは、VisualPresenterクラスのsetAttribute()メソッドによりPLAYER_MODE属性の設定を行なうことにより選択することができます。

・音声再生モードの指定

iモーションは、音声を内包することができるメディアデータです。インラインプレーヤーモードによるiモーション再生では、iモーションに含まれている音声データも再生するかどうかを選択することができます。iモーションに含まれている音声データを再生しないように設定されている場合はVisualPresenterはオーディオ再生に関するリソースを使用せず、その際iアプリはVisualPresenterとは別に用意されたAudioPresenterでメディアサウンドを同時に再生することができます。

音声再生モードは、VisualPresenterクラスのsetAttribute()メソッドによりAUDIO_MODE属性の設定を行なうことにより選択することができます。

・プログレッシブダウンロードへの対応

iモーションのダウンロードには、全てのデータを携帯電話上にダウンロードし終わった段階で再生が可能となる通常のダウンロード方法の他に、一定量のデータを取得した時点で再生を開始し、それ以降はデータのダウンロードと再生を並行して行うプログレッシブダウンロードがあります。DoJa-3.5プロファイル以降では、APIレベルでプログレッシブダウンロードに対応するために、MediaManagerクラスにgetStreamingImage()メソッドが追加されています。このメソッドを使用してプログレッシブダウンロード対応のiモーションコンテンツから生成されたMediaImageオブジェクトでは、VisualPresenterクラスで再生指示が行なわれるとデータのダウンロードと再生が並行して行われます。

なお、メーカーによっては、iアプリにおけるプログレッシブダウンロードをサポートしない場合があります。そのような機種ではgetStreamingImage()メソッドを使用した場合にもプログレッシブダウンロードとはならず、コンテンツ全体がダウンロードされた後に再生が行われます。

iアプリから再生可能なiモーションのデータサイズは、MediaManager.getImage()メソッドやMediaManager.getStreamingImage()メソッドにHTTPスキームのiモーションURLを指定する場合には最大150Kバイトまでです（本書5.1項に記載のHTTP(S)通信サイズの制約を受けることによります）。また、分割して入手したiモーションデータをScratchPadや実行メモリ上で結合するなどの方法により大きなiモーションデータをiアプリ内で生成するような場合は、500Kバイトまでのiモーションデータを再生することができます（メーカーによっては、より大きなサイズのiモーションデータに対応する場合もあります）。

なお、非活性化状態の待ち受けアプリケーションからネイティブプレーヤーモードでiモーションを再生することはできません。

【DoJa-5.0】

iアプリから再生可能なiモーションのデータサイズの規定（どの機種でも共通的に取り扱うことのできるサイズの規定）は、DoJaプロファイルバージョンにより異なります。DoJa-5.0プロファイルでは500Kバイトまでのiモーションデータを再生することができますが、それより古いプロファイルでは300Kバイトまでなどに制限されています。

注意事項：

- フル楽曲コンテンツとしての設定が行われているiモーションデータは、VisualPresenterクラスで再生することはできません。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

4.4.6 マルチメディアデータの外部メモリへの出力制御

ｉアプリで取り扱うマルチメディアデータ（MediaResource 継承インタフェースの実装オブジェクト）は、本書の第 11 章で解説されるようなアプリケーション連携機能（ImageStore クラスや MovieStore クラスなど）を利用して携帯電話ネイティブのデータ保存領域（以下、内蔵メモリと呼びます）に保存することができます。一方 FOMA 902i 以降の携帯電話では、コンテンツ保護技術を活用して内蔵メモリのコンテンツを外部メモリに移動したり、さらにそれを他の携帯電話の内蔵メモリに移動することができるようなものがあります。

DoJa-4.1 プロファイル（902iS）以降では、ｉアプリが内蔵メモリに保存したマルチメディアデータがユーザー操作で外部メモリに出力されるような状況において、そのコンテンツに対する保護のレベルを制御可能とする仕様が追加されています。

以下に、ｉアプリから内蔵メモリに保存されたマルチメディアデータの、コンテンツの保護レベルの取り扱いについて記載します。

・再配布不可情報が設定されていない（再配布が許可された）コンテンツ

再配布不可情報が設定されていないコンテンツは、外部メモリの持つ著作権保護機構の対象外となります。従って、ｉアプリが再配布不可情報の設定されていないマルチメディアデータを内蔵メモリに保存し、そのデータがユーザー操作によって外部メモリに移される際、そのコンテンツの内容は暗号化されていない平文の状態で外部メモリに記録されます。そのため、このようにして外部メモリに移されたデータは PC などのビューアなどでも閲覧することができます。

・再配布不可情報が設定された（再配布が許可されない）コンテンツ

再配布不可情報が設定されているコンテンツは外部メモリの持つ著作権保護機構の対象となり、外部メモリに出力される際に暗号化されます。通常、ｉアプリから内蔵メモリに保存したコンテンツを外部メモリに移すと、そのコンテンツはデータ保存時と同じ機種、かつ保存時と同じ UIM が挿入されている携帯電話上でなければ復号することはできません。なお、ｉアプリが内蔵メモリに保存しようとしているデータについて、保存前に MediaResource.setProperty()メソッドを

第 1 引数（プロパティ名）： “X-Dcm-Move”
第 2 引数（プロパティ値）： “0001”

を指定して呼び出しておくことで、そのデータについては保存時と同じ UIM が挿入された携帯電話上であれば、機種を問わずに復号できるようになります。

このプロパティ情報は内蔵メモリ上でも保持されており、ｉアプリが ImageStore クラスなどを使用してそのデータを再度内蔵メモリから読み込んだ際にも有効なまま引き継がれます（MediaManager クラスを使用してマルチメディアデータをオブジェクト化した際には、これらのプロパティは未設定の状態となります）。

4.4.7 メディアデータ利用時のメモリ管理

ｉアプリが画像やサウンドなどのメディアデータを利用する際には、比較的多くのアプリケーション実行メモリ（ヒープメモリ）が消費されます。一般的な実装ではメディアデータを取り扱う際には、GIF や JPEG などデータ形式毎に固有のオリジナルデータを保持するためのメモリや、それらを内部形式形式に変換したデータを保持するためのメモリが使用されます。

DoJa-5.0 プロファイルでは、メディアデータ使用時にｉアプリ実行環境が行うメモリ管理を以下のようにモデル化し、ｉアプリからそれらに対して一定の制御を行うことができるよう API が拡張されています。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

オリジナルデータ領域：

GIF や JPEG などのオリジナルデータを保持するために使用されるメモリです。原則として、メディアデータのオブジェクトが生成される時点で確保され、そのオブジェクトに対して dispose()メソッドが呼び出されるまで保持されます。

内部データ領域：

オリジナルデータを、システム内部で取り扱うことのできる形式に変換したものを保持するために使用されるメモリです。例えば画像の場合は、GIF や JPEG などのデータを画素毎のデータに展開したものが相当します。原則として、メディアデータのオブジェクトに対して use()メソッドが呼び出される時点で確保され、そのオブジェクトに対して unuse()メソッドが呼び出されるまで保持されます。

例えば、アプリケーションの起動から終了までの間に複数回 use()、unuse()を呼び出さない（利用可能状態と利用不可能状態を何度も行き来しない）ことが確実であるメディアデータに対しては、メディアデータを利用可能状態とする際にオリジナルデータを破棄し、その分のメモリを解放することを指示することができます（「一度きり」の利用指定）。

また、短期間だけ利用可能状態とすればよいメディアデータを複数取り扱うような場合には、複数のメディアデータのオブジェクトの間で 1 つの内部データ領域を引き継ぎながら使用するといったことを指示することもできます（内部データ領域の再利用）。

これらの機能は、各メディアデータインタフェースなどに規定されている use()メソッドの拡張として提供されています。

注意事項：

- i アプリ実行環境の実装によっては、本項で解説するメモリ管理のモデルがそのまま適用できない場合があります。そのような実装では i アプリ側でメモリ利用に関する制御を行っても、効果が少ないか、もしくは効果が現れない場合があります。
- 取り扱うメディアデータの種類によっては、内部データ領域の再利用は適用できない場合があります。内部データ領域の再利用が適用可能なメディアデータは、静止画像（アニメーション GIF は含まれません）とサウンド（MFi と SMF）です。
- 「一度きり」の利用指定が行われたメディアデータに対して複数回 use()メソッドを呼び出すと例外が発生します。また、この指定が行われたメディアデータオブジェクトはオリジナルデータを保持していないため、ImageStore などによるネイティブのデータ領域への保存はできないことに注意してください。
- 内部データ領域の再利用では、一度確保されたメモリを再利用時に拡張することはできません。内部データ領域が最初に確保される際に、再利用時にも十分なサイズとなる領域が確保されるように考慮しなければなりません。内部データ領域のサイズの考え方は、メディアデータの種別によって異なります。これらの詳細は、API リファレンスにおける各メディアデータインタフェースの use()メソッドの項を参照してください。
- MediaImage オブジェクトから取得（getImage()メソッド）した Image オブジェクトは、内部的に MediaImage オブジェクトの持つ内部データ領域を参照しています。従って、このようにして取得した Image オブジェクトを使用している間は、対応する MediaImage オブジェクトの内部データ領域が破棄されないようにしなければなりません。

【DoJa-5.0】【DoJa-5.1】

DoJa-5.0 プロファイルにおいては、メディアデータの生成時にストリームを指定した場合（MediaManager クラスのメディアデータ取得メソッドの引数にストリームを指定した場合）は、オリジナルデータ領域は使用されません。この場合は、メディアデータのオブジェクトに対して use()メソッドを呼び出す度にストリームから新たに 1 つ分のデータが読み込まれ、内部データが作成されます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

これに対して DoJa-5.1 プロファイルでは、メディアデータの生成時にストリームを指定した場合もオリジナルデータ領域が使用されるよう仕様が変更されています。DoJa-5.1 プロファイルでは、ストリームが指定されたメディアデータのオブジェクトに対して use()、unuse()メソッドを繰り返し呼び出した場合は、常に初回に読み込んだオリジナルデータから内部データが作成されます。

4.5 イメージ処理

i アプリ実行環境では、プロファイルのバージョンアップに伴いイメージ処理（イメージのエンコードや加工など）に関する機能が強化されています。この項では、それらの機能の概要について解説します。

4.5.1 イメージエンコーダ

DoJa-3.0 プロファイルで追加されたイメージエンコーダは、Image や Canvas に描画された内容をキャプチャーして、特定の画像フォーマットのバイトイメージにエンコードする機能を提供します。この機能を利用することで、画像ファイルの内容を Image 上や Canvas 上で加工し、その結果を再度画像ファイルに変換して ScratchPad に保存したりサーバーに送信したりすることができます。

イメージのエンコードを行うには、ImageEncoder クラスを使用します。static メソッド getEncoder()の引数に画像フォーマット名を指定して呼び出すことで、その画像フォーマットに対応したエンコーダオブジェクトを取得することができます。実際のエンコード処理は、エンコーダオブジェクトの encode()メソッドを使用して行います。

encode()メソッドを呼び出してイメージのエンコードを行うと、その結果として EncodedImage オブジェクトが返されます。EncodedImage はエンコードされた画像データをカプセル化したクラスであり、そのオブジェクトから画像のバイトイメージを取り出すことができます。

以下に、イメージのエンコード処理を行う例を示します。

例: イメージエンコーダの利用

```
import com.nttdocomo.ui.*;
class MainCanvas extends Canvas {

      public void paint(Graphics g) {
            // この Canvas に何かを描画する。
            …
      }

      public void processEvent(int type, int param) {
            // ソフトキーのイベントで、Canvas の描画内容をキャプチャーする。
            if (type==Display.KEY_RELEASED_EVENT && param==Display.KEY_SOFT1) {
                  // 画像フォーマットは JPEG 形式とする。
                  ImageEncoder ie = ImageEncoder.getEncoder("JPEG");
                  // この Canvas の座標(0,0)から、幅 50 ドット、高さ 50 ドットの領域をキャプチャー
                  //する。キャプチャーした結果は EncodedImage オブジェクトとして返される。
                  EncodedImage ei = ie.encode(this, 0, 0, 50, 50);
                  // 画像バイトイメージ（この場合 JPEG）を取り出すためのストリームを取得する。
                  InputStream is = ei.getInputStream();
                  …
            }
      }
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
        ...
}
```

注意事項：

- イメージエンコーダでどの機種でも共通的にサポートされる画像フォーマットは JPEG 形式（JFIF　Baseline エンコーディング）です。JPEG 形式のイメージエンコーダを取得するには、`ImageEncoder.getEncoder()` メソッドの引数に`"JPEG"`を指定します。また、エンコード可能な画像のサイズは、縦横とも携帯電話の描画エリアのサイズまでです。このサイズを超えた画像をエンコードできるかどうかはメーカーにより異なりますので注意してください。
- `Canvas` をキャプチャーする場合、その `Canvas` オブジェクトはカレントフレームに設定されていなければなりません。また、その `Canvas` オブジェクトで IME が起動されている状態ではキャプチャーを行うことはできません。
- `Image` をキャプチャーする場合、その `Image` オブジェクトは `getGraphics()`メソッドを呼び出せるものでなければなりません。つまり、`Image.createImage()`メソッドにより作成された `Image` オブジェクトである必要があります。
- i アプリオプション API、i アプリ拡張 API に属する `Canvas`、`Image` 継承クラスではキャプチャーがサポートされない場合があります。

4.5.2 ピクセル操作

ピクセル操作とは、グラフィックスコンテキスト上に並んでいる各ピクセル（画素）に対してアプリケーションプログラムから直接アクセスを行うための機能です。アプリケーションプログラムは、ピクセル操作として `Graphics` クラスの以下の機能を使用することができます。

- グラフィックスコンテキスト上の、特定ピクセル位置の色情報取得（`getPixel()`系メソッド）
- 特定ピクセル位置への点の描画（`setPixel()`系メソッド）

ピクセル操作機能は、DoJa-2.0 プロファイルで i アプリオプション API として規定されました。DoJa-3.0 プロファイルでは、それらの機能を一部強化した上で i アプリ基本 API に取り入れられています。

注意事項：

- 色情報取得メソッド（`getPixel()`系メソッド）において、範囲外の座標やカレントフレームに設定されていない `Canvas` に対応付けられる `Graphics` オブジェクトを使用した場合は、黒を表す色情報が返されます。この場合の振る舞いは、DoJa-2.0 プロファイルの i アプリオプション API におけるピクセル操作機能から変更されています。
- 色情報取得メソッドは、ダブルバッファリングが有効な状態ではバックバッファから情報を取得してアプリケーションプログラムに返します。このためダブルバッファリングが有効な状態では、ディスプレイで表示されている色とこれらのメソッドで取得する色が一致しているとは限りません。

4.5.3 イメージ回転・反転・拡大縮小表示

イメージの回転・反転表示を行うには、`Graphics.setFlipMode()`メソッドを呼び出します。`Graphics` オブジェクトに対して回転・反転方法を指定してこのメソッドを呼び出すと、以降のイメージ描画メソッド（`drawImage()`および `drawScaledImage()`メソッド）呼び出しでは回転・反転されたイメージが描画されます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

指定可能な回転・反転方法には以下のものがあります。

- Graphics.FLIP_HORIZONTAL : 横方向に鏡像反転することを示します。
- Graphics.FLIP_NONE : 反転表示しないことを示します。
- Graphics.FLIP_ROTATE : 縦横方向に鏡像反転（180 度回転）することを示します。
- Graphics.FLIP_ROTATE_LEFT : 左に 90 度回転することを示します。
- Graphics.FLIP_ROTATE_RIGHT : 右に 90 度回転することを示します。
- Graphics.FLIP_VERTICAL : 縦方向に鏡像反転することを示します。
- Graphics.FLIP_ROTATE_RIGHT_HORIZONTAL : 右に 90 度回転し、横方向に鏡像反転することを示します。
- Graphics.FLIP_ROTATE_RIGHT_VERTICAL : 右に 90 度回転し、縦方向に鏡像反転することを示します。

【DoJa-4.1】

FLIP_ROTATE_RIGHT_HORIZONTAL と FLIP_ROTATE_RIGHT_VERTICAL は、DoJa-4.1 プロファイルにて追加されました。

また、イメージの拡大縮小表示を行うには、Graphics.drawScaledImage()メソッドを呼び出します。このメソッドの引数には、イメージとグラフィックスコンテキスト上の描画位置の他に以下を設定します。

- イメージ描画先グラフィックスコンテキスト上の矩形領域の幅と高さ
- 描画元イメージのうち、描画対象とする矩形領域の左上座標、および幅と高さ

これにより、描画元の矩形領域内の内容が、描画先の矩形領域にちょうど収まるように拡大または縮小されて表示されます。

なお DoJa-3.5 プロファイル以降の Graphics クラスでは、イメージの回転や拡大縮小を任意のアフィン変換行列で表現し、指定することができる drawImage()メソッドが追加されています。このメソッドでは、イメージ上の各座標に以下のような 3 行 3 列のアフィン変換行列を適用し、求められた座標群への描画を行います。以下の式において、x、y は元のイメージ上の座標を、x′、y′は変換後の座標（描画対象における座標）を、m00～m12 はアプリケーションプログラムから指定可能なアフィン変換行列要素を示します。

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \frac{1}{4096} \begin{pmatrix} m00 & m01 & m02 \\ m10 & m11 & m12 \\ 0 & 0 & 4096 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

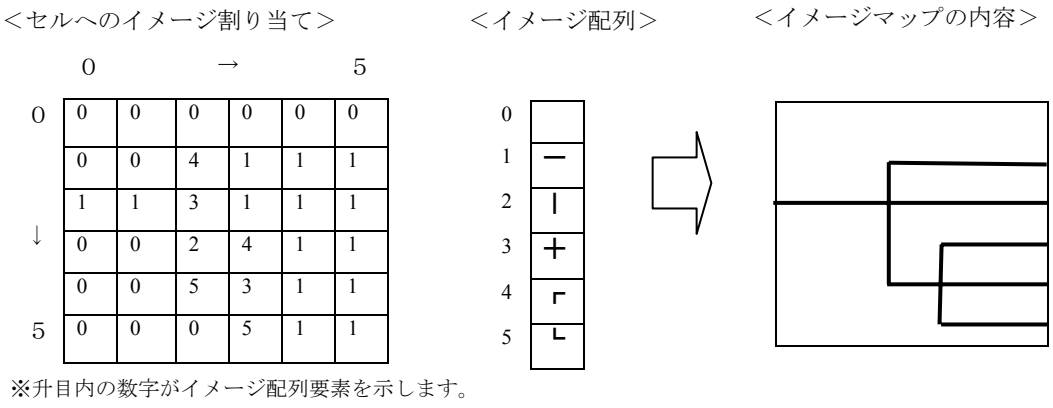
Graphics.setFlipMode()メソッドおよび Graphics.drawScaledImage()メソッドは、DoJa-2.0 プロファイルで i アプリオプション API として規定されました。DoJa-3.0 プロファイルでは、それらの機能を一部強化した上で i アプリ基本 API に取り入れられています。

4.5.4 イメージマップ

イメージマップ（com.nttdocomo.ui.ImageMap クラス）は、小さなイメージを縦横に並べて見かけ上大きなイメージを作成する機能を提供します。

イメージマップにイメージを割り当てる単位（イメージマップ上の升目）をセルと呼びます。イメージマップには、イメージマップを構成するイメージの集合（配列）と、個々のセルにどのイメー

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.
ジを割り当てるかを示すマップデータを指定します。以下は、6×6 個のセルから構成されるイメージマップにイメージを割り当てる様子を示しています。



※升目内の数字がイメージ配列要素を示します。

このようにイメージマップは、数種類の小さな画像の組み合わせで表現される大きな画像を最小限のデータ量で取り扱いたい場合に適しています。

イメージマップは`Graphics.drawImageMap()`メソッドを使用してグラフィックスコンテキスト上に描画します。イメージマップはその全体を描画対象とすることもできますが、イメージマップ上にウィンドウと呼ぶ矩形領域を設定することにより、その領域だけを描画対象とすることもできます。ウィンドウの位置はイメージマップ全体のサイズの範囲内で自由に変更することができるため、ディスプレイ内で大きなイメージマップがスクロールするようなアプリケーションを容易に記述することができます。

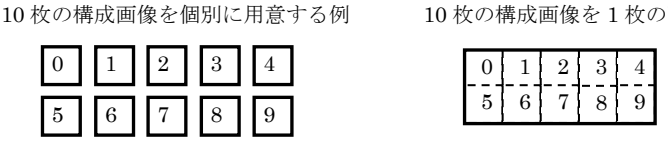
イメージマップは、DoJa-2.0 プロファイルで i アプリオプション API として規定されました。その後 DoJa-3.5 プロファイルにて i アプリ基本 API に取り入れられ、DoJa-4.0 プロファイルで以下の機能強化が図られています。

- **1 つのイメージマップを構成する画像の数（種類）の拡張**

  以前のプロファイルのイメージマップではセルにイメージを割り当てる際のインデックス配列に `byte` 配列を採用していたため、1 つのイメージマップを構成する画像の数は最大 128 種類までに制限されていました。DoJa-4.0 プロファイル以降ではインデックス配列に `int` 配列を使用するコンストラクタ・メソッドが追加されたため、この制限は撤廃されています。なお、インデックス配列への `int` 配列の採用に伴い、インデックス配列を `byte` 配列で指定するコンストラクタ・メソッドは非推奨となっています。

- **イメージマップ構成画像の連結形式での設定**

  以前のプロファイルのイメージマップでは、構成画像は 1 つ 1 つが独立したイメージのデータでなければならず、構成画像の数分イメージファイルを用意する必要がありました。DoJa-4.0 プロファイルではこれに加え、個々の構成画像を連結した大きなイメージで構成画像を与えることができるようになりました。構成画像を 1 つの画像ファイルにまとめることで、データ量をより小さく抑えることができる場合があります。



Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

4.5.5 スプライト

i アプリ実行環境では、イメージ処理機能の 1 つとして簡易なスプライト表示機能をサポートしています。スプライト機能は `com.nttdocomo.ui.Sprite` クラスと `com.nttdocomo.ui.SpriteSet` クラスから構成されます。

`Sprite` クラスは、画面上を移動する 1 つ 1 つのスプライトを表現し、属性としてイメージオブジェクトと表示位置を持ちます。また `Sprite` クラスの持つイメージを実際にディスプレイに表示する際には、以下の操作を行うことができます

- 可視・不可視設定
- 反転表示・回転表示設定

多くの場合、アプリケーションは 1 つの画面上にスプライトを複数表示します。このため、それら複数のスプライトを集合管理するための `SpriteSet` クラスを導入しています。アプリケーションは、同時に使用するスプライトを全てスプライトセットに登録します。画面へのスプライトの表示は、`Graphics.drawSpriteSet()`メソッドにスプライトセットを指定することで行います。`Sprite` オブジェクトをそのまま描画することはできません。

スプライトセットでは、内部で保持しているスプライト間の衝突判定を行うこともできます。衝突判定は、各々のスプライトの位置およびスプライトに登録されているイメージの矩形のサイズに基づいて行われます。

スプライトは、DoJa-2.0 プロファイルで i アプリオプション API として規定されました。その後 DoJa-3.5 プロファイルにて i アプリ基本 API に取り入れられています。ただし、i アプリオプション API において存在していたラスタオペレーション（`Sprite.setRenderMode()`：スプライトのピクセル色と背景のピクセル色の演算）は、 i アプリ基本 API のスプライト機能では定義されていません。

4.5.6 パレット付きイメージ

パレット付きイメージ（`com.nttdocomo.ui.PalettedImage` クラスと `com.nttdocomo.ui.Palette` クラス）とは、アプリケーションプログラムから色パレットを指定することのできるイメージです。

パレット付きイメージは static メソッド `PalettedImage.createPalettedImage()`を、パレット付きイメージとして取り扱いたい GIF データのバイト配列を引数に指定して呼び出すことにより取得します（DoJa-5.0 プロファイル以降では、GIF データ以外に 1、4、8 ビットの Windows BMP 画像を使用することもできます）。

パレット付きイメージからは、`getPalette()`メソッドおよび `setPalette()`メソッドを使用してパレットの参照や設定を行うことができます。パレットの色情報には、色を表す機種固有の整数値（`Graphics.getColorOfRGB()`メソッドなどで返される値）が使用されます。

`PalettedImage` クラスは `Image` クラスを継承しています。ただし `Image` クラスで定義されている `getGraphics()`メソッドは `PalettedImage` オブジェクトに対して呼び出すことはできません。アプリケーションがこのメソッドを呼び出した場合は例外が発生します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.
パレット付きイメージは、DoJa-2.0 プロファイルで i アプリオプション API として規定されました。その後 DoJa-3.5 プロファイルにて i アプリ基本 API に取り入れられています。

【DoJa-5.0】

DoJa-5.0 プロファイルにて、PalettedImage クラスに以下のメソッドが追加されました。

- 縦横サイズのみ指定し、空のイメージを生成するバリエーションの createPalettedImage()メソッド
- 一度生成された PalettedImage オブジェクトに対して画像データの内容を差し替えるための changeData()メソッド

なお、PalettedImage クラスでも透過色指定を行うことができますが、PalettedImage クラスでの透過色指定は、親クラスである Image クラスの透過色指定（次項参照）と異なり、setTransparentIndex()メソッドにパレットのインデックス値を指定することにより行うことに注意してください。

4.5.7 イメージに対する透過色指定と半透明描画指定

DoJa-5.0 プロファイル以降では、i アプリ基本 API に属する機能として、イメージに対する透過色指定、および半透明描画指定をサポートしています。これらの機能は、Image クラスに追加された API にて提供されます。

(1) イメージに対する透過色指定

イメージの描画時に、アプリケーションプログラムが指定した特定の色を透過色として取り扱い、イメージ上その色が配色されている部分を描画対象外とする機能です。

イメージに透過色を指定して描画するには、Image オブジェクトに対して setTransparentColor()メソッドを呼び出すことで透過させたい色を指定した後に、setTransparentEnabled()メソッドに引数 true を指定して呼び出すことでその透過色を有効化します。setTransparentColor()メソッドの引数に指定する色情報には、Graphics.getColorOfName()メソッドや Graphics.getColorOfRGB()メソッドで返される、メーカー固有の色値を使用します。

setTransparentEnabled()メソッドは、setTransparentColor()メソッドを使用して透過色を変更するたびに（透過色を変更した後に）呼び出す必要があることに注意してください。

現在透過色として指定されている色は、getTransparentColor()メソッドで確認することができます。また、setTransparentEnabled()メソッドに引数 false を指定して呼び出すことで、有効となっている透過色指定を無効化することができます。

注意事項：

- setTransparentColor()メソッドに渡す色値にアルファ値の情報が含まれている場合であってもその情報は無視され、常にアルファ値として 255 が指定されているものとして取り扱われます。
- 透過 GIF から作成された Image オブジェクトでは、setTransparentEnabled()メソッドに引数 true を指定して呼び出すまでは GIF データから引き継いだ透過色が有効です。この GIF データから引き継いだ透過色情報は、setTransparentEnabled()に引数 true を指定して呼び出すと失われます。
- i アプリオプション API では、本機能と同様に透過色付きのイメージを取り扱うための API として、com.nttdocomo.opt.ui.TransparentImage クラスが規定されています。DoJa-5.0 プロファイル以降では、透過色を取り扱うには TransparentImage クラスではなく本項で解説した機能を使用するようにしてください。DoJa-5.0 プロファイル以降では、TransparentImage クラスの使用は推奨されません。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

(2) イメージに対する半透明描画指定

イメージの描画において、そのイメージに作用させるアルファ値を指定する機能です。本機能を使用することで、イメージの描画時に描画元（イメージ）と描画先の各ピクセルについての透過演算を行わせることができます。

半透明描画を行うには、`Image` オブジェクトに対して `setAlpha()`メソッドを使用して、アルファ値の設定を行います。このメソッドの引数には 0～255 の範囲の値が設定できます。255 を指定した場合にはそのイメージを完全に不透明なものとして扱い、0 を指定した場合には完全に透明なものとして扱います。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

第 5 章
通信の制御

i アプリ実行環境では、ネットワーキング API が提供されます。これは、Web サーバー側コンポーネントと通信するときに開発者が使用する一連のクラスです。これらのクラスの目的は、Java クライアントアプリケーションの開発をできるだけ簡単にすることにあります（第 2 章の図 5「Generic Connection クラス階層」を参照）。 i アプリ実行環境では、アプリケーションプログラムは HTTP (S)プロトコルを使用して Web サーバーと通信を行うことができます。標準の J2SE ライブラリには、記憶装置との入出力や通信を処理する機能が豊富に揃っています。i アプリ実行環境では、2.2.2 項で述べた CLDC 仕様の Generic Connection フレームワークに基づいて携帯電話と Web サーバーを接続する基本的な機能が提供されます。以下の各節では、HTTP(S)接続インタフェースを使ってサーバーと通信する手順について説明します。

i アプリ実行環境のネットワーキング API と J2SE の `java.net.HttpURLConnection` は、次の点で大きく異なります。

- メッセージコンテンツのタイプを認識し、コンテンツを分析してオブジェクト形式に変換するハンドラをプラグインするためのコンテンツハンドラファクトリという概念はありません。
- メモリサイズを削減するため、サポートするコンテンツタイプを絞り込んでいます。一般的なアプリケーションでは、使用するコンテンツタイプは限られているため、多様なコンテンツタイプをサポートする総合的なフレームワークは採用していません。
- 汎用的な `URLConnection`（`java.net.URLConnection` クラス）はありません。クラスの数を最小限に抑えるため、汎用的な `URLConnection` API は `HttpConnection` インタフェースに統合されています。これによってメモリサイズが減少し、開発者にとっても使用すべき API が簡素になるというメリットがあります。
- `URLStreamHandlerFactory` や `URLStreamHandler` API はありません。 i アプリ実行環境では、使用できるネットワークプロトコルは HTTP、HTTPS に限定されており、多様な通信プロトコルに対応するための総合的なフレームワークは採用していません。これによってメモリサイズが減少し、セキュリティの問題も取り除かれます。
- i アプリ API の `HttpConnection` を使用する上では、Generic Connection フレームワーク API の厳密な呼び出しシーケンスが定義されています。詳細は、5.1 項を参照してください。

注意事項：

- HTTP(S)プロトコルの利用にあたり、PDC 携帯電話上の i アプリ実行環境では GET、POST、HEAD の各リクエストメソッドを使用することができます。これに対し FOMA 携帯電話上の i アプリ実行環境では HEAD メソッドはサポートされず、GET、POST の各メソッドのみ使用可能です。
- PDC 携帯電話では､HTTP プロトコルのバージョンは 1.0 となります｡これに対し FOMA 携帯電話では､HTTP プロトコルのバージョンは 1.1 となります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- 1 つの i アプリから同時に確立可能な通信リンクは 1 つだけです。また、 i アプリのバックグラウンドで動作しているネイティブ機能を含め、パケット通信が行われている状況では、 i アプリから HTTP 通信機能を利用することはできません。

5.1 クライアント/サーバープログラミング

多くのプログラムは、クライアントとサーバーが対になって動作します。 i アプリサービスでは、サーバーはインターネット経由でアクセスするリモートホストシステムであり、クライアントは携帯電話上で動作する Java アプリケーションです。クライアントとサーバーは、通常、IP アドレス（サーバー固有の識別子）とポート番号（接続を特定のプロセスにルーティングするための識別子）を知ることによって相互に通信します。構造をシンプルにするため、ネットワーク接続には CLDC 仕様で定義されている入出力ストリームが使用されます。データの読み書きは、一般的な Java のストリームクラスおよびメソッドを使って行います。そのため、通信対象への接続から入力ストリームを取得するメソッドと出力ストリームを取得するメソッドがそれぞれ用意されています。これによって、クライアントとサーバーは相互に通信することができます。クライアントは、HTTP サーバーから Web ページの内容を入手するなどのインターネットサービスが必要になると、URL を使ってこのサービスにアクセスします。 i アプリ実行環境では、J2SE における `HttpURLConnection` のサブセットとして `HttpConnection` インタフェースが提供されます。次のコードは、`HttpConnection` インタフェース（の実装オブジェクト）の設定と使用の例を示したものです。

例: 接続の確立と動作

```
package spaceinv;

import java.io.*;
import javax.microedition.io.*;
import com.nttdocomo.io.*;

private String[] getScoreList(int score) {
         try {
                  // HTTP 接続（HttpConnection）オブジェクトを取得する。
                  HttpConnection conn = (HttpConnection)Connector.open(
                       "http://backflip.sfbay:8080/scores",Connector.READ_WRITE,
                       true);

                  // リクエストメソッドとコンテンツタイプを設定する。
                  conn.setRequestMethod(HttpConnection.POST);
                  conn.setRequestProperty("Content-Type","text/plain");

                  // 出力ストリームを取り出す。
                  OutputStream out = conn.openOutputStream();

                  // データを書き込む（実際の接続がまだ確立されていないため、
                  // 書き込まれたデータは一時的にバッファに入れられる）。
                  out.write("Highscore-Game: Space Invaders¥n".getBytes());
                  out.write(("Highscore-Name: " + SpaceInvaders.USER.getName() +
                                                        "¥n").getBytes());
                  out.write(("Highscore-Value: " + score + "¥n").getBytes());

                  // 出力ストリームをクローズする。
                  out.close();

                  // リモート資源に実際に接続する。
                  conn.connect();
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
                String[] result = new String[HighScoreScreen.MAX_ENTRIES];

                // 入力ストリームを取り出す。
                InputStream in = conn.openInputStream();
                if (in == null)
                        throw new IOException("high score servlet unreachable");

                // データを読み取る。
                for (int i=0; i<HighScoreScreen.MAX_ENTRIES; i++) {
                        result[i] = readLine(in);
                }

                // 入力ストリームをクローズする。
                in.close();

                // HTTP 接続をクローズする。
                conn.close();

                return result;
        } catch (Exception x) {
                return new String[] {"Scores", "currently", "unavailable."};
        }
}
```

HttpConnection は、HTTP や HTTPS プロトコルのネットワーク資源にアクセスするために使用されるインタフェースです。このインタフェースを実装したクラスのインスタンスは、URL で指定する資源の読み取りと書き込みの両方に使用できます。一般に、URL との接続は、次の手順に従って行います。

1. Connector.open()メソッドを呼び出して接続オブジェクトを作成する。オープンモードとリクエストメソッドは以下のように対応する。

   Connector.READ　　：　GET メソッドおよび HEAD（PDC 携帯電話のみ）メソッドに対応

   Connector.READ_WRITE：　POST メソッドに対応

2. セットアップパラメータを設定する（リクエストメソッドとリクエストヘッダ）。アプリケーションプログラムから設定可能なリクエストヘッダは、If-Modified-Since と Content-Type の 2 つです。
3. HttpConnection.openOutputStream()メソッドを呼び出して、この接続の出力ストリームを取得する。
4. OutputStream.write()メソッドを使って出力ストリームに書き込む。
5. OutputStream.close()メソッドを使って出力ストリームをクローズする。
6. HttpConnection.connect()メソッドを使ってリモートオブジェクトに実際に接続する。この後、必要に応じてレスポンスヘッダを参照することができます。アプリケーションプログラムから参照可能なレスポンスヘッダについては、API リファレンス（HttpConnection.getHeaderField()メソッド）を参照してください。
7. HttpConnection.openInputStream()メソッドを使って入力ストリームを取得する。
8. InputStream.read()メソッドを使って入力ストリームから読み取る。
9. InputStream.close()メソッドを使って入力ストリームをクローズする。
10. HttpConnection.close()メソッドを使って接続をクローズする。

上記の 3～5 は、POST メソッド使用時における HTTP リクエストボディの書き込み処理に相当します。また 7～9 は、POST、GET 両メソッドにおける HTTP レスポンスボディの読み込み処理に相当します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.
これらの処理はそれぞれ必須ではなく必要に応じて行うものですが、処理を行う場合は全体として上記のシーケンスに合致するようにしてください。例えば`HttpConnection.connect()`を呼び出した後で`HttpConnection.openOutputStream()`を呼び出すと、機種によっては例外が発生することがあります。

次の図に、接続の始めから終わりまでの状態遷移を示します。

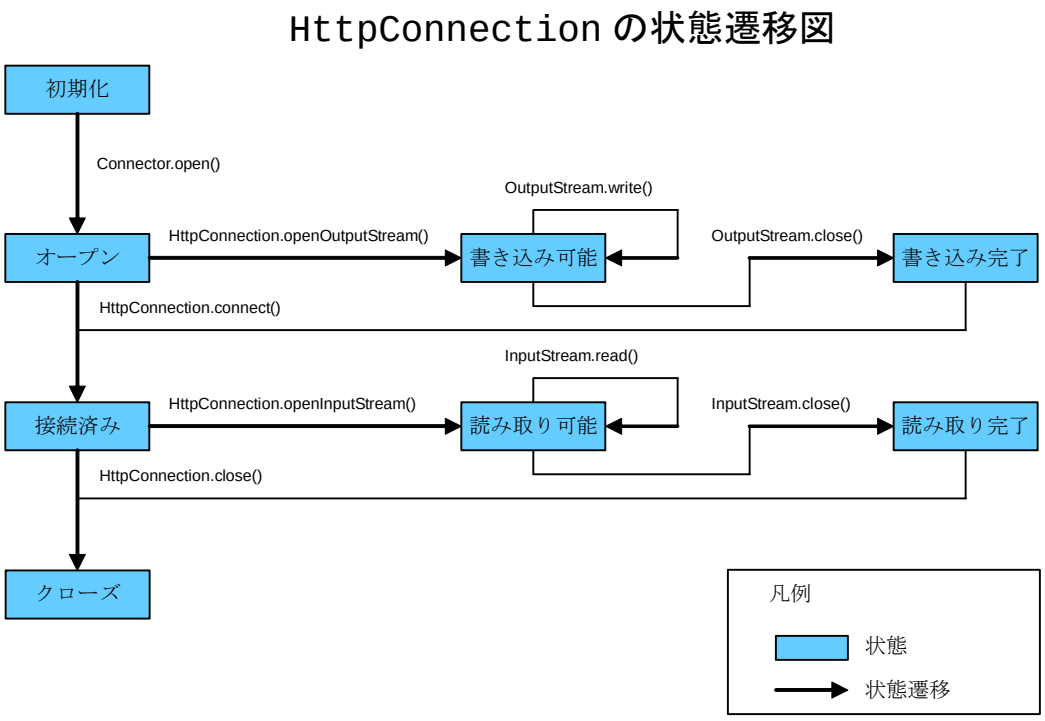


図 8: HttpConnection の状態遷移

注意事項：

- `HttpConnection` に渡す URL には、次の制約があります。
  - IP アドレスはシンボリックでなければなりません（"www.nttdocomo.co.jp"など）。`HttpConnection` では、数値 IP アドレス（"127.0.0.1"など）は使用できません。
  - `HttpConnection` に渡す URL のプロトコル、ホスト名、ポートは、ｉアプリのダウンロード元となった URL（ADF の `PackageURL` キーでの指定）と同じでなければなりません。
- 1 つのｉアプリから同時に接続可能な `HttpConnection` の数は 1 つだけです。
- ｉアプリサービスにおけるクライアント/サーバーシステムでは、クライアント（携帯電話）とサーバー（Web サーバー）の間にｉモードサーバーが介在しています。ｉモードサーバーでは、以下のような異常検出時にレスポンスステータスを 200 としてメッセージを返すことがあります。このような場合、クライアント側ではレスポンスステータスに基づいた正常性確認を行うことができません。クライアント側で厳密な正常性確認を行うには、独自の HTTP レスポンスヘッダの付加や HTTP レスポンスボディの内容による判断を行うようにしてください。
  - ｉモードサーバーのインターネット向けプロキシサーバー輻輳時
  - 公式メニューサイトのメンテナンス時（公式メニューサイトのコンテンツプロバイダより申請のあった場合）

  なお、異常を示すレスポンスステータスが返却された場合、ｉアプリ実行環境は例外をスローします。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- HTTP リクエスト/レスポンスボディについて x-www-form-urlencoded 形式のエンコード､デコードを行うには、それぞれ以下のユーティリティクラスを使用してください。
  - `com.nttdocomo.net.URLEncoder`
  - `com.nttdocomo.net.URLDecoder`
- HTTP(S)通信で送信可能なリクエストボディのサイズ、および受信可能なレスポンスボディのサイズには、プロファイルの世代により以下の制限があります。

プロファイル	PDC 携帯電話		FOMA 携帯電話	
	リクエスト	レスポンス	リクエスト	レスポンス
DoJa-1.x	5K バイト	10K バイト	80K バイト	100K バイト
DoJa-2.x	5K バイト	10K バイト	80K バイト	150K バイト
DoJa-3.x	10K バイト	20K バイト	80K バイト	150K バイト
DoJa-4.x / 5.x	――――	――――	80K バイト	150K バイト

この制限は、アプリケーションプログラムが直接 `HttpConnection` を操作する場合だけでなく、メディアデータを HTTP 通信を介して取得する際にも適用されます。

- i アプリがアクセスする URL には、サーバー側で基本認証（BASIC 認証）の設定が行われている場合があります。このような URL にアクセスした場合、携帯電話は自動的にユーザーID とパスワードの入力を促すダイアログを表示します。アプリケーションプログラムが独自の処理で基本認証の情報（`Authorization` リクエストヘッダなど）をサーバーに送信することはできません。
- i アプリが通信を行っている最中に通話着信があった場合、その動作は携帯電話の i モード着信設定に従います。
- Web サーバーに POST メソッドで HTTP リクエストを送信する場合、HTTP リクエストボディの設定内容に適したコンテンツタイプを指定するようにしてください。コンテンツタイプ（`Content-Type` ヘッダ）の設定には `HttpConnection.setRequestProperty()`メソッドを使用します。また、リクエストボディに書き込むデータが存在しない場合、POST メソッドではなく GET メソッドを使用します。GET メソッド使用時はコンテンツタイプの設定は行わないようにしてください。
- i モードサービスでは、Web サーバーからのレスポンスにはコンテンツ長（`Content-Length` ヘッダ）の設定を必須としています。 i アプリからの HTTP(S)通信においても、Web サーバーからのレスポンスにはコンテンツ長の設定を行うようにしてください。
- i アプリが通信を行っている状態で i アプリの中断（3.6 項参照）が発生すると、通信は中断されます。この場合、 i アプリの実行が再開された時点で例外が発生します。
- `HttpConnection` インタフェースには、`HTTP` レスポンスに含まれるレスポンスヘッダの内容を参照するための `getHeaderField()`メソッドがあります。このメソッドでは`"x-"`で始まるユーザ定義ヘッダも取得することができますが、 i モードにおいては i モードサーバーが携帯電話との通信を制御するために独自のユーザ定義ヘッダを使用する場合があるため、 i アプリではユーザ定義ヘッダは使わないことを強く推奨します。 i モードサーバーが使用しているものと同名のヘッダが i アプリの HTTP 通信で使用された場合、 i モードサーバーはそのヘッダを削除する場合があります。

【DoJa-2.0】

- DoJa-2.0 プロファイル以降に対応した携帯電話では、外部との通信機能を一切停止して、通信を使用しない機能のみ利用可能とするセルフモード機能が搭載されます。ユーザーが携帯電話をセルフモードにしている状態では、 i アプリの起動を行うことはできますが起動された i アプリが HTTP(S)通信を行うことはできません。
- DoJa-2.0 プロファイル以降では、HTTP(S)通信で指定する URL の長さとして、スキーム/パス部に 255 バイト、クエリー部に 512 バイトまで指定することができます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

・ DoJa-1.0 プロファイルでは、HTTP レスポンスのステータスコードについて以下のように定義されています。

- `HttpConnection.HTTP_SERVER_ERROR`（500）
- `HttpConnection.HTTP_INTERNAL_ERROR`（501）

これに対し、DoJa-2.0 プロファイル以降では

- `HttpConnection.HTTP_INTERNAL_ERROR`（500）
- `HttpConnection.HTTP_NOT_IMPLEMENTED`（501）

に改められます。なおコンパイル上の互換性を保持するため、DoJa-2.0 プロファイルでも `HttpConnection.HTTP_SERVER_ERROR`（500）は引き続き定義されます。

【DoJa-4.0】

通信例外が発生したことを示す `ConnectionException` クラスで定義されている例外ステータスについて、DoJa-4.0 にて他の例外クラスと統一するため一部名称の変更がありました。

旧：RESOURCE_BUSY　→　新：BUSY_RESOURCE
旧：NO_RESOURCE　→　新：NO_RESOURCES

互換性保持のため DoJa-4.0 では旧名称も定義されていますが、その使用は推奨されません。

5.1.1 接続の終了

サーバーとクライアント間の接続は、`HttpConnection` インタフェースの `close()`メソッドを使用してクローズします。ただし、このメソッドが実際の物理的な資源を解放するのは、`HttpConnection` から取得されたすべての入出力ストリームがクローズされた後になります。

なお DoJa-2.0 プロファイル以降では、`connect()`メソッドが通信を行っている状態で、他のスレッドからその `HttpConnection` インスタンスに対し `close()`メソッドを呼び出すことで通信を切断させることができます。

注意事項：

- `Connector.open()`を呼び出して取り出すすべての接続（`Connection` インタフェースの実装オブジェクト）と、そこから取得されたすべての入出力ストリームは明示的にクローズする必要があります。これは、`Connector.open()`メソッドで取得した `Connection` 実装オブジェクトを使用した結果として、通信エラーなどにより例外が発生した場合でも同様です。クローズしない場合、資源のリークが発生して新しい接続をオープンすることができなくなる場合がありますので注意してください。

5.2 セッション管理

HTTP プロトコルでは、永続的な接続は維持されません。接続は、クライアントがサーバーとデータを交換する必要があるときだけ確立されます。一方、特定のユーザーにのみ使用を許可するサーバーではユーザーの認証が必要になりますが、接続が行われるたびにユーザーがサーバーにログインすることは現実的ではありません。クライアントが繰り返しサーバーとデータを交換するためには、ユーザーが継続してシステムにログインした状態を保つことが必要となります。

この問題を解決するための手段がセッション管理と呼ばれている手法です。セッション管理の手法では、ユーザーがシステムにログインした際にセッションを一意に識別するための文字列（セッション ID）を作成します。そしてそれ以降は、サーバーとクライアントの間でセッション ID をやり取りして、個々の HTTP 接続の持ち主（ログインユーザー）を識別します。セキュリティを保つた

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.
めに、一定時間アクセスのなかったセッション ID は無効化されます。無効化されたセッション ID の持ち主は、再度システムを利用する際にログインする必要があります。

一般的には、セッション管理の具体的な実装例としてクッキーが広く利用されています。クッキーは HTTP リクエスト/レスポンスヘッダの特定のヘッダにユーザー情報（この場合セッション ID）を設定してクライアントとサーバーの間でセッションを保ちます。しかし i アプリ実行環境では HTTP リクエストヘッダに設定できる情報は `Content-Type` と `If-Modified-Since` の 2 つに制限されており、クッキーを使用することはできません。i アプリ実行環境ではクッキーを使用する代わりに、リクエストの対象 URL か HTTP のボディ部分にセッション ID を設定するようにしてください。

次の例では、クッキーを使用する代わりに HTTP リクエスト/レスポンスボディを使用してセッション ID のやり取りを行っています。

例: セッション管理の実装例

```
package banking;

import java.io.*;
import javax.microedition.io.*;
import com.nttdocomo.io.HttpConnection;

/**
 * このクラスにはバンクサーバーとの通信処理が実装されている。
 */
public class BankingClient {

        private static final String UNAUTHORIZED =
                        "verification of account data failed";
        private String url;
        private String cookie;
        private HttpConnection conn;
        private InputStream in;

        /**
        * このクラスをインスタンス化する。その際、入力された ID とパスワードをチェックする
        * ためにサーバーに初期接続を行う。
        * @param url バンクサーバーへの URL
        * @param acct ユーザーのアカウント名
        * @param pin ユーザーのアカウントパスワード
        * @exception サーバーに接続できないか入力された ID とパスワードが拒否された場合
        * に IOException がスローされる。
        */
        public BankingClient(String url, String acct, String pin)
                                                        throws IOException {
                this.url = url;
                conn = (HttpConnection)Connector.open(url,
                                                Connector.READ_WRITE, true);
                conn.setRequestMethod(HttpConnection.POST);
                conn.setRequestProperty("Content-Type","text/plain");
                OutputStream out = conn.openOutputStream();

                // HTTP リクエストボディにアカウント情報を設定する。
                out.write((VERIFY + " " + acct + " " + pin).getBytes());
                out.close();

                conn.connect();

                in = conn.openInputStream();
                if (getCookie(readLine(in)) == null) {
                        throw new IllegalArgumentException(UNAUTHORIZED);
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
                }
        }

        private String getCookie(String line) {
                String header = "Cookie: ";
                // HTTP レスポンスボディから読み込んだ行よりセッション ID を取得する。
                if (!line.startsWith(header) || line.length() <= header.length()))
                        return null;
                return line.substring(header.length());
        }

       private String contactServer(String cmd) {
                try {
                        // 古い接続が残っていたらそれをクローズする。
                        if (in != null) in.close();
                        if (conn != null) conn.close();

                        // サーバーとの接続を再度オープンする。その際に、この
                        // インスタンスのコンストラクタによる初期接続で得られた
                        // セッション ID を HTTP リクエストボディに設定する。
                        conn = (HttpConnection)Connector.open(url,
                                                Connector.READ_WRITE, true);
                        conn.setRequestMethod(HttpConnection.POST);
                        conn.setRequestProperty("Content-Type","text/plain");
                        OutputStream out = conn.openOutputStream();
                        out.write((cmd + "¥n").getBytes());
                        out.write(("Cookie: " + cookie + "¥n").getBytes());
                        out.close();

                        conn.connect();

                        in = conn.openInputStream();
                        if (in == null) {
                                throw new IOException(UNAUTHORIZED);
                        }

                       return readLine(in);

                       // 以降の処理でこの接続からさらにデータの読み込みが行われるので、
                       // ここでは入力ストリームと接続のクローズは行わない。
                       // 接続のクローズは、このメソッドが次に呼び出されるときか
                       // このクラスの close()メソッドを呼び出すことによって行われる。
                } catch (IOException ioe) {
                        return null;
                }
        }
}
```

5.3 HTTPS による安全な通信

安全な接続が必要なクライアント/サーバーアプリケーションでは、HTTPS プロトコルを使用します。HTTPS サーバーに接続するクライアントプログラムのコードは、URL 文字列のプロトコルセクションに”`http`”の代りに”`https`”と指定する以外、HTTP プロトコルの場合と全く同じです。たとえば、クライアントとサーバーの間でスペースインベーダのハイスコアと名前を安全に送受信する場合、必要な変更は URL 文字列だけです。つまり、サーバー（`backflip.sfbay`）のポート 8080 番において HTTP プロトコルを使用したサービスが提供されており、またポート 7070 番で HTTPS プロトコルを使用した同様のサービスが提供されている場合、URL ”`http://backflip.sfbay:8080/`” を ”`https://backflip.sfbay:7070/`” で置き換えます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

注意事項：

- `HttpConnection` に渡される URL は i アプリがダウンロードされたときの URL と同じプロトコル、ホスト、ポート番号が指定されていなければなりません。したがって、HTTPS を介してダウンロードされた i アプリでなければ HTTPS プロトコルを使用することはできません。これは、HTTPS を介してダウンロードされた i アプリは、HTTP プロトコルは使用できないということでもあります。
- HTTPS 通信で利用する Web サーバーには、 i アプリ対応携帯電話が対応している認証局によって署名されたサーバー証明書をインストールしておく必要があります。 i アプリ対応携帯電話では、以下のルート認証局の証明書に対応しています。

  1) VeriSign 社
     - VeriSign クラス 3 Primary CA ルート証明書
     - VeriSign クラス 3 Primary CA ルート証明書 G2
     - RSA セキュアーサーバー・ルート証明書
  2) Cybertrust（旧 Betrusted Japan）社
     - GTE CyberTrust Root CA (*2)
     - GTE CyberTrust Global Root
     - Baltimore CyberTrust Root (*1)
  3) GlobalSign（旧 GeoTrust）社
     - Equifax Secure Certificate Authority (*1)
     - Equifax Secure eBusiness CA-1 (*1)
     - GeoTrust Global CA (*1)
     - GlobalSign Root CA (*4)
     - GlobalSign Root CA – R2 (*4)
  4) RSA Security 社
     - ValiCert Class 3 Policy Validation Authority (*3)
     - RSA Security 2048 V3 (*3)
  5) セコムトラストシステムズ社
     - Security Communication Root CA 1 (*3)

  (*1) これらの証明書は、DoJa-4.0 プロファイル以降に対応した携帯電話にのみ搭載されています。

  (*2) GTE CyberTrust Root CA 向けに携帯電話に搭載されている証明書は、認証局のサービス終了に伴い 2006 年 2 月 24 日にエクスパイアされます。また、それ以降の機種ではこの認証局用の証明書は搭載されません。

  (*3) これらの証明書は、FOMA 902iS シリーズ以降の携帯電話に順次搭載されています。具体的な搭載機種は、別途 NTT ドコモよりアナウンスされます。

  (*4) これらの証明書は､FOMA 905i シリーズ以降の携帯電話に順次搭載されています｡具体的な搭載機種は、別途 NTT ドコモよりアナウンスされます。

【DoJa-2.0】

DoJa-2.0 プロファイル以降では、SSL 通信実行時に認証エラーなどの SSL エラーが発生した場合、`ConnectionException`（`SSL_ERROR`）がスローされます。例外ステータス `SSL_ERROR` は DoJa-2.0 プロファイルで追加されました。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

第 6 章
テキスト処理

一般にアプリケーションのテキスト処理では、ASCII 文字セット以外に各国固有の文字セットも多く使用されます。i モードサービスでは日本語テキストの符号化方式として Shift-JIS を採用していますが、i アプリ実行環境では CLDC の国際化されたストリーム入出力機構を利用して、Shift-JIS の日本語テキストを容易に取り扱うことができます。

6.1 テキスト処理

Java Virtual Machine では、すべてのテキスト処理は Unicode で行われます。このため、Unicode 以外の形式で符号化されたテキストデータを Java の文字ストリームとして使用するには、これを Unicode に変換する必要があります。CLDC には J2SE と同様に国際化されたストリーム入出力機構が備えられており、Unicode 文字ストリームとそれ以外のテキストのバイトストリームとの間の文字コード変換を行うクラスが `java.io` パッケージで提供されています。`InputStreamReader` クラスはバイトストリームを文字ストリームに、`OutputStreamWriter` クラスは文字ストリームをバイトストリームにそれぞれ変換します。次に、この変換プロセスを示します。

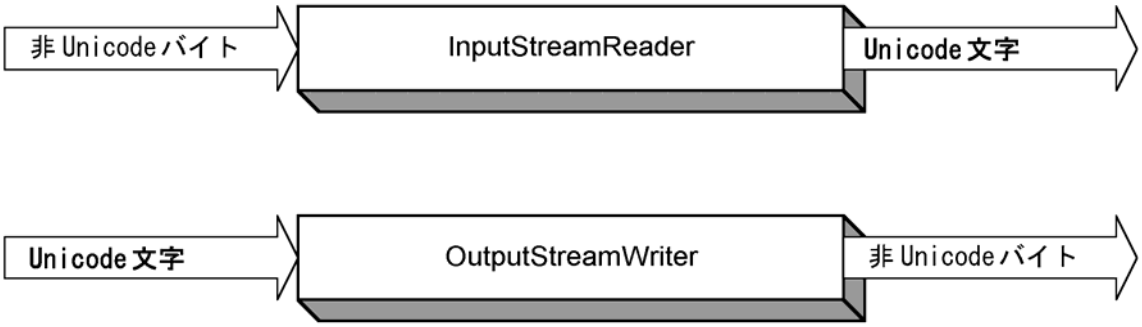


図 9: 入出力ストリームの変換

アプリケーションプログラムでは、`InputStreamReader` や `OutputStreamWriter` クラスを使ってネットワークから入出力するテキストを適切に変換する必要があります。なお、i モードサービスでは日本語テキストの符号化方式として Shift-JIS（SJIS）を採用しており、これらのクラスのデフォルトエンコーディングも SJIS となっています。

次に、これらの変換方法を使用するコード例を示します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

例: SJIS バイトデータを Unicode 文字に変換

```
java.io.*;
import com.nttdocomo.io.*;
import javax.microedition.io.*;

conn = (HttpConnection)Connector.open(url, Connector.READ, true);
conn.setRequestMethod(HttpConnection.GET);
conn.connect();

// SJIS バイトデータを Unicode 文字に変換する。（HTTP レスポンスボディ）
isr = new InputStreamReader (conn.openInputStream()); //デフォルトの符号化は SJIS
 ...
```

例： Unicode 文字を SJIS バイトデータに変換

```
java.io.*;
import com.nttdocomo.io.*;
import javax.microedition.io.*;

conn = (HttpConnection)Connector.open(url, Connector.READ_WRITE, true);
conn.setRequestMethod(HttpConnection.POST);

// Unicode 文字を SJIS バイトデータに変換する。（HTTP リクエストボディ）
osw = new OutputStreamWriter (conn.openOutputStream()); //デフォルトの符号化は SJIS
 ...
conn.connect();
```

【DoJa-3.0】

DoJa-3.0 プロファイルの com.nttdocomo.io パッケージには、以下の 2 つが追加されました。

- バッファ付きリーダ（BufferedReader）
- フォーマッティング出力用ライタ（PrintWriter）

これらのクラスは、J2SE の java.io パッケージに含まれる同名のクラスのサブセットに相当する機能を提供します。

6.2 i アプリ対応携帯電話の絵文字

i アプリ対応携帯電話では、従来の i モードコンテンツでもサポートされている絵文字（i モード用外字セット）を利用することができます。以下に NTT ドコモが定義した、各携帯電話に搭載されている絵文字のコード表（SJIS および Unicode の対応表）を示します。各コードの値は 16 進数の数値です。
なお、ここに記載したフォントは、実際の携帯電話に搭載されたフォントとは若干異なりますのでご注意ください。一般的に、フォントを構成するドット数の制約により、実際の携帯電話に搭載されるフォントはここに記載されたものより粗く表示されます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

＜基本絵文字＞

文字	SJIS	Unicode	タイトル	文字	SJIS	Unicode	タイトル
滴	F89F	E63E	晴れ	阡	F8CA	E669	ホテル
澪	F8A0	E63F	曇り	隩	F8CB	E66A	コンビニ
笏	F8A1	E640	雨	亳	F8CC	E66B	ガソリンスタンド
郤	F8A2	E641	雪	ゐ	F8CD	E66C	駐車場
菓	F8A3	E642	雷	佺	F8CE	E66D	信号
ゐ	F8A4	E643	台風	伜	F8CF	E66E	トイレ
儀	F8A5	E644	霧	豈	F8D0	E66F	レストラン
嬶	F8A6	E645	小雨	巷	F8D1	E670	喫茶店
欟	F8A7	E646	牡羊座	抔	F8D2	E671	バー
炕	F8A8	E647	牡牛座	曦	F8D3	E672	ビール
籠	F8A9	E648	双子座	楾	F8D4	E673	ファーストフード
閒	F8AA	E649	蟹座	珣	F8D5	E674	ブティック
靭	F8AB	E64A	獅子座	鬻	F8D6	E675	美容院
遜	F8AC	E64B	乙女座	釦	F8D7	E676	カラオケ
糲	F8AD	E64C	天秤座	耦	F8D8	E677	映画
	F8AE	E64D	蠍座	週	F8D9	E678	右斜め上
倣	F8AF	E64E	射手座	俍	F8DA	E679	遊園地
兎	F8B0	E64F	山羊座	倜	F8DB	E67A	音楽
則	F8B1	E650	水瓶座	偬	F8DC	E67B	アート
劬	F8B2	E651	魚座	僥	F8DD	E67C	演劇
勺	F8B3	E652	スポーツ	卽	F8DE	E67D	イベント
呀	F8B4	E653	野球	厎	F8DF	E67E	チケット
妥	F8B5	E654	ゴルフ	唗	F8E0	E67F	喫煙
宄	F8B6	E655	テニス	墉	F8E1	E680	禁煙
庾	F8B7	E656	サッカー	奩	F8E2	E681	カメラ
控	F8B8	E657	スキー	奮	F8E3	E682	カバン
摑	F8B9	E658	バスケットボール	奸	F8E4	E683	本
曆	F8BA	E659	モータースポーツ	岨	F8E5	E684	リボン
昞	F8BB	E65A	ポケットベル	嵗	F8E6	E685	プレゼント
暗	F8BC	E65B	電車	嶸	F8E7	E686	バースデー
楉	F8BD	E65C	地下鉄	拘	F8E8	E687	電話
潗	F8BE	E65D	新幹線	敏	F8E9	E688	携帯電話
璘	F8BF	E65E	車（セダン）	昏	F8EA	E689	メモ
玆	F8C0	E65F	車（RV）	昳	F8EB	E68A	TV
眸	F8C1	E660	バス	晌	F8EC	E68B	ゲーム
睎	F8C2	E661	船	晴	F8ED	E68C	CD
瞱	F8C3	E662	飛行機	栢	F8EE	E68D	ハート
穐	F8C4	E663	家	楦	F8EF	E68E	スペード
秤	F8C5	E664	ビル	櫌	F8F0	E68F	ダイヤ
簀	F8C6	E665	郵便局	泮	F8F1	E690	クラブ
纊	F8C7	E666	病院	濼	F8F2	E691	目
褚	F8C8	E667	銀行	熒	F8F3	E692	耳
覲	F8C9	E668	ATM	爀	F8F4	E693	手（グー）

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

文字	SJIS	Unicode	タイトル	文字	SJIS	Unicode	タイトル
牌	F8F5	E694	手（チョキ）	愀	F984	D6DF	フリーダイヤル
眛	F8F6	E695	手（パー）	憧	F985	E6E0	シャープダイヤル
紆	F8F7	E696	右斜め下	所	F986	E6E1	モバQ
緕	F8F8	E697	左斜め上		F987	E6E2	1
蒻	F8F9	E698	足	腑	F988	E6E3	2
衢	F8FA	E699	くつ	杕	F989	E6E4	3
訣	F8FB	E69A	眼鏡	鞠	F98A	E6E5	4
謁	F8FC	E69B	車椅子	滾	F98B	E6E6	5
｜	F940	E69C	新月	焔	F98C	E6E7	6
２	F941	E69D	やや欠け月	猶	F98D	E6E8	7
３	F942	E69E	半月	璢	F98E	E6E9	8
４	F943	E69F	三日月	甄	F98F	E6EA	9
５	F944	E6A0	満月	弥	F990	E6EB	0
６	F945	E6A1	犬	秖	F991	E6EC	黒ハート
７	F946	E6A2	猫	芾	F992	E6ED	揺れるハート
８	F947	E6A3	リゾート	苅	F993	E6EE	失恋
９	F948	E6A4	クリスマス	祚	F994	E6EF	ハートたち（複数ハート）
０	F949	E6A5	左斜め下	祇	F995	E6F0	わーい（嬉しい顔）
Ｇ	F950	E6AC	カチンコ	諸	F996	E6F1	ちっ（怒った顔）
Ｈ	F951	E6AD	ふくろ	通	F997	E6F2	がく～（落胆した顔）
Ｉ	F952	E6AE	ペン	鐘	F998	E6F3	もうやだ～（悲しい顔）
Ｌ	F955	E6B1	人影	付	F999	E6F4	ふらふら
Ｍ	F956	E6B2	いす	☎	F99A	E6F5	グッド（上向き矢印）
Ｎ	F957	E6B3	夜	巳	F99B	E6F6	るんるん
Ｒ	F95B	E6B7	soon	儀	F99C	E6F7	いい気分（温泉）
Ｓ	F95C	E6B8	on	劾	F99D	E6F8	かわいい
Ｔ	F95D	E6B9	end	鯉	F99E	E6F9	キスマーク
Ｕ	F95E	E6BA	時計	鎊	F99F	E6FA	ぴかぴか（新しい）
涙	F972	E6CE	phone to	仿	F9A0	E6FB	ひらめき
璢	F973	E6CF	mail to	住	F9A1	E6FC	むかっ（怒り）
艦	F974	E6D0	fax to	佣	F9A2	E6FD	パンチ
荵	F975	E6D1	ｉモード	律	F9A3	E6FE	爆弾
鄕	F976	E6D2	ｉモード（枠付き）	傭	F9A4	E6FF	ムード
霉	F977	E6D3	メール	倏	F9A5	E700	バッド（下向き矢印）
鵺	F978	E6D4	ドコモ提供	倅	F9A6	E701	眠い（睡眠）
修	F979	E6D5	ドコモポイント	凊	F9A7	E702	exclamation
團	F97A	E6D6	有料	剝	F9A8	E703	exclamation&question
垈	F97B	E6D7	無料	劾	F9A9	E704	exclamation×2
爹	F97C	E6D8	ID	卲	F9AA	E705	どんっ（衝撃）
	F97D	E6D9	パスワード	哆	F9AB	E706	あせあせ（飛び散る汗）
妛	F97E	E6DA	次項有	嗎	F9AC	E707	たらーっ（汗）
嫌	F980	E6DB	クリア	嗑	F9AD	E708	ダッシュ（走り出すさま）
屻	F981	E6DC	サーチ（調べる）	塾	F9AE	E709	ー（長音記号1）
庖	F982	E6DD	NEW	圳	F9AF	E70A	ー（長音記号2）
情	F983	D6DE	位置情報	坻	F9B0	E70B	決定

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

＜拡張絵文字＞

文字	SJIS	Unicode	タイトル	文字	SJIS	Unicode	タイトル
埒	F9B1	E70C	ｉアプリ	滿	F9D7	E732	トレードマーク
埜	F9B2	E70D	ｉアプリ（枠付き）	炎	F9D8	E733	走る人
堯	F9B3	E70E	Ｔシャツ（ボーダー）	昳	F9D9	E734	マル秘
妍	F9B4	E70F	がま口財布	炘	F9DA	E735	リサイクル
媛	F9B5	E710	化粧	焄	F9DB	E736	レジスタードトレードマーク
実	F9B6	E711	ジーンズ	牙	F9DC	E737	危険・警告
穷	F9B7	E712	スノボ	牏	F9DD	E738	禁止
窄	F9B8	E713	チャペル	玓	F9DE	E739	空室・空席・空車
専	F9B9	E714	ドア	珙	F9DF	E73A	合格マーク
峇	F9BA	E715	ドル袋	哪	F9E0	E73B	満室・満席・満車
嶚	F9BB	E716	パソコン	唪	F9E1	E73C	矢印左右
幪	F9BC	E717	ラブレター	囥	F9E2	E73D	矢印上下
建	F9BD	E718	レンチ	圯	F9E3	E73E	学校
彏	F9BE	E719	鉛筆	巏	F9E4	E73F	波
彣	F9BF	E71A	王冠	峆	F9E5	E740	富士山
憲	F9C0	E71B	指輪	帟	F9E6	E741	クローバー
怔	F9C1	E71C	砂時計	弹	F9E7	E742	さくらんぼ
怳	F9C2	E71D	自転車	徒	F9E8	E743	チューリップ
恇	F9C3	E71E	湯のみ	捨	F9E9	E744	バナナ
懲	F9C4	E71F	腕時計	揃	F9EA	E745	りんご
成	F9C5	E720	考えてる顔	时	F9EB	E746	芽
戝	F9C6	E721	ほっとした顔	膮	F9EC	E747	もみじ
捂	F9C7	E722	冷や汗	権	F9ED	E748	桜
拼	F9C8	E723	冷や汗２	俏	F9EE	E749	おにぎり
揭	F9C9	E724	ぷっくっくな顔	偰	F9EF	E74A	ショートケーキ
敷	F9CA	E725	ボケーとした顔	境	F9F0	E74B	とっくり（おちょこ付き）
暖	F9CB	E726	目がハート	璨	F9F1	E74C	どんぶり
曜	F9CC	E727	指で OK	臂	F9F2	E74D	パン
肖	F9CD	E728	あっかんべー	畔	F9F3	E74E	かたつむり
楣	F9CE	E729	ウィンク	癒	F9F4	E74F	ひよこ
樞	F9CF	E72A	うれしい顔	碘	F9F5	E750	ペンギン
橾	F9D0	E72B	がまん顔	礂	F9F6	E751	魚
檉	F9D1	E72C	猫２	祖	F9F7	E752	うまい！
欸	F9D2	E72D	泣き顔	秸	F9F8	E753	ウッシッシ
涵	F9D3	E72E	涙	稞	F9F9	E754	ウマ
溙	F9D4	E72F	NG	穴	F9FA	E755	ブタ
漳	F9D5	E730	クリップ	筠	F9FB	E756	ワイングラス
澧	F9D6	E731	コピーライト	筧	F9FC	E757	げっそり

注意事項：

- 拡張絵文字をサポートしていない携帯電話では、拡張絵文字はスペースなどに置換されて表示されます。どのような文字に置換されるかは機種により異なります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

第 7 章
ScratchPad とリソースの使用

この章では、i アプリ対応携帯電話上でアクセス可能な ScratchPad およびリソースの使用方法について解説します。

ScratchPad とは、携帯電話上に i アプリで使用する永続データを保存するための記憶領域です。またリソースとは、 i アプリの JAR ファイルに同梱されたデータファイル（画像ファイルやサウンドファイルなど）のことです。ScratchPad もリソースも、第 5 章で解説した HTTP 通信と同様に、Generic Connection フレームワークを使用してアクセスを行います。

7.1 ScratchPad に対する読み取りと書き込み

i アプリは、永続データ（長期間の保持・保存が必要なデータ）の主要な格納場所としてサーバーもしくは ScratchPad を使用します。i アプリ実行環境では、i アプリの実行終了後もデータを保存し続け、次にその i アプリが実行されたときにそれを再び利用できるようにするための、ScratchPad と呼ばれる特別な記憶領域が個々の i アプリに与えられます。使用可能な ScratchPad サイズには、3.2 項で解説した制限があります。さらに、ある i アプリに割り当てられた ScratchPad にアクセスできるのは、それを作成した i アプリだけです。

i アプリで ScratchPad を使用するには、ADF の `SPsize` キーを使ってその i アプリの使用する ScratchPad サイズを指定する必要があります。JAM は、指定された ScratchPad サイズと JAR ファイルサイズの両面から、携帯電話にその i アプリをダウンロードできるスペースがあるかどうかを判定します。ADF の記述方法については、15.5.1 項を参照してください。

ScratchPad の記憶モデルには、トランザクションの概念の一部が導入されています。ScratchPad に対する変更は、必ず全部が書き込まれるか、全部が拒否されます。

JAM は、ADF に ScratchPad サイズが指定されている i アプリごとに記憶領域を作成し、それを管理します。JAM が記憶装置にデータを実際に書き込むタイミングは、メーカーのメモリやシステムモデルの最適化方法によって異なります。

ScratchPad へのアクセスは Generic Connection フレームワークを使用して行います。

ScratchPad の URL の書式は以下の通りです。

`scratchpad:///<`*number*`>` `[;pos=<`*start-point*`>` `[,length=<`*access-length*`>]` `]`

アプリケーションに割り当てられた ScratchPad は、ADF の `SPsize` キーの指定により最大 16 個まで分割管理することができます。**<*number*>**には、何番目の ScratchPad にアクセスするかを 0～15 までの数字で指定します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

また、pos オプションの値<*start-point*>には指定された番号の ScratchPad におけるアクセス開始オフセットを、length オプションの値<*access-length*>にはアクセス開始オフセット位置からのバイト数を指定します。pos オプションを指定した場合は、その位置からデータの読み取りまたは書き込みが行えます。また pos オプションとともに length オプションを指定した場合は、そのストリームの長さは length オプションで指定された長さとなります（pos オプションを指定せずに length オプションを指定することはできません）。pos オプション、length オプションとも、それらを適用した結果 ScratchPad の領域（分割管理されている場合は分割された個々の領域）を逸脱する値を指定することはできません。

Connector.open()に対して ScratchPad の URL を指定すると、このメソッドは StreamConnection の実装インスタンスを返します。アプリケーションプログラムは、この StreamConnection インスタンスから入出力ストリームを取り出して ScratchPad へのアクセスを行います。

【DoJa-3.0】

ScratchPad の分割管理および length オプションは、DoJa-3.0 プロファイルにて追加されました。DoJa-2.0 プロファイル以前のプロファイルでは、ScratchPad の URL に指定する<*number*>の値は 0 固定となります。ScratchPad を分割管理するための ADF（SPsize キー）の宣言方法については、15.5.1 項を参照してください。

なお ScratchPad アクセスは、性能やリソースに制約のある携帯電話にとってコストの高い処理です。大きな ScratchPad 領域を使用する i アプリでは、ScratchPad アクセス時の URL に pos オプションおよび length オプションを適切に指定することで、必要なエリアだけを最低限のメモリ使用量でアクセスできるようにしてください。機種によっては、メディアデータを ScratchPad から読み込む場合（MediaManager クラスのメディアデータ取得メソッドに ScratchPad の URL を指定する場合）、その URL に適切な length オプションを設定することでアプリケーションの実行パフォーマンスが向上する場合があります。

次の 2 つの例は、ScratchPad への書き込みと ScratchPad からの読み取りをどのように行うかを示したものです。

例: ScratchPad 領域への書き込み

```
import java.io.*;
import javax.microedition.io.*;

public void save() {
        try {
              OutputStream out = Connector.openOutputStream("scratchpad:///0")
              out.write((hi_score >> 24) & 0xff);
              out.write((hi_score >> 16) & 0xff);
              out.write((hi_score >> 8) & 0xff);
              out.write(hi_score & 0xff);
              out.close();
         } catch (IOException e) {
                 // ScratchPad 入出力の例外処理
         }
}
```

例: ScratchPad 領域の読み取り

```
import java.io.*;
import javax.microedition.io.*;

public void load() {
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
        try {
              InputStream in= Connector.openInputStream("scratchpad:///0")
              hi_score = in.read() << 24;
              hi_score |= (in.read() << 16);
              hi_score |= (in.read() << 8);
              hi_score |= in.read();
              in.close();
         } catch (IOException e) {
                // ScratchPad 入出力の例外処理
         }
}
```

例: ScratchPad 領域の読み取り

```
// ScratchPad の 4 番めのバイトから始まる UTF（つまり Unicode）文字列を取り出す。
// 最初の 4 バイトは他の用途（前の例では 4 バイトの数値）で使用されている。
package spaceinv;

import java.io.*;
import javax.microedition.io.*;

public class UserNameScreen extends Panel implements TimerListener {
        private String name;

        /** このクラスのインスタンスを作成する。 */
        public UserNameScreen() {
                try {
                        DataInputStream in = Connector.openDataInputStream(
                                                "scratchpad:///0;pos=4");
                        name = in.readUTF();
                        in.close();
                } catch (IOException e) {
                        // ScratchPad 入出力の例外処理
                }
                …
         }
         …
}
```

注意事項：

- ScratchPad への `StreamConnection` をオープンする場合、オープンモードとして `Connector.READ`、`Connector.WRITE`、`Connector.READ_WRITE` の各モードが指定できます。`Connector.READ_WRITE` モードでオープンした ScratchPad からは入力ストリーム、出力ストリームのいずれも取得できます。ただし、1 つの `StreamConnection` について、入力ストリームと出力ストリームを同時に有効な状態にすることはできません。一方をオープンする際には、必ず他方がクローズされた状態となるようにしてください。

  【DoJa-2.0】

  `Connector.READ_WRITE` での ScratchPad のオープンは、DoJa-2.0 プロファイル以降でのみ可能です。DoJa-1.0 プロファイルでは、ScratchPad を `Connector.READ_WRITE` でオープンした際の動作はメーカーにより異なります。

- DoJa-3.0 プロファイル以降では、ユーティリティの位置付けとして Jar インフレータ機能（`com.nttdocomo.util.JarInflater` クラス）をサポートしており、Jar 形式で圧縮されたファイルのエントリを i アプリで展開することができます。`JarInflater` に Jar ファイルを ScratchPad からの読み込みストリームとして与える場合、ScratchPad の URL には `pos` オプションおよび `length` オプションを使用して、厳密な Jar ファイル格納位置を指定するようにしてください。これを行わない場合、ファイルフォーマットエラーとして例外がスローされることがあります。

  以下に、`ScratchPad` に格納された Jar ファイルの内容を Jar インフレータ機能を使用して読み込む例を示します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

例: ScratchPad に格納された Jar ファイルの利用

```
// ScratchPad 先頭から 1K バイトの位置に格納された、長さ 1867 バイトの
// Jar ファイルを使用する。
try {
    String url = "scratchpad:///0;pos=1024,length=1867";

    // ScratchPad からの読み込みストリームをオープンする。
    InputStream is = Connector.openInputStream(url);
    // Jar インフレータオブジェクトを生成する。
    JarInflater ji = new JarInflater(is);
    // Jar インフレータオブジェクトが生成された後は、コンストラクタに指定した
    // 読み込みストリームは不要となる。
    is.close();

    // Jar ファイルに含まれるファイルのエントリ名を指定して、目的のファイルに
    // アクセスするための読み込みストリームをオープンする。
    InputStream is2 = ji.getInputStream("datafile.bin");
    for (;;) {
        // 読み込んだデータを処理する。
        …
    }
    // Jar エントリ読み込み用のストリームをクローズする。
    is2.close();
    // Jar インフレータをクローズする。
    ji.close();
} catch (IOException ioe) {
    // ScratchPad アクセスでエラーが発生した場合はこの例外が発生する。
    …
} catch (JarFormatException jfe) {
    // Jar インフレータが Jar のファイルフォーマットエラーを検出した場合は
    // この例外が発生する。
    …
}
```

なお、JarInflater クラスの詳細については API リファレンスを参照してください。

7.2 リソースの読み取り

JAR ファイルに同梱される画像ファイルやサウンドファイルなどのデータファイルをリソースと呼びます。リソースに対しても Generic Connection フレームワークの枠組みでアクセスすることができます。リソースは、読み取り専用でのみオープン可能です。リソースにアクセスするには、Connector.open()メソッドのパラメータで READ モードであることを明示的に指定するか、Connection オブジェクトを得ることなく入力ストリームを得るための簡易 API である Connector.openInputStream()を使用しないと例外がスローされます。static メソッド Connector.open()には、3 つのパラメータがあります。

1) 接続先の URL
2) オープンモード
3) タイムアウト例外要求フラグ（true または false）

リソースにアクセスする場合、最初のパラメータは”resource:///<*jar-relative-path*>”という形式の URL 文字列です。<*jar-relative-path*>の部分には、JAR ファイルに格納されたリソースの相対パスを指定します。2 つめのパラメータには、Connector.READ を指定します（リソースは読み取り専用であるため）。3 つめのパラメータは、HTTP(S)接続および OBEX 接続をオープンする場合を除き、通常 false を指定します。短縮形式の open()メソッド（パラメータが URL だけのもの）はオープンモードが Connector.READ_WRITE になりますので、リソースアクセス時には使用しないでください。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

リソースのURLは、メディアデータにアクセスする際のURL文字列にも指定することができます。これにより、JARファイルに同梱された画像ファイルやサウンドファイルを、アプリケーションから容易に再生することができます。

次に、メディアデータとしてリソースを使用する例を示します。この例では、JARファイルに同梱されたbank.gifというファイルを取り出して、`VisualPresenter`に設定する処理を行っています。

例:リソースの利用

```
package banking;

import com.nttdocomo.io.ConnectionException;
import com.nttdocomo.ui.*;
…
/** welcome 画面を表示する。 */
class WelcomePanel extends Panel {

        /** このクラスの新しいインスタンスを作成する。 */
        public WelcomePanel() {
                final Panel instance = this;
                MediaImage bank;

                VisualPresenter vp1 = new VisualPresenter();
                try {
                        bank = MediaManager.getImage(
                        "resource:///BankingDemo/img/bank.gif");
                        bank.use();
                } catch (ConnectionException ce) {
                        // リソースアクセス時の例外処理
                }
                vp1.setImage(bank);
                …
         }
         …
}
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

7.3 エラー処理

ScratchPad またはリソースの取得先を指定した URL を使用する i アプリでは、次の例外を処理する必要があります。

ConnectionNotFoundException

URL が無効であったときにスローされます。無効とは、指定された URL の形式を i アプリ実行環境が解釈できない、または指定された URL の種別とオープンモードが矛盾していることを指します。

ConnectionException

ステータス： SCRATCHPAD_OVERSIZE

ScratchPad のサイズを越えて書き込みを行おうとした場合にスローされます。

ステータス： ILLEGAL_STATE

接続オブジェクトの状態に反したメソッド呼び出しが行われた場合にスローされます。接続オブジェクトの状態に反したメソッド呼び出しとは、例えば READ モードで作成した接続オブジェクトに対して `openOutputStream()`メソッドを呼び出すようなことを指します。

ステータス： BUSY_RESOURCE

許された数以上のスレッドが同時に ScratchPad やリソースをオープンしようとしたときにスローされます。

その他の例外条件が発生すると、`IOException` または `IOException` のサブクラスがスローされます。

【DoJa-2.0】

DoJa-1.0 プロファイルでは、ScratchPad の末尾を越えて読み取りを行った場合には `EOFException` が発生することが規定されています。これに対し DoJa-2.0 プロファイル以降では、ScratchPad の末尾を越えて読み取りを行った場合の動作も、CLDC で定義されている各ストリームクラスのメソッドの動作規定に従います。例えば、`InputStream.read()`メソッドは、ScratchPad の末尾を越えた位置のバイト読み取りを指示されると-1 を返します。

【DoJa-4.0】

`ConnectionException` クラスで定義されている例外ステータスについて、DoJa-4.0 にて他の例外クラスと統一するため一部名称の変更がありました。

旧：RESOURCE_BUSY → 新：BUSY_RESOURCE

旧：NO_RESOURCE → 新：NO_RESOURCES

互換性保持のため DoJa-4.0 以降では旧名称も定義されていますが、その使用は推奨されません。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

第 8 章
プラットフォームリソースのアクセス

i アプリから携帯電話のプラットフォームリソースを制御するために、com.nttdocomo.ui パッケージに PhoneSystem クラスが定義されています。PhoneSystem クラスには、プラットフォームリソースの属性値を設定する setAttribute()メソッド、プラットフォームリソースの属性値を参照する getAttribute()メソッド、および特定のプラットフォームリソースをその機種上で制御可能かテストする isAvailable()メソッドがあります。プラットフォームリソースを制御するには、プラットフォームリソースタイプとそれに対応する属性値を指定します。i アプリ実行環境で規定されている標準的なプラットフォームリソースタイプには以下のようなものがあります。

- ディスプレイのバックライト（オンまたはオフに切り替え可能）
- バイブレータ（オンまたはオフに切り替え可能）
- 機体の折りたたみ状態（属性値の参照のみ　折りたたみ型、フリップ型の携帯電話のみ有効）
- メール・メッセージ受信状態（属性値の参照のみ）
- 電池残量（属性値の参照のみ）
- 電波強度（属性値の参照のみ）
- マナーモード設定状況（属性値の参照のみ）

これらの属性は全て PhoneSystem クラスで定義されています。

また PhoneSystem クラスには、携帯電話にあらかじめ組み込まれた効果音を発音させるための playSound()メソッドが用意されています。効果音の種類は PhoneSystem クラスに定義されていますが、実際に発音される音はメーカーにより異なります。

【DoJa-2.x】【DoJa-3.x】【DoJa-4.x】【DoJa-5.x】

各プラットフォームリソースの導入プロファイルと位置付けは以下の通りです。表中において、「基本」は i アプリ基本 API の位置付けであることを、「オプション」は i アプリオプションの位置付けであることを示しています。また、「×」はそのプロファイルにおいて機能が規定されていないことを示します。

プラットフォームリソース	DoJa-1.x	DoJa-2.x	DoJa-3.x	DoJa-4.x/5.0	DoJa-5.1
ディスプレイのバックライト	基本	基本	基本	基本	基本
バイブレータ	×	オプション	基本	基本	基本
機体の折りたたみ状態	×	オプション	基本(*1)	基本(*1)	基本(*1)
メール・メッセージ受信状態	×	基本	基本	基本	基本
電池残量	×	×	基本	基本	基本
電波強度	×	×	基本	基本	基本(*2)

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

マナーモード設定状況	×	×	基本	基本	基本
効果音発音	×	オプション	基本	基本	基本
追加キーグループ	×	×	オプション	オプション	オプション
画面可視・不可視状態	×	×	オプション	オプション	オプション
オーディオ出力のサラウンド制御	×	×	×	オプション	オプション
エリア情報	×	×	×	×	基本(*2)

(*1) 機体の折りたたみ状態は、折りたたみ型およびフリップ型の携帯電話でのみ有効です。

(*2) 電波強度リソース（DEV_SERVICEAREA）の利用は、DoJa-5.1 プロファイル以降では推奨されません。DoJa-5.1 プロファイル以降では、電波状態をより包括的に調べることができるエリア情報リソース（DEV_AREAINFO）が新たに規定されました。

注意事項：

- メール・メッセージ受信状態、電池残量、電波強度（および関連機能であるエリア情報）、マナーモード設定状況は、携帯電話のディスプレイ上に表示される各種アイコンの表示内容に連動します。 i アプリがこれらを参照するためには、ADF の GetSysInfo キーにてその i アプリが携帯電話のアイコン情報を参照することを宣言し、かつユーザーがその i アプリに対しアイコン情報の参照を許可する必要があります。ADF の記述方法については、15.5.1 項を参照してください。
- メール・メッセージ受信状態は携帯電話のメール・メッセージ受信アイコンに連動するため、電波状況により携帯電話がメール・メッセージ着信の通知を受けられないようなケースでは必ずしも正しい結果が得られない場合がありますのでご注意ください。
- 画面のバックライトやバイブレータなど、電力消費の大きいプラットフォームリソースを i アプリが長時間オンにし続けた場合、機種によっては省電力のためシステムにより自動的にオフの状態に戻されます。PhoneSystem.getAttribute()メソッドの返すプラットフォームリソース状態には、このようなシステムによる自動的な状態変更の結果も反映されます。
- 非活性化状態の待ち受けアプリケーションからは、バックライトの制御を行うことはできません。
- 携帯電話の設定により携帯電話のキー操作が無効化されている状態では、 i アプリはバイブレータの制御を行うことはできません。
- DoJa-5.1 プロファイルにて、電波状況を包括的に調べることができるエリア情報リソース（DEV_AREAINFO）が新たに規定されました。エリア情報リソースでは、FOMA の HSDPA エリアに在圏しているかどうかといった、詳細な電波状況も調べることができます。ただし HSDPA エリアに在圏している場合でも、実際に HSDPA の高速通信が行えるかどうかは、その時点におけるエリアの基地局の利用状況に依存します。他のユーザが多数 HSDPA を利用している場合など、HSDPA エリアに在圏していても HSDPA の高速通信は行われない場合がありますので注意してください。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

第 9 章
待ち受けアプリケーション

DoJa-2.0 プロファイル以降では携帯電話の待ち受け画面に、カレンダーや画像などを表示するかわりにシステムに常駐した i アプリの実行画面を表示する、待ち受けアプリケーション機能をサポートします。DoJa-1.0 プロファイルでは、i アプリを常駐させておきたいという要求に対しては、通常の i アプリを常に実行し続けるという形で解決する必要がありました。しかしこの方法では、i アプリおよび携帯電話機能自身の操作性を低下させるいくつかの問題があります。DoJa-2.0 プロファイル以降の待ち受けアプリケーション機能では、このような問題点を解決し、より操作性のよい常駐アプリケーションを実現するための機構が採用されています。

9.1 待ち受けアプリケーションの概要

9.1.1 待ち受けアプリケーションの特徴

待ち受けアプリケーション機能を使用して常駐アプリケーションを作成した場合、DoJa-1.0 プロファイルの機能の範囲で常駐アプリケーションを作成した場合と比較して以下のようなメリットがあります。

操作性の向上

通常の i アプリを実行している状態では、携帯電話のキー操作は i アプリのイベントとして受信されます。例えば、携帯電話は待ち受け状態で数字キーを押されるとネイティブオペレーティングシステムのユーザーインタフェース機構（以降ネイティブ OS とします）がイベントを捕捉して自動的に通話発信を行うためのモードに移りますが、i アプリ実行中は数字キーの操作は全てキーイベントとして i アプリに配送されます。従って、従来の手法で作成された常駐アプリケーションのユーザーは、電話を掛けたり Web ブラウザを使用する場合には、手動操作で一旦その i アプリを終了させる必要がありました。また、それらの作業が終了した後は、手動操作で再度 i アプリを起動し直さなければなりませんでした。

これに対し待ち受けアプリケーション機能では、i アプリがキーイベントを受け取るモード（活性化状態と呼びます）と携帯電話のネイティブ OS がキーイベントを受け取るモード（次項参照）をユーザーが簡単なキー操作で切り替えることができます。ユーザーは、通常はネイティブ OS がキーイベントを受け取るモードで i アプリを常駐させておき、必要の都度モードを切り替えて i アプリと対話する、というスタイルで待ち受けアプリケーションを利用します。

なお、待ち受けアプリケーションが常駐している状態でユーザーが通話発信や Web ブラウザなどの他のネイティブ機能を起動すると、待ち受けアプリケーションは自動的に終了または中断します。またユーザーがそれらの作業を終了して再度待ち受け状態に復帰する際、待ち受けアプリケーションは自動的に再起動または再開されます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

電力消費の抑制

通常のｉアプリを実行していたり待ち受けアプリケーションが活性化状態である場合には、それらの実行時間はそのまま電力消費に結びつき、長時間駆動させ続けることは困難です。待ち受けアプリケーションでは活性化状態の他に、

- キーイベントの発生がｉアプリに通知されないだけで、アプリケーションプログラムの処理の続行は可能な非活性化状態
- キーイベントの発生がｉアプリに通知されず、アプリケーションプログラムの処理も停止（スリープ）して余分な電力消費を最低限に抑える休眠状態

の 2 つを使い分けることで、長時間駆動が可能な常駐アプリケーションを設計することができます。

休眠状態ではアプリケーションプログラムの実行は停止しますが、特定のシステムイベント（一定時間が経過した場合など）が発生した際に非活性化状態に復帰し、再びアプリケーションプログラムを実行できるようになります。

メール・メッセージ着信のリアルタイム化

通常のｉアプリを実行している状態では、メールやメッセージの着信通知を受けても、ｉアプリの実行を優先させるためそれらの受信は行いません(*1)。このため、従来の手法で作成された常駐アプリケーションのユーザーは、メール・メッセージの着信に気づかないことがありました。

待ち受けアプリケーションでは、活性化状態では通常のｉアプリ実行中と同様にメール・メッセージの着信通知を受けてもそれらの受信は行いませんが(*1)、非活性化状態または休眠状態では受信処理を行います（その際待ち受けアプリケーションの実行は中断されます）。

なお DoJa-2.0 プロファイル以降に対応した携帯電話では、活性化状態および通常のｉアプリを実行している状態でも、メール・メッセージの着信通知を受けた際は画面上のマークなどによりリアルタイムにユーザーに通知することが規定されています。

(*1)FOMA 携帯電話には、通常のｉアプリを実行している状態や待ち受けアプリケーションの活性化状態でも、それらと並行してメール、メッセージの受信を行えるものもあります。

待ち受けアプリケーションの例として、次のような株価アラームを考えます。

1. ユーザーは、あらかじめ関心のある銘柄の株価をいくつか選択し、それぞれの銘柄でアラームを発生させるための株価条件を設定しておく。
2. アプリケーションプログラムは、通常は 1 時間に 1 度、選択された銘柄の最新の株価をサーバーから取得して画面に表示する。株価条件を超えた銘柄が発生した場合は、ユーザーに音やバイブレータでアラームを伝える。
3. アラームの通知を受けたユーザーは、キー操作によりｉアプリと対話して該当する銘柄の詳細情報を見ることができる。

この例では、2.の処理は非活性化状態および休眠状態で構成することができます。アプリケーションプログラムは、非活性化状態の時にサーバーとの通信や画面の更新を行い、その後 1 時間の間休眠状態に入ることを繰り返します。この処理を行っている間はユーザーのキー操作は全てネイティブ OS の方に伝えられるため、例えばユーザーが数字キーを押すと携帯電話は通話発信のモードに移行します。

また 3.の処理は、ユーザー操作により待ち受けアプリケーションを活性化状態に移行させることで実現します。活性化状態ではユーザーのキー操作はｉアプリに伝えられるため、ユーザーはｉアプリと対話することができます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

注意事項：

- ユーザーによるモード切り替えキー（以降、切り替えキーと呼びます）操作は、非活性化状態または休眠状態から活性化状態への遷移のみサポートされます。その他の状態遷移については、アプリケーションプログラムからメソッドを呼び出すことで行います。
- 切り替えキーのキーアサインはメーカーにより異なります。

9.1.2 待ち受けアプリケーションのユーザー設定

待ち受けアプリケーションは、通常の i アプリのようにユーザーが手動操作で起動することも、待ち受けアプリケーションとして携帯電話に登録することによりシステムに常駐させることもできます。システムに常駐した待ち受けアプリケーションは、ユーザー操作により通話発信や Web ブラウザなどの他のネイティブ機能が起動されると自動的に終了または中断します。またユーザーがそれらの作業を終了して再度待ち受け状態に復帰する際は、自動的に再起動または再開されます。このような起動形態を、通常の i アプリ起動と区別して「待ち受け起動」と呼びます。

ユーザーが待ち受けアプリケーションをダウンロードしてから待ち受け設定を行い、実際に待ち受け起動されるまでの流れはおおよそ以下の通りです。なお、ここで示す流れは典型的なものであり、携帯電話の操作方法の詳細はメーカーにより異なります。

1. **待ち受けアプリケーションのダウンロード**

   待ち受けアプリケーションは、通常の i アプリと同様の手順でダウンロードします。ブラウザで i アプリダウンロード用のリンクをクリックすると、まず ADF の内容がチェックされ、携帯電話上にインストール可能であると判断されれば JAR ファイルがダウンロードされます（詳細は 1.4.1 項を参照してください）。

   なお、ここでダウンロードされる ADF には、その i アプリが待ち受けアプリケーションであることの宣言（`MyConcierge` キー）が含まれます。`MyConcierge` キーおよび ADF については、9.2.4 項および 15.5.1 項を参照してください。

2. **アプリケーション設定**

   i アプリがダウンロードされた後、ADF の内容に応じてアプリケーション設定を行います。アプリケーション設定の項目には、待ち受け設定、ネットワーク許可設定、アイコン情報参照許可設定などがあります。ADF にて待ち受けアプリケーションであることが宣言されている場合、ユーザーは待ち受け設定を行うことができます。

   待ち受けアプリケーションを待ち受け起動可能とするには、待ち受け設定にてその i アプリを携帯電話に登録します。なお、複数の i アプリに対し、同時に待ち受け起動可能とすることはできません。また、ユーザーはここで設定した待ち受け設定内容を後で変更することもできます。

3. **待ち受け状態への復帰と待ち受けアプリケーションの起動**

   2.により i アプリが待ち受け設定された後、ユーザーの終話キー操作などにより携帯電話が待ち受け状態に復帰すると、自動的に待ち受けアプリケーションが待ち受け起動されます。

注意事項：

- 「待ち受け起動」に対する通常の起動とは、待ち受け起動以外の全ての起動形態を指します。具体的には、以下が通常起動に該当します。
  - i アプリ一覧のメニュー操作など、ユーザーの直接的な i アプリ起動指示により i アプリが起動される場合
  - ADF の `LaunchAt` キー（15.5.1 項参照）により i アプリがタイマー起動される場合
  - アプリケーション連携（第 11 章参照）により他のアプリケーションや外部機器から i アプリが起動される場合
  - ダウンロード即起動 i アプリがダウンロード直後に自動的に起動される場合

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

9.2 待ち受けアプリケーションの作成

この項では、待ち受けアプリケーションを作成する上で、通常の i アプリを作成する場合と異なる点、留意すべき点について解説します。

9.2.1 待ち受けアプリケーションクラス

待ち受けアプリケーションでは、 i アプリのメインクラス（ADF の AppClass キーに指定するクラス）は MApplication クラスを継承して作成します。MApplication クラスは com.nttdocomo.ui.IApplication を継承したクラスであり、通常の i アプリのライフサイクル制御に加えて待ち受けアプリケーションの複雑なライフサイクルを制御するためのメソッドを備えています。

MApplication クラスでは、IApplication クラスに加え以下のようなメソッドを使用することができます。

- 状態遷移制御
  - deactivate() : 活性化状態から非活性化状態に遷移します。
  - sleep() : 非活性化状態から休眠状態に遷移します。
  - isActive() : 現在、活性化状態であるか非活性化状態であるかを調べます。
- システムイベント制御
  - setWakeupTimer() : ウェイクアップタイマーを開始します。
  - getWakeupTimer() : ウェイクアップタイマーの残り時間を取得します。
  - resetWakeupTimer() : ウェイクアップタイマーを解除します。
  - setClockTick() : 時計イベントの発生有無を設定します。
  - processSystemEvent() : システムイベントを処理するためのハンドラメソッドです。

MApplication を継承した i アプリは待ち受け起動だけでなく、通常の i アプリと同様にユーザーの手動操作やタイマー起動などによっても起動することが可能です。しかしその場合、アプリケーションプログラムからこれらのメソッドを呼び出すことはできません。待ち受け起動されていない i アプリがこれらのメソッドを呼び出した場合、例外が発生します。

待ち受けアプリケーションのライフサイクルおよび状態遷移については 9.2.2 項で、システムイベントについては 9.2.3 項で解説します。

【DoJa-5.1】

isActive()メソッドは DoJa-5.1 プロファイルにて新設されました。このメソッドも本項に記載した他のメソッドと同様に、待ち受けアプリケーションが待ち受け起動されている場合のみ使用することができます。

9.2.2 待ち受けアプリケーションのライフサイクルと状態遷移

以下に、待ち受けアプリケーションの取る 3 つの状態について解説します。

1. 活性化状態

通常の i アプリを実行している時と同様に、待ち受けアプリケーションがキーイベントおよびソフトキーイベントを受信することのできる状態です。ソフトキーイベントの処理が待ち受けアプリケーション側に委ねられるため、ソフトキーラベルの表示も待ち受けアプリケーション側で行います。

2. 非活性化状態

待ち受けアプリケーションは動作を継続しており、画面の表示も待ち受けアプリケーションの指示通り行われますが、キーイベントおよびソフトキーイベントは待ち受けアプリケーションには通知されず、ネイティブ

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

OS に通知されます。ソフトキーイベントの処理はネイティブ OS 側に委ねられるため、待ち受けアプリケーション側ではソフトキーラベルを設定することはできません。

待ち受けアプリケーションが待ち受け起動された直後は、非活性化状態になっています。

3. 休眠状態

電池の消費を抑えるため、待ち受けアプリケーションの動作は停止しています。画面には、待ち受けアプリケーションが休眠状態に入る直前の描画内容が保持されています。休眠状態への遷移は非活性化状態からのみ可能であり、活性化状態から休眠状態に遷移することはできません。

注意事項：

- 非活性化状態ではキーイベントはネイティブ OS 側に通知されますが、画面の表示は待ち受けアプリケーションに委ねられています。従って、非活性化状態の待ち受けアプリケーションがユーザーにキー操作を促すような画面表示を行っていると、ユーザーが誤った操作を行う可能性もあり得ます。使いやすい待ち受けアプリケーションを作成するためには、非活性化状態における画面表示内容についても留意する必要があります。
- ｉアプリ実行中には、アプリケーションプログラムではなくシステム（ネイティブコンポーネントである JAM）がダイアログを表示することがあります。
  - HTTP 通信時に、BASIC 認証により ID・パスワードが求められた場合
  - HTTPS 通信時に、サーバーの証明書が完全には安全ではないと判断され、ユーザーに継続可否を確認する必要が生じた場合
  - アプリケーション連携機能など、機能の実行にユーザー確認が必要な API をアプリケーションプログラムが呼び出した場合

  待ち受けアプリケーションが非活性化状態にある場合、システムはダイアログを表示することはできません。その場合は例外が発生しますので注意してください。

待ち受けアプリケーション実行中は、9.2.1 項で解説した状態遷移制御メソッドの呼び出しや各種システムイベントの発生などによりこれら 3 つの状態の間の遷移が行われます。

以下に、活性化状態、非活性化状態、休眠状態の間の状態遷移の様子を図示します。

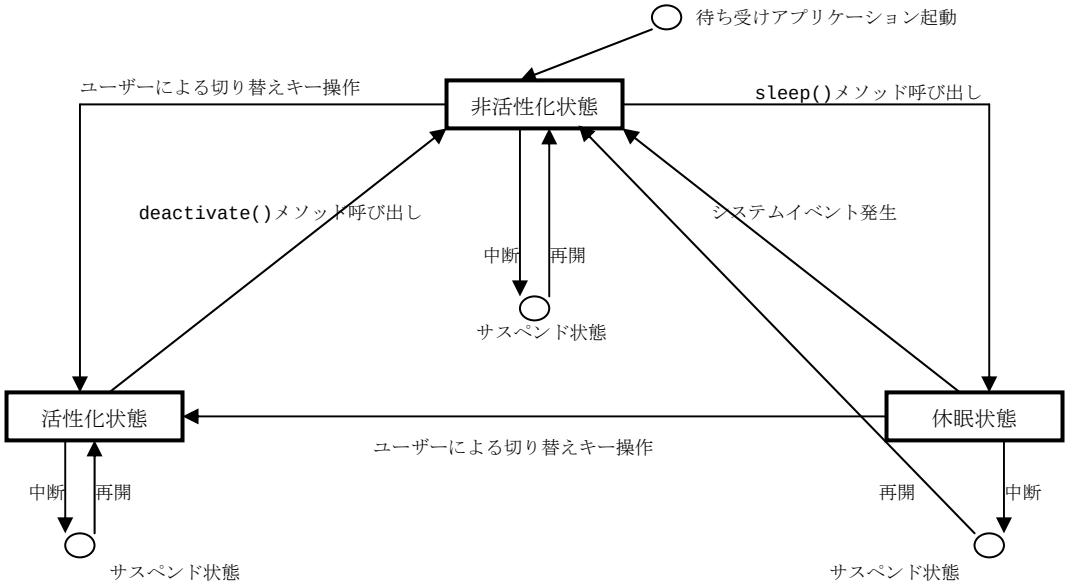


図 10：待ち受けアプリケーションにおける各状態間の状態遷移

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

活性化状態、非活性化状態、休眠状態の間の状態遷移について以下に解説します。

1. 非活性化状態から活性化状態への遷移

待ち受けアプリケーションが非活性化状態のときに、ユーザーにより切り替えキー操作が行われると活性化状態へ遷移します。このとき、待ち受けアプリケーションにはシステムイベント MODE_CHANGED_EVENT が通知されます。

2. 活性化状態から非活性化状態への遷移

活性化状態の待ち受けアプリケーションが、MApplication.deactivate()メソッドを呼び出すことで非活性化状態へ遷移します。非活性化状態の待ち受けアプリケーションがこのメソッドを呼び出しても何もしません。

なお、非活性化状態および休眠状態では切り替えキー操作はキーイベントとして待ち受けアプリケーションに通知されませんが、活性化状態では切り替えキー操作はキーイベント（KEY_IAPP）として待ち受けアプリケーションに通知されます。アプリケーションプログラムは、このキーイベントを受けて MApplication.deactivate()メソッドを呼び出すようにすることで、ユーザーは切り替えキーを状態切り替えのためのトグルキーのように使用することができます。その際、MApplication.deactivate()メソッドの呼び出しは KEY_IAPP のリリースイベントを受けて行うようにしてください。KEY_IAPP のプレスイベントを受けてこのメソッドを呼び出すようにしている場合、メーカーによってはプレスイベントに続いて発生するリリースイベントをシステムが受信し、再度活性化状態に戻ってしまうことがあります。

3. 非活性化状態から休眠状態への遷移

非活性化状態の待ち受けアプリケーションが、MApplication.sleep()メソッドを呼び出すことで休眠状態へ遷移します。その際、イベントキューに未処理のイベントが蓄えられている場合はそれらは全て破棄されます。休眠状態に入った待ち受けアプリケーションは、システムイベントが発生するまで停止します。sleep()メソッドを呼び出したスレッドは、システムイベントが発生すると sleep()メソッドから復帰し、それに続くロジックの処理を再開します。

特定の時間が経過した際に非活性化状態に復帰させたい場合は、sleep()メソッドの前に MApplication.setWakeupTimer()メソッドを呼び出します。またネイティブの時計の時刻変化に合わせて非活性化状態に復帰させたい場合は、あらかじめ MApplication.setClockTick()メソッドにより時計イベントを有効化しておく必要があります。

なお、休眠状態に入ると待ち受けアプリケーションのスレッドは全て停止します。イベント処理や描画メソッド（Canvas.paint()）など、非同期的に実行される可能性のあるスレッドの実行中に休眠状態に入ると、それらのスレッドも停止しますので注意してください。また、アプリケーションプログラムが通信処理を行っている際に休眠状態に入ると、その通信処理は失敗します。

4. 休眠状態から活性化状態への遷移

待ち受けアプリケーションが休眠状態のときに、ユーザーにより切り替えキー操作が行われると活性化状態へ遷移します。このとき、待ち受けアプリケーションにはシステムイベント MODE_CHANGED_EVENT が通知されます。sleep()メソッドを呼び出したスレッドは、休眠状態から活性化状態に遷移すると sleep()メソッドから復帰し、それに続くロジックの処理を再開します。sleep()メソッドからの復帰は、MODE_CHANGED_EVENT の通知に先立って行われます。

なお、システムイベントの詳細については 9.2.3 項を参照してください。

5. 休眠状態から非活性化状態への遷移

待ち受けアプリケーションが休眠状態のときに、MODE_CHANGED_EVENT および FOLD_CHANGED_EVENT（開状態から閉状態への変化）の 2 ケース以外システムイベントが発生すると非活性化状態へ遷移します。sleep()メソッドを呼び出したスレッドは、休眠状態から非活性化状態に遷移すると sleep()メソッドから復帰し、それに続くロジックの処理を再開します。sleep()メソッドからの復帰は、システムイベントの通知に先立って行われます。

なお、システムイベントの詳細については 9.2.3 項を参照してください。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

活性化状態から休眠状態へ遷移することはできません。活性化状態の待ち受けアプリケーションが sleep()メソッドを呼び出すと例外が発生します。活性化状態から休眠状態に遷移させたい場合には、deactivate()メソッドにより一旦非活性化状態に遷移した後で sleep()メソッドを呼び出すようにしてください。

待ち受けアプリケーションは、自身が現在どのような状態にあるかについて把握する必要があります。MODE_CHANGED_EVENT が通知される契機および deactivate()メソッドを呼び出す契機にて、自身の状態を適切に管理するようにしてください。

各状態からの待ち受けアプリケーションの中断は、以下のような契機で発生します。中断された待ち受けアプリケーションが再開される際、通常の i アプリの実行が再開される場合と同様に MApplication.resume()メソッドが呼び出されます。

状態	待ち受けアプリケーションの中断が発生する契機
活性化状態	携帯電話が通話着信を受けた場合など、通常の i アプリ実行より高優先度のイベントが発生した場合
	その待ち受けアプリケーションが、MApplication.launch()メソッド以外の、ユーザー操作（確認ダイアログ操作等）を伴うアプリケーション連携機能のメソッドを呼び出した場合 （このようなメソッドは、非活性化状態で呼び出すことはできません）
非活性化状態 ・休眠状態	携帯電話が通話着信またはメール・メッセージ着信を受けた場合や、スケジュールアラームのアラーム通知が発生した場合など、非活性化状態・休眠状態の i アプリ実行より高優先度のイベントが発生した場合
	ユーザー操作により、通話発信が行われた場合
	ユーザー操作により、パケット通信を使用しないアプリケーション(*1)が起動された場合

(*1)パケット通信を使用しないアプリケーションとは、携帯電話に搭載されているネイティブのスケジューラや電卓アプリケーションなど、そのアプリケーションの中でパケット通信を行う機能を持たないもの（パケット通信を行う可能性がないもの）を指します。

中断時の待ち受けアプリケーションの状態が活性化状態または非活性化状態であれば、再開後の状態は中断前の状態と同じとなります。これに対し、中断時の待ち受けアプリケーションの状態が休眠状態の場合、再開後の状態は非活性化状態に変化しますので注意してください。休眠状態にある待ち受けアプリケーションが中断・再開を受けると、MApplication.sleep()メソッドでブロックされているスレッドは sleep()メソッドから復帰し、後続の処理を再開します。

また、各状態にある待ち受けアプリケーションは、以下のような契機で終了します。終了した待ち受けアプリケーションは、再度待ち受け状態に復帰した際、自動的に再起動されます。

状態	待ち受けアプリケーションが終了する契機
活性化状態	その待ち受けアプリケーションから MApplication.terminate()が呼び出された場合
	その待ち受けアプリケーションが MApplication.launch()メソッドによりアプリケーション連携起動を行った場合(*1)
	ユーザーの強制終了操作が行われた場合
	接続ケーブルにより外部機器が接続された場合
	その待ち受けアプリケーションの内部で、キャッチされない例外・エラーが発生した場合(*2)(*3)
非活性化状態	その待ち受けアプリケーションから MApplication.terminate()が呼び出された場合
	ユーザーのメニュー操作により実行中待ち受けアプリケーションの強制終了が指示された場合

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

	ユーザー操作により、パケット通信を使用するアプリケーション(*4)が起動された場合
	接続ケーブルにより外部機器が接続された場合
	その待ち受けアプリケーションの内部で、キャッチされない例外・エラーが発生した場合(*2)(*3)
	ユーザーのメニュー操作により i モード機能やキー操作のロックが掛けられた場合 この場合、ロックが解除されるまで待ち受けアプリケーションは待ち受け起動されません
休眠状態	ユーザーのメニュー操作により実行中待ち受けアプリケーションの強制終了が指示された場合
	ユーザー操作により、パケット通信を使用するアプリケーション(*4)が起動された場合
	接続ケーブルにより外部機器が接続された場合
	ユーザーのメニュー操作により i モード機能やキー操作のロックが掛けられた場合 この場合、ロックが解除されるまで待ち受けアプリケーションは待ち受け起動されません

(*1)DoJa-4.0 プロファイル以前の携帯電話では、`MApplication.launch()`メソッドによりブラウザを起動しようとした場合を除きます。DoJa-4.0 プロファイル以前の携帯電話では、待ち受け実行されている待ち受けアプリケーションは、ブラウザを起動することはできません。DoJa-4.1 プロファイル以降は、活性化状態であれば待ち受け実行されている待ち受けアプリケーションであってもブラウザを起動することができます（その際は、他のアプリケーションを連携起動した場合と同様に、待ち受けアプリケーションが終了してブラウザが起動します）。

(*2)キャッチされない例外・エラーにより待ち受けアプリケーションが終了する場合、 i アプリ実行環境はその待ち受けアプリケーションを再度待ち受け起動するかどうかをユーザーに問い合わせます。その結果、ユーザーが起動を行わないことを選択した場合、その携帯電話の待ち受けアプリケーション設定は解除されます。待ち受けアプリケーションは、自身のライフサイクルで発生する例外を適切にハンドリングする必要があります。
（ただし、待ち受けアプリケーションが重大なセキュリティ違反を行った場合はユーザー確認が行われることなくその携帯電話の待ち受け設定は解除されます。）

なお DoJa-2.0 プロファイルでは、キャッチされない例外・エラーにより待ち受けアプリケーションが終了すると、ユーザー確認が行われることなくその携帯電話の待ち受け設定は解除されます。

(*3)非活性化状態の待ち受けアプリケーションがキャッチされない例外・エラーにより終了した場合、どの待ち受けアプリケーションがいつ終了したかなどを示す情報が携帯電話に記録されます。ユーザーは、携帯電話のメニュー操作などによりこの記録を参照することができます。

(*4)パケット通信を使用するアプリケーションとは、ブラウザ、メーラ、 i アプリなど、そのアプリケーションの中にパケット通信を行う機能を持つもの（パケット通信を行う可能性があるもの）を指します。なおユーザー操作によるブラウザ、メーラの起動については、ハードウェア上の能力（搭載メモリ量など）によっては非活性化状態、休眠状態の待ち受けアプリケーションを終了させるのではなく中断させる機種もあります。

表のように、実行中の待ち受けアプリケーションはさまざまな外部要因により終了する可能性があります。保存する必要のあるデータが生じた際は、その都度 ScratchPad またはサーバーに保存するようにしてください。

9.2.3 システムイベント

待ち受けアプリケーションでは、ライフサイクルを制御する特殊なイベント（システムイベント）をハンドリングすることができます。システムイベントは、待ち受けアプリケーションのメインクラス（`MApplication` のサブクラス）のインスタンスに対し、`processSystemEvent()`メソッドで通知されます。待ち受けアプリケーションは、このメソッドをオーバーライドすることでシステムイベント処理を実装することができます。このメソッドの第 1 引数で発生したイベントの種別が、第 2 引数で発生したイベントに付随するイベントパラメータが通知されます。

システムイベントのイベント種別は以下の通りです。`MApplication` クラスでは、これらのイベント種別を示す定数が定義されています。

- `MODE_CHANGED_EVENT`

  ユーザーにより切り替えキー操作が行われ、非活性化状態または休眠状態から、活性化状態への遷移が行われたことを示すイベントです。他の状態遷移ではこのイベントは発生しません。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

このイベントでは、イベントパラメータは使用しません。

- CLOCK_TICK_EVENT

  ネイティブの時計の時刻が変化したことを示すイベントです。現在のプロファイルでは、分単位のイベント発生をサポートします。つまり、ネイティブの時計の時刻が毎分正秒（00 秒）を指した時にこのイベントが発生します。このイベントを発生させるためには、MApplication.setClockTick()メソッドを呼び出してこのイベントを有効化する必要があります。

  このイベントは待ち受けアプリケーションがどの状態にある場合でも発生しますが、待ち受けアプリケーションが中断されている期間に発生したイベントは破棄されます。

  イベントパラメータには、その日の 0 時 0 分 0 秒からの経過秒数が渡されます。

- WAKEUP_TIMER_EVENT

  ウェイクアップタイマー（待ち受けアプリケーションが設定した時間が経過したことをシステムイベントとして通知するタイマー）がエクスパイアしたことを示すイベントです。ウェイクアップタイマーは待ち受けアプリケーションが中断・再開されても停止しません。ただし、中断されている期間でタイマーがエクスパイアされた場合、イベントの通知は待ち受けアプリケーションの実行が再開されるまで遅延します。

  ウェイクアップタイマーは、MApplication.setWakeupTimer()メソッドを呼び出すことで動作を開始します（動作は 1 回限りであり、インターバルタイマーとしては動作しません）。また、MApplication.resetWakeupTimer()メソッドを呼び出すと、動作中のウェイクアップタイマーを停止させることができます。ウェイクアップタイマーが動作している状態で MApplication.getWakeupTimer()メソッドを呼び出すと、その時点からウェイクアップタイマーがエクスパイアするまでの時間を取得することができます。

  このイベントは、待ち受けアプリケーションがどの状態にある場合でも発生します。

  このイベントが発生した際、時間通りにイベントが発生した場合には値 0 が、待ち受けアプリケーションの実行中断により遅延してイベントが発生した場合には値 1 が、それぞれイベントパラメータとして渡されます。

- FOLD_CHANGED_EVENT

  折りたたみ型またはフリップ型の携帯電話において、折りたたみ状態またはフリップの開閉状態が変化したことを示すイベントです。このイベントは折りたたみ型またはフリップ型の携帯電話でのみ発生します。

  このイベントは、待ち受けアプリケーションがどの状態にある場合でも発生しますが、待ち受けアプリケーションが中断されている期間に発生したイベントは破棄されます。

  このイベントが発生した際、閉じた状態に変化した場合には値 0 が、開いた状態に変化した場合には値 1 が、それぞれイベントパラメータとして渡されます。

9.2.4 ADF による待ち受けアプリケーションの宣言

待ち受けアプリケーションは、ADF の MyConcierge キーを使用して、自身が待ち受けアプリケーションであることを宣言（値"Yes"を設定）する必要があります。このキーが設定されていない i アプリは、待ち受けアプリケーションとして携帯電話に登録することはできません。

なお、携帯電話にダウンロードされている通常の i アプリを、新たに MyConcierge キーが設定された ADF を使用して待ち受けアプリケーションとして更新することはできません。またその逆に、携帯電話にダウンロードされている待ち受けアプリケーションを、新たに MyConcierge キーが削除された ADF を使用して通常の i アプリとして更新することもできません。このような場合は、携帯電話にインストールされた i アプリを一旦削除し、再度新たな i アプリとしてダウンロードし直す必要があります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

9.3 待ち受けアプリケーション実行中の他機能の起動

待ち受けアプリケーションの実行中に携帯電話の他機能を起動させようとする場合、待ち受けアプリケーションが活性化状態でなければ原則的に他機能の方が優先されます。以下に、待ち受けアプリケーション実行中に他機能を起動しようとした場合の、待ち受けアプリケーションの振る舞いを示します。

起動させる機能	待ち受けアプリケーションの振る舞い	
	活性化状態の場合	非活性化状態または休眠状態の場合
ユーザー操作による他機能の起動		
ブラウザ起動	不可 (*1)	待ち受けアプリケーションを終了し、ブラウザを起動します。ユーザーが作業を終えて待ち受け状態に戻ると、待ち受けアプリケーションは再起動されます。
メーラ起動	不可 (*1)	待ち受けアプリケーションを終了し、メーラを起動します。ユーザーが作業を終えて待ち受け状態に戻ると、待ち受けアプリケーションは再起動されます。
通話発信	不可 (*1)	待ち受けアプリケーションを中断し、通話発信動作に移ります。通話が終了すると待ち受けアプリケーションは再開され、非活性化状態になります。
iアプリ起動	不可 (*1)	待ち受けアプリケーションを終了し、指定されたiアプリを起動します。ユーザーが作業を終えて待ち受け状態に戻ると、待ち受けアプリケーションは再起動されます。
アプリケーション連携による、待ち受けアプリケーション内部からの他機能の起動		
ブラウザ起動	DoJa-4.0 プロファイル以前：不可 DoJa-4.1 プロファイル以降 : 待ち受けアプリケーションを終了し、ブラウザを起動します。ユーザーが作業を終えて待ち受け状態に戻ると、待ち受けアプリケーションは再起動されます。 (*2)	不可
通話発信	待ち受けアプリケーションを中断し、通話発信動作に移ります。通話が終了すると待ち受けアプリケーションは再開され、活性化状態になります。	不可 (*3)
iアプリ起動	待ち受けアプリケーションを終了し、指定されたiアプリを起動します。ユーザーが作業を終えて待ち受け状態に戻ると、待ち受けアプリケーションは再起動されます。	不可（*3）
iアプリ更新機能起動	待ち受けアプリケーションを終了し、JAM（iアプリ更新機能）を起動します。ユーザーが作業を終えて待ち受け状態に戻ると、待ち受けアプリケーションは再起動されます。（*4）	不可（*3）
ユーザー操作を伴うネイティブ機能の呼び出し	待ち受けアプリケーションを中断し、ネイティブ機能を呼び出します。ユーザーが操作を終えてネイティブ機能を終了すると待ち受けアプリケーションは再開され、活性化状態になります。	不可（*3）
上記以外の方法による他機能の起動		
通話着信	待ち受けアプリケーションを中断し、通話着信動作に移ります。通話が終了すると待ち受けアプリケーションは再開され、活性化状態になります。	待ち受けアプリケーションを中断し、通話着信動作に移ります。通話が終了すると待ち受けアプリケーションは再開され、非活性化状態になります。
メール・メッセージ着信	メール・メッセージの受信は行わず、待ち受けアプリケーションの実行を継続します。携帯電話は画面にマークを表示し、iモードサーバーに未受信のメールがあることをユーザーに知らせます。なお、DoJa-3.5 プロファイル以降に対応したFOMA 携帯電話では、活性化状態の待ち受けアプリケーションを実行しながらバックグラウンドでメール・メッセージの受信を行います。	待ち受けアプリケーションを中断し、メール・メッセージ受信動作に移ります。受信が終了すると待ち受けアプリケーションは再開され、非活性化状態になります。 ただし、非活性化状態の待ち受けアプリケーションがHTTP 通信処理を行っている状態でメール・メッセージが着信した場合は、活性化状態での動作に準じます。
iアプリのタイマー起動	iアプリのタイマー起動は行わず、待ち受けアプリケーションの実行を継続します。ユーザーはメニュー操作などにより、タイマー起動できなかったiアプリがあることを知ることができます。	待ち受けアプリケーションを終了し、iアプリをタイマー起動します。タイマー起動されたiアプリが終了して待ち受け状態に戻ると、待ち受けアプリケーションは再起動されます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

	アラームやスケジューラなど、スケジュール起動されるネイティブアプリケーションの自動起動	待ち受けアプリケーションを中断し、ネイティブアプリケーションが起動されます。ネイティブアプリケーションが終了すると待ち受けアプリケーションは再開され、活性化状態になります。	待ち受けアプリケーションを中断し、ネイティブアプリケーションが起動されます。ネイティブアプリケーションが終了すると待ち受けアプリケーションは再開され、非活性化状態になります。

(*1) 待ち受けアプリケーションが活性化状態である場合、キーイベントは待ち受けアプリケーションに通知されるため、ユーザーはこれらの操作を行うことはできません。ただし複数タスクを同時に実行可能な FOMA 携帯電話では、待ち受けアプリケーションが活性化状態であっても、タスク切り替え機能を使用してこれらの機能を使用できる場合があります。

(*2) DoJa-4.0 プロファイル以前の携帯電話では、待ち受け実行されている待ち受けアプリケーションは、ブラウザを起動することはできません。DoJa-4.1 プロファイル以降は、活性化状態であれば待ち受け実行されている待ち受けアプリケーションであってもブラウザを起動することができます。

(*3) 非活性化状態の待ち受けアプリケーションは、これらの機能を起動することはできません。

(*4) i アプリ更新機能の起動による i アプリの更新が完了すると、その i アプリは自動的に起動されます。待ち受けアプリケーションの場合、この起動は通常起動として行われます。通常起動された待ち受けアプリケーションが終了し、待ち受け状態に復帰した後でその待ち受けアプリケーションは待ち受け起動されます。

9.4 i アプリ API 利用上の注意点

待ち受けアプリケーションが待ち受け起動されている場合、その特性から、i アプリ API を利用する上で通常の i アプリとは異なる注意点があります。本項では、これらの注意点について解説します。

9.4.1 ユーザーインタフェース

- フレームに設定されたソフトキーラベルは、待ち受けアプリケーションが非活性化状態または休眠状態にある間は表示されません。これらの状態では、ソフトキーラベル表示領域はネイティブ OS が使用します。待ち受けアプリケーションが活性化状態に移行した際に、カレントフレームに設定されているソフトキーラベルがソフトキーラベル表示領域に表示されます。
- `Panel` 使用時、非活性化状態にある待ち受けアプリケーションからフォーカスに関する操作（フォーカスマネージャの設定やコンポーネント間のフォーカス移動）を行うことはできません。非活性化状態の待ち受けアプリケーションがフォーカス操作のメソッドを呼び出しても、そのメソッドは何もせずアプリケーションプログラムに復帰します。
- `Canvas` 使用時、非活性化状態ではキーパッドの状態を取得（`Canvas.getKeyPadState()`メソッド）することはできません。非活性化状態の待ち受けアプリケーションがこのメソッドを呼び出すと、何もキーが押されていない状態を示す値 0 が返されます。
- 高レベルイベント、低レベルイベントそれぞれについて、非活性化状態の待ち受けアプリケーションへの通知の有無は以下の通りです。原則的にキー操作を伴うイベントは、非活性化状態の待ち受けアプリケーションには通知されません。

高レベルイベント

イベントの種類		通知の有無	備考
コンポーネントイベント			
	BUTTON_PRESSED	×	
	SELECTION_CHANGED	○	`ListBox` クラスの `select()`、`deselect()`メソッドにより選択状態が変化した場合に通知されます。
	TEXT_CHANGED	○	`TextBox` クラスの `setText()`メソッドによりテキストが再設定された場合に通知されます。
キーイベント		×	

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

ソフトキーイベント	×	
メディアイベント	○	
タイマーイベント	○	

低レベルイベント

イベントの種類	通知の有無	備考
キープレスイベント	×	ソフトキー操作を含みます。
キーリリースイベント	×	ソフトキー操作を含みます。
レジュームイベント	○	
アップデートイベント	○	
タイマーイベント	○	

- `Ticker` は、活性化状態および非活性化状態ではスクロール動作を行いますが、休眠状態に入るとスクロール動作は停止します。休眠状態から活性化状態または非活性化状態に遷移することで、再度スクロール動作が行われるようになります。
- プレゼンタ（`VisualPresenter`、`AudioPresenter`）オブジェクトは、活性化状態および非活性化状態では再生処理を行いますが、再生中のプレゼンタオブジェクトは待ち受けアプリケーションが休眠状態に入ると停止します。その際、再生停止イベントは発生しません。また、待ち受けアプリケーションが休眠状態から活性化状態または非活性化状態に遷移しても、自動的に再生を再開することはありません。
- タイマー（`Timer` および `ShortTimer`）は、待ち受けアプリケーションが活性化状態および非活性化状態にある場合に動作します。タイマーが動作している待ち受けアプリケーションが休眠状態に遷移すると、使用中のタイマーは停止します（`stop()`メソッドが呼び出された状態）。その後、待ち受けアプリケーションが休眠状態から活性化状態または非活性化状態に遷移しても、自動的にタイマーの動作を再開することはありません。
- `Canvas` からの IME 起動（`Canvas.imeOn()`メソッドの呼び出し）は、非活性化状態の待ち受けアプリケーションから行うことはできません。非活性化状態の待ち受けアプリケーションが IME を起動しようとすると例外が発生します。

9.4.2 入出力

- 非活性化状態の待ち受けアプリケーションが HTTP(S)通信を行っている場合、通話着信を受けた場合だけではなく、ユーザー操作により通話発信が行われた場合にも通信が中断されます。この場合、通話着信により通信が中断された場合と同様に、通信処理のリトライを行う必要があります。 i アプリの中断、再開が与える影響については、3.6 項を併せて参照してください。
- 待ち受けアプリケーションは、活性化状態および非活性化状態の場合のみ HTTP(S)通信を行うことができます。あるスレッドが HTTP(S)通信を行っているときに他のスレッドが `MApplication.sleep()`メソッドを呼び出して休眠状態に入ると、その通信は中断され例外が発生します。その後、待ち受けアプリケーションが休眠状態から他の状態に復帰すると、通信を行っていたスレッドは通信の例外処理から動作を再開します。
- ScratchPad が不揮発メモリにデータを書き込んでいるときに他のスレッドが `MApplication.sleep()`メソッドを呼び出して休眠状態に入ろうとすると、実際の休眠状態への遷移は ScratchPad への書き込みが完了するまで遅延します。なお、ScratchPad をオープンしたまま待ち受けアプリケーションが休眠状態に移行した場合は、活性化状態または非活性化状態に復帰した後も ScratchPad はオープンされたままの状態を保っており、継続してデータの書き込みまたは読み込みを行うことができます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- OBEX 外部接続による外部機器との通信は、待ち受けアプリケーションが活性化状態にある場合でのみ可能です。OBEX 外部接続により通信を行っている待ち受けアプリケーションが非活性化状態に遷移すると、通信は中断され例外が発生します。これは、待ち受けアプリケーションが OBEX クライアントとして動作している場合、OBEX サーバーとして動作している場合の双方に適用されます。

9.4.3 ハードウェア制御

- 非活性化状態にある待ち受けアプリケーションは、バックライト制御を行うことはできません。非活性化状態にある待ち受けアプリケーションが `PhoneSystem.setAttribute()`メソッドでバックライト状態の変更を指示しても、このメソッドは何も行わず復帰します。
- 活性化状態にある待ち受けアプリケーションはバックライト制御を行うことができますが、待ち受けアプリケーションにより点灯されたバックライトは、一定時間経過後システムにより自動的に消灯されます。
- バイブレータ制御は、待ち受けアプリケーションが活性化状態または非活性化状態にある場合に行うことができます。バイブレータがオンになっている状態で待ち受けアプリケーションが休眠状態に遷移すると、バイブレータは停止します。待ち受けアプリケーションが活性化状態や非活性化状態に復帰しても、自動的にバイブレータ動作を再開することはありません。
- その他、ユーザーがメニュー操作などで状態を設定できるハードウェアリソースは、原則的に待ち受けアプリケーションから状態を変更しようとしてもユーザーの設定が優先されます。

9.4.4 アプリケーション連携

- 非活性化状態の待ち受けアプリケーションは、ユーザー操作（`Phone.call()`メソッドにおける通話発信の許可確認など）を伴うアプリケーション連携機能のメソッドを呼び出すことはできません。非活性化状態の待ち受けアプリケーションがこれらのメソッドを呼び出した場合は例外が発生します。
- DoJa-4.0 プロファイル以前の i アプリ実行環境では、待ち受けアプリケーションが待ち受け起動されている場合、どのような状態であっても `MApplication.launch()`メソッドを使用してブラウザを起動することはできません。待ち受け起動された待ち受けアプリケーションがブラウザを起動しようとすると例外が発生します。

  これに対し DoJa-4.1 プロファイル以降の i アプリ実行環境では、待ち受け起動された待ち受けアプリケーションであっても、活性化状態であれば `MApplication.launch()`メソッドによるブラウザ起動を行うことができます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

第 10 章
OBEX 外部接続

DoJa-2.0 プロファイル以降では、赤外線ポートを介して i アプリと外部機器を接続し、通信を行うための OBEX 外部接続機能が i アプリ基本 API としてサポートされます。外部機器には DoJa-2.0 プロファイル以降に対応した携帯電話自身も含まれ、また i アプリ仕様では OBEX 外部接続におけるクライアント API およびサーバーAPI の両方をサポートするため、携帯電話間で名刺やスケジュールなどの小さなデータを交換しあう i アプリを容易に開発することができます。

本章では、 i アプリ実行環境における OBEX クライアント API、OBEX サーバーAPI、および携帯電話以外の外部機器をターゲットとしたシステムを開発する際のヒントについて解説します。

なお、OBEX 外部接続機能は IrDA 規格に基づいて実装されますが、本書では IrDA 規格そのものに関する解説は行いません。これらの規格に関する詳細は、Infrared Data Association の制定している各種規格書を参照してください。なお、DoJa-2.0 プロファイルから現在のプロファイルまでにおける OBEX 外部接続機能は、以下のバージョンの各規格に準拠しています。

- Infrared Data Association "Link Manager Protocol version 1.1"
- Infrared Data Association "Serial Infrared Physical Layer Specification version 1.3"
- Infrared Data Association "Tiny TP: A Flow-Control Mechanism for use with IrLMP version 1.1"
- Infrared Data Association "Object Exchange Protocol IrOBEX version 1.2"
- Infrared Data Association "Specification for Ir Mobile Communications(IrMC) version 1.1"
- Infrared Data Association "Serial Infrared Link Access Protocol version 1.1"

10.1 OBEX によるデータ送受信

OBEX（IrOBEX）は、赤外線ポートを介したデータ送受信の手順を「オブジェクトの交換」という形式で規格化したものです。ここでいうオブジェクトとは、一塊となることで意味を持つ、例えばファイルなどの一般的なデータ実体のことを指します。

OBEX 自体はデータ送受信の手順（プロトコル）を規格化したものであり API を規格化したものではありません。 i アプリ仕様では、Generic Connection フレームワーク上に、OBEX によるデータ送受信を取り扱うための API を規定しています。

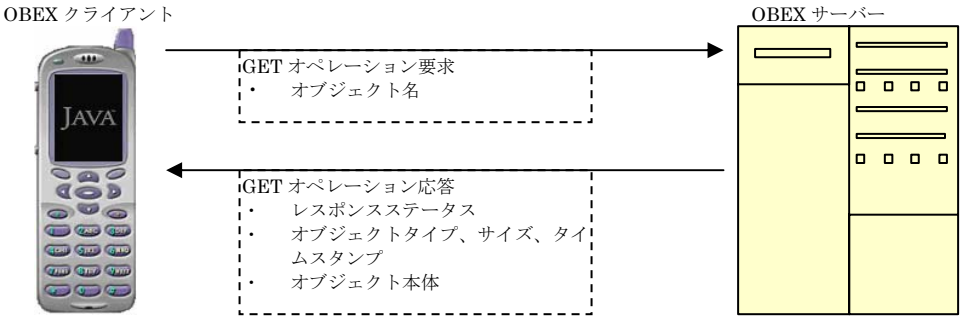
OBEX の仕様は HTTP を参考として策定されており、OBEX クライアントが送信するリクエストに対して OBEX サーバーがレスポンスを返すクライアント/サーバーモデルに従っています。OBEX クライアントが OBEX サーバーにオペレーション（HTTP におけるメソッドに相当）の処理を要求し、OBEX サーバーはその結果としてレスポンスステータスや、処理結果としてのオブジェクトの内容を OBEX クライアントに返します。クライアントとサーバーの間でオブジェクトを送受するためのオペレーションとして、 i アプリ実行環境は以下の 2 つを提供しています。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

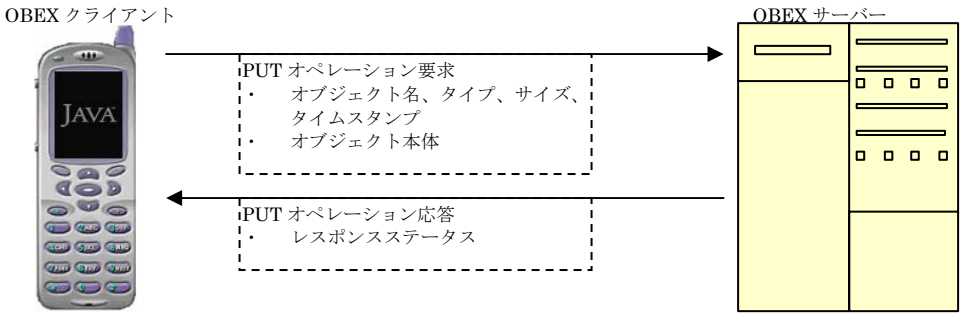
- GET オペレーション： OBEX クライアントが OBEX サーバーにオブジェクト送信を要求する
- PUT オペレーション： OBEX クライアントから OBEX サーバーにオブジェクトを送信する

プロトコル自体がバイナリ形式で表現されることを除き、リクエストおよびレスポンスがヘッダやオブジェクト本体（ボディ）から構成されること、および規定されているレスポンスステータスの内容などは HTTP と類似しています。

OBEX 規格では、赤外線通信のパケット長の制限などに起因するオブジェクトの分割や再構成といった処理を行うための詳細な手順が規定されています。しかし i アプリ実行環境の提供する API がこれらの複雑な規定を隠蔽するため、開発者はより容易に赤外線通信を行う i アプリを開発することができます。



(1) GET オペレーションによるオブジェクト取得

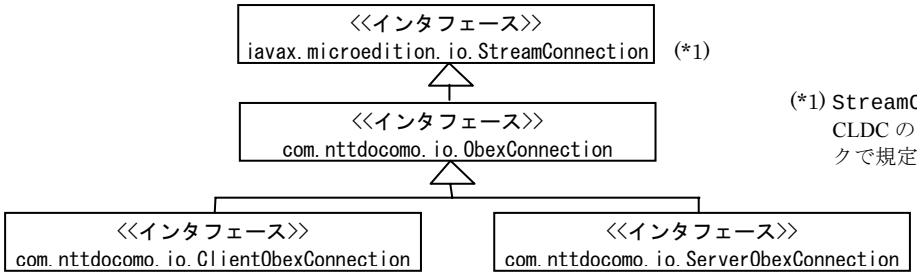


(2) PUT オペレーションによるオブジェクト送信

図 11: OBEX クライアントと OBEX サーバーの利用イメージ

10.2 OBEX 外部接続 API

OBEX 外部接続 API は、Generic Connection フレームワーク上に構築された、 i アプリ独自の API セットとして提供されます。OBEX 外部接続 API の構成を以下に図示します。



(*1) StreamConnection インタフェースは、CLDC の Generic Connection フレームワークで規定されているインタフェースです。

図 12: OBEX 外部接続 API の構成

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

ObexConnectionインタフェースはOBEXクライアントとOBEXサーバーで共通に使用される定数やアクセスメソッドを定義するインタフェースであり、それを継承するサブインタフェースとしてClientObexConnectionおよびServerObexConnectionが用意されています。これら2つのサブインタフェースでは、OBEXクライアントまたはOBEXサーバーに固有のメソッドが定義されています。

OBEX外部接続で使用するコネクションオブジェクトはHTTP通信やScratchPadの場合と同様Connector.open()メソッドを使用して取得しますが、このコネクションオブジェクトはClientObexConnectionまたはServerObexConnectionを実装しています。Connector.open()メソッドに指定するURLは、OBEXクライアントアプリケーションとOBEXサーバーアプリケーションでは異なります。

以降、OBEXクライアントAPIおよびOBEXサーバーAPIそれぞれについて解説します。

10.2.1 OBEXクライアントAPI

OBEXクライアントが使用するコネクションオブジェクトはClientObexConnectionインタフェースを実装しています。このコネクションオブジェクトを取得するには、アプリケーションプログラムはURL文字列 "obex:/irclient"を指定してConnector.open()メソッドを呼び出します。

以下に、アプリケーションプログラムでClientObexConnection実装インスタンスを取得、使用する例を示します。

例：ClientObexConnectionの利用

```
ClientObexConnection coc;
int response;
try {
          // ClientObexConnectionインスタンスを取得し、OBEXサーバーに接続する
          coc = (ClientObexConnection)Connector.open("obex:/irclient",
                                                     Connector.READ_WRITE, true);
          coc.connect();

          // GETオペレーションの実行
          coc.setOperation(ObexConnection.GET);       // オペレーションにGETを設定
          coc.setName("userdata_1");                  // GET対象オブジェクトの名前を設定
          coc.sendRequest();                          // オペレーションを送信

          // OBEXサーバーから返却されたレスポンスコードをチェックし、正常であれば受信した
          // オブジェクト本体を読み込む
          response = coc.getResponseCode();
          if (response==ObexConnection.SUCCESS) {
                   InputStream in = coc.openInputStream();
                            :       // GETオペレーションで要求したデータの読み込み処理
                   in.close();
          } else {
                   throw new IOException();
          }

          // PUTオペレーションの実行
          coc.setOperation(ObexConnection.PUT);       // オペレーションにPUTを設定
          coc.setName("userdata_2");                  // PUT対象オブジェクトの名前を設定
          coc.setType("text/plain");                  // PUT対象オブジェクトのタイプを設定
          coc.setTime(System.currentTimeMillis());    // PUT対象オブジェクトのタイムスタンプを設定
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
            OutputStream out = coc.openOutputStream();
                      :        // PUT オペレーションで送信するデータの書き込み処理
            out.close();
            coc.sendRequest();                                  // オペレーションを送信
            // レスポンスコードが正常であれば ClientObexConnection を閉じて処理終了
            response = coc.getResponseCode();
            if (response!=ObexConnection.SUCCESS) {
                    throw new IOException();
            }
            coc.close();

} catch (IOException ioe) {
            // 例外処理
            // API がスローする IOException、ConnectionException、
            // アプリケーションプログラムがスローする IOException の処理を行う
                      :
}
```

アプリケーションプログラムは ClientObexConnection を使用する際に、まず connect()メソッドを呼び出して OBEX サーバーとの間に赤外線通信のリンクを確立します。いったん確立された通信リンク上では、複数のオペレーションを連続して処理することができます。前出のサンプルコードでは、1 回確立した通信リンク上で GET オペレーションと PUT オペレーションを 1 回ずつ処理し、その後 ClientObexConnection.close()を呼び出して通信リンクを切断しています。

OBEX クライアントでは、オペレーションの処理は以下の手順に従って行います。

1. setOperation()メソッドを使用してオペレーション種別を設定します。
2. ObexConnection で定義されているアクセスメソッドを使用して、OBEX リクエストヘッダの設定を行います。アクセスメソッドを使用して設定することのできる OBEX リクエストヘッダには、Name ヘッダ、Type ヘッダ、Time ヘッダがあります。
3. PUT オペレーションの場合、送信するオブジェクトボディの内容を書き込みます。この操作は、コネクションオブジェクトの openOutputStream()または openDataOutputStream()メソッドで取り出した出力ストリームを使用して行います。
4. sendRequest()メソッドを使用してオペレーション処理要求を OBEX サーバーに送信します。このメソッドは OBEX サーバーから応答が返されるまでブロックします。このメソッド呼び出しから復帰すると、必要に応じて OBEX サーバーから返却された OBEX レスポンスコードや OBEX レスポンスヘッダの内容を参照することができます。OBEX クライアントが参照することのできる OBEX レスポンスヘッダには、Name ヘッダ、Type ヘッダ、Time ヘッダがあります。また、OBEX レスポンスコードは ObexConnection で定義されています。ObexConnection で定義されている OBEX レスポンスコードは、OBEX 規格におけるレスポンスコードの規定に準拠しています。各コードの意味などの詳細については OBEX 規格を参照してください。OBEX サーバーが OBEX クライアントの要求を正常に処理できた場合、一般的には OBEX クライアントアプリケーションは、OBEX レスポンスコードとして i アプリ実行環境(CrientObexConnection.getResponseCode()メソッド)から ObexConnection.SUCCESS を受け取ります。
5. GET オペレーションの場合、返却されたオブジェクトボディの内容を読み出します。この操作は、コネクションオブジェクトの openInputStream()または openDataInputStream()メソッドで取り出した入力ストリームを使用して行います。オブジェクトボディの長さは、コネクションオブジェクトの getContentLength()メソッドを使用して取得することができます。

OBEX クライアントの処理は、全体として上記のシーケンスに合致するようにしてください。例えばオペレーション処理要求を OBEX サーバーに送信していない状態で OBEX レスポンスヘッダを取得しようとすると例外が発生します。

次の図に、ClientObexConnection の状態遷移を示します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

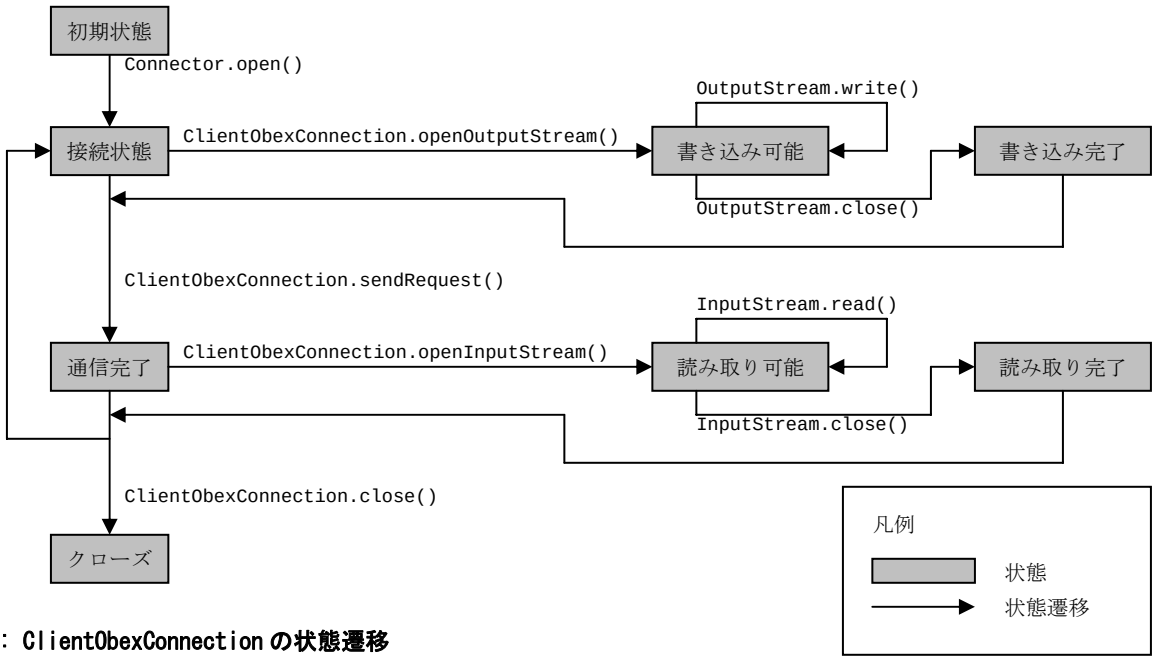


図 13: ClientObexConnection の状態遷移

注意事項：

- OBEX 外部接続においては、Connector クラスから直接 openInputStream()、openDataInputStream()、openOutputStream()、openDataOutputStream()の各メソッドにより入出力ストリームを取り出すことはできません。アプリケーションプログラムは Connector.open()メソッドで ClientObexConnection オブジェクトを取得し、そのオブジェクトから入出力ストリームを取り出す必要があります。
- 携帯電話の赤外線ポートへの通電は、アプリケーションプログラムから ClientObexConnection.connect()を呼び出した際に、システムにより自動的に行われます。
- OBEX サーバーに対し 1 度のオペレーションで、オブジェクトを送信するとともに OBEX サーバーからオブジェクトを受信することはできません。OBEX クライアントは、PUT オペレーションではオブジェクトの送信、GET オペレーションではオブジェクトの受信のみ行うことができます。
- OBEX クライアントにおけるコネクションオブジェクトのオープンモードは、READ モードが GET オペレーションに、WRITE モードが PUT オペレーションに対応します。1 回の接続で GET、PUT の両オペレーションを使用する場合は、READ_WRITE モードを使用するようにしてください。
- Name ヘッダ、Type ヘッダの値にはそれぞれ最大 63 文字まで指定可能です。また、Type ヘッダの値には ASCII 文字のみ使用することができます。
- 1 つの i アプリから同時に確立可能な通信リンクは 1 つだけです。また、 i アプリのバックグラウンドで動作しているネイティブ機能を含め、パケット通信が行われている状況では、OBEX 外部接続の通信リンクを確立することはできません。
- あるスレッドが connect()メソッドや sendRequest()メソッドを呼び出して OBEX サーバーのアクションを待っている時、他のスレッドが ClientObexConnection オブジェクトの close()メソッドを呼び出すと通信は切断されます。また、タイムアウト発生やユーザーの中断操作といった要因により、通信が切断されることがあります。
- GET、PUT の両オペレーションとも、1 回につき 100K バイトまでのオブジェクトを取り扱うことができます。この制限を越えたサイズのオブジェクトを送受信することはできません。

  【DoJa-3.0】

  送受信可能なオブジェクトサイズの制限は、DoJa-3.0 プロファイルで緩和されました。DoJa-2.0 プロファイルでは、送受信可能なオブジェクトサイズは 30K バイトまでに制限されています。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- アプリケーションプログラムが設定したオペレーション種別やOBEXヘッダ､オブジェクトデータの内容は、1回のオペレーションでのみ有効です。1回の接続で連続してオペレーションを実行する場合、その都度これらを設定しなおすようにしてください。
- OBEXサーバーから返却されたレスポンスコードが異常を示す値（0x20番台以外のコード）であった場合、iアプリ実行環境は例外をスローします。
- DoJa-2.0プロファイル以降に対応した携帯電話では、外部との通信機能を一切停止して、通信を使用しない機能のみ利用可能とするセルフモード機能が搭載されます。ユーザーが携帯電話をセルフモードにしている状態では、iアプリの起動を行うことはできますが起動されたiアプリがOBEX外部接続を行うことはできません。
- 非活性化状態の待ち受けアプリケーションは、OBEX外部接続を行うことはできません。また、活性化状態の待ち受けアプリケーションがOBEX外部接続を行っている際に非活性化状態に遷移しようとすると、OBEX外部接続は失敗します。
- iアプリが赤外線通信を行っている間は、iアプリ実行環境は通信を実行中である旨のメッセージを表示します。また通信リンクの確立が行えなかった場合や、タイムアウト、ユーザー操作などにより通信が中断された場合には、iアプリ実行環境はその旨をユーザーに伝えるメッセージを表示します。これらのメッセージを非表示とすることはできません。

10.2.2 OBEXサーバーAPI

OBEXサーバーが使用するコネクションオブジェクトは`ServerObexConnection`インタフェースを実装しています。このコネクションオブジェクトを取得するには、アプリケーションプログラムはURL文字列`"obex:/irserver"`を指定して`Connector.open()`メソッドを呼び出します。

以下に、アプリケーションプログラムで`ServerObexConnection`実装インスタンスを取得、使用する例を示します。

例：`ServerObexConnection`の利用

```
ServerObexConnection soc;
try {
          // ServerObexConnectionインスタンスを取得する
          soc = (ServerObexConnection)Connector.open("obex:/irserver",
                                                     Connector.READ_WRITE, true);
          // OBEXクライアントからの通信リンク確立要求を待ち合わせる
          soc.accept();

          // OBEXクライアントからのオペレーション要求を処理する
          while (true) {
                   soc.receiveRequest();              // オペレーション要求を待つ
                   int operation = soc.getOperation(); // 到着したオペレーションの種別を取得する

                   if (operation==ObexConnection.GET) {         // GETオペレーション要求
                            // 要求されたオブジェクトの名前を取得
                            String name = soc.getName();
                            // 指定された名前に対応するオブジェクトのヘッダ情報を設定
                            soc.setType("text/plain");
                            soc.setTime(System.currentTimeMillis());
                            // 指定された名前に対応するオブジェクトの書き込み処理
                            OutputStream out = soc.openOutputStream();
                                     :
                            out.close();
                            // 正常終了を示すステータスコードをOBEXクライアントに返す
                            soc.sendResponse(ObexConnection.SUCCESS);
                   } else if (operation==ObexConnection.PUT) { // PUTオペレーション要求
                            // 送信されたオブジェクトの各種情報を取得
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
                        String name = soc.getName();
                        String type = soc.getType();
                        long time = soc.getTime();
                        // 送信されたオブジェクトの読み出し処理
                        InputStream in = soc.openInputStream();
                                :
                        in.close();
                        // 正常終了を示すステータスコードを OBEX クライアントに返す
                        soc.sendResponse(ObexConnection.SUCCESS);
                } else if (operation==ObexConnection.DISCONNECT) {    // 切断要求
                        break;
                }
        }

        // 通信リンクを切断し、通信を終了する
        soc.close();
} catch (IOException ioe) {
        // 例外処理
        // API がスローする IOException、ConnectionException の処理を行う
                :
}
```

アプリケーションプログラムは ServerObexConnection を使用する際に、まず accept()メソッドを呼び出して OBEX クライアントからの赤外線通信のリンク確立要求（OBEX クライアントの connect()メソッド実行）を待ちます。いったん確立された通信リンク上では、複数のオペレーションを連続して処理することができます。前出のサンプルコードでは、1 回確立した通信リンク上で、OBEX クライアントからの切断要求を受け付けるまで繰り返しオペレーションの処理を行っています。

OBEX サーバーでは、オペレーションの処理は以下の手順に従って行います。

1. receiveRequest()メソッドを使用して要求を受け付けます。このメソッドは、OBEX クライアントからの要求を受け付けるまでブロックします。
2. getOperation()メソッドを使用して要求のオペレーション種別を取得します。オペレーション種別として ObexConnection.DISCONNECT を受け取った場合は、OBEX クライアントが通信リンクの開放を通知したことを示しています。この要求を受け取った場合、OBEX サーバーもコネクションオブジェクトをクローズするようにしてください。
3. ObexConnection で定義されているアクセスメソッドを使用して、OBEX リクエストヘッダの内容を取得します。アクセスメソッドを使用して取得することのできる OBEX リクエストヘッダには Name ヘッダ、Type ヘッダ、Time ヘッダがあります。
4. PUT オペレーションの場合、OBEX クライアントから送信されたオブジェクトボディの内容を読み出します。この操作は、コネクションオブジェクトの openInputStream()または openDataInputStream()メソッドで取り出した入力ストリームを使用して行います。オブジェクトボディの長さは、コネクションオブジェクトの getContentLength()メソッドを使用して取得することができます。
5. ObexConnection で定義されているアクセスメソッドを使用して、OBEX レスポンスヘッダの内容を設定します。アクセスメソッドを使用して設定することのできる OBEX レスポンスヘッダには Name ヘッダ、Type ヘッダ、Time ヘッダがあります。
6. GET オペレーションの場合、OBEX クライアントが要求しているオブジェクトのオブジェクトボディを書き込みます。この操作は、コネクションオブジェクトの openOutputStream()または openDataOutputStream()メソッドで取り出した出力ストリームを使用して行います。
7. sendResponse()メソッドを使用して、OBEX クライアントにレスポンスを返します。このメソッドの引数には、ObexConnection で定義されている OBEX レスポンスコードを指定します。なお、ObexConnection で定義されている OBEX レスポンスコードは、OBEX 規格におけるレスポンスコードの規定に準拠しています。各コードの意味などの詳細については OBEX 規格を参照してください。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

OBEX サーバーが OBEX クライアントの要求を正常に処理できた場合、一般的には OBEX サーバーアプリケーションは、OBEX レスポンスコードとして i アプリ実行環境（ServerObexConnection.sendResponse()メソッド）に ObexConnection.SUCCESS を返します。

OBEX サーバーの処理は、全体として上記のシーケンスに合致するようにしてください。例えばオペレーション処理要求を受け付けていない状態で OBEX リクエストヘッダを取得しようとすると例外が発生します。

次の図に、ServerObexConnection の状態遷移を示します。

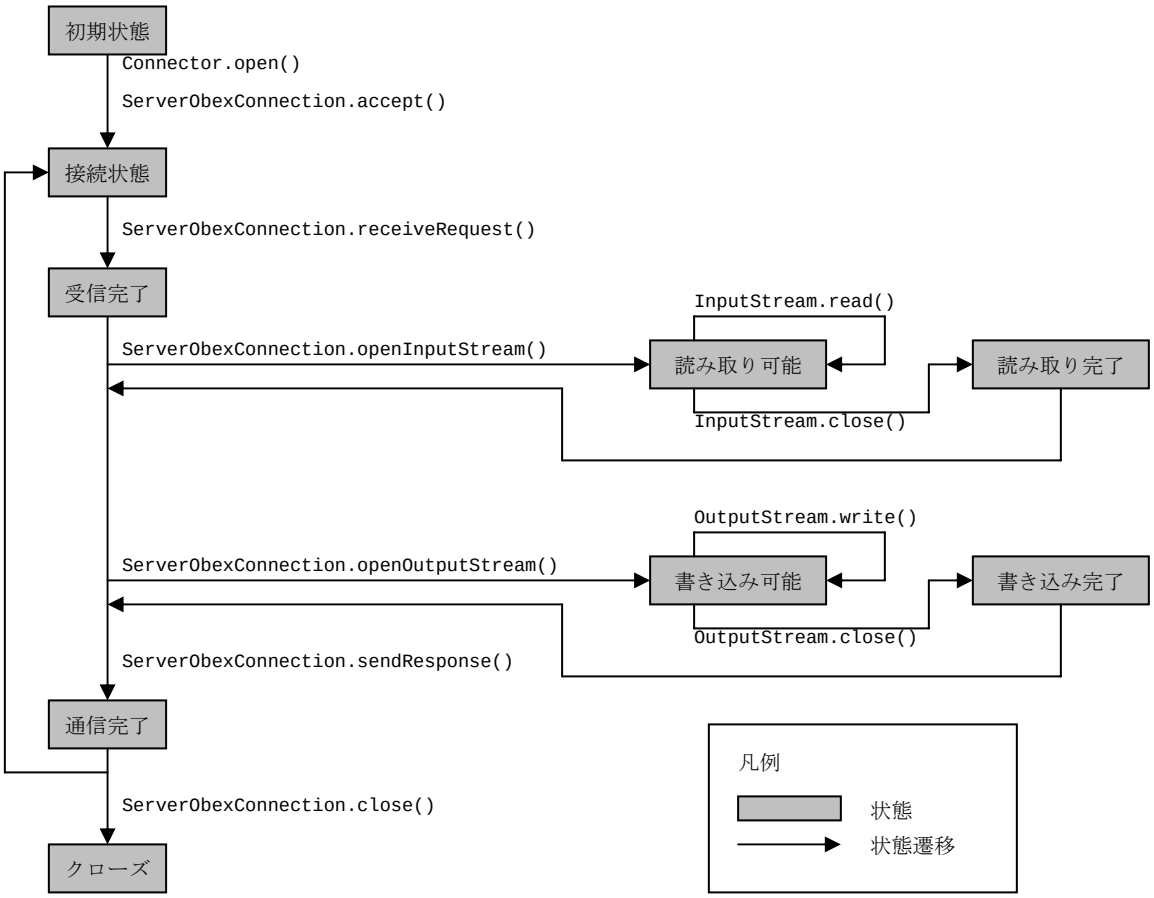


図 14: ServerObexConnection の状態遷移

注意事項：

- OBEX 外部接続においては、Connector クラスから直接 openInputStream()、openDataInputStream()、openOutputStream()、openDataOutputStream()の各メソッドにより入出力ストリームを取り出すことはできません。アプリケーションプログラムは Connector.open()メソッドで ServerObexConnection オブジェクトを取得し、そのオブジェクトから入出力ストリームを取り出す必要があります。
- 携帯電話の赤外線ポートへの通電は、アプリケーションプログラムから ServerObexConnection.accept()を呼び出した際に、システムにより自動的に行われます。
- OBEX クライアントからの PUT オペレーションに対して、OBEX サーバーがオブジェクトを送信することはできません。OBEX サーバーは GET オペレーションに対してのみオブジェクトを送信することができます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- OBEX サーバーにおけるコネクションオブジェクトのオープンモードは、READ モードが PUT オペレーションに、WRITE モードが GET オペレーションに対応します。1 回の接続で GET、PUT の両オペレーションを受け付ける可能性がある場合は READ_WRITE モードを使用するようにしてください。
- Name ヘッダ、Type ヘッダの値にはそれぞれ最大 63 文字まで指定可能です。また、Type ヘッダの値には ASCII 文字のみ使用することができます。
- 1 つの i アプリから同時に確立可能な通信リンクは 1 つだけです。また、 i アプリのバックグラウンドで動作しているネイティブ機能を含め、パケット通信が行われている状況では、OBEX 外部接続の通信リンクを確立することはできません。
- あるスレッドが `accept()`メソッドや `receiveRequest()`メソッドを呼び出して OBEX クライアントのアクションを待っている時、他のスレッドが `ServerObexConnection` オブジェクトの `close()`メソッドを呼び出すと通信は切断されます。また、タイムアウト発生やユーザーの中断操作といった要因により、通信が切断されることがあります。
- GET、PUT の両オペレーションとも、1 回につき 100K バイトまでのオブジェクトを送受信することができます。この制限を越えたサイズのオブジェクトを送受信することはできません。

  【DoJa-3.0】

  送受信可能なオブジェクトサイズの制限は、DoJa-3.0 プロファイルで緩和されました。DoJa-2.0 プロファイルでは、送受信可能なオブジェクトサイズは 30K バイトまでに制限されています。
- アプリケーションプログラムが設定した OBEX ヘッダやオブジェクトデータの内容は、1 回のオペレーションでのみ有効です。1 回の接続で連続してオペレーションを実行する場合、その都度これらを設定しなおすようにしてください。
- DoJa-2.0 プロファイル以降に対応した携帯電話では、外部との通信機能を一切停止して、通信を使用しない機能のみ利用可能とするセルフモード機能が搭載されます。ユーザーが携帯電話をセルフモードにしている状態では、 i アプリの起動を行うことはできますが起動された i アプリが OBEX 外部接続を行うことはできません。
- 非活性化状態の待ち受けアプリケーションは、OBEX 外部接続を行うことはできません。また、活性化状態の待ち受けアプリケーションが OBEX 外部接続を行っている際に非活性化状態に遷移しようとすると、OBEX 外部接続は失敗します。
- i アプリが赤外線通信を行っている間は、 i アプリ実行環境は通信を実行中である旨のメッセージを表示します。また通信リンクの確立が行えなかった場合や、タイムアウト、ユーザー操作などにより通信が中断された場合には、 i アプリ実行環境はその旨をユーザーに伝えるメッセージを表示します。これらのメッセージを非表示とすることはできません。

10.3 外部機器接続のヒント

以下に、i アプリ仕様における赤外線通信の下位レイヤ仕様、および携帯電話上の i アプリと携帯電話以外の外部機器との間で赤外線通信を行う上での注意点を示します。

携帯電話間で赤外線通信を行う i アプリを開発する場合はこれらを意識する必要はありません。携帯電話上の i アプリと他の外部機器との間で赤外線通信を行うシステムを構築するようなケースにおいて、外部機器側の設計を行う際の参考としてください。

なお、下記の記載内容に関する規定の詳細については IrDA 規格を参照してください。

＜物理的条件＞

通信速度： 共通的にサポートされる最大通信速度は 115.2kbps です。

通信距離： 携帯電話に搭載される赤外線デバイスは、IrDA 規格における IrMC Low Power Option に対応しています。このため、通信距離は屋内環境で 20cm 程度に制限されます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

＜IrLAP　接続パラメータ＞

Baud Rate：　9600bps、38400bps および 115200bps をサポートします。

Data Size：　64byte、128byte、256byte をサポートします。

Link Disconnect / Threshold Time：　3 秒（Threshold=0 秒）、8 秒（Threshold=3 秒）をサポートします。

＜IrLMP　デバイス名およびデバイスニックネーム＞

デバイス名およびデバイスニックネームには、以下のような文字列が設定されます。

`DoCoMo/<機種名>/j`　　（例：　`DoCoMo/X505i/j`）

<機種名>には、ユーザーエージェントに含まれている名前と同等のものが設定されます。外部機器からこの名前を参照することで、現在通信を行っている携帯電話の機種を判別することができます。

＜IrLMP　サービスヒント＞

サービスヒントのビットマップには、以下のフラグが ON に設定されます。

```
Byte1：Bit1   PDA/Palmtop
Byte1：Bit7   Extension
Byte2：Bit8   Telephony
Byte2：Bit13  IrOBEX
```

【DoJa-3.0】

IrLAP 接続パラメータのうち、Baud Rate 115200bps は DoJa-3.0 プロファイルにて追加されました。DoJa-2.0 プロファイルで共通的にサポートされる Baud Rate は 9600bps および 38400bps であり、115200bps での通信は保証されないため注意してください。

また、IrLMP のサービスヒントの設定内容は、DoJa-2.0 プロファイルではメーカーにより異なる場合があります。

注意事項：

- OBEX で規定されているオペレーションのうち、ｉアプリ実行環境は OBEX クライアント、OBEX サーバーとも SetPath オペレーションをサポートしません。また、OBEX クライアントは ABORT オペレーションをサポートしません。
- ｉアプリ実行環境は、OBEX 認証をサポートしません。
- ｉアプリ実行環境において、OBEX クライアントおよび OBEX サーバーが受け付けることのできる OBEX ヘッダは Name、Type、Time、Body、End of Body の 5 つです。これら以外の OBEX ヘッダを使用することはできません。また、オブジェクトボディ（Body ヘッダおよび End of Body ヘッダ）は他のヘッダより後に送信してください。
- 携帯電話に搭載される赤外線通信機能は IrDA の相互接続認定を受けています。しかしこれは、IrDA 規格に準拠した全ての外部機器との接続性を保証するものではありません。使用する環境や相手側機器のハードウェアおよびソフトウェア特性などにより、接続することができない場合があります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

10.4 IrSimple

DoJa-5.1 プロファイル以降の携帯電話の中には、オプショナル機能として i アプリからの IrSimple の利用をサポートするものがあります。IrSimple を利用すると高速、大容量の赤外線通信が行える一方、一部機能の利用に制限が生じるなどの注意事項もあります。

以下に、 i アプリからの IrSimple の利用方法とその注意点について記載します。

IrSimple をサポートする機種のために、DoJa-5.1 プロファイルでは OBEX 外部接続 API に以下のメソッド、フィールドが追加されています。

- `ClientObexConnection.connect(int mode)`

  OBEX クライアント側から、通信モードを指定して赤外線通信のリンクを確立します（通信モードとは `ObexConnection` の `COMM_MODE` 定数のことです）。IrSimple を利用するための指定は、`connect()` メソッドに設定する通信モード引数で行います。

- `ObexConnection.getCommMode()`

  リンクが確立されたコネクションオブジェクトから、現在の通信モードを取得します。

- `ObexConnection.COMM_MODE_IRDA`

  通信モード定数で、従来互換の IrDA 通信であることを表します。

- `ObexConnection.COMM_MODE_IRSIMPLE_INTERACTIVE`

  通信モード定数で、IrSimple 双方向通信であることを表します。IrSimple 双方向通信では、従来の IrDA などと同様に、OBEX サーバーと OBEX クライアントの間でデータ転送の送達確認や再送制御などを行い、信頼性の高い通信を実現します。

- `ObexConnection.COMM_MODE_IRSIMPLE_UNILATERALLY`

  通信モード定数で、IrSimple 片方向通信であることを表します。IrSimple 片方向通信では、OBEX クライアント側から一方通行的にデータを送信し、クライアント・サーバー相互の送達確認や再送制御は行いません。このため、双方向通信のような高い信頼性は得られませんが、より高速、高効率な通信を行うことができます。

IrSimple を使用する場合には、OBEX クライアント側 i アプリの通信リンク確立処理において、`connect(int)`メソッドの引数に `ObexConnection.COMM_MODE_IRSIMPLE_INTERACTIVE` もしくは `ObexConnection.COMM_MODE_IRSIMPLE_UNILATERALLY` を指定してください。

IrSimple を使用する場合、従来の IrDA と比較して以下の点が異なります。

- OBEX クライアントは、GET オペレーションを使用することはできません。IrSimple を使用する場合、OBEX クライアントは PUT オペレーションのみ使用することができます。さらに IrSimple 片方向通信では、1 回の通信リンク確立につき、実行可能なオペレーションの回数は `1` 回に限られます。
- OBEX クライアントが一度の PUT オペレーションで送信可能なデータサイズの制限は、2M バイト以内までに緩和されます。

なお、OBEX サーバーが IrSimple に対応していない状況で、OBEX クライアントが IrSimple の利用を試みた場合は以下のような動作となることに注意してください。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

1. OBEX クライアントが IrSimple 双方向通信で接続しようとした場合

IrSimple をサポートしていない OBEX サーバーに対して、OBEX クライアントが IrSimple 双方向通信で接続しようとすると、IrSimple ではなく、従来互換の IrDA 通信の通信リンクが確立されます。

2. OBEX クライアントが IrSimple 片方向通信で接続しようとした場合

IrSimple 片方向通信では、OBEX クライアントは OBEX サーバーが IrSimple をサポートしているかどうか、さらには対向側に IrDA 機器が存在しているかどうかにも関係なく、一方的にデータを送信し、例外等も発生することなく正常に処理を完了させます。対向側の OBEX サーバーが IrSimple をサポートしていない場合、OBEX クライアントは正常にデータ転送を完了させますが、OBEX サーバー側は接続要求に応答せず、受信待ち状態で待ち続けます。

注意事項：

- IrSimple を使用する際も、`Connector.open()`メソッドに指定する URL 文字列は、通常の IrDA 通信を行う際と同じもの（`"obex:/irclient"`、`" obex:/irserver"`）を指定します。
- IrSimple をサポートしていない携帯電話で `ClientObexConnection.connect(int)`メソッドや `ObexConnection.getCommMode()`メソッドを使用した場合、`UnsupportedOperationException` がスローされます。
- IrSimple 片方向通信では、OBEX クライアントから OBEX サーバーに向けて一方通行的に通信が行われるため、OBEX クライアントのリクエストに対して OBEX サーバーがレスポンスを返すことはありません。しかし、 i アプリの OBEX サーバーに IrSimple 片方向通信を処理させる場合には、`ServerObexConnection.receiveRequest()`メソッドでリクエストを受け取ったら、それに対応して `ServerObexConnection.sendResponse()`メソッドを使用するようにしてください（これらのメソッドの利用方法については 10.2.2 項も参照してください）。

【DoJa-5.1】

IrSimple のための API は、DoJa-5.1 プロファイルにて新設されました。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

第 11 章
アプリケーション連携

DoJa-2.0 プロファイルでは、i アプリと、ブラウザやメーラなど携帯電話上のネイティブアプリケーションとの間での連携起動がサポートされました。また DoJa-3.0 プロファイル以降では、連携起動の範囲を拡大（i アプリー i アプリ間など）するとともに、i アプリからブックマーク管理機能や画像データ管理機能などネイティブ機能の呼び出しをサポートするなどアプリケーション連携の拡張が図られました。これらの機能を利用することで、i アプリの機能だけでなく携帯電話の機能をフル活用したサービスを i アプリで提供することができます。

なお、機能の性格上、アプリケーション連携機能の一部はトラステッド i アプリの範囲に含まれます。本書ではトラステッド i アプリに含まれる機能の解説は行いません。本書で解説される機能は、トラステッド i アプリ以外の一般の i アプリでも使用することができます。

各プロファイルでサポートされているアプリケーション連携機能の内容は以下の通りです。

※　アプリケーション連携起動

連携機能の概要	DoJa-2.0 以降	DoJa-3.0 以降	DoJa-4.1 以降	DoJa-5.0 以降
ブラウザからの i アプリ起動	○	○	○	○
i アプリからのブラウザ起動	○	○	○	○
メーラからの i アプリ起動	○	○	○	○
外部機器（赤外線ポート）からの i アプリ起動	○	○	○	○
i アプリからの i アプリ起動	×	○	○	○
i アプリからの i アプリ更新機能（JAM）の起動	×	○	○	○

※　i アプリからのネイティブ機能呼び出し

連携機能の概要	DoJa-2.0 以降	DoJa-3.0 以降	DoJa-4.1 以降	DoJa-5.0 以降
通話機能の呼び出し（通話発信）	○	○	○	○
通話機能の呼び出し（個体識別情報の参照）	△(*1)	○	○	○
電話帳管理機能の呼び出し（電話帳エントリ、電話帳グループの新規登録）	×	○	○	○
ブックマーク管理機能の呼び出し（ブックマークの新規登録）	×	○	○	○
スケジュール管理機能の呼び出し（スケジュールの新規登録、i アプリからのスケジューラの起動）	×	○(*3)	○(*3)	○(*3)
画像データ管理機能の呼び出し（画像の新規登録・選択読み込み・ID 指定読み込み）	×	○	○	○
カメラ機能の呼び出し（撮影および撮影画像の取得）	△(*2)	○	○	○
映像データ管理機能の呼び出し（映像データ（i モーション）の新規登録）	×	×	○	○

(*1) 個体識別情報の参照は、DoJa-2.1 プロファイルで i アプリ基本 API に追加されました。

(*2) カメラ機能の呼び出しは、DoJa-2.0 プロファイルでは i アプリオプション API として一部の機種でサポートされました。DoJa-3.0 プロファイルでは、それらの機能を一部強化した上で i アプリ基本 API に取り入れられています。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

(*3) スケジュール管理機能の呼び出しのうち、i アプリからのスケジューラの起動については DoJa-4.1 プロファイルまでは i アプリオプション API の位置づけです。この機能は、DoJa-5.0 プロファイルにて i アプリ基本 API の位置づけに変更されました。

本章では、アプリケーション連携機能の利用方法、および各種制限事項などについて解説します。

11.1 ブラウザ連携起動

ブラウザ連携起動では、ブラウザからの i アプリ起動および i アプリからのブラウザ起動を行うことができます。なお、ブラウザからの i アプリ起動においては、起動対象の i アプリはあらかじめ携帯電話にインストールされていなければなりません。

11.1.1 ブラウザからの i アプリの起動

ブラウザから i アプリを起動させるためには、開発者は以下の 2 つを行います。

(1) インターネット経由でアクセス可能な Web サイト上に i アプリ起動用の HTML ファイルを作成し、その中に i アプリ起動タグを記述します。i アプリ起動タグの記述内容により、そのタグを選択した際に携帯電話上で起動される i アプリが決定されます。

(2) ADF の `LaunchByBrowser` キーにて、その i アプリが(1)で作成された HTML の URL からの起動を許可することを宣言します。`LaunchByBrowser` キーで許可を受けていない URL の HTML からは、その i アプリを起動することはできません。なお、`LaunchByBrowser` キーで指定された URL と HTML の URL は前方一致で比較され、`LaunchByBrowser` キーに指定された内容が HTML の URL に全て含まれていれば、その HTML に対し起動許可が与えられます。

ブラウザから i アプリを起動するには、まず i アプリ起動用 HTML をブラウザに表示させます。i アプリ起動タグはリンク形式で表示されます。ユーザーがこのリンクを選択することにより、携帯電話は対応する i アプリを起動します。

以下に、i アプリ起動用 HTML におけるタグの記述例を示します。

```
<OBJECT declare id="application.declaration"
data="http://www.nttdocomo.co.jp/java/abc.jam" type="application/x-jam">
<PARAM name=”Param1” value=”i-mode”>
<PARAM name=”Param2” value=”i アプリ”>
</OBJECT>
i アプリを起動するには
<A ista=”#application.declaration” href=”notapplicable.html”>ここ</A>をクリック。
```

i アプリ起動のために使用されるタグは以下の通りです。

- `A` タグおよび `ista` 属性

  `A` タグは、起動対象の i アプリに対応する `OBJECT` タグを参照するために使用されます。ユーザーがこの `A` タグを選択することで、携帯電話にインストールされている i アプリが起動されます。`A` タグの `ista` 属性には、`OBJECT` タグの `id` 属性に指定された名前を指定します。携帯電話上のどの i アプリが起動されるかは、後述の `OBJECT` タグに指定された ADF URL により決定されます。

  `ista` 属性は DoJa-2.0 プロファイルにて新設された属性であり、DoJa-1.0 プロファイルに対応した携帯電話では解釈されません。

  なお、`ista` 属性を持つ `A` タグでは、必ずペアで `href` 属性を指定するようにしてください。また、アプリケーションのダウンロードで使用する `A` タグおよび属性（`ijam` 属性、`ilet` 属性）については、第 16 章を参照してください。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

● OBJECT タグ

OBJECT タグは、 i アプリに対応する ADF を特定するために使用されます。OBJECT タグの id 属性には、A タグの ista 属性で参照される名前（HTML 内で一意）を指定します。また data 属性には ADF の位置を示す URL を指定します。携帯電話は i アプリをダウンロードした際に、その i アプリに対応する ADF の URL を記憶しています。ブラウザからの i アプリ起動では、data 属性に指定された URL と同じ ADF を参照している i アプリが起動対象となります。type 属性には data 属性で示されたデータ（この場合 ADF）のコンテンツタイプを指定します。ADF のコンテンツタイプは"application/x-jam"です。

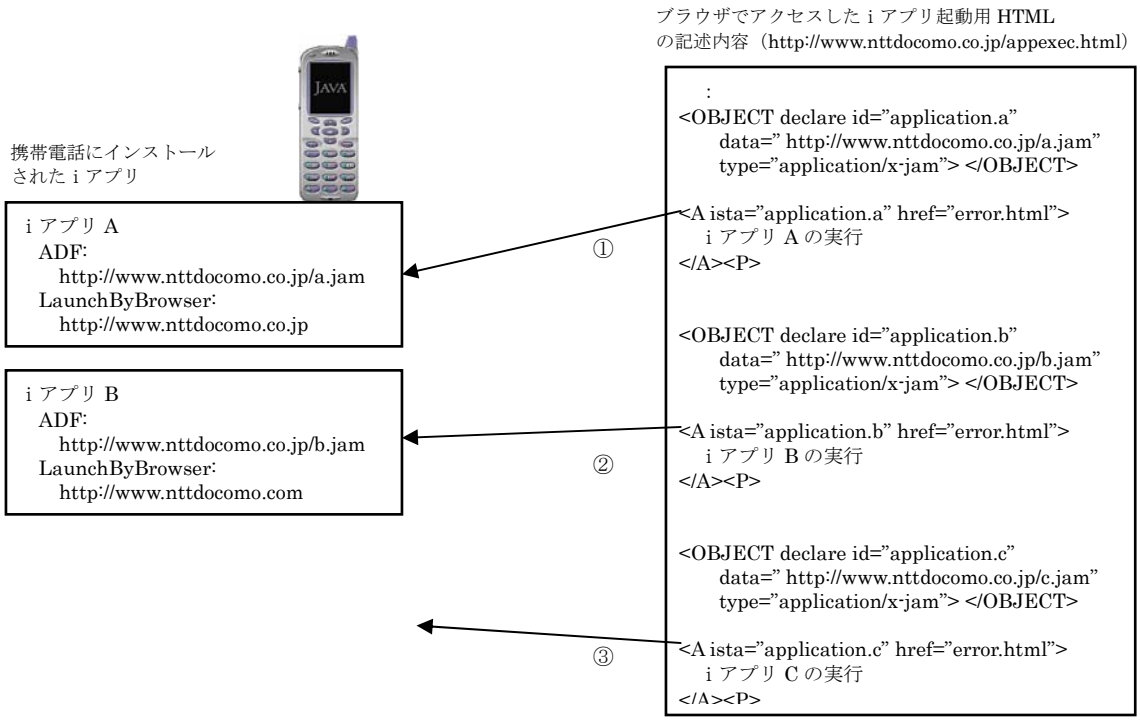
OBJECT タグの内部には、下記に示す PARAM タグを記述することができます。

● PARAM タグ

PARAM タグは、ブラウザからの i アプリ起動時に、IApplication.getParameter ()メソッドにより取得できるパラメータを指定するために使用されます。

PARAM タグは、OBJECT タグ中に最大 16 個含めることができます。また、1 つの OBJECT タグ中に指定された全 PARAM タグの name 属性の値の長さと value 属性の値の長さの合計は最大 255 バイトに制限されます。name 属性の値および value 属性の値には、それぞれ SJIS で日本語テキストを指定することができます。

以下に、携帯電話にダウンロードされている i アプリと、 i アプリ起動用 HTML の内容の関係について図示します。



① 指定された ADF を使用してダウンロードされた i アプリが携帯電話上に存在し、かつ i アプリ起動用 HTML の URL が LaunchByBrowser キーの値に前方一致するため、 i アプリ A はユーザーの許可が得られれば起動可能。
② 指定された ADF を使用してダウンロードされた i アプリが携帯電話上に存在するが、 i アプリ起動用 HTML の URL が LaunchByBrowser キーの値に前方一致しないため、 i アプリ B は起動不可能。
③ 指定された ADF を使用してダウンロードされた i アプリが携帯電話上に存在しないため、 i アプリ C は起動不可能。

図 15: ダウンロード済み i アプリと i アプリ起動用 HTML の記述内容との関係

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

注意事項：

- 携帯電話ユーザーは、携帯電話の設定によりブラウザからの i アプリ起動を禁止することができます。またブラウザから i アプリを起動する前に、 i アプリ実行環境はユーザーに i アプリを起動してよいか確認を行います。これらの機構によりユーザーの許可が得られなかった場合、その i アプリの起動は行われません。
- ブラウザから起動された i アプリの動作が終了（IApplication.terminate()により終了）すると、 i アプリを起動する前のブラウザの状態に復帰します。ただし、ブラウザから起動された i アプリをユーザーが強制終了した場合、およびブラウザから起動された i アプリがアプリケーション連携機能によって他のネイティブアプリケーションの機能の起動または呼び出しを行った場合は i アプリ終了後にどのような状態に移るかはメーカーにより異なります。

【DoJa-5.0】

DoJa-5.0 プロファイル以降、トルカのバージョン 2.0 フォーマットに対応している携帯電話では、トルカ（詳細）の HTML コンテンツ部分に本項で解説したタグを記述することで、トルカビューアから i アプリを起動することもできます。この場合、トルカ（詳細）の HTML コンテンツ部分に記述する OBJECT タグ data 属性に記載する ADF URL には相対 URL は指定しないようにしてください。

トルカビューアからの i アプリ起動を許可するには、その i アプリの ADF に、LaunchByBrowser キーの代わりに LaunchByToruCa キーを指定します。各々のトルカデータにはトルカ（詳細）を取得するための URL が対応付けられていますが、トルカ（詳細）の取得先 URL に LaunchByToruCa キーの値が前方一致する場合は、そのトルカから該当の i アプリを起動することができます。

なお、ブラウザから i アプリを起動した際の i アプリの起動種別（IApplication.getLaunchType()メソッドで取得する起動種別）が IApplication.LAUNCHED_FROM_BROWSER となるのに対し、トルカビューアから i アプリを起動した際の i アプリの起動種別は IApplication.LAUNCHED_FROM_TORUCA となります。

【DoJa-5.1】（FOMA 906i 以降）

i アプリ連携起動用の A タグに、ista 属性に加え ijam 属性も指定した場合は、 i アプリを連携起動しようとした際にその i アプリが更新されていることが検出されると i アプリの更新処理が行われます。そのような状況において、i アプリの更新処理に引き続いて起動された i アプリは、FOMA 906i シリーズ以降では起動種別は IApplication.LAUNCHED_FROM_BROWSER（ブラウザからの起動の場合）もしくは IApplication.LAUNCHED_FROM_TORUCA（トルカビューアからの起動の場合）となります。しかしそれより以前の機種では、更新処理に引き続いて起動された i アプリの起動種別はメーカーにより異なります。

なお ista 属性と ijam 属性が併記されていても、携帯電話上に対象の i アプリが存在しない場合には、 i アプリのダウンロード処理に引き続いて起動された i アプリの起動種別は IApplication.LAUNCHED_AFTER_DOWNLOAD となります。

11.1.2 i アプリからのブラウザ起動

i アプリからブラウザを起動するには、IApplication.launch()メソッドを使用します。ブラウザを起動するには、このメソッドの第 1 引数（起動対象となるアプリケーションの種別）に IApplication.LAUNCH_BROWSER を、また第 2 引数（起動引数を示す String 配列）の先頭要素に接続先の URL（http スキームまたは https スキームをサポート）を設定します。URL は最長 255 バイトまで指定できます。

なおブラウザを起動する i アプリは、ADF の UseBrowser キーでブラウザ起動を行うことを宣言（値"launch"を設定）する必要があります。この宣言を行っていない i アプリはブラウザを起動することはできません。

IApplication.launch()メソッドを呼び出すと、 i アプリはサスペンド状態に入ります。その後、以下の 2 点を満たすことでブラウザが起動されます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- パケット通信が可能であること（圏外など、パケット通信ができない状態でないこと）。
- ブラウザ起動前、ｉアプリ実行環境がユーザーに求める起動確認に対しユーザーが同意すること。

いずれかを満たしていない場合、ｉアプリはレジュームし、`IApplication.launch()`メソッドは例外をスローすることなく呼び出し元に復帰します。また、両方の条件を満たしてブラウザが起動される際、ｉアプリは終了します。

【DoJa-5.0】

DoJa-5.0 プロファイル以降では、ｉアプリからブラウザを起動する際にｉアプリを終了させず、サスペンド状態を継続することを指示するための API が用意されています。ただしこの機能はオプショナルであり、メーカーによってはサポートされない場合があります。

この機能を使用するには上記の手順において、`launch()`メソッドの第１引数に、`IApplication.LAUNCH_BROWSER` の代わりに `IApplication.LAUNCH_BROWSER_SUSPEND` を指定します。この方法でブラウザを起動した場合はブラウザ起動後もｉアプリはサスペンド状態を保っており、ブラウザ終了時にレジューム復帰します。

注意事項：

- DoJa-4.0 プロファイル以前のｉアプリ実行環境では、待ち受けアプリケーションが待ち受け起動されている場合、どのような状態であっても `MApplication.launch()`メソッドを使用してブラウザを起動することはできません。待ち受け起動された待ち受けアプリケーションがブラウザを起動しようとすると例外が発生します。

  これに対し DoJa-4.1 プロファイル以降のｉアプリ実行環境では、待ち受け起動された待ち受けアプリケーションであっても、活性化状態であれば `MApplication.launch()`メソッドによるブラウザ起動を行うことができます。
- ｉアプリから起動されたブラウザがユーザー操作により終了すると、待ち受け状態に復帰します。
- `HttpConnection` を使用した HTTP 通信時と異なり、ｉアプリからのブラウザ起動では、ｉアプリダウンロード元サイト以外のサイトにアクセスすることができます。ただしリクエストメソッドは GET に限定され、ｉアプリ実行環境によるユーザーへのブラウザ起動確認時に URL の内容がユーザーに提示されます。

11.2 メール連携起動

メール連携起動では、ユーザーに配信されたメールからｉアプリの起動を行うことができます。なお、起動対象のｉアプリはあらかじめ携帯電話にインストールされていなければなりません。

11.2.1 メールからのｉアプリの起動

メールからｉアプリを起動させるためには、開発者は以下の 2 つを行います。

(1) ADF の `LaunchByMail` キーにて、対応するｉアプリが、特定の送信元から受け取ったメールからの起動を許可することを宣言します。`LaunchByMail` キーで許可を受けていないメール送信元のメールからは、そのｉアプリを起動することはできません。なお、`LaunchByMail` キーで指定されたメールアドレスとメール送信元アドレスは後方一致で比較され、`LaunchByMail` キーに指定された内容がメール送信元アドレスに全て含まれていれば、そのメールに対し起動許可が与えられます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

(2) (1)の指定が行われたｉアプリをダウンロードしているユーザーの携帯電話に対し、起動許可を得られるメールアドレスからメールを送信します。このメールには、起動するｉアプリを決定するための情報やｉアプリの起動パラメータなど（ｉアプリ起動情報）を記述します。

ｉアプリ起動情報が記述されたメールをメーラで閲覧すると、ｉアプリ起動用のリンクなどが表示されます。ユーザーがこのリンクを選択することにより、携帯電話は対応するｉアプリを起動します。

以下に、ｉアプリ起動情報を含むメールの記述例を示します。

```
・・・・・・・・
・・・・・・・・・・
・・・・・
（ここまでメーラで閲覧可能なメール本文）
--B:A
TEXT="ｉアプリＡ"
ADF="http://www.nttdocomo.co.jp/java/abc.jam"
"Param1"="i-mode"
"Param2"="ｉアプリ"
```

（注）各行とも行末は<CR><LF>（0x0d0a）です。

ｉアプリ起動情報の記法は以下の通りです。

- --B:A

  "—-B:A"は、メール本文とｉアプリ起動情報の境界を示すバウンダリ識別子です。この識別子（アルファベット部は大文字である必要があります）のみを含む行以降はｉアプリ起動情報と見なされます。ｉアプリ起動情報は、ユーザーがメールを閲覧する際にその内容を直接見ることはできません。また、メールの返信や転送などにおいても、ｉアプリ起動情報は引用されません。

- TEXT キー

  TEXT キーには、ｉアプリ起動用リンクで表示される文字列の設定を行います。先の記述例ではメール閲覧時に"ｉアプリＡ"という文字列がリンクとして表示され、ユーザーがそのリンクを選択することでｉアプリが起動されます。

- ADF キー

  ADF キーには、起動したいｉアプリに対応する ADF の URL を指定します。携帯電話はｉアプリをダウンロードした際に、そのｉアプリに対応する ADF の URL を記憶しています。メール連携においては、メールに記述された ADF キーの URL と同じ ADF を参照しているｉアプリが起動対象となります。

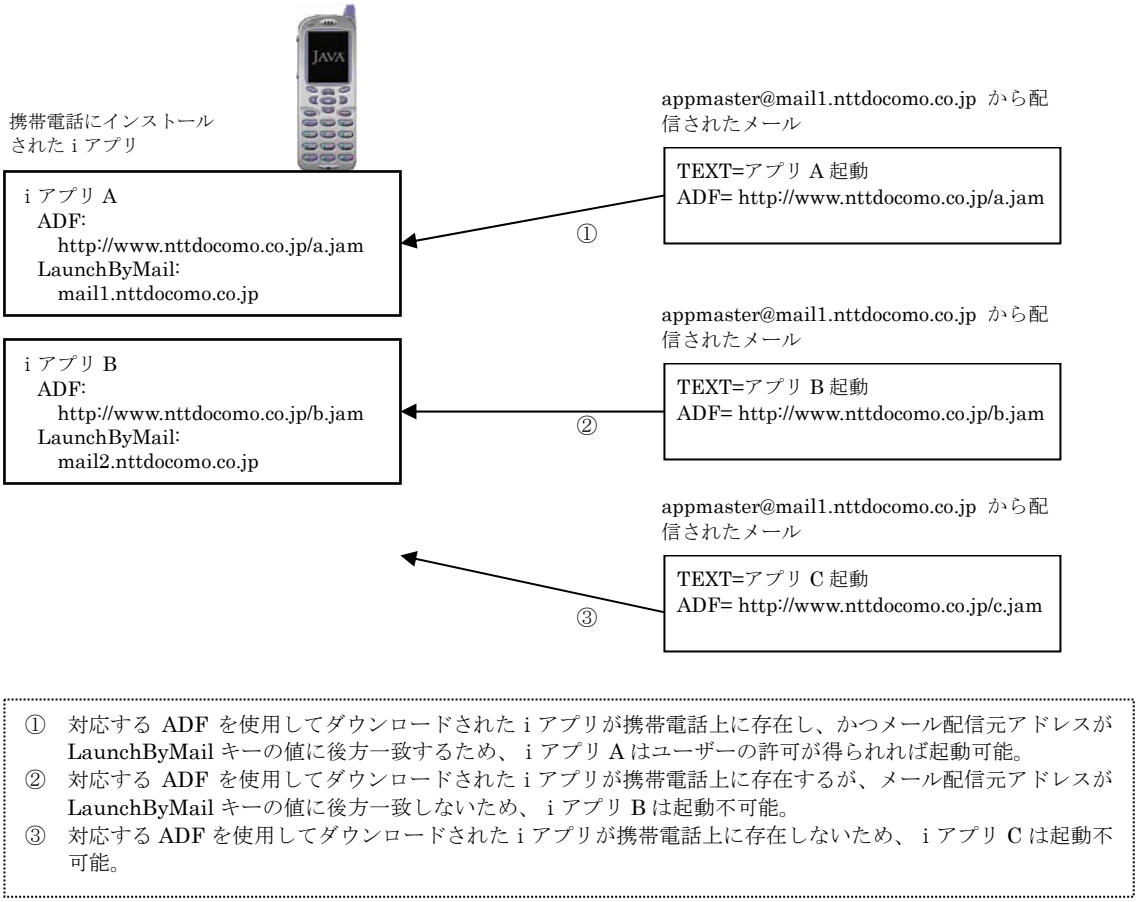
- パラメータ指定キー

  開発者は、キーの名前と値をそれぞれダブルクォートで囲むことにより、ｉアプリに引き渡すことのできるパラメータを定義することができます。このように指定されたパラメータは、IApplication.getParameter()メソッドで取得することができます（ブラウザからのｉアプリ起動における PARAM タグと同等）。

  パラメータ指定キーは、1 つのメール内に最大 16 個含めることができます。また、1 つのメール内で使用されている全パラメータ指定キーの名前の長さと値の長さの合計は最大 255 バイトに制限されます。キーの名前および値には、それぞれ日本語テキストを指定することができます。パラメータに日本語テキストを使用する場合、長さの制限は SJIS 表現に対し適用されます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

以下に、携帯電話にダウンロードされている i アプリと、メールに記述された i アプリ起動情報の内容の関係について図示します。



① 対応する ADF を使用してダウンロードされた i アプリが携帯電話上に存在し、かつメール配信元アドレスが LaunchByMail キーの値に後方一致するため、 i アプリ A はユーザーの許可が得られれば起動可能。
② 対応する ADF を使用してダウンロードされた i アプリが携帯電話上に存在するが、メール配信元アドレスが LaunchByMail キーの値に後方一致しないため、 i アプリ B は起動不可能。
③ 対応する ADF を使用してダウンロードされた i アプリが携帯電話上に存在しないため、 i アプリ C は起動不可能。

図 16: ダウンロード済み i アプリと i アプリ起動情報の記述内容との関係

注意事項：

- 携帯電話ユーザーは、携帯電話の設定によりメールからの i アプリ起動を禁止することができます。またメールから i アプリを起動する前に、 i アプリ実行環境はユーザーに i アプリを起動してよいか確認を行います。これらの機構によりユーザーの許可が得られなかった場合、その i アプリの起動は行われません。
- メールから起動された i アプリの動作が終了（`IApplication.terminate()`により終了）すると、 i アプリを起動する前のメーラの状態（元のメールが表示されている状態）に復帰します。ただし、メールから起動された i アプリをユーザーが強制終了した場合、およびメールから起動された i アプリがアプリケーション連携機能によって他のネイティブアプリケーションの機能の起動または呼び出しを行った場合は i アプリ終了後にどのような状態に移るかはメーカーにより異なります。
- メールに i アプリ起動情報と他の添付情報（バウンダリ識別子でメール本文と区切られた添付情報　メロディ添付など）を同時に設定することはできません。これらを同時に設定した場合は、いずれもが無効なものとして無視されます。
- メールのサイズは、 i アプリ起動情報を含めて i モードメールの制限の範囲内でなければなりません。本機能は i モードメール分割受信には対応しないため、メールの分割が発生しないサイズで送信する必要があります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

11.3 外部機器連携起動

外部機器連携起動では、外部機器から赤外線ポートを介して携帯電話に起動指示を行うことにより、携帯電話上の i アプリの起動を行うことができます。なお、起動対象の i アプリはあらかじめ携帯電話にインストールされていなければなりません。

11.3.1 外部機器からの i アプリの起動

外部機器から i アプリを起動させるためには、開発者は以下の 2 つを行います。

(1) 外部機器上に、赤外線ポートを介して携帯電話へ i アプリの起動を指示するための OBEX アプリケーションを作成・配置します。外部機器側の赤外線通信に関する要件は 10.3 項を参照してください。外部機器からの i アプリ起動指示は、後述の vTrigger オブジェクトを携帯電話に送信することにより行います。

(2) 外部機器から起動させたい i アプリに対応する ADF の AllowPushBy キーにて、 i アプリが特定の起動コマンドを含んだ vTrigger オブジェクトを送信する外部機器からの起動を許可することを宣言します。
AllowPushBy で宣言する起動コマンドと外部機器が送信する起動コマンドが完全に一致していない場合、その i アプリを起動することはできません。AllowPushBy キーは以下のように指定します。ADF の記述方法の詳細については 15.5.1 項を参照してください。

```
AllowPushBy = Irda:<起動コマンド>
```

携帯電話の赤外線ポートは、ユーザーによるメニューやボタンなどの操作により通電されます。この状態でユーザーが外部機器と携帯電話の赤外線ポートを互いに対向させ、vTrigger オブジェクトを送受することにより i アプリが起動されます。

vTrigger とは、 i アプリ対応携帯電話において、外部機器連携起動をサポートするために規定されたオブジェクトフォーマットです。vTrigger は複数のプロパティ（属性名と値のペア）の集合として表現されるテキストデータです。赤外線ポートを介して vTrigger オブジェクトを携帯電話に送信することにより、携帯電話上の i アプリに起動指示を出すことができます。

以下に、vTrigger の書式について解説します。

```
BEGIN:VTRG
VERSION:<vTrigger フォーマットバージョン>
ADFURL:<起動対象 i アプリの ADF URL>
COMMAND:<起動コマンド>
<パラメータ>
END:VTRG
```

※行末は<CR><LF>（0x0d0a）としてください。

記法は以下の通りです。

- BEGIN:VTRG

  vTrigger オブジェクトの開始を表します。

- VERSION:<vTrigger フォーマットバージョン>

  使用している vTrigger オブジェクトのフォーマットバージョンを指定します。現在までのプロファイルにおいては、vTrigger フォーマットバージョンとして"1.0"を指定します。

- ADFURL:<起動対象アプリケーションの ADF URL>

  この vTrigger オブジェクトで起動させたい i アプリに対応する ADF の URL を ASCII 形式で指定します（最大 255 バイト）。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- COMMAND:<起動コマンド>

  起動コマンドを ASCII 形式で指定します（最大 250 バイト）。ここで指定されている起動コマンドと、 i アプリの ADF の AllowPushBy キーで起動許可が与えられている起動コマンド（AllowPushBy キーの値として"irda:"以降に指定されるコマンド文字列）が完全に一致していなければ、 i アプリを起動することはできません。

- <パラメータ>

  起動された i アプリに引き渡すパラメータを指定します（オプショナル）。1 つのパラメータは以下のフォーマットを取ります。

  PARAM [;<エンコード指定>][;<文字セット指定>]:<パラメータ名>;<パラメータ値>

  ※[ ]は省略可能であることを示します。なお、エンコード指定と文字セット指定の位置は入れ替わっていても構いません。

  パラメータ名およびパラメータ値は、 i アプリ起動時に IApplication.getParameter()メソッドにより取得できるパラメータを指定するために使用されます。<パラメータ>は 1 つの vTrigger オブジェクトに最大 16 個まで指定することができます。ただし、全ての<パラメータ>の名前と値の合計長は 255 バイトに制限されます。

  エンコード指定および文字セット指定は以下のように指定します。

  ー エンコード指定

  ENCODE=<エンコード名> の形式で指定します。エンコード名には"QUOTED-PRINTABLE"または"BASE64"のいずれかが指定できます。指定がない場合はエンコード指定が行われていないものと見なされます。

  ー 文字セット指定

  CHARSET=<文字セット名> の形式で指定します。現在までのプロファイルにおいて、有効な文字セット名は"SHIFT_JIS"のみです。また、文字セット指定がない場合も"SHIFT_JIS"が指定されたものと見なされます。

  なお vTrigger オブジェクトでは、ボディの内容は 7 ビットコードで記述されている必要があります。パラメータの名前や値に日本語（SHIFT_JIS）を指定する場合は、必ず QUOTED-PRINTABLE 形式または BASE64 形式でエンコードするようにしてください。エンコードされた日本語パラメータは、指定されたエンコード種別に従って i アプリ実行環境で自動的にデコードされ、さらに Java の String に変換された上で IApplication.getParameter()メソッドに渡されます。

- END:VTRG

  vTrigger オブジェクトの終了を表します。

以下に、vTrigger オブジェクトの記述例を示します。

```
BEGIN:VTRG
VERSION:1.0
ADFURL:http://www.nttdocomo.co.jp/java/a.jam
COMMAND:startup_Application_A
PARAM:Param1;i-mode
PARAM;ENCODING=BASE64;CHARSET=SHIFT_JIS:Param2;gomDQYN2g4o=
END:VTRG
```

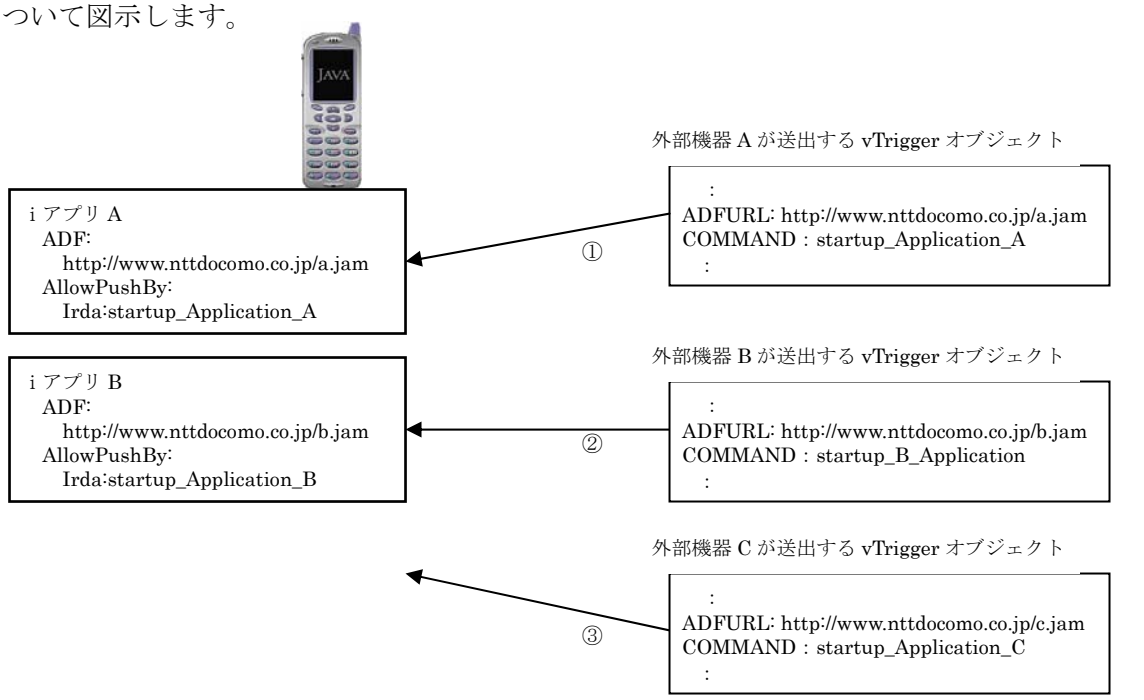
OBEX レイヤにおける vTrigger オブジェクトの送信とは、Body ヘッダに vTrigger オブジェクトの内容を設定し、PUT オペレーションにて送信することをいいます。携帯電話は外部機器からの i アプリ起動要求に対し、 i アプリの起動可否を判断して以下の OBEX レスポンスコードを返します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

レスポンスコード	OBEX における規定	外部機器連携における規定
0xA0	OK, Success	指定された i アプリの起動が可能である
0xC4	Not Found	指定した i アプリが存在しない
0xE1	Database Locked	指定した i アプリの外部機器からの起動が禁止されている、 または AllowPushBy キーの起動コマンドと vTrigger オブジェクトで受信した起動コマンドが一致しない

レスポンスコード 0xA0 が外部機器に返された時点では、携帯電話上ではまだ i アプリは起動されていません。 i アプリの起動は、携帯電話がレスポンスコードを返却し、赤外線通信が終了された後に行われます。

以下に、携帯電話にダウンロードされている i アプリと、vTrigger オブジェクトの記述内容の関係について図示します。



① 対応する ADF を使用してダウンロードされた i アプリが携帯電話上に存在し、かつ COMMAND の値が AllowPushBy キーの値に完全一致するため、 i アプリ A は起動可能。
携帯電話を外部機器の赤外線ポートにかざすことがユーザーの i アプリ起動への同意となるため、携帯電話（i アプリ実行環境）は i アプリの起動確認は行いません。

② 対応する ADF を使用してダウンロードされた i アプリが携帯電話上に存在するが、COMMAND の値が AllowPushBy キーの値に完全一致しないため、 i アプリ B は起動不可能。

③ 対応する ADF を使用してダウンロードされた i アプリが携帯電話上に存在しないため、 i アプリ C は起動不可能。

図 17: ダウンロード済み i アプリと vTrigger オブジェクトの記述内容との関係

注意事項：

- 携帯電話ユーザーは、携帯電話の設定により外部機器からの i アプリ起動を禁止することができます。外部機器からの起動が禁止されている場合、vTrigger オブジェクトを受信しても i アプリの起動は行われません。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- 外部機器から起動コマンド（vTrigger オブジェクト）を受信するためには、ユーザーはあらかじめ携帯電話の赤外線ポートへの通電操作を行っている必要があります。赤外線ポートへの通電操作の方法はメーカーにより異なります。
- 外部機器から起動された i アプリの動作が終了すると、待ち受け状態に復帰します。ただし、外部機器から起動された i アプリをユーザーが強制終了した場合、 i アプリ終了後にどのような状態に移るかはメーカーにより異なります。
- 外部機器から vTrigger オブジェクトを携帯電話に送信する際、その vTrigger オブジェクトの Name ヘッダの値は末尾（拡張子）が`".vtg"`となるようにしてください。携帯電話は、受信したオブジェクトの名前の拡張子が`".vtg"`である場合に、そのオブジェクトを vTrigger オブジェクトと認識します。

11.4 i アプリ連携起動

i アプリ連携起動では、アプリケーションプログラムからの API 呼び出しによって他の i アプリの起動を行うことができます。 i アプリ連携起動には、さらに以下の 2 種類の形態があります。

- 連携モード

  連携モードでは i アプリが他の i アプリを起動する際に、起動先 i アプリを ADF の URL により直接指定することができます。また、起動元 i アプリは起動先 i アプリにパラメータを渡すことができます。ただし連携モードでは、起動元 i アプリと起動先 i アプリは同一ホストからダウンロードされたものでなければなりません。

- ランチャーモード

  ランチャーモードでは連携モードと異なり、起動元 i アプリとはダウンロード元ホストが異なる i アプリでも起動することができます。ただし i アプリを起動する際、起動先 i アプリは ADF の URL ではなく、携帯電話内にダウンロードされている個々の i アプリに対応付けられたエントリ ID を使用して指定します。このため、ランチャーモードで i アプリを起動するには、ターゲットとする起動先 i アプリのエントリ ID をあらかじめ取得する必要があります。 i アプリのエントリ ID の取得は、API 呼び出しによりユーザーの確認の下で行われます。ランチャーモードにおいては、起動元 i アプリから起動先 i アプリにパラメータを渡すことはできません。

なお、起動対象の i アプリはあらかじめ携帯電話にインストールされていなければなりません。

11.4.1 連携モードによる i アプリ連携起動

i アプリを連携モードで起動するには、`IApplication.launch()`メソッドを使用します。 i アプリを連携モードで起動する場合、このメソッドの第 1 引数（起動対象となるアプリケーションの種別）に `IApplication.LAUNCH_IAPPLI` を、また第 2 引数（起動引数を示す `String` 配列）の先頭要素に起動したい i アプリの ADF の URL を指定します。

起動先 i アプリにパラメータを渡す場合には、第 2 引数の `String` 配列の 2 番目以降（配列添え字が 1 以上）の要素を使用します。起動元 i アプリから渡されたパラメータは、起動先 i アプリ側で `IApplication.getParameter()`メソッドを使用して取得することができます。1 つのパラメータは、変数名と値を示す 2 つの配列要素から構成されます。以下に、起動元 i アプリと起動先 i アプリでパラメータを授受する例を示します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

※起動元 i アプリでのパラメータ設定

```
String[] args = new String[5];
// 起動先 i アプリの ADF URL 設定
args[0] = "http://www.nttdocomo.co.jp/app.jam"

// パラメータ設定
args[1] = "Param1";  // 1 つ目のパラメータの名前
args[2] = "i-mode";  // 1 つ目のパラメータの値
args[3] = "Param2";  // 2 つ目のパラメータの名前
args[4] = "ｉアプリ"; // 2 つ目のパラメータの値

// ｉアプリ連携起動（連携モード）
IApplication.getCurrentApp().launch(
         IApplication.LAUNCH_IAPPLI, args);
```

起動

※起動先 i アプリでのパラメータ取得

```
String param1, param2;

// param1 には"i-mode"が格納される
param1 = IApplication.getCurrentApp().
         getParameter("Param1");

// param2 には"ｉアプリ"が格納される
param2 = IApplication.getCurrentApp().
         getParameter("Param2");
```

図 18：連携モードにおける起動元 i アプリから起動先 i アプリへのパラメータ授受

パラメータは最大 16 個（パラメータ名と値の組み合わせで 16 セット）まで指定することができます。また、指定するパラメータ名と値の長さの合計は最大 20480 バイトに制限されます（バイト数はデフォルトエンコーディングで評価されます）。

なお i アプリ連携起動を行う i アプリは、ADF の LaunchApp キーで i アプリ連携起動を行うことを宣言（値"yes"を設定）する必要があります。この宣言を行っていない i アプリは、i アプリ連携起動を行うことはできません。

注意事項：

- 起動元 i アプリの ADF URL と起動先 i アプリの ADF URL のホスト名は同一でなければなりません。また、起動元 i アプリと同一の ADF URL を持つ i アプリを起動することはできません。
- ダウンロード即起動 i アプリ、および非活性化状態の待ち受けアプリケーションからはこの機能を使用することはできません。
- 起動先 i アプリが起動される際、起動元 i アプリは終了します。
- 起動先 i アプリの動作が終了すると、携帯電話は待ち受け状態に復帰します。ただし、起動先 i アプリをユーザーが強制終了した場合、i アプリ終了後にどのような状態に移るかはメーカーにより異なります。
- 起動先 i アプリが待ち受けアプリケーションであっても、i アプリ連携起動で起動された場合は待ち受け起動ではなく通常起動となります。
- 起動元 i アプリがトラステッド i アプリの場合、起動先 i アプリもトラステッド i アプリでなければなりません。

【DoJa-5.0】

DoJa-4.x プロファイル以前のプロファイルでは、i アプリを連携起動しようとした際には起動元 i アプリが一旦サスペンド状態に入り、i アプリの起動確認を行うためのダイアログがシステムにより表示されます。これに対し DoJa-5.0 プロファイル以降では、この起動確認は行われません。

【DoJa-5.1】

連携モードでの i アプリ連携起動について、起動元 i アプリから起動先 i アプリに渡すことのできるパラメータの長さに関する制限は DoJa-5.1 プロファイルにて緩和されました。DoJa-5.1 プロファイル以降ではパラメータ名と値の長さの合計で 20480 バイトまでのパラメータを渡すことができますが、DoJa-5.0 プロファイル以前ではこの制限は 255 バイトまでとなっています。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

11.4.2 ランチャーモードによる i アプリ連携起動

ランチャーモードによる i アプリ連携起動を行う場合、開発者は以下の 2 つの処理を起動元 i アプリ（以下ランチャー i アプリとします）に実装します。

なお連携モードと同様、ランチャーモードでの i アプリ連携起動を行う場合も、ADF の LaunchApp キーで i アプリ連携起動を行うことを宣言（値"yes"を設定）する必要があります。

(A) 起動先 i アプリのエントリ ID の取得

ランチャーモードで i アプリを起動するには、その携帯電話にダウンロードされている各 i アプリと 1 対 1 に対応付けられているエントリ ID を使用します。エントリ ID はランダムな整数値であり、有効なエントリ ID を得るためにはユーザーによる i アプリ選択プロセスを経る必要があります（アプリケーションプログラムが無効なエントリ ID を指定して i アプリの起動を試みた場合、 i アプリ実行環境はユーザーに警告を表すダイアログを表示します）。 i アプリ選択プロセスは、com.nttdocomo.system.ApplicationStore クラスを使用して行います。このクラスの static メソッド selectEntry()を呼び出すと、 i アプリはサスペンドして画面上に i アプリ一覧が表示されます。その i アプリ一覧からユーザーがいずれかの i アプリを選択すると i アプリはレジュームし、selectEntry()メソッドは選択された i アプリに対応する ApplicationStore オブジェクトを返します。返却された ApplicatoinStore オブジェクトに対して getId()メソッドを呼び出すことで、選択された i アプリのエントリ ID を取得することができます。

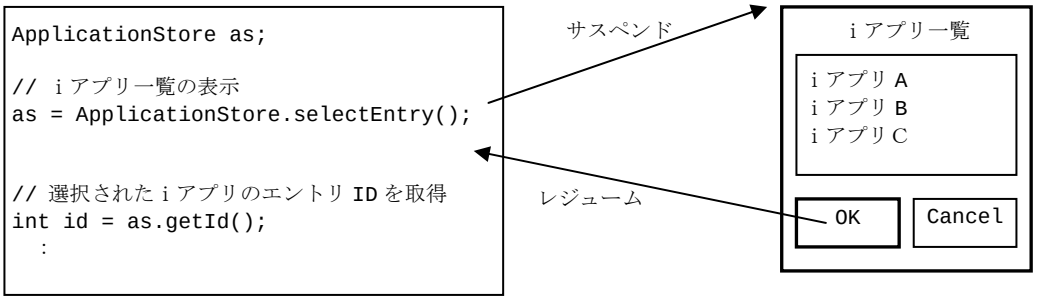


図 19: i アプリのエントリ ID の取得

i アプリに対応付けられたエントリ ID は、その i アプリが携帯電話上に存在する限り有効であり、 i アプリが更新された場合でも更新後の i アプリに引き継がれます。

なお、ユーザーが選択した i アプリについて、エントリ ID 以外の情報（例えば i アプリ名など）を取得することはできません。

(B) エントリ ID を指定しての i アプリの起動

エントリ ID を指定してランチャーモードで i アプリを起動するには、IApplication.launch()メソッドを使用します。launch()メソッドの第 1 引数（起動対象となるアプリケーションの種別）には IApplication.LAUNCH_AS_LAUNCHER を、また第 2 引数（起動引数を示す String 配列）の先頭要素に文字列化したエントリ ID を指定します。

連携モードと異なり、ランチャーモードでは起動先 i アプリにパラメータを渡すことはできません。

注意事項：

- ダウンロード即起動 i アプリ、および非活性化状態の待ち受けアプリケーションからはこれらの機能を使用することはできません。
- 起動先 i アプリが起動される際、起動元 i アプリは終了します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- 起動先 i アプリの動作が終了すると、携帯電話は待ち受け状態に復帰します。ただし、起動先 i アプリをユーザーが強制終了した場合、 i アプリ終了後にどのような状態に移るかはメーカーにより異なります。
- 起動先 i アプリが待ち受けアプリケーションであっても、 i アプリ連携起動で起動された場合は待ち受け起動ではなく通常起動となります。
- 通常起動されることを想定していない待ち受けアプリケーションなど、 i アプリの内容によってはランチャーモードで起動されることが適切でない場合があります。そのような i アプリは、ランチャーモードで連携起動されることを拒否することができます。その場合、ADF の `LaunchByApp` キーで連携起動の拒否を宣言（値`"deny"`を設定）します。この設定が行われている i アプリは、 i アプリ選択プロセスでの選択対象外となります。
- i アプリ選択プロセスにおいて、選択可能な i アプリが携帯電話上に存在しない場合やユーザーが選択操作をキャンセルした場合は、`ApplicationStore.selectEntry()`メソッドは null を返却します。
- 新規にダウンロードされた i アプリにはその時点でランダムなエントリ ID が割り当てられ、その i アプリを削除しない限り（ i アプリの更新を行っても）有効なまま存続します。ダウンロードされた i アプリを削除して再度ダウンロードした場合は、新規ダウンロードと見なされ再度ランダムなエントリ ID が割り当てられます。

【DoJa-5.0】

DoJa-4.x プロファイル以前のプロファイルでは、 i アプリを連携起動しようとした際には起動元 i アプリが一旦サスペンド状態に入り、 i アプリの起動確認を行うためのダイアログがシステムにより表示されます。これに対し DoJa-5.0 プロファイル以降では、この起動確認は行われません。

11.5 i アプリ更新機能連携起動

i アプリ更新機能連携起動では、 i アプリから JAM の持つ i アプリ更新機能を起動することができます。JAM の i アプリ更新機能とは、携帯電話の i アプリ一覧のサブメニューなどで提供される「バージョンアップ」項目に相当する機能です。 i アプリはこの機能を利用して、自身の更新をユーザーに促すことができます。

i アプリ更新機能を起動するには、`IApplication.launch()`メソッドを使用します。第 1 引数（起動対象となるアプリケーションの種別）には `IApplication.LAUNCH_VERSIONUP` を、また第 2 引数（起動引数を示す `String` 配列）には null を指定します。

i アプリ更新機能を起動すると、携帯電話は以下のような動作を行います。

(1) アプリケーションプログラムから `IApplication.launch()`メソッドを呼び出します。このメソッドが呼び出されると i アプリはサスペンドします。

(2) i アプリ実行環境は、パケット通信が可能であること（携帯電話のディスプレイに、通信状態を示す"i"マークが点灯または点滅している状態）を確認し、パケット通信可能であれば、ユーザーに i アプリ更新の確認を求めます。パケット通信が不可能な場合やユーザーの同意が得られなかった場合、 i アプリはレジュームし、`IApplication.launch()`メソッドは例外をスローすることなく呼び出し元に復帰します。

(3) `(2)`でユーザーの同意が得られると、 i アプリは終了して i アプリ更新機能が起動し、呼び出し元 i アプリに対応する ADF のダウンロードが行われます。ADF をダウンロードした結果、その i アプリを更新する必要がないと判断された場合（ADF の `LastModified` キーが更新されていないなど）は i アプリの更新は行われません。

(4) JAR ファイルをダウンロードし、携帯電話上の i アプリを更新します。その後、ユーザー確認が必要な ADF 項目（HTTP 通信の有無など）の内容に変更があれば、それらの個別設定画面に移ります。

(5) JAR ファイルのダウンロードおよび個別設定が完了すると、更新後の i アプリが自動的に再起動されます。待ち受け起動された待ち受けアプリケーションがこの機能を使用して更新された場合は、この再起動も待ち受け起動となります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

注意事項：

- ダウンロード即起動 i アプリ、および非活性化状態の待ち受けアプリケーションからは i アプリ更新機能を連携起動することはできません。

11.6 通話機能の呼び出し

通話機能の呼び出しでは、i アプリから携帯電話の通話発信機能（ダイヤラー）を呼び出すことができます。また、その携帯電話のハードウェアが持つ個体識別情報を参照することもできます。

11.6.1 i アプリからの通話発信

i アプリから通話発信機能を呼び出すには、`com.nttdocomo.util.Phone` クラスの `call()`メソッドを使用します。このメソッドを、引数に電話番号を表現する文字列を指定して呼び出すことにより通話発信機能が呼び出されます。テレビ電話をサポートしている FOMA 携帯電話では、引数の指定によりテレビ電話の発信を行わせることもできます。また、プッシュトークをサポートしている携帯電話では、呼び出された通話発信機能を操作することによってプッシュトーク発信を行うこともできます。

電話番号文字列には、`0`～`9` までの数字、 `'#'` 、 `'*'` 、 `','` （ポーズ：1 秒間）、 `'/'` （ポーズ：キー入力待ち）を指定することができます。ポーズ指定は電話番号文字列中に連続して指定することはできません。また、電話番号文字列中に現れる `'-'` 、 `'('` 、 `')'` 、 `'.'` は無視されます。

なお、通話発信機能を使用する i アプリは、ADF の `UseTelephone` キーで通話発信機能を使用することを宣言（値`"call"`を設定）する必要があります。この宣言を行っていない i アプリは通話発信機能を使用することはできません。

注意事項：

- 非活性化状態にある待ち受けアプリケーションは、通話発信機能を起動することはできません。
- `Phone.call()`メソッドを呼び出すと、i アプリはサスペンド状態に入ります。その後、i アプリ実行環境がユーザーに求める発信確認に対しユーザーが同意することによりダイヤル操作が行われます。通話が終了すると i アプリはレジュームし、`call()`メソッドは呼び出し元に復帰します。

11.6.2 i アプリからの個体識別情報の参照

個体識別情報とは、デバイスの各個体に割り当てられるユニークな ID です。携帯電話の個体識別情報には以下の 2 種類があり、それぞれ同じ値が複数の個体に割り当てられることはありません。

- 携帯電話本体の識別情報（製造番号） ・・・ `Phone.TERMINAL_ID`
- UIM カードを搭載する携帯電話での、UIM カードの識別情報（製造番号） ・・・ `Phone.USER_ID`

これらの情報は、それ単独でユーザー個人を特定するなどの用途に使用することはできません（これらの情報を元にユーザー個人を特定する手段はありません）。しかし適切なユーザー管理システムとこれらの情報を組み合わせて利用することで、サーバーへのアクセス管理などのシステム管理を容易に実現することができます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

個体識別情報を取得するには、com.nttdocomo.util.Phone クラスの static メソッド getProperty()を使用します。このメソッドの引数には、上記のうちいずれの個体識別情報を取得するかを指定します。

なお、 i アプリが個体識別情報を参照するには、ADF の GetUtn キーを使用して個体識別情報を参照することを宣言しなければなりません。GetUtn キーに値"terminalid"を指定すると携帯電話本体の識別情報を、値"userid"を指定すると UIM カードの識別番号を参照することができます。両方を参照する場合は、これら 2 つをカンマ区切りで併記します。

注意事項：

- ダウンロード即起動 i アプリからは、個体識別情報を参照することはできません。
- UIM カードを搭載していない携帯電話では、UIM カードの識別情報を取得することはできません。

11.7 電話帳管理機能の呼び出し

電話帳管理機能の呼び出しでは、i アプリから携帯電話のネイティブの電話帳管理機能を呼び出し、ユーザーの同意の下で電話帳エントリ（電話番号、氏名、メールアドレスなどのセット）の新規登録および電話帳グループの新規登録を行なうことができます。

なお、これらの機能を使用する i アプリは、ADF の AccessUserInfo キーにてこれらの機能を使用することを宣言（値"yes"を設定）する必要があります。

11.7.1 電話帳グループの新規登録

電話帳グループを新規登録するには、com.nttdocomo.system.PhoneBookGroup クラスの static メソッド addEntry()を使用します。このメソッドの引数には、登録したい電話帳グループの名前または null を指定します。

電話帳グループの新規登録を行うと、携帯電話は以下のような動作を行います。

(1) アプリケーションプログラムから PhoneBookGroup.addEntry()メソッドを呼び出します。このメソッドが呼び出されると i アプリはサスペンドします。

(2) 電話帳グループの登録を行うためのネイティブのユーザーインタフェース画面が起動します。この画面では、ユーザーは i アプリからの電話帳グループの新規登録を許可、キャンセルするだけでなく、登録するグループ名を変更することもできます。

(3) ユーザーが電話帳グループの新規登録を許可するとそのときに設定されていた名前で電話帳グループの新規登録が行われ、i アプリはレジュームして addEntry()メソッドはアプリケーションプログラムに復帰します。その際このメソッドは、新規登録された電話帳グループ名にその携帯電話内で 1 対 1 に対応するエントリ ID を返します。エントリ ID は、後述する電話帳エントリの新規登録において、電話帳エントリのグループ指定を行うために利用することができます。

ユーザーが電話帳グループの新規登録をキャンセルしても例外は発生せず、このメソッドは-1 を返します。

注意事項：

- ダウンロード即起動 i アプリ、および非活性化状態の待ち受けアプリケーションからは電話帳グループの新規登録を行うことはできません。
- メーカーによっては電話帳グループの新規登録は、グループの追加ではなく携帯電話にプリセットされたグループの名称変更により行われる場合があります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

11.7.2 電話帳エントリの新規登録

電話帳エントリを新規登録するには、com.nttdocomo.system.PhoneBook クラスの static メソッド addEntry()を使用します。このメソッドは、電話帳エントリに登録する情報として以下のものをサポートします。

- 名前
- 名前の読み仮名
- 電話番号（複数指定可能）
- メールアドレス（複数指定可能）
- 登録先の電話帳グループ（グループ名またはグループのエントリ ID を指定）

これらの情報をパラメータとして addEntry()メソッドに与えるには、以下の 2 つの方法があります。

(1) これらの項目を、addEntry()メソッドの引数として列挙します。

(2) 電話帳エントリに指定する各項目をカプセル化する com.nttdocomo.system.PhoneBookParam オブジェクトを作成し、addEntry()メソッドの引数に指定します。

電話帳エントリの新規登録を行うと、携帯電話は以下のような動作を行います。

(1) アプリケーションプログラムから PhoneBook.addEntry()メソッドを呼び出します。このメソッドが呼び出されると i アプリはサスペンドします。

(2) 電話帳エントリの登録を行うためのネイティブのユーザーインタフェース画面が起動します。この画面では、ユーザーは i アプリからの電話帳エントリの新規登録を許可、キャンセルするだけでなく、登録するエントリの各項目を変更することもできます。

(3) ユーザーが電話帳エントリの新規登録を許可するとそのときに設定されていた名前で電話帳エントリの新規登録が行われ、i アプリはレジュームして addEntry()メソッドはアプリケーションプログラムに復帰します。

ユーザーが電話帳エントリの新規登録をキャンセルしても、特に例外は発生せずアプリケーションプログラムに復帰します。

注意事項：

- ダウンロード即起動 i アプリ、および非活性化状態の待ち受けアプリケーションからは電話帳エントリの新規登録を行うことはできません。
- 1 つの電話帳エントリに対して登録可能な電話番号やメールアドレスの数は機種により異なりますが、どの機種でも各々最低 3 つまでは登録可能です。
- 機種によっては、電話帳エントリの名前および読み仮名の項目を姓と名に分けて管理できるものがあります。PhoneBookParam オブジェクトを使用しない場合、そのような機種では名前、読み仮名とも姓の方にまとめて設定されます。

  PhoneBookParam オブジェクトを使用する場合は、名前や読み仮名を姓と名に分けて設定することができます。
- 電話帳グループ名に null を設定すること、および電話帳グループのエントリ ID に-1 を設定することは、登録先の電話帳グループを指定しないことを意味します。この場合、その携帯電話の標準の場所に電話帳エントリが登録されます。
- 電話帳グループのエントリ ID を指定して電話帳エントリの登録を行う際、指定したエントリ ID に対応する電話帳グループが存在しない場合は例外が発生します。これに対して電話帳グループの名前を指定して電話

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

帳エントリの登録を行う際、その名前の電話帳グループが存在しない場合は、 i アプリ実行環境はユーザーにその電話帳グループを作成するかどうかを問い合わせます。

11.8 ブックマーク管理機能の呼び出し

ブックマーク管理機能の呼び出しでは、i アプリから携帯電話のネイティブのブックマーク管理機能を呼び出し、ユーザーの同意の下でブックマーク（接続先 URL とタイトル）の新規登録を行なうことができます。

なお、ブックマーク管理機能を使用する i アプリは、ADF の `AccessUserInfo` キーにてこの機能を使用することを宣言（値`"yes"`を設定）する必要があります。

ブックマークを新規登録するには、`com.nttdocomo.system.Bookmark` クラスの static メソッド `addEntry()`を使用します。このメソッドの引数には、登録したいブックマークのタイトルおよび接続先 URL を指定します。

ブックマークの新規登録を行うと、携帯電話は以下のような動作を行います。

(1) アプリケーションプログラムから `Bookmark.addEntry()`メソッドを呼び出します。このメソッドが呼び出されると i アプリはサスペンドします。

(2) ブックマークの登録を行うためのネイティブのユーザーインタフェース画面が起動します。この画面では、ユーザーは i アプリからのブックマークの新規登録を許可、キャンセルするだけでなく、登録する URL やタイトルを変更することもできます。

(3) ユーザーがブックマークの新規登録を許可するとそのときに設定されていた内容でブックマークの新規登録が行われ、 i アプリはレジュームして `addEntry()`メソッドはアプリケーションプログラムに復帰します。

ユーザーがブックマークの新規登録をキャンセルしても、特に例外は発生せずアプリケーションプログラムに復帰します。

注意事項：

- ダウンロード即起動 i アプリ、および非活性化状態の待ち受けアプリケーションからはブックマークの新規登録を行うことはできません。
- ブックマークとして登録可能な URL のスキームは HTTP または HTTPS のみです。

11.9 スケジュール管理機能の呼び出し

スケジュール管理機能の呼び出しでは、i アプリから携帯電話のネイティブのスケジュール管理機能（メーカーによってはカレンダーなど別の名称で呼んでいる場合があります）を呼び出し、ユーザーの同意の下でスケジュール（説明文や日時）の新規登録を行なうことができます。また、DoJa-5.0 プロファイル以降では、i アプリからスケジューラを連携起動することもできます（スケジューラの連携起動は、DoJa-4.x プロファイル以前では i アプリオプション API の位置づけとなっています）。

なお、スケジュール管理機能を使用する i アプリは、ADF の `AccessUserInfo` キーにてこの機能を使用することを宣言（値`"yes"`を設定）する必要があります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

11.9.1 スケジュールの新規登録

スケジュールを新規登録するには、com.nttdocomo.system.Schedule クラスの static メソッド addEntry()を使用します。このメソッドの引数には、登録したいスケジュールの説明文、日時およびアラーム音の有無を指定します。

スケジュールの日時指定には、com.nttdocomo.util パッケージの ScheduleDate クラスを使用します。ScheduleDate クラスでは、1 回限りのスケジュール、および毎日、毎週、毎月、毎年といった定期的なスケジュールを表現（スケジュールの日時指定タイプの設定）することができます。

スケジュールの新規登録を行うと、携帯電話は以下のような動作を行います。

(1) アプリケーションプログラムから Schedule.addEntry()メソッドを呼び出します。このメソッドが呼び出されると i アプリはサスペンドします。

(2) スケジュールの登録を行うためのネイティブのユーザーインタフェース画面が起動します。この画面では、ユーザーは i アプリからのスケジュールの新規登録を許可、キャンセルするだけでなく、登録する説明文などを変更することもできます。

(3) ユーザーがスケジュールの新規登録を許可するとそのときに設定されていた内容でスケジュールの新規登録が行われ、 i アプリはレジュームして addEntry()メソッドはアプリケーションプログラムに復帰します。

ユーザーがスケジュールの新規登録をキャンセルしても、特に例外は発生せずアプリケーションプログラムに復帰します。

注意事項：

- ダウンロード即起動 i アプリ、および非活性化状態の待ち受けアプリケーションからはスケジュールの新規登録を行うことはできません。
- スケジュールの登録において、どのような日時指定タイプが使用可能かはメーカーにより異なります。どの機種でも共通的に指定可能な日時指定タイプは「1 回限り」です。また、特定の機種でどのような日時指定タイプが使用できるかは Schedule.getSupportedTypes()メソッドを使用して調べることができます。

【DoJa-5.1】

DoJa-5.1 プロファイルにおいて、ScheduleDate クラスのコンストラクタに、タイムゾーン（java.util.TimeZone クラスのオブジェクト）を指定できるバリエーションが追加されました。

11.9.2 スケジューラの連携起動

i アプリからスケジューラを起動するには、IApplication.launch()メソッドを使用します。スケジューラを起動するには、このメソッドの第 1 引数（起動対象となるアプリケーションの種別）に IApplication.LAUNCH_SCHEDULER を指定します。スケジューラ連携起動においては launch()メソッドの第 2 引数（起動引数を示す String 配列）は使用しないため、第 2 引数には null を指定してください。

注意事項：

- 非活性化状態の待ち受けアプリケーションからは、スケジューラを起動することはできません。
- IApplication.launch()メソッドを呼び出すと、 i アプリはサスペンド状態に入ります。その後、 i アプリ実行環境がユーザーに求めるスケジューラ起動確認に対しユーザーが同意することによりスケジ

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

ューラ起動が行われます。スケジューラ連携起動では、スケジューラの起動が行われてもiアプリは終了しません（サスペンド状態を保ちます）。ユーザー操作によりスケジューラが終了すると、iアプリはレジュームし launch()メソッドは制御をアプリケーションプログラムに返します。

11.10 画像データ管理機能の呼び出し

DoJa-3.0 プロファイル以降に対応した携帯電話は、マイピクチャと呼ばれるネイティブの画像管理領域を持っています。画像データ管理機能の呼び出しでは、iアプリから携帯電話のネイティブの画像データ管理機能を呼び出し、マイピクチャ領域からの画像の入出力を行うことができます。画像データ管理機能の呼び出しには、com.nttdocomo.system.ImageStore クラスを使用します。

なお、これらの機能を使用するiアプリは、ADF の AccessUserInfo キーにてこれらの機能を使用することを宣言（値"yes"を設定）する必要があります。また、iアプリはブラウザを使用してダウンロードおよび保存された画像にはアクセスすることはできません。iアプリからアクセス可能な画像は以下に限定されます。

- その携帯電話のカメラ機能により撮影・保存された画像
- iアプリにより保存された画像（他のiアプリから保存された画像を含みます。ただしそのiアプリが画像を保存する際に、他のiアプリからその画像にアクセスすることを禁じている(*1)場合を除きます。）
- ネイティブの赤外線通信ツールにより、その携帯電話の外部から取り込まれた画像
- 携帯電話出荷時にプレインストールされている画像
- メモリカードなどの外部メモリを搭載している携帯電話では、外部メモリからマイピクチャ領域に移された画像
- FOMA 携帯電話においては、ブラウザからダウンロード・保存された画像のうちコンテンツプロバイダにより再配布可能と設定されているもの

(*1) DoJa-4.0 プロファイルより、iアプリからネイティブの画像管理領域に画像を保存する際に、他のiアプリからその画像にアクセスすることを禁止することができる API が追加されています。詳細は 11.10.1 項を参照してください。

注意事項：

- FOMA 携帯電話ではネイティブの画像データ管理機能で映像データ（iモーション形式データ）を保存、管理できるものがありますが、iアプリから本機能を使用してiモーション形式データを取り扱うことはできません。本機能では、静止画像のみが取り扱い対象となります。

11.10.1 画像データの新規登録

画像データを新規登録するには、ImageStore クラスの static メソッド addEntry()を使用します。このメソッドの引数には、登録したい画像を表す MediaImage オブジェクトを指定します。

画像データの新規登録を行うと、携帯電話は以下のような動作を行います。

(1) アプリケーションプログラムから ImageStore.addEntry()メソッドを呼び出します。このメソッドが呼び出されるとiアプリはサスペンドします。

(2) 画像データの登録を行うためのネイティブのユーザーインタフェース画面が起動します。この画面では、ユーザーはiアプリからの画像データの新規登録を許可、キャンセルすることができます。また、iアプリがどのような画像データを登録しようとしているかを確認することもできます。

(3) ユーザーが画像データの新規登録を許可すると画像データの新規登録が行われ、iアプリはレジュームして addEntry()メソッドはアプリケーションプログラムに復帰します。その際このメソッドは、新規登録された画像データにその携帯電話内で 1 対 1 に対応するエントリ ID を返します。エントリ ID は、後述する画像

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

データの ID 指定読み込みにおいて、ユーザー操作なしに画像データを読み込むために使用することができます。

ユーザーが画像データの新規登録をキャンセルしても例外は発生せず、このメソッドは-1 を返します。

【DoJa-4.0】

DoJa-4.0 プロファイル以降、 i アプリからネイティブの画像管理領域に画像を保存する際に、他の i アプリからのその画像へのアクセスを禁止する API（`ImageStore.addEntry(MediaImage, boolean)`メソッド）が追加されています。このメソッドを使用して画像を保存した場合、他の i アプリからその画像にアクセスしたり、携帯電話ネイティブの機能を使用してその画像を携帯電話外部に持ち出すことが禁止されます。

また、DoJa-4.0 プロファイルでは、複数の画像を一度の操作（ユーザー確認等）で保存するための `ImageStore.addEntry()`メソッドのバリエーション（`ImageStore.addEntry(MediaImage[])`）が追加されました。ただし `ImageStore.addEntry(MediaImage[])`は i アプリオプション API に属するメソッドであり、メーカーによってはサポートされない場合があります。

【DoJa-4.1】

DoJa-4.1 プロファイル以降、メーカーによっては Flash コンテンツをネイティブの画像管理領域に保存することをサポートする場合があります。このような機種では、`MediaManager.getImage()`メソッドの引数に Flash コンテンツのデータを指定して呼び出すことで `MediaImage` オブジェクトを取得し、その `MediaImage` オブジェクトを `ImageStore.addEntry()`メソッドの引数に指定することで Flash コンテンツを保存します。

なお、Flash コンテンツのネイティブ画像管理領域への保存をサポートする機種であっても、Flash コンテンツから生成した `MediaImage` オブジェクトを上記以外の目的（例えば `VisualPresenter` で再生するなど）で使用することはできません。また、以下に解説する画像データの選択読み込みや画像データの ID 指定読み込みで、ネイティブ画像管理領域に保存されている Flash コンテンツを取り扱うことはできません`(*1)`。

`(*1)` DoJa-5.1 プロファイル以降では、メーカーの実装選択によっては、ネイティブ画像管理領域に保存されている Flash コンテンツを画像データの選択読み込みや画像データの ID 指定読み込みで取り出すことができるようになりました。ただし、取り出された Flash コンテンツを再生することは、従来同様サポートされません。

【DoJa-5.0】

903i シリーズ以降の携帯電話では、デコメールに対して小さな画像（20x20 ドットの JPEG 画像または GIF 画像）を i モード絵文字風にインライン埋め込みするデコメ絵文字の機能が搭載されています。メーカーによっては、 i アプリから以下の両方の条件を満たす画像を新規登録しようとした場合、その画像はデコメ絵文字用の画像フォルダに格納されることがあります。

- 20x20 ドットの GIF 画像（アニメ-ション GIF を含む）もしくは JPEG 画像である。
- 再配布可能である（再配布不可の設定が行われていない）。

注意事項：

- ダウンロード即起動 i アプリ、および非活性化状態の待ち受けアプリケーションからは画像データの新規登録を行うことはできません。
- `addEntry()`メソッドに指定する `MediaImage` オブジェクトは、使用可能な状態（`use()`メソッドが呼び出された状態）でなければなりません。
- `ImageStore.addEntry(MediaImage)`メソッドを使用して登録した画像データは、携帯電話ネイティブの機能（赤外線通信や i ショットメール送信など）により携帯電話の外部に保存することができます。携帯電話の外部に保存されることが好ましくない画像データは、以下のいずれかの方法で保存することを検討してください。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- 画像データを、マイピクチャ領域ではなく ScratchPad に保存する。
- 画像データを、DoJa-4.0 プロファイルにて新設された `ImageStore.addEntry(MediaImage, boolean)`メソッドを使用してマイピクチャ領域に保存する。
- 画像データに対応する `MediaImage` オブジェクトを、DoJa-4.0 プロファイルにて新設された `MediaResource.setRedistributable()`メソッドにより明示的に再配布不可と設定した上で、`ImageStore.addEntry(MediaImage)`メソッドを使用してマイピクチャ領域に保存する。

11.10.2 画像データの選択読み込み

画像データの選択読み込みとは、アプリケーションプログラムからの画像データ要求（メソッド呼び出し）に対し、その都度画面に表示される画像一覧などからユーザーが選択操作した画像データを与える機能です。画像データの選択読み込みを行うには、`ImageStore` クラスの static メソッド `selectEntry()`を使用します。

画像データの選択読み込みを行うと、携帯電話は以下のような動作を行います。

(1) アプリケーションプログラムから `ImageStore.selectEntry()`メソッドを呼び出します。このメソッドが呼び出されると i アプリはサスペンドします。

(2) 画像データの選択を行うためのネイティブのユーザーインタフェース画面が起動します。この画面を使用して、ユーザーは i アプリにアクセスを行わせる画像データを選択します。また、操作をキャンセルして処理を中止することもできます。

(3) ユーザーが選択する画像データを決定すると、 i アプリはレジュームして `selectEntry()`メソッドはアプリケーションプログラムに復帰します。その際このメソッドは、選択された画像データに対応する `ImageStore` オブジェクトを返します。

ユーザーが画像データの選択操作をキャンセルしても例外は発生せず、このメソッドは null を返します。

`selectEntry()`メソッドが返した `ImageStore` オブジェクトからは、選択された画像のバイトイメージや `MediaImage` オブジェクトを取得することができます。

また、この `ImageStore` オブジェクトからは画像データのエントリ ID を取得することもでき、後述する画像データの ID 指定読み込みにおいて、ユーザー操作なしに再度その画像データを読み込むために使用することができます。

注意事項：

- ダウンロード即起動 i アプリ、および非活性化状態の待ち受けアプリケーションからは画像データの選択読み込みを行うことはできません。

11.10.3 画像データの ID 指定読み込み

画像データのエントリ ID は、画像がマイピクチャ領域に保存される際にシステムにより割り当てられる一意な整数値であり、その画像データが削除または別画像により上書きされるまで変わることはありません。 i アプリは画像データの新規登録または選択読み込みで一度有効なエントリ ID を取得すると、それ以降はユーザー操作を経ることなくその画像データにアクセスすることができます。画像データの ID 指定読み込みを行うには、`ImageStore` クラスの static メソッド `getEntry()`を使用します。

`getEntry()`メソッドの引数に画像データのエントリ ID を指定して呼び出すと、その画像データに対応する `ImageStore` オブジェクトが返されます。この `ImageStore` オブジェクトからは、画

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.
像データの選択読み込み時と同様に画像のバイトイメージや MediaImage オブジェクトを取得することができます。

注意事項：

- ダウンロード即起動 i アプリからは画像データの ID 指定読み込みを行うことはできません。また、この機能はユーザーの操作を必要としないため、非活性化状態の待ち受けアプリケーションも使用することができます。

11.11 カメラ機能の呼び出し

DoJa-3.0 プロファイル以降に対応した携帯電話ではカメラ機能が標準的に搭載されています。カメラ機能の呼び出しでは、このカメラ機能に i アプリからアクセスする手段を提供します。動画撮影に対応している機種では、静止画像だけでなく動画像の取り扱いをサポートする場合もあります。

カメラ機能の呼び出しには、com.nttdocomo.device.Camera クラスを使用します。

11.11.1 カメラの制御

カメラ制御機能は com.nttdocomo.device.Camera クラスで提供されます。携帯電話には複数のカメラデバイスが搭載されていることがありますが、個々のデバイスはカメラ ID（個々のデバイスに固定的に割り当てられた、0 から始まる整数）により識別されます。アプリケーションプログラムは、Camera.getCamera()メソッドにカメラ ID を指定することにより、カメラオブジェクトを取得することができます。

この機能を使用することで、i アプリはカメラオブジェクトに対し以下のような制御を行うことができます。

- カメラデバイスの設定実行
- ネイティブのカメラ機能の起動
- 撮影画像の取得

(A) カメラデバイスの設定実行

カメラデバイスの設定内容には、デバイス属性と撮影画像サイズ、フレーム画像の 3 つがあります。

デバイス属性の設定は、Camera.setAttribute()メソッドにより行います。規定されている属性には以下のようなものがあります。

- 連写モード（静止画撮影時のみ有効）
- 画質
- マイク音量（動画撮影時のみ有効）

実際のカメラデバイスがどの属性に対応しているかは機種により異なりますが、アプリケーションプログラムは Camera.isAvailable()メソッドにより、その携帯電話で対応している属性を判定することができます。また、Camera.getAttribute()により各属性の現在の設定値を知ることができます。

撮影画像サイズの設定には Camera.setImageSize()メソッドを使用します。撮影画像サイズは縦横のドット数で表します。指定された撮影画像サイズがその携帯電話でサポートされていない場合、システムは指定されたサイズをサポートしているサイズに補正します。また、フレーム画像の設定には Camera.setFrameImage()メソッドを使用します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

カメラデバイスは複数の撮影画像サイズをサポートすることがありますが、その全てのサイズでフレーム付き撮影を行えるとは限りません。カメラデバイスの撮影画像サイズに関する情報は、以下のメソッドを使用することで取得することができます。

- `Camera.getAvailablePictureSizes()`：撮影可能な静止画像のサイズの取得
- `Camera.getAvailableMovieSizes()`　：撮影可能な動画像のサイズの取得
- `Camera.getAvailableFrameSizes()`　：フレーム付きで撮影可能な静止画像のサイズの取得

【DoJa-4.1】【DoJa-5.1】

DoJa-4.1 プロファイル（902iS）以降では上記に加え、接写モードと通常モードをシステム的に切り替え可能な機種向けに、オプショナルとして i アプリからフォーカスモードを設定・取得する機能が追加されました。

この機能に対応している機種では、`Camera` クラスの `setFocusMode()`メソッドを使用して、カメラ起動時点で適用されるフォーカスモードを設定することができます。ただしフォーカスモードは、カメラ起動後にユーザー操作で変更することもできます。その場合、ユーザーが設定したフォーカスモードの内容は `Camera` オブジェクトにはフィードバックされません。

また、`getFocusMode()`メソッドで現在設定されているフォーカスモードを確認したり、`getAvailableFocusModes()`メソッドでその機種でサポートされているフォーカスモードのリストを取得することができます。

DoJa-4.1 プロファイルでは、フォーカスモードとして `FOCUS_NORMAL_MODE`（通常モード）と `FOCUS_MACRO_MODE`（接写モード）の 2 つが規定されました。また DoJa-5.1 プロファイルでは、フォーカスモードの切り替えがハードウェアのユーザ操作によってのみ可能であり，i アプリからの制御ができないことを表す `FOCUS_HARDWARE_SWITCH` が追加されました。

(B) ネイティブのカメラ機能の起動

アプリケーションプログラムは、`Camera.takePicture()`メソッドを呼び出すことで、静止画撮影モードでカメラ機能を起動することができます。また i アプリからの動画撮影に対応している機種では、`Camera.takeMovie()`メソッドを呼び出すことで動画撮影モードでカメラ機能を起動することもできます。

カメラ機能を起動すると i アプリの実行は中断し、カメラ操作画面がディスプレイに表示されます。実際の撮影はユーザーの操作に基づいてのみ行われ、アプリケーションプログラムがシャッターの操作などを行うことはできません。

ユーザーがカメラ操作画面を終了させると、i アプリの実行が再開されます。カメラ機能終了時にユーザーが撮影画像の保存を行っていた場合、その画像はカメラオブジェクト内に保持されています。

注意事項：

- 非活性化状態の待ち受けアプリケーションからはカメラ機能を起動することはできません。

(C) 撮影画像の取得

カメラ機能で撮影された画像は、次にカメラ機能が起動されるか、i アプリが終了するか、または `Camera.disposeImages()`メソッドにより明示的に破棄されるまでカメラオブジェクト内に保持されており、アプリケーションプログラムからメディアイメージまたはバイトストリームの形で参照することができます。

カメラオブジェクトに保持されている画像をメディアイメージの形式で取得するには、`Camera.getImage()`メソッドを使用します。また、画像をバイトストリーム（静止画像の場合は `JPEG` フォーマット、動画像の場合は i モーションフォーマット）の形式で取得するには、`Camera.getInputStream()`メソッドを使用します。これらのメソッドの引数には、カメラオブジェクト内に保存されている画像のインデックスを指定します。画像のインデックスには通常は 0 を指定しますが、静止画の連写撮影を行っている場合には、インデックスで目的の画像を指定することができます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

11.11.2 撮影画像の携帯電話間での交換

DoJa-3.0 プロファイル以降に対応した携帯電話の中には、携帯電話ネイティブの機能として、携帯電話間で赤外線ポートを介して撮影画像を交換できる機能を持った機種があります。この機能は OBEX 外部接続機能に基づいて実現されているため、OBEX 外部接続機能を使用する i アプリとこのネイティブアプリケーションの間でも画像の交換を行うことができます。

ネイティブアプリケーションが画像を交換する際には、IrMC vNote バージョン 1.1 に準拠したオブジェクトを使用します。ただし、本機能では画像交換に必要な追加情報を送受信するために vNote のオブジェクトプロパティを拡張しています。

以下に、本機能で使用するオブジェクトプロパティの内容を示します。

項番	プロパティ	値	備考
1	VERSION	1.1	v Note のバージョン
2	X-DOCOMO-TYPE	JPEG または GIF	画像データのフォーマット種別
3	X-DOCOMO-SIZE	<width>X<height>	画像のサイズ（横 X 縦） X はアルファベット半角大文字（ASCII コード 0x58）
4	X-DOCOMO-FILESIZE	<filesize>byte	画像のファイルサイズ（バイト単位）
5	X-DOCOMO-FILENAME	備考欄参照	ファイル名（拡張子を含め半角英数で 24 バイト以内の文字列）
6	SUMMARY	備考欄参照	画像タイトル（QUOTED-PRINTABLE エンコーディングを適用した、SHIFT-JIS を含む 18 バイト以内の文字列）
7	DATE	備考欄参照	撮影または編集日時（UTC 表記）
8	X-DOCOMO-BODY	備考欄参照	Base64 形式でエンコードされた JPEG または GIF データ 本プロパティ値の終端には<CR><LF>を付加してください

以下に、画像データを交換するための vNote オブジェクトの記述例を示します。

```
BEGIN:VNOTE
VERSION:1.1
X-DOCOMO-TYPE:JPEG
X-DOCOMO-SIZE:120X120
X-DOCOMO-FILESIZE:3000byte
X-DOCOMO-FILENAME:picture_001.jpg
SUMMARY;CHARSET=SHIFT_JIS;ENCODING=QUOTED-PRINTABLE:=89=E6=91=9C
DATE:20020715T153530Z
X-DOCOMO-BODY;ENCODING=BASE64:<Base64 形式でエンコーディングされた画像データ本体＋<CR><LF>>
END:VNOTE
```

※各行の行末は<CR><LF>（0x0d0a）としてください。X-DOCOMO-BODY についてはデータ末尾の<CR><LF>と合わせて 2 つの<CR><LF>が連続することになります。

i アプリ側からネイティブアプリケーション側に画像データを送信する場合、ネイティブアプリケーション側は OBEX サーバーとして動作します。 i アプリ側は OBEX クライアントとして実装し、画像データ（vNote オブジェクト）を PUT オペレーションで送信するようにします。

またネイティブアプリケーション側から i アプリ側に画像データを送信する場合、ネイティブアプリケーション側は OBEX クライアントとして動作します。 i アプリ側は OBEX サーバーとして実装し、ネイティブアプリケーションから PUT オペレーションで送信される画像データ（vNote オブジェクト）を受信するようにします。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

注意事項：

- 画像の交換においては、vNote オブジェクトの複数送受信を行うことはできません。画像データは 1 回のオペレーションで 1 つだけ送信することができます。
- 現在のプロファイルでは、 i アプリが OBEX 外部接続機能を使用して送受信できるデータのサイズは 100KByte まで（DoJa-2.0 プロファイルでは 30KByte まで）に制限されています。画像の交換においては、上記の vNote オブジェクト全体のサイズが制限以下である必要があります。これは。 i アプリ側が画像を送信する場合も受信する場合も同様です。
- `X-DOCOMO-FILESIZE` プロパティには、Base64 形式にエンコードする前のファイルサイズを指定します。
- i アプリからネイティブアプリケーション側に画像データを送信する際、PUT オペレーションで送信するオブジェクトの Name ヘッダは`"mypic.vnt"`としてください。
- i アプリからネイティブアプリケーション側に画像データを送信する際、送信可能な画像サイズ（`X-DOCOMO-SIZE` で示される画像の縦横サイズ）は受信側携帯電話のカメラがサポートする撮影画像サイズに制限されます。

11.11.3 コード認識

コード認識機能では、携帯電話に搭載されたカメラデバイスを使用して画像で表現されたデータ（典型的にはバーコードなど）を取り込み、その内容を認識するための機能を提供します。コード認識機能は、`com.nttdocomo.device.CodeReader` クラスで提供されます。

現在のプロファイルでは、識別可能なコードのコード種別として以下を規定しています。

- `CodeReader.CODE_JAN13`（JAN コード標準タイプ）
- `CodeReader.CODE_JAN8`（JAN コード短縮タイプ）
- `CodeReader.CODE_OCR`（テキスト認識）
- `CodeReader.CODE_QR`（QR コード）
- `CodeReader.CODE_MICRO_QR`（マイクロ QR コード）
- `CodeReader.CODE_39`（CODE-39 コード）
- `CodeReader.CODE_NW7`（NW-7 コード）

ただし、どの機種でも認識可能なコードは `CODE_JAN13`、`CODE_JAN8`、`CODE_QR` の 3 つです。

【DoJa-4.1】

`CODE_MICRO_QR`、`CODE_39`、`CODE_NW7` は、DoJa-4.1 プロファイル（902iS）以降、オプショナルのコード種別として追加されました。

コード認識は以下の手順で行います。

1) `CodeReader` オブジェクトの取得と設定
2) カメラデバイスによるコードの取り込みと認識
3) コード認識の結果取得

＜CodeReader オブジェクトの取得と設定＞

`CodeReader` オブジェクトを取得するには、`CodeReader` クラスの static メソッド `getCodeReader()`を呼び出します。このメソッドの引数には、カメラ制御機能でも使用するカメラ ID を指定します（複数のカメラ

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

デバイスを搭載している機種では、コード認識に使用可能な解像度を持つデバイスがどれか 1 つに制限される場合があります）。

コードの取り込みを行う前には、あらかじめ CodeReader オブジェクトに、これから取り込ませようとしているコードのコード種別を設定する必要があります。コード種別の設定は、CodeReader.setCode()の引数に、上述のコード種別を指定することで行います。携帯電話各機種で取り扱うことのできるコード種別は、CodeReader.getAvailableCodes()メソッドで取得することができます。またメーカーによっては、コード種別に CodeReader.CODE_AUTO を指定することにより、コード種別の自動認識を行わせることができるものがあります。

【DoJa-5.0】

DoJa-5.0 プロファイル以降では、CodeReader クラスでも Camera クラスと同様に、オプショナルとしてフォーカスモードを設定・取得する機能が追加されました。

＜カメラデバイスによるコードの読み取りと認識＞

カメラデバイスを起動してコードの読み取りと認識を行うには、CodeReader.read()メソッドを使用します。このメソッドを呼び出すと i アプリはサスペンドし、カメラデバイスの起動が行われます。

ユーザーがカメラデバイスを操作し、撮影およびコード認識処理を行わせることで、撮影されたコードがデータに変換されて CodeReader オブジェクトに保持されます。ユーザーが操作をキャンセルしたり、コード認識に失敗した場合は、コードのデータは保持されません。いずれの場合でも、ユーザーがカメラデバイスの操作を終了させると i アプリはレジュームし、アプリケーションプログラムに復帰します。

＜コード認識の結果取得＞

カメラデバイスによるコードの読み取り・認識に成功すると、アプリケーションプログラムは CodeReader オブジェクトからその結果を取得することができます。アプリケーションプログラムは、コード認識の結果として以下の情報を参照することができます。

- 認識したコードのコード種別（getResultCode()メソッド）
- コード認識の結果得られたデータ（getString()メソッドまたは getBytes()メソッド）
- コード認識の結果得られたデータのデータ型（getResultType()メソッド）

データ型には以下の 4 種類があります。

- 数字文字列（CodeReader.TYPE_NUMBER）
- ASCII 文字列（CodeReader.TYPE_ASCII）
- 非 ASCII 文字列（CodeReader.TYPE_STRING）
- バイナリ（CodeReader.TYPE_BINARY）

JAN コードでは規格上の制限により数字文字列のみ取り扱うことができますが、QR コードでは非 ASCII 文字を含む文字列やバイナリデータも取り扱うことができます。

コード認識機能は、DoJa-3.0 プロファイルで i アプリオプション API として規定されました。その後 DoJa-3.5 プロファイルにて i アプリ基本 API に取り入れられています。

注意事項：

- 非活性化状態の待ち受けアプリケーションからは、CodeReader.read()メソッドを呼び出すことはできません。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- コード認識機能はカメラ制御機能と同じカメラデバイスを使用するため、アプリケーションプログラムが同一のカメラ ID でこれらの機能を同時に使用しようとすると、開発者の意図しない結果を引き起こす場合があります。例えば、Camera オブジェクトが画像を保持している状態で CodeReader.read()メソッドを呼び出すと、Camera オブジェクトが保持していた画像は破棄されることがあります。
- JAN コードを読み取る場合、読み取った結果はチェックディジットを含めた全ての文字がアプリケーションプログラムに返されます。つまり、JAN コード標準タイプでは 13 桁、JAN コード短縮タイプでは 8 桁の数字文字列がアプリケーションプログラムに返されます。
- QR コードについては、以下の仕様に従ったものを取り扱うことができます。

  － バージョン　　：1～10（バージョン 11 以上の使用可否はメーカーにより異なります）
  － モデル　　：モデル 2（モデル 1、および MicroQR はサポートしません）
  － 誤り訂正レベル　：L、M、Q、H

  また、漢字データを扱う際には、i アプリ実行環境のデフォルトエンコーディングである Shift-JIS コードを使用することができます。
- CodeReader.getResultType()メソッドにより、読み取ったデータのデータ型がどれだけ詳細に得られるかは、メーカーにより異なります。例えば数字文字列を読み込ませた場合、あるメーカーの携帯電話ではその型を詳細に CodeReader.TYPE_NUMBER と判断するかもしれませんが、別のメーカーの携帯電話では文字列の内容判定までは行わず CodeReader.TYPE_ASCII や CodeReader.TYPE_STRING と判定するかもしれません。このメソッドの返す値は、コード認識の結果得られたデータをアプリケーションプログラムが解析する上でのヒントとして使用するようにしてください。

なお、DoJa-3.5 プロファイル以降に対応した携帯電話では、特定の情報を含む 2 次元バーコードを携帯電話ネイティブのコード認識アプリケーションに読み取らせることによりその携帯電話内にダウンロードされている特定の i アプリを連携起動する機能が搭載されています（DoJa-3.0 プロファイルの携帯電話ではオプショナル機能）。

i アプリを連携起動するための 2 次元バーコードに含まれる情報（i アプリ起動情報）は、以下の例に示すようなテキストデータです。2 次元バーコード化された i アプリ起動情報をネイティブのコード認識アプリケーションで読み取ると、対応する i アプリを起動する旨の確認画面が表示されます。

＜ i アプリ起動情報のテキスト例＞

```
LAPL:
ADFURL:http¥://www.nttdocomo.co.jp/java/a.jam;
CMD:startup_Application_A;
PARAM:param1,i-mode;
PARAM:param2,iappli;
;
```

※便宜上折り返し表示を行なっていますが、テキスト内に改行コードは含みません。

以下に、i アプリ起動情報の書式および記法を解説します。

```
LAPL:
ADFURL:<起動対象 i アプリの ADF URL(*1)>;
CMD:<起動コマンド(*1)>;
PARAM:<パラメータ名(*2)>, <パラメータ値(*2)>;
    :（必要に応じて PARAM を繰り返し）
;
```

※記述例同様、便宜上折り返し表示を行なっていますがテキスト内に改行コードは含みません。

(*1) これらに現れる特殊文字 '¥' ':' ';' には、直前にエスケープ文字 '¥' を付加します。

(*2) これらに現れる特殊文字 '¥' ':' ';' ',' には、直前にエスケープ文字 '¥' を付加します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- LAPL:

  i アプリ起動情報の開始を表す識別子です。

- ADFURL:<起動対象 i アプリの ADF URL>;

  この i アプリ起動情報で起動させたい i アプリに対応する ADF の URL を ASCII 形式で指定します(最大 255 バイト）。末尾には必ずセミコロンを指定します。起動対象 i アプリは、あらかじめその携帯電話内にダウンロードされている必要があります。

- CMD:<起動コマンド>;

  起動コマンドを ASCII 形式で指定します（最大 250 バイト）。末尾には必ずセミコロンを指定します。ここで指定されている起動コマンド（末尾のセミコロンは除きます）と、i アプリの ADF の AllowPushBy キーで起動許可が与えられている起動コマンド（AllowPushBy キーの値として"Code:"以降に指定されるコマンド文字列）が完全に一致していなければ、 i アプリを起動することはできません。ADF の AllowPushBy キーについては、15.5.1 項を参照してください。

- PARAM:<パラメータ名>, <パラメータ値>;

  起動された i アプリに引き渡すパラメータを指定します（オプショナル）。1 つのパラメータは以下のフォーマットを取ります。

  PARAM <パラメータ名>, <パラメータ値>;

  ※末尾には必ずセミコロンを指定します。

  パラメータ名およびパラメータ値は、 i アプリ起動時に IApplication.getParameter()メソッドにより取得できるパラメータを指定するために使用されます。パラメータ指定は 1 つの i アプリ起動情報に最大 16 個まで指定することができます。ただし、全てのパラメータの名前と値の合計長は 255 バイトに制限されます。またパラメータの名前および値には ASCII 文字のみ使用可能です。

- ;

  i アプリ起動情報の終了を表す識別子です。開始識別子（LAPL:）と対となります。

これらの規定に従って作成した i アプリ起動情報のテキストを 2 次元バーコードに変換する方法については、2 次元バーコード編集のための各ツールのマニュアル等を参照してください。

注意事項：

- 携帯電話ユーザーは、携帯電話の設定により 2 次元バーコードからの i アプリ起動を禁止することができます。2 次元バーコードからの起動が禁止されている場合、2 次元バーコードを読み取らせても i アプリの起動は行われません。
- 2 次元バーコードから起動された i アプリの動作が終了すると、 i アプリ起動前の状態（コード認識結果表示）に復帰します。ただし、2 次元バーコードから起動された i アプリをユーザーが強制終了した場合、 i アプリ終了後にどのような状態に移るかはメーカーにより異なります
- 2 次元バーコードは人間が視覚的に直接解読することは困難ですが、対応機器（読み取り専用機器だけでなく 2 次元バーコード対応の一般的な携帯電話なども含みます）を介することで容易に読み取ることができます。機密の保持が求められるようなデータは、 i アプリ起動情報の起動コマンドやパラメータには設定しないなどの配慮が必要です。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

11.12 映像データ管理機能の呼び出し

DoJa-4.0 プロファイル以降、FOMA 携帯電話向けに映像データ（i モーション）を取り扱うための機能が追加されています。DoJa-4.0 プロファイルでは、一般 i アプリがネイティブの映像データ管理機能を利用することはできませんでしたが、DoJa-4.1 プロファイル以降では一般 i アプリがこの機能を利用して、映像データをネイティブ保存領域に新規登録することができます。

なお、本機能を使用する i アプリは、ADF の `AccessUserInfo` キーにてこの機能を使用することを宣言（値`"yes"`を設定）する必要があります。

【DoJa-4.1】

映像データ管理機能による映像データの新規登録は、DoJa-4.1 プロファイルから一般 i アプリでも使用可能となりました。これより前のプロファイルでは、この機能はトラステッド i アプリでのみ使用可能です。

また、i アプリから呼び出せる映像データ管理機能は、DoJa-4.1 プロファイルの i アプリ基本 API としては映像データの新規登録のみです。ネイティブ保存領域に保存されている映像データを i アプリから取得する機能（選択読み込みや ID 指定読み込みなど）は、DoJa-4.1 プロファイルでは i アプリオプション API の位置づけとなっています。

映像データをネイティブ保存領域に新規登録するには、`com.nttdocomo.system.MovieStore` クラスの static メソッド `addEntry()`を使用します。このメソッドの引数には、登録したい映像データ（i モーション形式データ）を表す `MediaImage` オブジェクトを指定します。

映像データの新規登録を行うと、携帯電話は以下のような動作を行います。

(1) アプリケーションプログラムから `MovieStore.addEntry()`メソッドを呼び出します。このメソッドが呼び出されると i アプリはサスペンドします。

(2) 映像データの登録を行うためのネイティブのユーザーインタフェース画面が起動します。この画面では、ユーザーは i アプリからの映像データの新規登録を許可、キャンセルすることができます。また、i アプリがどのような映像データを登録しようとしているかを確認することもできます。

(3) ユーザーが映像データの新規登録を許可すると映像データの新規登録が行われ、i アプリはレジュームして `addEntry()`メソッドはアプリケーションプログラムに復帰します。その際このメソッドは、新規登録された映像データにその携帯電話内で 1 対 1 に対応するエントリ ID を返します。ネイティブ保存領域からの映像データ取得をサポートする機種では、エントリ ID は映像データ取得のキーとして使用することができます。

ユーザーが映像データの新規登録をキャンセルしても例外は発生せず、このメソッドは-1 を返します。

注意事項：

- ダウンロード即起動 i アプリ、および非活性化状態の待ち受けアプリケーションからは映像データの新規登録を行うことはできません。
- `addEntry()`メソッドに指定する `MediaImage` オブジェクトは、使用可能な状態（`use()`メソッドが呼び出された状態）でなければなりません。またこの `MediaImage` オブジェクトは、i モーション形式の映像データから生成されたものでなければなりません（静止画やアニメーション GIF から生成された `MediaImage` オブジェクトを `MovieStore.addEntry()`メソッドの引数に指定することはできません）。
- 映像データから生成した `MediaImage` オブジェクトを `MovieStore` クラスで保存する際には、元の映像データに設定されていた再配布可否情報もそのままの形で保存されます。また、DoJa-4.0 プロファイルから追加された `MediaResource. setRedistributable()`メソッドを使用して、i アプリから再配布可否情報の設定を行うこともできます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

第 12 章
赤外線リモコン

DoJa-3.0 プロファイルより、携帯電話の赤外線ポートから一般的な赤外線リモコン対応機器に対して制御信号を送出するための API が追加されました。 i アプリから適切な制御信号を組み立てて送出することにより、それらの赤外線リモコン対応機器を外部から操作することができます。

注意事項：

- i アプリの赤外線リモコン機能では、赤外線信号の物理的な波形を組み立てて送出するためのプリミティブな API が規定されています。一方、一般的な赤外線リモコン対応機器では、具体的にどのような波形を制御信号として受け付けるかは対応機器のメーカーが独自に設計しています。 i アプリ仕様の範囲には、赤外線リモコンの制御信号の仮想化・標準化などは含みません。開発者は、ターゲットとする赤外線リモコン対応機器の仕様を確認の上、それに適した制御信号を送出するよう i アプリを設計する必要があります。
- i アプリの赤外線リモコン機能では、携帯電話からおおむね 2～3 メートル程度離れた場所にある機器を制御可能とすることを想定しています。ただし、実際にどの程度離れた機器を制御できるかは、携帯電話からの信号の出力や指向性だけでなく、周囲の環境や赤外線リモコン対応機器側での受光能力などにより影響を受けます。

12.1 制御信号の構成

i アプリの赤外線リモコン機能では、以下に示すような赤外線リモコンの制御信号を取り扱うことができます。日本国内で販売されている一般的な家電製品の多くは、制御信号を下記の枠組みで表現することができます。ただし、前述の通り制御信号の設計はその機器のメーカーが独自に行っているため、下記内容が全ての赤外線リモコン対応機器に適用されることを保証するものではありません。

キャリア周波数と変調方式：

赤外線リモコンの制御信号はキャリア周波数 25kHz～50kHz の搬送波（赤外線発光の ON と OFF で表現される矩形波）で搬送されるパルス位相変調信号（PPM 信号　以下、搬送波で搬送される論理的な波形を制御信号とします）として取り扱われます。制御信号の High 区間と Low 区間の長さの組み合わせによってビット値（論理 0 および論理 1）を表現します。

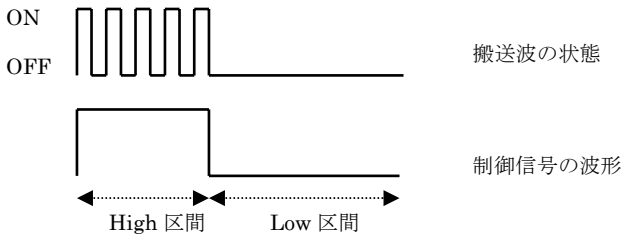
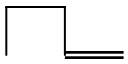
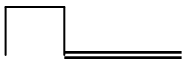
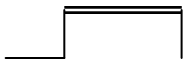


図 20：搬送波による制御信号の構成

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

ビット値の制御信号波形上での表現方法には以下の 2 つのパターンがあり、どちらの方法を取るかは赤外線リモコン対応機器側の設計により異なります。 i アプリの赤外線リモコン機能では、下記のいずれの方法を使用するかを指定することができます（波形図の二重線部の長さで論理 0 と論理 1 が識別されます）。

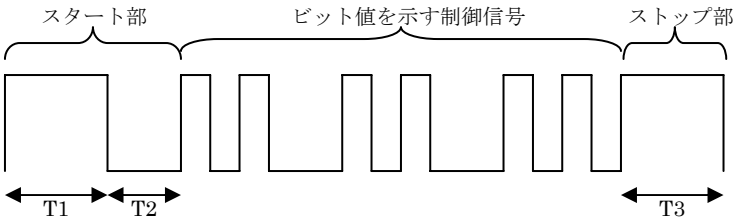
	論理 0 の波形の例	論理 1 の波形の例
High 先行パターン Low 区間の長さにより論理 0、論理 1 を表現		
Low 先行パターン High 区間の長さにより論理 0、論理 1 を表現		

上記において、以下の項目は赤外線リモコン対応機器の設計に依存しており、アプリケーションプログラムはこれらを機器に適した値に設定することができます。

- 搬送波 1 波形あたりの High 区間および Low 区間の長さ（これにより搬送波のキャリア周波数が決定付けられます）
- ビット値の表現方法（High 先行パターンまたは Low 先行パターン）
- 制御信号上の、論理 0 の High 区間の長さおよび Low 区間の長さ
- 制御信号上の、論理 1 の High 区間の長さおよび Low 区間の長さ

データフレームの構成：

赤外線リモコンの送出データは、上述の方法で変調されたビットデータの制御信号にスタート部（一定の長さを持つ High 区間と Low 区間各々1 つずつの制御信号の組み合わせ）およびストップ部（一定の長さを持つ 1 つの High 区間から構成される制御信号）を付加したものとなります。



T1： スタート部 High 区間
T2： スタート部 Low 区間
T3： ストップ部 High 区間

図中のビットデータは High 先行パターンの例：

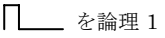
 を論理 0、  を論理 1 とすると、上記のデータフレームには０１０１００ という 6 ビットのビット列が含まれることになります。

図 21：データフレームの構成

スタート部の High 区間と Low 区間、およびストップ部の High 区間の長さは赤外線リモコン対応機器の設計に依存しており、アプリケーションプログラムはこれらを機器に適した値に設定することができます。また、1 つのデータフレームに含めることのできるビットデータのビット数には、DoJa プロファイルバージョンごとに制限があります。データフレームは、12.2.2 項で述べる `com.nttdocomo.device.IrRemoteControlFrame` クラスで表現されます。データフレーム中のビット数の制限についても 12.2.2 項を参照してください。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

赤外線リモコン送出データの全体構成：

赤外線リモコンの送出データは、データフレームの集合として表現されます。全く別のデータフレームが連続して送られることも、あるデータフレームが繰り返し送られることもあります。

また一般に赤外線リモコンは、規定されたデータフレームを一通り送信し終わると、再度先頭のデータフレームから送信を繰り返します。これは、例えばテレビなどのリモコンの音量ボタンを押し続けると、その間制御信号が送出され続けることと対応しています。

データフレームの繰り返し送信は、データ送出時に指定した時間（タイムアウト時間）または繰り返し回数が経過するか、アプリケーションプログラムが明示的にデータ送出をストップするまで行われます。

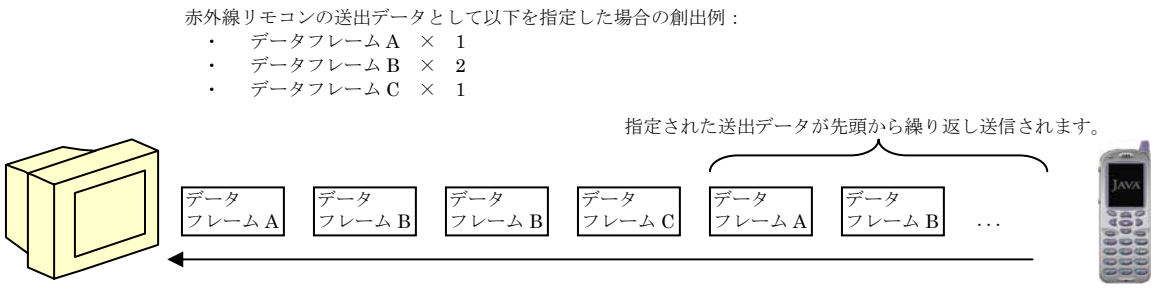


図 22：携帯電話からのデータフレームの送出例

アプリケーションプログラムは、作成したデータフレーム情報の送出順序、送出間隔、データ送出を終えるまでのタイムアウト時間または繰り返し回数を、ターゲットとする機器に適した内容に設定することができます。赤外線リモコン送出データは、12.2.1 項で述べる `com.nttdocomo.device.IrRemoteControl` クラスを使用して送出します。

12.2 赤外線リモコン API

前項に記載した赤外線リモコンの機能を i アプリから利用するためのクラスとして、以下のものがあります。

- `com.nttdocomo.device.IrRemoteControl`
- `com.nttdocomo.device.IrRemoteControlFrame`

`IrRemoteControl` クラスは赤外線ポートから制御信号を送信するための送出器としての役割を持ちます。また `IrRemoteControlFrame` クラスは、送出データを構成するデータフレームを表現します。アプリケーションプログラムは、`IrRemoteControlFrame` オブジェクトの形でデータフレームのセットを作成し、そのセットを `IrRemoteControl` オブジェクトにより赤外線ポートから送出します。

12.2.1 IrRemoteControl

`IrRemoteControl` クラスは、`IrRemoteControlFrame` の形で指定されたデータフレームを送出する機能を持ちます。また、リモコン機能全体に渡るコンフィグレーションを行うためのメソッドを備えています。

アプリケーションプログラムは、以下の手順で `IrRemoteControl` クラスを使用します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

(1) IrRemoteControl.getIrRemoteControl()メソッドを使用して、IrRemoteControl オブジェクトを取得します。IrRemoteControl オブジェクトは携帯電話の赤外線リモコンポートと 1 対 1 に対応付けられており、アプリケーションプログラム中で複数のオブジェクトを持つことはできません。

(2) 赤外線リモコンの物理的なコンフィグレーションを行います。赤外線リモコンの設定項目には以下のものがあります。

- 搬送波の High 区間長、Low 区間長（キャリア周波数）の設定（setCarrier()メソッド）
- 制御信号上の論理 0 波形の表現パターンと High 区間長、Low 区間長（setCode0()メソッド）
- 制御信号上の論理 1 波形の表現パターンと High 区間長、Low 区間長（setCode1()メソッド）

論理 0 波形と論理 1 波形の表現パターンには、IrRemoteControl クラスの定数 PATTERN_HL（High 先行）または PATTERN_LH（Low 先行）を指定します。

(3) 送信したいデータフレームを表す IrRemoteControlFrame オブジェクトのセットを生成し、send()メソッドを使用して送信します。send()メソッドでは、データフレームのセットを繰り返し自動送信する時間（タイムアウト時間）を指定することができます。デフォルトのタイムアウト時間は 10 秒です。
send()メソッドは、データフレームの送信要求を受け付けた後すぐにアプリケーションプログラムに復帰し、その後アプリケーションプログラムとは非同期にデータフレームの送信が行われます。データフレームの送信は、以下のいずれかの契機まで繰り返し続けられます。
・タイムアウト時間の経過
・アプリケーションプログラムからの明示的な送信停止指示（stop()メソッド）
・通話着信の発生などによる中断
・アプリケーションプログラムからの HTTP 通信の実行
・アプリケーションプログラムの終了（異常終了も含みます）

【DoJa-4.1】

DoJa-4.1 プロファイルでは send()メソッドについて、データ送出を停止するまでのタイムアウト時間だけでなく、データフレームのセットを送出する際の繰り返し回数も指定することのできるバリエーションが追加されました。このメソッドを使用することにより、タイムアウト時間内であっても、指定した繰り返し回数で送信を停止することができます。

注意事項：

- 以下の場合には、アプリケーションプログラムは send()メソッドを呼び出すことはできません。
  ・待ち受けアプリケーションが非活性化状態にある場合
  ・携帯電話がセルフモードに設定されている場合
  ・ｉアプリが赤外線ポートを使用している状態（赤外線リモコン、OBEX 外部接続）
  ・パケット通信中の場合
- 赤外線リモコンの設定において、設定値の単位や設定可能な値の範囲はメソッドにより異なっています。メソッド毎の設定値の単位などについては API リファレンスを参照してください。
- メーカーの採用する赤外線デバイスの特性によっては、API で指定した波形パラメータ（High 区間長、Low 区間長）の値がそのデバイスで取り扱うことのできる値（単位）に補正されることがあります。

12.2.2 IrRemoteControlFrame

IrRemoteControlFrame クラスは、個々のデータフレームを表現します。アプリケーションプログラムはこのクラスのコンストラクタを使用してオブジェクトを生成し、スタート部、ビットデータ部、ストップ部を設定することでデータフレームを構成します。

データフレームの設定項目には以下のものがあります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

- データフレームの送信間隔（`setFrameDuration()`メソッド）
  そのデータフレームの送信開始時点から、次のデータフレームの送信を開始するまでの間隔を指定します。
- データフレームの繰り返し送信回数（`setRepeatCount()`メソッド）
  1 つのデータフレームセットの中で、同じデータフレームを連続して送信する場合は、その回数を指定します。同じデータフレームを連続して送信しない場合は 1 を指定します。
- スタート部の形式（`setStartHighDuratoin()`および`setStartLowDuration()`メソッド）
  そのデータフレームにおけるスタート部の High 区間および Low 区間の長さを指定します。
- ストップ部の形式（`setStopHighDuratoin()`メソッド）
  そのデータフレームにおけるストップ部の High 区間の長さを指定します。
- ビットデータの値（`setFrameData()`メソッド）
  `byte`配列または`long`値の形で、ビットデータとして送出する値を指定します。

  【DoJa-3.0】【DoJa-4.0】【DoJa-4.1】

  DoJa-3.0 プロファイルでは、どの機種でも送信することのできるビットデータ長は、1 フレームあたり 48 ビットです。またこの値は、DoJa-4.0 プロファイルでは 1 フレームあたり 512 ビット、DoJa-4.1 プロファイルでは 1 フレームあたり 1024 ビットに拡大されています。

注意事項：

- データフレームの設定において、設定値の単位や設定可能な値の範囲はメソッドにより異なっています。メソッド毎の設定値の単位などについては API リファレンスを参照してください。
- メーカーの採用する赤外線デバイスの特性によっては、API で指定したスタート部、ストップ部およびデータフレーム送信間隔に関するパラメータの値がそのデバイスで取り扱うことのできる値（単位）に補正されることがあります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

第 13 章
3D グラフィックス・3D サウンド

DoJa-4.0 プロファイル以降では、 i アプリの演出表現能力の向上を目指し、 i アプリから 3D グラフィックスおよび 3D サウンドを制御するための API が追加されています。

従来、3D グラフィックス描画機能は i アプリ拡張 API として提供されていましたが、本プロファイルではこれを機能強化した上で、新たに i アプリ基本 API として規定しています。また、携帯電話に搭載されたステレオスピーカーを有効に利用するため 3D サウンド制御機能を規定しています。

本章では、これら 3D グラフィックス描画機能、3D サウンド制御機能の概要について解説します。なお、本プロファイルに対応した携帯電話では、標準の 3D グラフィックスエンジンとして株式会社エイチアイの Mascot Capsule Engine Micro3D Edition のバージョン 4 が搭載されています。このエンジンを活用した i アプリプログラミング（DoJa-5.0 プロファイルで追加された衝突判定機能も含みます）の詳細については、株式会社エイチアイより提供されるドキュメント「Micro3D Programming for i アプリ」を参照してください。

13.1 3D グラフィックス描画機能

13.1.1 3D グラフィックス描画機能

本プロファイルにおける 3D グラフィックス描画機能は、以前のプロファイルにおいて i アプリ拡張 API として規定されていた高レベル 3D グラフィックス描画機能をベースとして、機能強化および一部機能の削除、およびそれらに伴うクラス構成、メソッド構成の変更が加えられています。高レベル 3D グラフィックス描画機能がベースとなっていますので、比較的簡単な API 呼び出しで 3D モデルの描画を行うことができるようになっています。

3D グラフィックス描画機能は、`com.nttdocomo.ui.graphics3d` パッケージで提供されます。また、3D に関する数値演算のために `com.nttdocomo.ui.util3d` パッケージで提供されるユーティリティ機能を使用します。

3D グラフィックス描画機能を構成するクラス群を以下に示します。これらクラス構成やクラス内のメソッド、フィールド構成はオプションパッケージに含まれる（ i アプリ拡張 API の）高レベル 3D グラフィックス描画機能がベースになっていますが変更が加えられており、従来の API とはその名前や使用方法も含めて互換性がありません。従来の高レベル 3D グラフィックス描画機能を使用している既存アプリケーションを簡便に動作させるには、本プロファイルにおいてもオプションパッケージに含まれる高レベル 3D グラフィックス描画機能を使用することを推奨します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

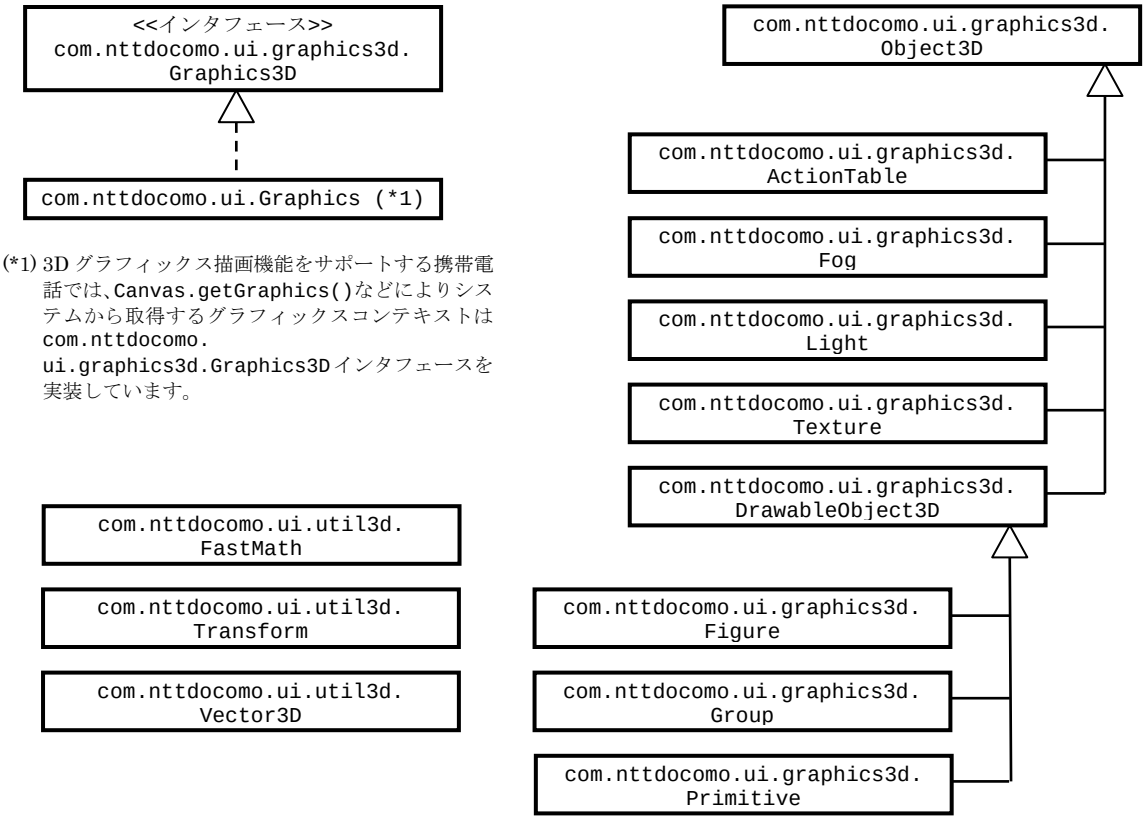


図 23: 3D グラフィックス描画機能のクラス構成

各クラス、インタフェースの機能概要は以下の通りです。

クラス・ インタフェース	機能概要
Graphics3D	3D グラフィックス描画機能をサポートするグラフィックスコンテキストが備えるべきメソッドを定義するインタフェースです。Image に対応するグラフィックスコンテキストがこのインタフェースをサポートするかどうかはメーカーにより異なります。
Object3D	全ての 3D オブジェクトの基底となるクラスです。3D オブジェクトとは、3D グラフィックスのレンダリング結果に影響を与えるオブジェクト全般を指します。3D オブジェクトには、フィギュアやプリミティブのように直接レンダリングの対象物とすることのできるオブジェクトと、ライトやフォグのようにレンダリング結果に効果を与える性質を持つオブジェクトに大別されます。このクラスでは 3D オブジェクトの生成方法として createInstance()を規定していますが、3D オブジェクトの種類によってはコンストラクタを使用してオブジェクトを生成するものもあります。
ActionTable	3D モデルのアクション（アニメーションデータ）を保持するクラスです。アニメーションデータは市販の 3D オーサリングツールなどを利用して作成します。 ある特定の瞬間に 3D モデルが取るべきポーズをフレームと呼び、アクションは複数のフレームのシーケンスとして構成されます。ActionTable には複数のアクションを保持することができます。フレームは全ての 3D オブジェクトが内部に保持している「現在時刻」と対応しており、後述の Figure オブジェクトに現在時刻を設定することで、3D モデルにポーズを取らせることができます。
Fog	フォグ効果を与えるためのデータを保持するクラスです。線形フォグモードと指数フォグモードをサポートします。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

Light	光源データを保持するクラスです。いくつまでの光源をサポートするかはメーカーにより異なります（getMaxLights()メソッドで確認することができます）。環境光源、平行光源、点光源、スポット光源をサポートします。
Texture	テクスチャのデータを保持するクラスです。テクスチャオブジェクトを使用することで、モデルマッピングに使用するテクスチャと環境マッピングに使用するテクスチャの両方を取り扱うことができます。なお、本機能における環境マッピング用テクスチャの使用方法はｉアプリ拡張 API の高レベル 3D グラフィックス描画機能における使用方法から変更されています。本機能では、環境マッピング用のテクスチャは 3D グラフィックスコンテキスト（Graphics3D オブジェクト）ではなく各モデルマッピング用テクスチャオブジェクトに設定して使用します。
DrawableObject3D	レンダリング可能な 3D オブジェクトの基底となるクラスです。このクラスを継承する Figure クラス、Group クラス、Primitive クラスの各オブジェクトは、Graphics3D.renderObject3D()メソッドによるレンダリング対象とすることができます。また、3D 空間内で 2 つのオブジェクト同士が衝突しているかどうかを簡易に判定するためのメソッドや、ブレンド機能、テクスチャ歪み補正機能などのメソッドを備えています。
Figure	3D モデルのデータを保持するクラスです。このオブジェクトはレンダリング可能であり、Graphics3D.renderObject3D()メソッドにレンダリング対象として指定することができます。
Group	3D オブジェクトの集合体であるグループを表現するクラスです。このオブジェクトはレンダリング可能であり、Graphics3D.renderObject3D()メソッドにレンダリング対象として指定することができます。複数の Figure オブジェクトや Primitive オブジェクトを属させて一度にレンダリングを行わせたり、Fog オブジェクトや Light オブジェクトを属させてグループ内のレンダリングに効果を与えたりすることができます。
Primitive	プリミティブ（点や線、面などの図形）を描画する際に用いる頂点情報などを格納するための、プリミティブ配列を保持するクラスです。このオブジェクトはレンダリング可能であり、Graphics3D.renderObject3D()メソッドにレンダリング対象として指定することができます。
FastMath	3D 関連機能での使用を前提とした高速数値演算ユーティリティです。本プロファイルに対応する携帯電話では CLDC-1.1 が搭載されているため浮動小数点数演算が使用できますが、このクラスでは浮動小数点数演算を内部的に整数演算に置き換えます。このため、通常の浮動小数点数演算より大きな誤差を含む可能性はありますが高速に演算を行うことができます。
Transform	3 次元アフィン変換用の行列を扱うクラスです。3D モデルの回転や視点の移動には、このクラスのオブジェクトを使用します。
Vector3D	3 次元ベクトルを表すクラスです。内積・外積計算、正規化といったユーティリティメソッドを備えています。

注意事項：

- 本プロファイルに対応した携帯電話では、標準の 3D グラフィックスエンジンとして株式会社エイチアイの Mascot Capsule Engine Micro3D Edition のバージョン 4 が搭載されています。このエンジンを活用したｉアプリプログラミングの詳細については、株式会社エイチアイより提供されるドキュメント「Micro3D Programming for ｉアプリ」を参照してください。
- ｉアプリ拡張 API の高レベル 3D グラフィックス描画機能（com.nttdocomo.opt.ui.j3d パッケージ）と本機能を 1 つのｉアプリの中で同時に使用することはできません。一方を使用中に他方を使用しようとすると例外がスローされます。
- 本機能では光源に関する機能強化（複数光源や点/スポット光源への対応）やフォグ効果、オブジェクトグルーピング機能の新規導入など、従来の高レベル 3D グラフィックス描画機能と比較して機能強化が図られています。一方、ｉアプリ拡張機能の高レベル 3D グラフィックス描画機能におけるコマンドリスト、およびトゥーンシェーディングの機能はサポートされておらず使用することはできません。
- 従来の高レベル 3D グラフィックス描画機能では、浮動小数点型の存在しない CLDC-1.0 上で 3D 機能の API を定義するために、整数型を使用して小数点以下の値を扱うための独自の単位系を使用していました。これに対し本機能では、CLDC-1.1（本プロファイルのプラットフォーム）の浮動小数点型を使用したインタフェースに改められています。ただしシステム内部の演算では、性能を確保するために従来通り整数型を使用した演算が行われます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

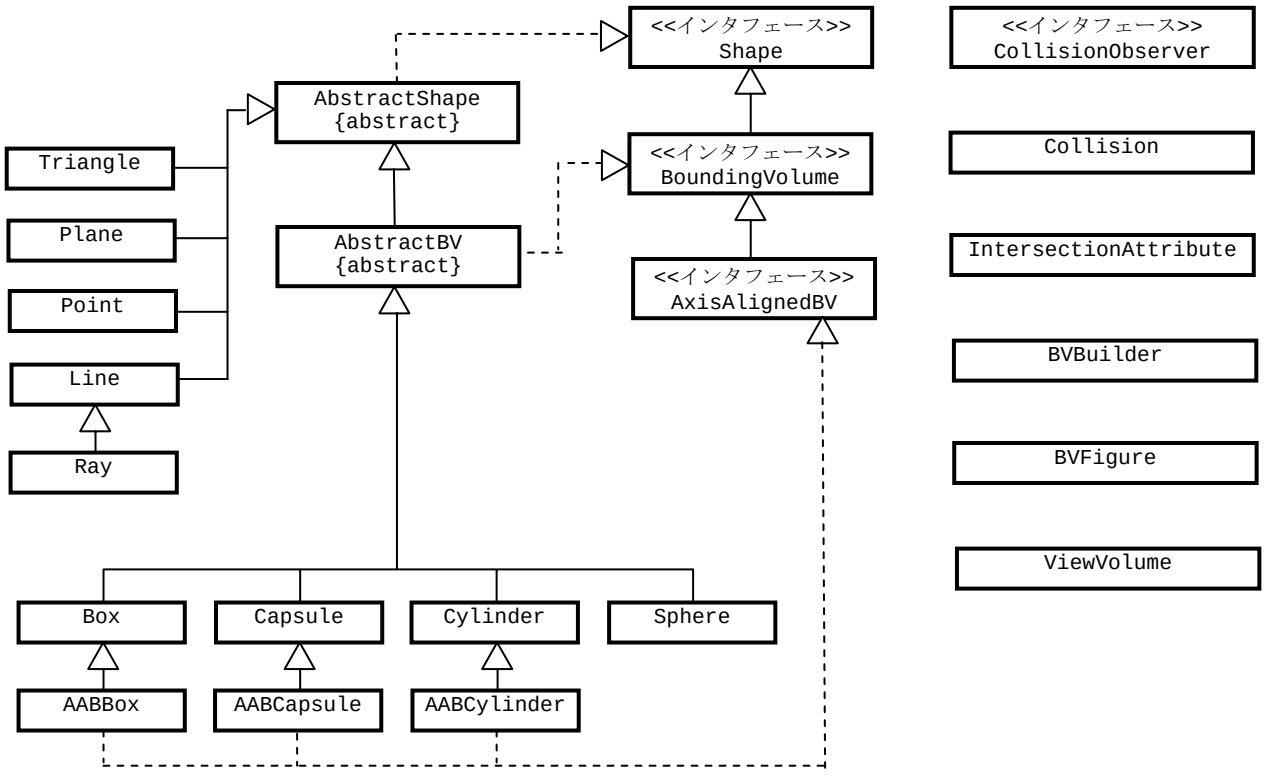
- 携帯電話の中には、３Ｄグラフィックス関連の処理を高速化するため専用のハードウェアを搭載しているものがあります。そのような機種では、ハードウェア実装の制限から他の機種やエミュレータと比較して演算精度が落ちる場合があります。特に、Z 方向の位置関係の演算（デプス・バッファ処理）にこの問題が作用した場合、ごく似通った前後関係のオブジェクト同士が意図通りの前後関係で描画されないことがありますので注意してください。

13.1.2 衝突判定機能

DoJa-5.0 プロファイルにて、前項に記載した 3D グラフィックス描画機能と併用して 3D 空間上での物体の衝突判定を行うための `com.nttdocomo.ui.graphics3d.collision` パッケージが新設されました。

前項に記載した `DrawableObject3D` クラスでも衝突判定の機能（`isCross()`メソッド）が用意されていますが、そちらの機能では複雑なポリゴン構成を持つ 3D オブジェクトでも各ポリゴン単位で厳密に衝突判定を行うため、複雑な 3D オブジェクトを高速に処理するには適さない場合があります。本項で解説する衝突判定機能では、実際の 3D オブジェクトよりも簡略化されたシンプルな図形を用いて衝突判定を行うため、精度は `DrawableObject3D.isCross()`メソッドに劣りますが高速に判定を行うことができます。処理速度を重視するコンテンツでは、`DrawableObject3D.isCross()`メソッドではなく本項に解説する衝突判定機能を使用することを推奨します。

衝突判定機能を構成する、`com.nttdocomo.ui.graphics3d.collision` パッケージのクラス、インタフェース群を以下に示します。



各クラス、インタフェースの機能概要は以下の通りです。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

クラス･インタフェース	機能概要
AABBox	ワールド座標軸に平行な Box を表すクラスです。
AABCapsule	ワールド座標の Y 軸に中心軸が平行となる Capsule を表すクラスです。
AABCylinder	ワールド座標の Y 軸に中心軸が平行となる Cylinder を表すクラスです。
AbstractBV	形状のうち、立体を表す抽象クラスです。このクラスは BoundingVolume インタフェースを実装しています。立体形状は、このクラスを継承した具象クラスとして提供されます。
AbstractShape	全ての形状の基底となる抽象クラスであり、Shape インタフェースを実装しています。非立体形状は、このクラスを継承した具象クラスとして提供されます。また立体形状は、このクラスを継承した抽象クラス AbstractBV を継承した具象クラスとして提供されます。
AxisAlignedBV	比較判定で使用される立体の中でも、ワールド座標に平行に配置される特性を持つ立体が実装するインタフェースです。
BoundingVolume	比較判定で使用される形状のうち、立体が実装するインタフェースです。本インタフェースは Shape インタフェースを継承しています。本パッケージでは、具体的な立体形状として直方体、円柱、球形、カプセル形をサポートしています。
Box	比較判定で使用される立体のうち、直方体を表すクラスです。
BVBuilder	フィギュアから、BVFigure オブジェクトや、BVFigure オブジェクトに貼り付けることのできる BoundingVolume オブジェクト（BoundingVolume インタフェースを実装した各種立体のオブジェクト）を生成するためのクラスです。
BVFigure	ボーン構造を持ち、複数の BoundingVolume で表現されたフィギュアです。通常のフィギュア（Figure クラス）ではポリゴンを貼り付けることによって形状を表現しますが、BVFigure は BoundingVolume を貼り付けることによって形状を表現します。
Capsule	比較判定で使用される立体のうち、カプセル形を表すクラスです。
Collision	衝突判定に関する機能を提供するクラスです。衝突判定、交差判定、距離算出などの機能が本クラスで提供されます。
CollisionObserver	Collision クラスでの衝突判定の結果、衝突したと判定された場合に、衝突情報とともに通知を受けるためのオブザーバクラスが実装すべきインタフェースです。通知を受けたいアプリケーションは、このインタフェースを実装したオブザーバクラスのオブジェクトを作成して Collision オブジェクトにセットします。
Cylinder	比較判定で使用される立体のうち、円柱を表すクラスです。
IntersectionAttribute	Figure と Ray の交差判定において、交差したと判断された場合に CollisionObserver の通知メソッドに渡される交点情報を表すクラスです。
Line	比較判定で使用される非立体形状のうち、線分を表すクラスです。
Plane	比較判定で使用される非立体形状のうち、無限平面を表すクラスです。
Point	比較判定で使用される非立体形状のうち、点を表すクラスです。
Ray	比較判定で使用される非立体形状のうち、半直線を表すクラスです。
Shape	衝突判定に使用される全ての形状が実装するインタフェースです。このパッケージで提供される衝突判定は Figure 同士を直接比較判定するのではなく、Figure の位置に近似的に配置されたシンプルな形状同士を比較判定することにより行われます。Shape インタフェースは、この比較判定で使用される全ての形状で実装されています。
Sphere	比較判定で使用される立体のうち、球を表すクラスです。
Triangle	比較判定で使用される非立体形状のうち、三角形を表すクラスです。
ViewVolume	視錐台（カメラから見た視界）における、BoundingVolume の可視判定を行う機能を提供するクラスです。このクラスを使用して非可視と判断されたものは配置上カメラの視界に入らないため、多くの物体の中から真に描画すべき物体を絞り込むことができます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

注意事項：

- 衝突判定機能を活用したｉアプリプログラミングの詳細については、株式会社エイチアイより提供されるドキュメント「Micro3D Programming for ｉアプリ」を参照してください。

13.2 3D サウンド制御機能

本プロファイルでは、携帯電話に搭載されたステレオスピーカーを応用して、左右だけでなく上方や後方などさまざまな方向から立体的にサウンドが聴こえるような効果を与えることのできる 3D サウンド制御機能を定義しています。この機能を使用することにより、聴者と仮想音源の位置関係（定位と呼びます）を指定してユーザーに特定の方向から音が聴こえるように感じさせたり、時間の経過とともに定位を変更して仮想音源が音を出しながら移動しているように感じさせたりすることができます。

本項では、3D サウンド制御 API の使用方法について解説します。

13.2.1 3D サウンド制御機能のクラス構成

3D サウンド制御機能は、`com.nttdocomo.ui` パッケージおよび `com.nttdocomo.ui.sound3d` パッケージで提供されます。また 3D グラフィックス描画機能と同様、3D に関する数値演算のために `com.nttdocomo.ui.util3d` パッケージで提供されるユーティリティ機能を使用します。

3D サウンド制御機能を構成する、または関連するクラス群を以下に示します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

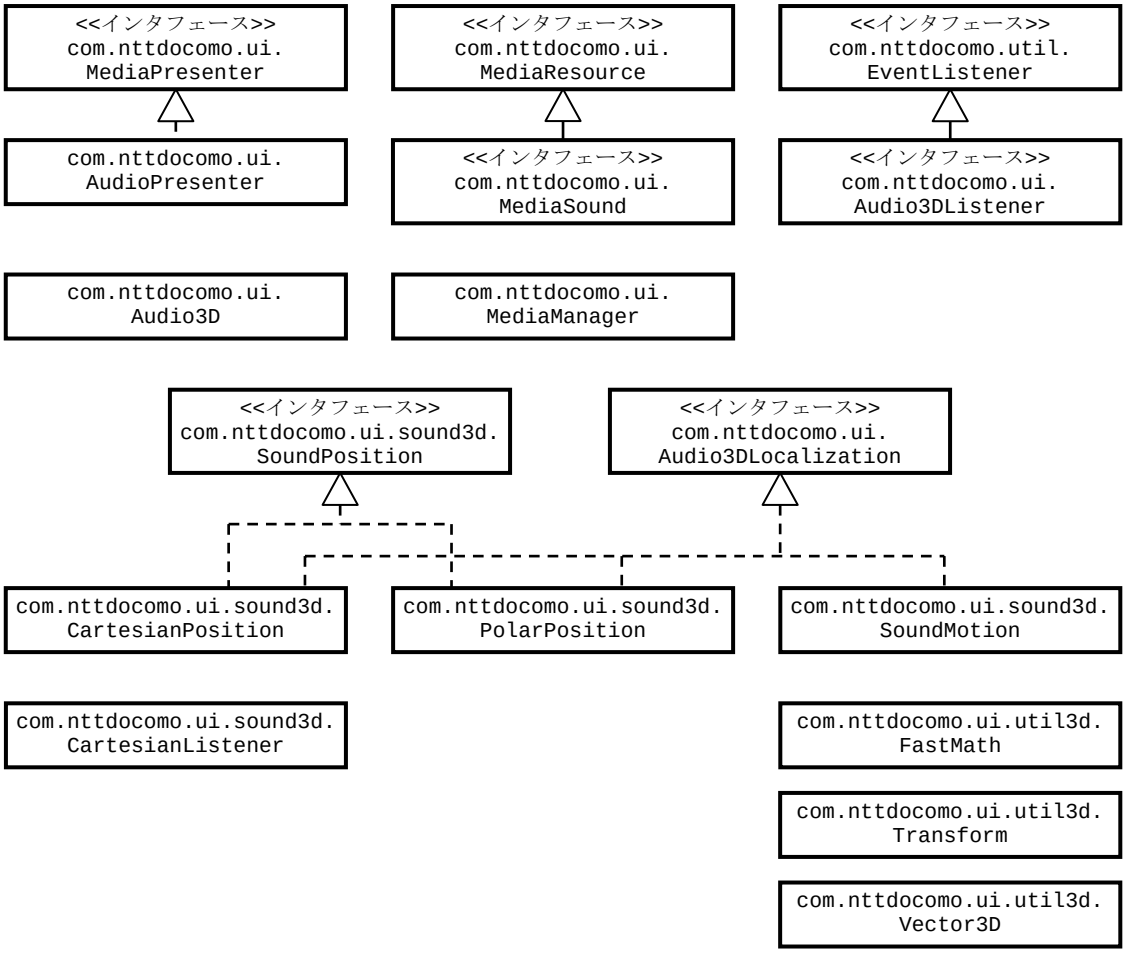


図 24: 3D サウンド制御機能のクラス構成

以下に 3D サウンド制御機能を構成する各クラス、インタフェースの機能概要を示します。

クラス・ インタフェース	機能概要
AudioPresenter	オーディオプレゼンタです。3D サウンド制御ではオーディオプレゼンタが仮想音源に相当し、各オーディオプレゼンタに対応付けられたコントローラ（Audio3D オブジェクト）によって定位（聴者と仮想音源の位置関係）を制御します。本プロファイルでは、AudioPresenter クラスにはそのプレゼンタに対応する Audio3D オブジェクトを取り出すための getAudio3D()メソッドが追加されています。
Audio3D	各オーディオプレゼンタ毎に 3D サウンド制御を行うためのコントローラです。Audio3D オブジェクトを使用して 3D サウンド制御の有効/無効を切り替えたり、定位の制御を行います。i アプリの中でいくつの仮想音源を使用できるか（3D サウンド制御リソース数）はメーカーにより異なりますが、どの機種でも利用可能な 3D サウンド制御リソース数は 4 つです。i アプリはこの制限の範囲内で Audio3D オブジェクトを使用することができます。 【DoJa-5.0】どの機種でも利用可能な 3D サウンド制御リソース数の規定は、DoJa-5.0 プロファイルで拡大されました。DoJa-4.x プロファイルでは、どの機種でも利用可能な 3D サウンド制御リソース数は 2 つです。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

MediaSound	サウンドデータを表現するメディアデータです。3D サウンド制御では i アプリから定位を指示することで 3D 制御を行う方法の他に、あらかじめサウンドデータに定位情報を埋め込んでおき、サウンド再生時にそれを使用して自動的に 3D 制御を行う方法を取ることもできます。サウンドデータに定位情報を埋め込む場合は 1 つのサウンドに複数の 3D サウンド制御リソースを割り当てることができます。本プロファイルの MediaSound クラスでは、対応するサウンドデータに埋め込まれた定位情報にいくつの 3D サウンド制御リソースが割り当てられているかを取得する機能が追加されています。
Audio3DLocalization	定位を表すインタフェースです。3D サウンド制御機能では定位を表現する方法（クラス）が複数ありますが、どのクラスもこのインタフェースを実装しています。 なお、このインタフェースではメソッドやフィールドは定義されていません。
SoundPosition	聴者の位置と仮想音源の位置によって表される、固定された定位を表すインタフェースです。このインタフェースを実装したクラスでは、聴者の位置と仮想音源の位置を指定することによって両者の位置関係を表します。 なお、このインタフェースではメソッドやフィールドは定義されていません。
PolarPosition	SoundPosition インタフェースを実装するクラスで、極座標系を使用して聴者と仮想音源の位置関係を表します。極座標系ではその原点に一定の方向を向いた聴者がいるものとし、そこからの方位角、仰角、距離を指定して仮想音源の位置を表現します。
CartesianPosition	SoundPosition インタフェースを実装するクラスで、x 軸、y 軸、z 軸の 3 軸から構成されるデカルト座標系を使用して聴者と仮想音源の位置関係を表します。 PolarPosition が常に聴者が原点に存在するローカル座標系と捉えることができるのに対し、CartesianPosition は任意の位置に存在する聴者と仮想音源を包含するいわゆるワールド座標系と捉えることができます。ただし CartesianPosition でも最終的には聴者からみた仮想音源の位置関係が計算され、使用されます。
CartesianListener	CartesianPosition クラスで定位を表現する際に、デカルト座標系上に置く聴者を表現するクラスです。デカルト座標系で定位を表すために必要となる聴者の向きの設定は、このクラスを使用して行います。
SoundMotion	SoundMotion クラスは「移動する定位」を表現するためのクラスです。移動開始からの経過時間とその時点での定位を経路情報として設定することで、その経路に沿って仮想音源が移動していくようにユーザーに感じさせることができます。
Audio3DListener	3Dサウンド制御に関するイベントをアプリケーションプログラムに通知するためのイベントリスナを定義するインタフェースです。本プロファイルでは、SoundMotion（移動する定位）における移動完了イベントが定義されています。
FastMath	3D 関連機能での使用を前提とした高速数値演算ユーティリティです。本プロファイルに対応する携帯電話では CLDC-1.1 が搭載されており、このバージョンの KVM がサポートする浮動小数点数演算が使用できますが、このクラスでは浮動小数点数演算を内部的に整数演算に置き換えます。このため、通常の浮動小数点数演算より大きな誤差を含む可能性はありますが高速に演算を行うことができます。
Transform	3 次元アフィン変換用の行列を扱うクラスです。CartesianListener に聴者の向きを与える際に、視点座標への変換行列を設定して使用することができます。
Vector3D	3 次元ベクトルを表すクラスです。内積・外積計算、正規化といったユーティリティメソッドを備えています。座標系上の位置や向きなどを表現するために使用されます。

13.2.2 Audio3D

Audio3D クラスは、3D サウンド制御を行うためのコントローラの役割を持ちます。Audio3D オブジェクトは AudioPresenter オブジェクト毎に割り当てられており、AudioPresenter.getAudio3D()メソッドを使用して取得することができます。

3D サウンド制御のオン、オフを切り替えるには、Audio3D.enable()メソッドおよび Audio3D.disable()メソッドを使用します。3D サウンド制御では i アプリからの定位の指示に

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.
より 3D 制御を行う方法の他に、あらかじめサウンドデータに定位情報を埋め込んでおき、サウンド再生時にその情報を使用して自動的に 3D 制御を行う方法を取ることもできます。3D 制御をサウンドデータに埋め込まれた定位情報を使用して行うか、あるいはサウンドデータに埋め込まれた定位情報を無視して i アプリからの定位の指示を有効とするかは、`enable()`メソッドの引数で指定します。

`Audio3D.enable()`を使用して 3D サウンド制御を有効にすると、システムはサウンドを 3D 制御するためのリソース（3D サウンド制御リソースと呼びます）を消費します。このリソースの総数はメーカーにより異なりますが、DoJa-5.0 プロファイルにおいてはどの機種でも最低限 4 つは備えています（DoJa-4.x プロファイルでは、どの機種でも備えている最低限の数は 2 つです）。3D サウンド制御リソースとは仮想音源、言い換えると仮想的に移動するスピーカーを表現するためのリソースであり、i アプリが定位を指示して制御を行うケースでは 1 つの `AudioPresenter` につき 1 つのリソースが消費されます。しかしサウンドデータに埋め込まれた定位情報を使用して制御を行うケースでは、サウンドデータの内容によっては 1 つの `AudioPresenter` が複数のリソースを使用することがあります。あるサウンドデータがいくつの 3D サウンド制御リソースとを使用するかは、`MediaSound.getProperty()`メソッドに引数 `MediaSound.AUDIO_3D_RESOURCES` を指定することで知ることができます。

3D 制御を有効にした後、定位を指定して具体的に聴者と仮想音源の位置関係を設定するには、`Audio3D.setLocalization()`メソッドを使用します。移動する定位（`SoundMotion`）で定位を制御する場合には、`Audio3D.setListener()`を呼び出すことで定位の移動完了をイベントとして通知を受けるためのイベントリスナーを使用することができます。

定位の指定方法は次項にて解説します。

注意事項：

- 移動する定位（`SoundMotion`）を使用しなくても、i アプリが一定の時間間隔ごとに定位の設定変更を繰り返すことで定位が移動しているように感じさせることができます。この場合、定位を変更する間隔は目安として `Audio3D.setTimeResolution()`が返す時間（単位 msec）以上を空けるようにしてください。定位の変更には一定の負荷が伴うため、このメソッドが返す間隔より小さな間隔で定位の変更を行っているとシステムに過度の負荷がかかり、意図通りの動作とならない場合があります。

13.2.3 定位の指定

聴者と仮想音源の位置関係を表す定位を表現するには、以下の 3 種類の方法があります。

`CartesianPosition / CartesianListener` を使用する

x 軸、y 軸、z 軸の 3 軸から構成されるデカルト座標系を用いて聴者と仮想音源の位置を指定します。聴者、仮想音源とも座標系内の任意の位置に存在することができますが、最終的には聴者からみた音源の位置（方向と距離）が計算されて実際の効果に反映されます。`CartesianListener` は、座標上に配置する聴者を表すクラスです。`CartesianListener` オブジェクトに聴者の位置と向きを設定し、そのオブジェクト、および仮想音源の位置を `CartesianPosition` オブジェクトに設定することで、両者の位置関係を表現します。

`PolarPosition` を使用する

Z 軸の正方向を向いた聴者を球状の座標の中心に置き、その聴者からの方位角、仰角および距離を指定することで仮想音源の位置を指定します。常に聴者が座標原点に存在しているため、指定した仮想音源の位置がそのまま聴者との位置関係を表します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

SoundMotion を使用する

CartesianPosition や PolarPosition が、アプリケーションプログラムからパラメータを設定変更しない限り自動的には移動しない固定された定位を表すのに対し、SoundMotion は指定された経路をたどって自動的に移動する定位を表します。経路情報は、移動開始からの経過時間とその時点での定位（CartesianPosition または PolarPosition）をセットとして与えることで指定します。経路情報を複数与えることで、複雑な経路をたどらせることもできます。SoundMotion を使用する場合、Audio3D.setListener()メソッドを使用して 3D サウンド制御リスナーを設定することで、定位の移動完了の通知をイベントとして受け取ることができます。

これらのクラスは、定位の表現のためのインタフェース Audio3DLocalization を実装しており、Audio3D.setLocalization()の引数に設定することができます。

SoundMotion は、定位の移動を行う前にその経路があらかじめ決まっている場合に使用します。これに対し、移動中の定位の経路がその場その場でのユーザー操作によって動的に決定付けられるような場合には SoundMotion は使用せず、CartesianPosition や PolarPosition を使用してアプリケーションプログラムが動的に位置の設定変更を行うようにします。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

第 14 章
ユーティリティ API

この章では、これまでに解説した機能カテゴリに含まれないユーティリティ機能について解説します。

14.1 デジタル署名 API

本項では、DoJa-4.1 プロファイルにて導入されたデジタル署名 API について解説します。

初期の機種を除く FOMA 携帯電話では、ユーザー毎に発行されたクライアント証明書を UIM カードに格納し、インターネットアクセスの際に SSL クライアント認証機能を提供している FirstPass を利用することができます。デジタル署名 API ではこの FirstPass のクライアント証明書を使用して、任意のデータにデジタル署名を施す（PKCS#7 SignedData 形式への変換）機能を提供します。本 API を使用することにより、署名者の認証とデータの完全性の確認を行うことができます。

注意事項：

- デジタル署名 API で署名を行う際は、FirstPass にて発行されたクライアント証明書を使用します。このため、デジタル署名を行う携帯電話では、あらかじめ UIM カードに FirstPass のクライアント証明書がダウンロードされている必要があります。また、FirstPass のサービスを受けるためには、バージョン 2 以降の UIM カードを使用している必要があります（バージョン 1 の UIM カードでは FirstPsss のサービスを受けることはできません）。現在携帯電話に挿入されている UIM カードのバージョン情報は、`com.nttdocomo.util.Phone` クラスの static メソッド `getProperty()`の引数に、`Phone.UIM_VERSION` を指定することにより調べることができます。
- デジタル署名 API で施された署名を検証するには、FirstPass における認証局の証明書（ドコモルート CA 証明書）を使用する必要があります。携帯電話上では、ドコモルート CA 証明書は工場出荷時に組み込まれています。また、サーバー側でドコモルート CA 証明書を利用するには、別途 NTT ドコモにサービスの申し込みを行う必要があります。
- FirstPass の詳細や、サーバー側でドコモルート CA 証明書を利用するための申し込みなどについては NTT ドコモの以下のサイトを参照してください。

  http://www.nttdocomo.co.jp/service/anshin/firstpass/index.html

【DoJa-4.1】

デジタル署名 API、および `Phone.UIM_VERSION` は DoJa-4.1 プロファイルにて新設されました。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

14.1.1 デジタル署名 API のクラス構成

デジタル署名 API を構成するクラスは、com.nttdocomo.security パッケージで提供されます。以下に、本機能のクラス構成を示します。

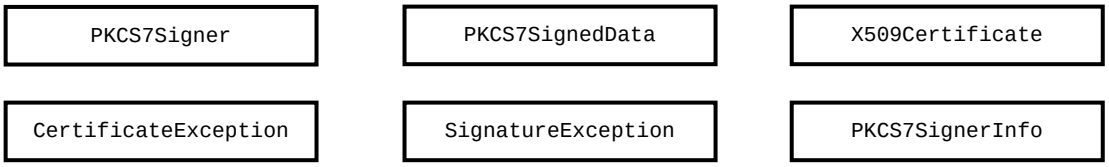


図 25: デジタル署名 API のクラス構成

各クラスの機能概要は以下の通りです。

クラス	機能概要
PKCS7Signer	デジタル署名付きデータを生成する機能を提供するクラスです。このクラスのオブジェクトに署名対象データを与えることでデジタル署名を行います。デジタル署名付きデータは、次に記載する PKCS7SignedData クラスで表現されます。
PKCS7SignedData	デジタル署名付きデータを表現するクラスです。このクラスのオブジェクトからは、デジタル署名付きデータのバイナリデータ表現を取り出したり、デジタル署名付きデータに含まれているコンテンツ（署名前のデータ）を取り出したりすることなどができます。
X509Certificate	X509 証明書を表現するクラスです。デジタル署名付きデータには、署名者の証明書情報が含まれています。デジタル署名 API では、デジタル署名付きデータ（PKCS7SignedData オブジェクト）に含まれる証明書情報を本クラスのオブジェクトとして取得することができます。
PKCS7SignerInfo	デジタル署名付きデータに含まれる署名者情報を表現するクラスです。このクラスのオブジェクトは、PKCS7SignedData オブジェクトから getSignerInfos()メソッドを使用して取り出すことができます。
CertificateException	証明書に関する例外を定義します。検証対象の署名に対応するルート証明書が端末にない場合や、ルート証明書は存在するが有効期限切れなどの理由で失効している場合などに発生します。
SignatureException	署名に関する処理の例外を定義します。デジタル署名実行時にユーザーに求められる暗証番号入力がキャンセルや失敗した場合、またはデジタル署名の検証結果が失敗だった場合などに発生します。

14.1.2 デジタル署名 API の利用

以下に、デジタル署名 API の利用方法の概要を示します。デジタル署名 API の利用は、主に以下の 2 つに大別されます。

- 署名対象データへのデジタル署名の実施（PKCS7Signer クラスの利用）
- デジタル署名付きデータの操作（PKCS7SignedData クラスの利用）

署名対象データへのデジタル署名の実施

署名対象データにデジタル署名を行う手順を以下に示します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

1) PKCS7Signer オブジェクトを、コンストラクタを使用して生成します。

2) PKCS7Signer オブジェクトの設定を行います。設定項目には、ダイジェストアルゴリズムの選択と、署名対象データのコンテンツタイプの設定の 2 つがあります。ダイジェストアルゴリズムの設定は、setDigestAlgorithm()メソッドに引数"SHA-1"もしくは"MD5"を指定することで行います。デフォルトのダイジェストアルゴリズムは SHA-1 です。またコンテンツタイプの設定は、setContentType()メソッドの引数に PKCS7Signer で定義される定数 DATA もしくは SIGNED_DATA を指定することで行います。通常のデータに署名を行う際は DATA を指定しますが、署名対象データ自身がデジタル署名付きデータである場合（二重にデジタル署名するような場合）には SIGNED_DATA を指定します。

3) update()メソッドを使用して、PKCS7Signer オブジェクトに署名対象データを設定します。update()メソッドを複数回呼ぶことで署名対象データを複数回に分けて設定したり、reset()メソッドを使用してそれまでに設定した署名対象データを破棄することができます。

4) 署名対象データを設定し終わった後で、sign()メソッドを呼び出します。このメソッドは署名対象データにデジタル署名を実施し、その結果として PKCS7SignedData オブジェクトを返します。デジタル署名処理において、ダイジェストアルゴリズムは 2)のフェーズで SHA-1 もしくは MD5 を選択することができますが、暗号方式は RSA 固定となります。

上記により取得した PKCS7SignedData オブジェクトは、次に示すデジタル署名付きデータの操作方法に従って操作することができます。

デジタル署名付きデータの操作

デジタル署名付きデータ（PKCS7SignedData オブジェクト）に対しては、以下のような操作を行うことができます。

・デジタル署名付きデータの検証

verify()メソッドを使用して、デジタル署名付きデータの検証を行うことができます。検証を行うことにより、署名者の認証とデータの完全性の確認を行うことができます。

・デジタル署名付きデータのバイナリデータ表現の取得

getEncoded()メソッドを使用して、デジタル署名付きデータを DER エンコードされた ASN.1 形式のバイナリデータとして取得することができます。このメソッドを使用することで、デジタル署名付きデータを、HTTP 通信などの手段により外部に送信可能なバイト配列として取得することができます。

PKCS7SignedData オブジェクトは、このバイナリデータを引数としてコンストラクタを呼び出すことでも生成することができます。このことを利用して、署名されたデータを別の携帯電話に送信し、送信先の i アプリでオブジェクト化した上で検証などの操作を行うといったことも可能です。

・デジタル署名付きデータに含まれているコンテンツ（署名前のデータ）の取得

getContent()メソッドを使用して、デジタル署名付きデータから署名前のコンテンツデータを取得することができます。

・デジタル署名付きデータに含まれている証明書情報の取得

getCertificates()メソッドを使用して、デジタル署名付きデータに含まれている証明書の情報を取得することができます。証明書の情報は X509Certificate クラスのオブジェクトとして表現されます。デジタル署名付きデータに複数の証明書が含まれている場合、それら全てについて証明書情報を取得することができます。X509Certificate オブジェクトからは、証明書の発行者と被認証者についての情報（一般名や国名、組織名など）や証明書の有効期間終了日といった情報を得ることができます。

・デジタル署名付きデータに含まれている署名者情報の取得

getSignerInfos()メソッドを使用して、デジタル署名付きデータに含まれている署名者情報を取得することができます。署名者の情報は PKCS7SignerInfo クラスのオブジェクトとして表現されます。
getSignerInfos()メソッドは、デジタル署名付きデータに含まれている全ての署名者情報を返します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

PKCS7SignerInfo オブジェクトからは、署名者情報の発行者識別名の設定値（一般名、組織名、部門名）やシリアルナンバーの情報を得ることができます。

以下に、デジタル署名 API で取り扱うデータの相関関係を示します。

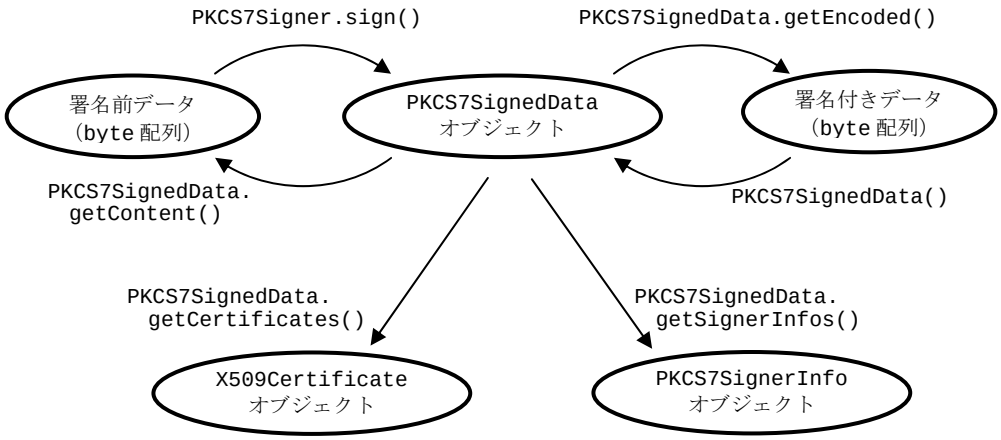


図 26: デジタル署名 API で取り扱うデータの相関関係

【DoJa-5.0】

DoJa-5.0 プロファイル以降の携帯電話の中には、FirstPass で提供されるクライアント証明書以外に、i モードサイト（携帯電話側では i モードブラウザ）を介してコンテンツプロバイダがユーザーにクライアント証明書を配布し、ユーザー側でそのクライアント証明書を利用可能とする機能が搭載される機種があります。DoJa-5.0 プロファイルでは、そのようにしてダウンロードされたクライアント証明書でデジタル署名を行えるようにするため、以下のような API が i アプリオプション API として規定されています。

CertificateStore クラス（com.nttdocomo.system パッケージ）：

クライアント証明書のネイティブ保存領域を操作するためのクラスです。このクラスの static メソッド selectEntryId()を呼び出すことで、携帯電話に保存されているクライアント証明書の中から任意の 1 つをユーザーが選択し、そのクライアント証明書のエントリ ID を i アプリに知らせることができます。

PKCS7Signer.sign(int)メソッド：

CertificateStore.selectEntryId()で取得したクライアント証明書のエントリ ID を指定することで、そのクライアント証明書を利用してデジタル署名を行います。

なお、DoJa-5.0 プロファイル対応の携帯電話のうち FOMA904i シリーズ以降のものには、X509Certificate クラスに証明書のシリアルナンバーを取得するための getSerialNumber()メソッドが追加されています。このメソッドは FOMA903i シリーズ以前の DoJa-5.0 プロファイル対応携帯電話には存在しないため、DoJa-5.0 プロファイルにおいてはあくまでも i アプリオプション API の位置づけとなります。FOMA903i シリーズ以前の携帯電話でも i アプリを動作させる前提では、X509Certificate.getSerialNumber()メソッドは使用しないようにしてください。

また DoJa-5.0 プロファイルにて、SignatureException に規定されている例外ステータスに以下の 2 つが追加されました。

- ENCRYPTED_DIGEST_ERROR
- SECURITY_CODE_REJECTED

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

【DoJa-5.1】

`PKCS7SignerInfo` クラスおよび `PKCS7SignedData.getSignerInfos()`メソッドは、DoJa-5.1 プロファイルにて追加されました。また DoJa-5.1 プロファイルにおいて、`CertificateException` のステータス定数に、デジタル署名付きデータに含まれる証明書が有効期限外であった場合のエラーを表す `INVALID` が追加されました。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

第 15 章
i アプリのビルド、テスト、およびパッケージ化

この章では、 i アプリの開発サイクルについて概要を説明します。NTT ドコモでは、PC 上で i アプリを開発するための開発環境（以降 i アプリ開発環境とします）を提供していますが、ここでは i アプリ開発環境および付属のツールの個々の詳細な情報には触れません。i アプリ開発環境の詳細については、 i アプリ開発環境のユーザーズガイドを参照してください。

開発者は、Java 2 SDK, Standard Edition（バージョン 1.3.1　以降 J2SE SDK とします）を利用したことがあり、J2SE SDK に含まれるツールやドキュメントの知識があるものとします。

なお、開発した i アプリをテストのために実際の携帯電話にインストールするには、i アプリを配布するための Web サイト（インターネットに接続されている必要があります）を設定する必要があります。 i アプリの配布については、第 15 章を参照してください。

15.1 i アプリ開発環境のインストール

i アプリ開発環境を使用するには、NTT ドコモが配布する i アプリ開発環境のアーカイブファイルを入手し、PC 上でインストール作業を行います。 i アプリ開発環境のインストール手順の詳細については、 i アプリ開発環境のユーザーズガイドを参照してください。

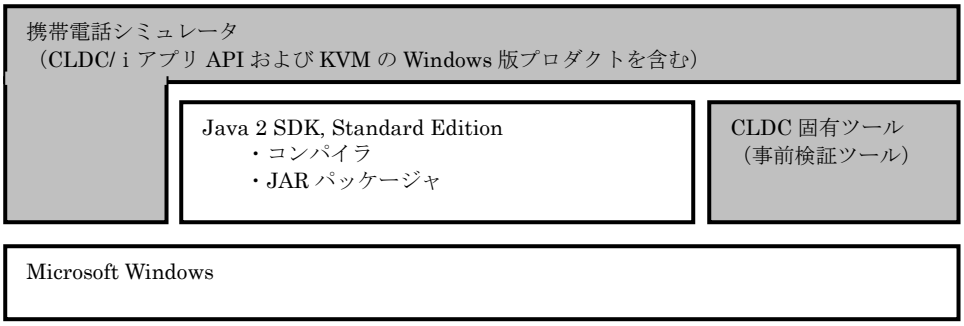
なお、 i アプリ開発環境をインストールするには、 i アプリ開発環境のほかに J2SE SDK を用意する必要があります。J2SE SDK をインストールした上で i アプリ開発環境をインストールしてください。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

15.2 i アプリ開発環境のコンポーネント

i アプリ開発環境は、PC 上で i アプリを開発するための簡易な開発環境です。これを使用することで、PC 上で i アプリをビルド、シミュレート実行することができます。

i アプリ開発環境は以下のようなコンポーネント構成を取っています。



注意： ハッチングされたコンポーネントが i アプリ開発環境に含まれています。
また、i アプリ開発環境がサポートする動作環境（Microsoft Windows や Java 2 SDK のバージョンなど）は、i アプリ開発環境のバージョン（対応プロファイル）により異なります。詳細については、i アプリ開発環境のユーザーズガイドを参照してください。

図 27: i アプリ開発環境の構成

以下に i アプリ開発環境に含まれる各コンポーネントの概要を示します。

- 携帯電話シミュレータ
  i アプリ開発環境のメインコンポーネントです。開発者は携帯電話シミュレータを使用して i アプリのビルド（コンパイル、事前検証、JAR パッケージング）とテスト実行を行うことができます。なお i アプリのビルドは、各ビルド用コマンドをコマンドラインから呼び出して行うこともできます。
  本書では、携帯電話シミュレータをエミュレータと表記することがあります。

- CLDC 固有ツール
  CLDC 固有ツールには事前検証ツール（preverify）があります。事前検証とは、標準的な Java 仮想マシンで行われるバイトコード検証作業の一部を事前（アプリケーションビルド時）に済ませておくことを言います。CLDC では事前検証方式を採用することにより、アプリケーション実行時の検証作業実行コストを削減しています。 i アプリ実行環境では、事前検証を行っていないクラスを使用することはできません。
  事前検証ツールは携帯電話シミュレータの i アプリビルド機能からだけでなく、コマンドラインから単独のツールとして使用することもできます。

【DoJa-2.0】

DoJa-2.0 プロファイル以降向けに提供される i アプリ開発環境は、DoJa-1.0 プロファイル向けの i アプリ開発環境と若干コンポーネント構成が変更されています。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

15.3 開発サイクルの概要

次に、一般的な開発サイクルを示します。

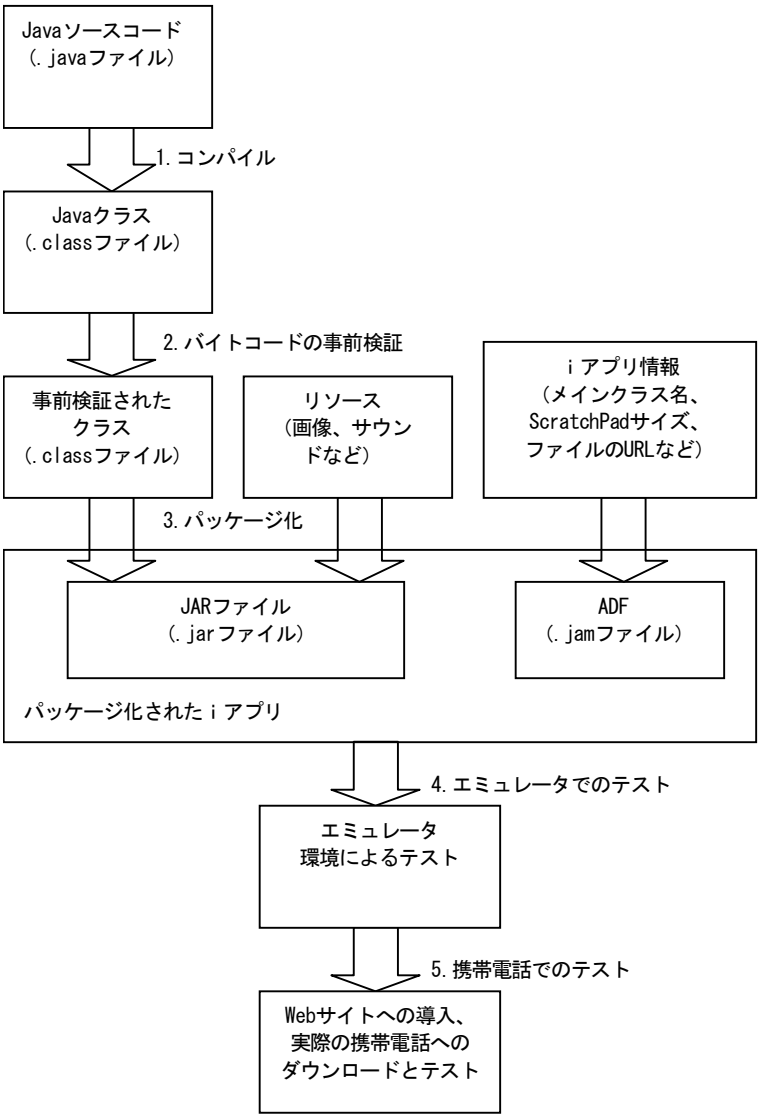


図 28. 開発サイクル

コンパイル – ソースファイルをクラスファイルにコンパイルするには、J2SE SDK の Java コンパイラ（javac）を使用します。このときコンパイラは、 i アプリ実行環境に属していないパッケージやクラス、メソッドがアプリケーションプログラムで使用されていないか検査します。

バイトコードの事前検証 – エミュレータのバイトコード事前検証ツール（preverify）では、Java クラスに対する事前検証プロセスを行います。事前検証ツールを実行すると追加の属性情報がクラスファイルに書き込まれるため、CLDC 仮想マシンは実行時のバイトコード検査を効率的に行うことができます。さらに、事前検証ツールでは、アプリケーションプログラムでファイナライズメソッドなど CLDC の制約機能が使用されていないか検査します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

パッケージ化 – 事前検証済みのすべてのクラスファイルとリソースファイルを JAR ファイルとしてパッケージ化するには、J2SE SDK のアーカイブツール（jar）を使用します。jar コマンドの使用方法は、J2SE SDK における jar コマンドの標準的な使用方法と同じです。携帯電話に保存される JAR ファイルサイズを最小限に保つため、非圧縮オプションは指定しないようにしてください。

エミュレータでのテスト – i アプリの機能が正しく動作するかどうかをエミュレータ環境で確認します。

実際の i アプリ対応携帯電話でのテスト – i アプリ（JAR ファイル、ADF）およびダウンロードナビゲート用 HTML コンテンツを Web サイトに配置し、実際の携帯電話にダウンロードします。そして、携帯電話で i アプリを実行して機能が正しく動作するか確認します。使用可能なメモリ容量など、携帯電話によってハードウェア・リソースの容量が異なる場合があるので、ターゲットとする全てのメーカーの携帯電話で i アプリの動作確認を行うようにしてください。

コンパイル、事前検証、パッケージ化の一連の流れ（ビルド作業）は個々のツールを独立して使用する以外に、エミュレータの持つ i アプリビルド機能を使用して一括して行うこともできます。エミュレータの i アプリビルド機能については、i アプリ開発環境のユーザーズガイドを参照してください。

15.4 通信先アプリケーションの作成

クライアント/サーバー型 i アプリでは、 i アプリの通信相手となるサーバー側アプリケーションを作成する必要があります。サーバー側アプリケーションは、クライアント側アプリケーション（i アプリ）と HTTP プロトコルで通信するものであればどのような技術を使用したものでも構いません。

また i アプリが OBEX 外部接続を使用する場合、通信先となる OBEX アプリケーションを用意する必要があります。 i アプリ対応携帯電話間で OBEX 外部接続を利用するケースでは、 i アプリ API を使用して OBEX クライアント、OBEX サーバーとも開発することができます。携帯電話とその他の機器を接続する場合、通信先アプリケーションの開発については、通信先となるプラットフォームで利用する OBEX アプリケーション開発環境のドキュメントを参照してください。

15.5 i アプリのパッケージ化

図 24 に示したように、i アプリは、ADF と JAR ファイル（Java クラスファイルおよびリソースファイルを含んでいる）の 2 つのファイルから構成されます。

15.5.1 ADF の作成

ADF は、SJIS コードで記述されたテキストファイルです。このファイルは、”=”で分離されたキーと値のペアからなる行で構成されます。ファイルの編集には、任意のテキストエディターを使用することができます。このファイルには、次のキーのエントリ（行）を指定します。

AppName: 必須キー

i アプリ名（最大 16 バイト）。携帯電話で i アプリ一覧を表示させた際に、ここに指定された名前が表示されます。SJIS で日本語名を指定することができます。

AppVer: オプショナルキー

i アプリのバージョン（最大 10 バイト）。開発者はこのキーを利用して i アプリのバージョン管理を行うことができます。ASCII 文字のみ指定可能です。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

PackageURL: 必須キー

iアプリのJARファイルのURL（最大255バイト）。相対パスを指定した場合、そのベースディレクトリはADFの位置となります。ASCII文字のみ指定可能です。なお、URLに指定するホスト名はシンボリックでなければなりません（"www.nttdocomo.co.jp"など）。数値IPアドレス（"127.0.0.1"など）は使用できません。

AppSize: 必須キー

JARファイルのサイズ。バイト単位の整数値を指定します。JARファイルサイズには、3.2項で解説した制限があります。

AppClass: 必須キー

iアプリの起動に使われるメインクラス名（フルパッケージ名付き・最大255バイト）。ASCII文字のみ指定可能です。

AppParam: オプショナルキー

メインクラスの起動パラメータ。パラメータ間はスペースで区切ります（最大255バイト）。これらのパラメータをアプリケーションプログラムから取得するには、`com.nttdocomo.ui.IApplication.getArgs()`メソッドを使用します。ASCII文字のみ指定可能です。省略時は起動パラメータがないものとします。

ConfigurationVer(KvmVer): オプショナルキー

J2MEコンフィグレーションバージョン。このキーが指定されていない場合は、そのiアプリがCLDC仮想マシンのすべてのバージョンで動作することを示します。`ConfigurationVer`と`KvmVer`は同じ意味を持ちますが、1つのADFに両者を同時に指定することはできません。`KvmVer`はDoJa-1.0プロファイルとの互換性のために用意されています。このキーの設定値は以下のフォーマットに従います。

`CLDC-x.x`　（xは`0`〜`9`の数字。本バージョンでは`"1.0"`および`"1.1"`をサポート）

ProfileVer: オプショナルキー（DoJa-2.0プロファイルにて新設）

iアプリ実行環境のプロファイルバージョン。このキーが指定されていない場合は、そのiアプリがiアプリ実行環境のすべてのバージョンで動作することを示します。このキーの設定値は以下のフォーマットに従います。

`DoJa-x.x`　（xは`0`〜`9`の数字）

本バージョンでは以下をサポートしています。

- `DoJa-1.0`（503iシリーズ対応）
- `DoJa-2.0`（504iシリーズ対応）
- `DoJa-2.1`（FOMA02モデル対応）
- `DoJa-2.2`（FOMA03前期モデル対応）
- `DoJa-3.0`（505iシリーズ対応）
- `DoJa-3.5`（FOMA900iシリーズ対応）
- `DoJa-4.0`（FOMA901iシリーズ対応）
- `DoJa-4.1`（FOMA902iシリーズ対応）
- `DoJa-5.0`（FOMA903iシリーズ、FOMA904iシリーズ対応）
- `DoJa-5.1`（FOMA905iシリーズ、FOMA906iシリーズ対応）

SPsize: オプショナルキー

ScratchPadのサイズ。バイト単位の整数値を指定します。iアプリが使用可能なScratchPadサイズには、3.2項で解説した制限があります。省略時はScratchPadを使用しないものとします。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

DoJa-3.0 プロファイル以降では、ScratchPad を内部的に最大 16 個まで分割管理することができます。ScratchPad を分割するには、SPsize キーの値として、個々のバイト数をカンマ区切りで列挙します（カンマの前後に空白は含めないでください）。ScratchPad にアクセスする際は、URL に ScratchPad の番号（0～最大 15 まで）を指定しますが、その番号はこのキーで指定されたサイズの並びに対応します。

（"SPsize = 1024,2048" と記載されている場合、0 番目の ScratchPad のサイズが 1024 バイトに、1 番目の ScratchPad のサイズが 2048 バイトになります。）

ScratchPad を分割管理する場合は、分割された個々の合計サイズがサイズ制限の範囲内に収まるようにしてください。

LastModified: 必須キー

i アプリの最終変更日時。i アプリの更新は、このキーの内容に基づいて行われます。フォーマットは以下の通りです。

```
Dow, DD Mon YYYY HH:MM:SS TimeZone
```

Dow：曜日　DD：日　Mon：月　YYYY：年

HH：時間　MM：分　SS：秒　TimeZone：タイムゾーン

タイムゾーンは省略可能です。DoJa-1.0 プロファイルではタイムゾーンの指定はサポートされないため、DoJa-1.0 プロファイル対応の携帯電話と DoJa-2.0 プロファイル対応の携帯電話で ADF を共有する場合は、タイムゾーンは省略するようにしてください（エミュレータが作成する ADF では、タイムゾーンの設定は行われません）。

タイムゾーンは"GMT"および"JST"に対応しています。また、"+0900"というようにタイムゾーンを時差で指定することもできます。タイムゾーンが指定されていない場合、"JST"が指定されたものとします。

指定例：　Fri, 25 Feb 2000 12:04:25 JST

UseNetwork: オプショナルキー

ネットワーク機能を使用する場合に、通信方式を指定します。i アプリ実行環境では"http"が唯一の有効な値です（HTTPS 使用時も"http"を指定します）。この指定がない i アプリは、ネットワーク接続を要求することはできません。

TargetDevice: オプショナルキー

ターゲットとする携帯電話の機種名を指定します（最大 128 バイト）。すべての機種をターゲットとする場合はエントリ自体を指定しません。このキーには以下のいずれかの形式で値を設定します。

- ターゲットとなる機種名の個別指定。機種名には、ユーザーエージェントに含まれている名前と同等のものを使用します。ターゲットとなる機種が複数存在する場合は、コンマ区切りで列挙します。

  例：　X901i,Y901i,Z901i

- ターゲットとなるシリーズ全機種の一括指定。"all901"のように、キーワード"all"の後にシリーズ名を指定します。ターゲットとなるシリーズが複数存在する場合は、コンマ区切りで列挙します。なお、この指定方法は DoJa-1.0 プロファイルのみに対応した携帯電話では解釈できないため注意してください。

LaunchAt: オプショナルキー

i アプリに自動起動を行わせる場合、その起動タイミングを指定します。先頭の 1 文字は形式指示子です。i アプリ実行環境では、形式指示子としてインターバルを表す"I"をサポートします。

✧　時間間隔指定形式（形式指示子：I）

フォーマット：　I h

（h：　時間　有効桁数は 2 桁　"I"と数字の間にはスペースが 1 つ必要です）

指定例：　I 6　（6 時間毎に起動する）

時間間隔の起点は、i アプリが携帯電話にダウンロードされた時刻となります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

iアプリが自動起動するタイミングで他のアプリケーションが実行されていたり、通話中であったりした場合、そのiアプリは自動起動できません。その場合、携帯電話は各機種固有の方法で、自動起動できなかったiアプリがあることを利用者に通知します。

なおメーカーによっては携帯電話の機能として、iアプリを特定の時刻に自動起動するようユーザーによる登録設定が行える機能を搭載していることがあります。このような機能で自動起動される場合、iアプリの起動形態はタイマー起動（`IApplication.LAUNCHED_FROM_TIMER`）ではなく通常起動（`IApplication.LAUNCHED_FROM_MENU`など）となります。

`MyConcierge:` オプショナルキー（DoJa-2.0プロファイルにて新設）

待ち受けアプリケーションであることを宣言します。値に`"Yes"`が指定されている場合、そのiアプリは待ち受けアプリケーションとして利用可能であることを示します。この指定がないiアプリは、待ち受けアプリケーションとして携帯電話に登録することはできません。

`UseTelephone:` オプショナルキー（DoJa-2.0プロファイルにて新設）

通話機能を利用するiアプリであることを宣言します。値に`"call"`が指定されている場合、そのiアプリは通話発信（`com.nttdocomo.util.Phone.call()`メソッド）を使用することができます。この指定がないiアプリは、通話発信を利用することはできません。

`UseBrowser:` オプショナルキー（DoJa-2.0プロファイルにて新設）

ブラウザ機能を利用するiアプリであることを宣言します。値に`"launch"`が指定されている場合、そのiアプリは`com.nttdocomo.ui.IApplication.launch()`メソッドを使用してブラウザを起動することができます。この指定がないiアプリは、ブラウザを起動することはできません。また、iアプリが待ち受け実行されている場合は、このキーの設定に関わらずブラウザを起動することはできません。

`LaunchByMail:` オプショナルキー（DoJa-2.0プロファイルにて新設）

このiアプリがメールからの起動を許可することを宣言します。値には、起動を許可するメールの送信元メールアドレスを1つだけASCII形式で指定します（最大50バイト）。このキーの値が、後方一致比較の結果として送信元メールアドレスに全て含まれていた場合、そのメールからiアプリを起動することを許可します。なお、このキーに値`"any"`が指定されている場合、送信元メールアドレスの制限は行いません。

`LaunchByBrowser:` オプショナルキー（DoJa-2.0プロファイルにて新設）

このiアプリがブラウザからの起動を許可することを宣言します。値には、起動を許可するWebページのURLを1つだけASCII形式で指定します（最大255バイト）。このキーの値が、前方一致比較の結果としてブラウザで表示中のURLに全て含まれていた場合、そのWebページからiアプリを起動することを許可します。このキーに値`"any"`が指定されている場合、URLの制限は行いません。

なお、トルカビューアからのiアプリ起動を行う場合は`LaunchByBrowser`キーの代わりに`LaunchByToruCa`キーを使用します。前方一致比較の適用先がWebページのURLではなくトルカ（詳細）の取得先URLとなる他は、`LaunchByToruCa`キーの使用方法は`LaunchByBrowser`キーと同じです。

`AllowPushBy:` オプショナルキー（DoJa-2.0プロファイルにて新設）

このiアプリが、外部インタフェースからの起動を許可することを宣言します。外部インタフェースからの起動には、赤外線ポートからのvTriggerオブジェクト受信による起動、およびコード認識機能における2次元バーコードからの起動の2種類があります。

＜赤外線ポートからのvTriggerオブジェクト受信による起動＞

`AllowPushBy`キーの値には、以下の形式のパラメータを1つだけASCII形式で指定します（接頭辞`"Irda:"`を含めて最大255バイト）。

```
Irda:<command>
```

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

<*command*>には、 i アプリ起動を許可する vTrigger オブジェクトに指定される起動コマンドを指定します。このキーの値が、 i アプリに対して起動要求を行った vTrigger オブジェクトの起動コマンドと完全一致した場合、その vTrigger オブジェクトによって i アプリを起動することを許可します。

なお、<*command*>には空白およびタブを含めることはできません。

＜コード認識機能における 2 次元バーコードからの起動＞

AllowPushBy キーの値には、以下の形式のパラメータを 1 つだけ ASCII 形式で指定します（接頭辞"Code:"を含めて最大 255 バイト）。

Code:<*command*>

<*command*>には、 i アプリ起動を許可する 2 次元バーコードに指定される起動コマンドを指定します。このキーの値が、 i アプリに対して起動要求を行った 2 次元バーコードの起動コマンドと完全一致した場合、その 2 次元バーコードによって i アプリを起動することを許可します。

なお、<*command*>には空白およびタブを含めることはできません。

AppTrace: オプショナルキー（DoJa-2.0 プロファイルにて新設）

i アプリ開発のフェーズにおいて、携帯電話上で簡易なデバッグ機能を提供するためのキーです。このキーに値"on"を指定すると、携帯電話上で以下の機能が有効となります。

- キャッチされない例外により i アプリが異常終了した場合、どのような例外により i アプリが異常終了したかを表示します。
- i アプリから標準出力または標準エラー出力に出力した文字列をロギングし、 i アプリ終了後に表示させることができます。ただしロギング可能な文字列長には制限があり（最大 512 バイト）、新しく書き込まれたものから順に制限の範囲内のものが表示されます。なお、java.lang.Throwable.printStackTrace()を呼び出した結果はこのログには保存されません。
- JAM の i アプリ情報表示機能が詳細モードとなります。

i アプリの開発が完了した際には、本キーは ADF から削除してください。

DrawArea: オプショナルキー（DoJa-2.0 プロファイルにて新設）

この i アプリが前提としている画面サイズ（ i アプリの描画エリアサイズ）を指定します。大きな描画エリアを持つ携帯電話で小さい描画エリアを前提とした i アプリを実行すると、ディスプレイに対しセンタリングされた、指定サイズの矩形領域内で i アプリが動作します。矩形領域の外側の部分に i アプリから描画を行うことはできません（その部分の表示色はメーカーにより異なります）。

キーの値は以下のように指定します。

<*width*>x<*height*>　　（<*width*>は横、<*height*>は縦のドット数　"x"は小文字のみ有効）

例：　DrawArea = 120x120

<*width*>の設定可能最小値は 96、<*height*>の設定可能最小値は 72 です。また、このキーで指定された描画エリアサイズは、 i アプリをダウンロードしようとしている携帯電話の描画エリアサイズ以下である必要があります（横または縦のいずれも、ダウンロードしようとしている携帯電話のサイズ以下でなければなりません）。これらの条件を満たさない設定が行われている場合、その i アプリをダウンロードすることはできません。

なお、このキーが指定されていない場合は、その携帯電話の描画エリアを全て使用して i アプリを実行します。

【DoJa-5.0】

DoJa-5.0 プロファイル以降、メーカーによっては ADF の DrawArea キーを使用することで、デフォルト（DrawArea キー未使用時）の表示領域よりも広い表示領域を使用できる場合があります。各機種で利用可能な最大の表示領域サイズは、別途 NTT ドコモより公開されます。なお、デフォルトの表示領域よりも広い表示領域を使用する i アプリでは、Panel は使用せず Canvas のみで画面を構成することを推奨します。この機能は Canvas と組み合わせて使用することを想定して実装されており、Panel と組み

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

合わせて使用した場合には画面表示やコンポーネント操作などの点でアプリ開発者の意図とは異なった動作をする可能性があります。

GetSysInfo: オプショナルキー（DoJa-2.0 プロファイルにて新設）

このｉアプリが、携帯電話のアイコン情報を参照することを宣言します。このキーに値"yes"が指定されているｉアプリがダウンロードされると、携帯電話はｉアプリにアイコン情報の参照を許可するかユーザーに確認を求めます。このキーが指定されていないｉアプリやユーザーから許可を得ていないｉアプリは、アイコン情報を参照することはできません。

なおアイコン情報の取得には、`PhoneSystem.getAttribute()`メソッドを使用します。

【DoJa-2.0】【DoJa-3.0】【DoJa-5.1】

DoJa-2.0 プロファイルでは、参照可能なアイコン情報はメール・メッセージ受信状態です。また DoJa-3.0 プロファイル以降では、メール・メッセージ受信状態の他に電波強度、電池残量、マナーモード設定状態を参照することもできます。

なお、DoJa-5.1 プロファイルにおいて、電波強度リソースはエリア情報リソースに置き換えられました。詳細は、本書 第 8 章を参照してください。

LaunchApp: オプショナルキー（DoJa-3.0 プロファイルにて新設）

アプリケーション連携により他のｉアプリを起動（連携モードまたはランチャーモード）するｉアプリであることを宣言します。値に"yes"が指定されている場合、そのｉアプリは`com.nttdocomo.ui.IApplication.launch()`メソッドを使用して他のｉアプリを起動することができます。この指定がないｉアプリは、他のｉアプリを起動することはできません。

LaunchByApp: オプショナルキー（DoJa-3.0 プロファイルにて新設）

アプリケーション連携により他のｉアプリから起動されようとする場合の動作を制御します。値に"deny"を指定することで、そのｉアプリが他のｉアプリからランチャーモードで連携起動されることを拒否します。アプリケーション連携によるｉアプリの起動は、起動先ｉアプリが待ち受けアプリケーションであっても通常起動となり待ち受け起動とはなりません。通常起動されることを前提としていない待ち受けアプリケーションなどは、このキーを使用して連携起動を禁止することができます。

AccessUserInfo: オプショナルキー（DoJa-3.0 プロファイルにて新設）

アプリケーション連携により、携帯電話のユーザーデータ領域にアクセスするｉアプリであることを宣言します。値に"yes"が指定されている場合、そのｉアプリは以下のユーザーデータ操作を行うことができます。

- 電話帳エントリ、電話帳グループの登録
- ブックマークの登録
- スケジュールの登録
- 画像の登録・選択・取得

GetUtn: オプショナルキー（DoJa-2.1 プロファイルにて新設）

このｉアプリが、携帯電話の個体識別情報を参照することを宣言します。このキーに値"terminalid"が指定されている場合は携帯電話本体の識別情報を、また値"userid"が指定されている場合は UIM カードの識別情報を参照することができます。携帯電話本体の識別情報と UIM カードの識別情報を両方参照する場合は、これら 2 つをカンマ区切りで両方指定します（カンマの前後に空白は含めないでください）。

このキーが指定されたｉアプリがダウンロードされると、携帯電話はｉアプリに個体識別情報の参照を許可するかユーザーに確認を求めます。このキーが指定されていないｉアプリやユーザーから許可を得ていないｉアプリは、個体識別情報を参照することはできません。また、UIM カードを搭載していない携帯電話では、UIM の識別情報を取得することはできません。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

IletPreserve: オプショナルキー（DoJa-3.0 プロファイルにて新設）

ダウンロード即起動 i アプリが、自身の携帯電話への保存を禁止することを宣言します。このキーに値 "deny"が指定されている場合、ユーザーはその i アプリを携帯電話に保存することはできません。

このキーが設定されていない場合、ダウンロード即起動 i アプリの実行が終了した後、携帯電話はその i アプリを通常の i アプリとして携帯電話に保存するかどうかをユーザーに確認します。

ダウンロード即起動 i アプリの詳細については第 16 章を参照してください。

AppIcon: オプショナルキー（DoJa-4.1 プロファイルにて IC アプリ向けに新設、DoJa-5.0 プロファイルにて IC アプリ以外にも適用）

メーカーによっては、携帯電話ネイティブの i アプリ一覧表示画面（IC カードメニューなども含みます）において、各 i アプリの名称の他にその i アプリ独自のアイコンを表示できる機種があります。AppIcon キーは、そのような機種においてアイコンとして使用する画像を指定するために使用します（アイコンを持たない i アプリについては、システムが持つデフォルトのアイコンが表示されます）。

アイコンデータの画像ファイルは、アプリケーション Jar ファイル内のルートディレクトリの位置に同梱します。AppIcon キーでは、この画像ファイルのファイル名を以下の書式で指定します。

```
AppIcon = <ファイル名 1> [,<ファイル名 2>]
```

ファイル名 1 には縦横ドットサイズが 48×48 ドットの画像ファイルのファイル名を、またファイル名 2 を指定する場合は 96×96 ドットの画像ファイルのファイル名を指定します（カンマの前後に空白は含めないでください）。メーカーにより、アイコンとしてサポートする画像のサイズがこれら 2 通りに分かれますが、AppIcon キーにファイル名を 2 つ指定している場合には、その i アプリがダウンロードされた機種に応じて適切なサイズのものが選択されます。

なお、アイコンデータには、アニメーション GIF やインターレース GIF を除く GIF87a、GIF89a データ、および JPEG データを使用することができます。またファイル名には ASCII 文字のみ使用可能であり、パス区切りを表す“¥”や“/”は使用できません。

【DoJa-5.1】

アイコンへの JPEG 形式データの利用は、DoJa-5.1 プロファイルにてサポートされました。DoJa-5.0 プロファイル以前では、アイコンに利用できる画像は GIF 形式データのみです。

AppMainTitle: オプショナルキー（DoJa-5.1 プロファイルにて新設）

DoJa-5.1 プロファイル（FOMA905i シリーズ）以降、携帯電話の i アプリ一覧表示画面の表示スタイルに、従来の一覧表示形式に加えグラフィカル表示形式の機能がオプショナルとして追加されました。グラフィカル表示形式の i アプリ一覧画面では、ユーザーの選択操作でフォーカスが当たっている i アプリについて、AppIcon キーのアイコンよりも大きな（160×160 ドット）画像でメインタイトル画像を表示することができます（メインタイトル画像を持たない i アプリについては、システムが持つデフォルトの画像が表示されます）。

メインタイトル画像の画像ファイルは、アプリケーション Jar ファイル内のルートディレクトリの位置に同梱します。AppMainTitle キーでは、この画像ファイルのファイル名を以下の書式で指定します。

```
AppMainTitle = <ファイル名>
```

ファイル名には縦横ドットサイズが 160×160 ドットの画像ファイルのファイル名を指定します。

なお、メインタイトル画像のデータには、アニメーション GIF やインターレース GIF を除く GIF87a、GIF89a データ、および JPEG データを使用することができます。またファイル名には ASCII 文字のみ使用可能であり、パス区切りを表す“¥”や“/”は使用できません。

DeniedMultiApp: オプショナルキー（DoJa-5.1 プロファイルにて新設）

FOMA 携帯電話の中には、複数の端末アプリケーション（例えばブラウザと通話機能など）を同時に起動できるものがあります。機種によっては i アプリと音楽プレーヤー、もしくは i アプリとデジタルテレビを同時に起動して、音楽やテレビ音声を聞きながら i アプリを利用するといった使い方をサポートするものもあります。しかし、音楽プレーヤーやデジタルテレビ機能はそれら単体でも相応の CPU パワーを必要とし、 i

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

アプリとそれらを同時に利用している場合にはiアプリに十分なCPUパワーを提供できない場合があります（このような状況が問題となるiアプリは、ユーザには一般にiアプリの動作が重くなったり、動作速度が不安定になったりするように見えます）。

このためDoJa-5.1プロファイルでは、CPUパワーをフルに使用する必要があるiアプリについて、音楽プレーヤーやデジタルテレビなどとの同時実行を抑止するための`DeniedMultiApp`キーが新設されました。`DeniedMultiApp`キーの書式は以下の通りです。

```
DeniedMultiApp = <アプリケーション名> [,<アプリケーション名> ... ]
```

アプリケーション名には、そのiアプリと同時に起動することを抑止するアプリケーション名を指定します。現在のプロファイルでは、アプリケーション名には`music`（音楽プレーヤー）、`dtv`（デジタルテレビ）の2つが規定されています。また`all`を指定することで、音楽プレーヤーとデジタルテレビの両方を対象とすることもできます。

なお、`IletPreserve`キーによって携帯電話への保存を禁じているダウンロード即起動iアプリでは、このキーを併用しないことを推奨します。これらを併用したダウンロード即起動iアプリは、同時起動抑止の対象となっている機能が実行されている携帯電話上ではダウンロードしても利用することができません。

以下にADFの記述例を示します。ADFは下記に示すとおり、1つのキーエントリを1行で記述しなければなりません。各行の行末は、最終行も含めCR LF（0x0d0a）とします。

```
AppName = バンキングデモ
ConfigurationVer = CLDC-1.0
ProfileVer = DoJa-3.0
AppClass = banking.BankingDemo
AppVer = 1.0
PackageURL = http://aserver.com:8080/BankingDemo.jar
AppSize = 17556
LastModified = Fri, 25 Feb 2000 12:04:25
```

15.5.2 JARファイルの作成

JARは､Javaアプリケーションを構成するクラスファイルとリソースファイルを圧縮形式で格納するための、標準のJavaアーカイブ機能です。JARファイルでは、そのサイズが圧縮前のファイルの合計サイズと比較しておよそ40%から50%削減されます（JARファイルに含まれる内容により圧縮率は変動します）。

iアプリもJARファイルにパッケージングされた形で携帯電話に配布されます。これによって携帯電話に必要なiアプリ保存容量が減るだけでなく、iアプリのダウンロード時間も少なくなります。JARファイルにパッケージングされた個々のファイルは、必要に応じてJAMによりアンパックされます。

15.6 iアプリのテスト

iアプリのテストは、エミュレータおよび実際のiアプリ対応携帯電話を使って行います。

開発の初期段階では、iアプリのテストやデバッグをエミュレータを使って行う方が効率的であり、コスト的にも有利です。エミュレータでは、ユーザーインタフェースや各種入出力、アプリケーションロジックなどのデバッグ作業において、アプリケーションプログラムを修正した結果をPC上ですぐに確認することができます。

しかしエミュレータ上では、プログラムの実行速度やネットワークの帯域幅などに関して、ハードウェアの動作を完全にエミュレートできるわけではありません。また実際のハードウェアには、メ

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.
モリ容量などの機種固有の制限事項もあります。したがって、実際の i アプリ対応携帯電話で最終的なテストをして、 i アプリの動作が適切かどうかを確認することは不可欠となります。

なおエミュレータの使用方法の詳細については、i アプリ開発環境のユーザーズガイドを参照してください。

エミュレータ環境には、 i アプリのテストに関して、さらにいくつかの制約があります。エミュレータでは、i アプリの実行環境は提供されますが、実際の携帯電話に備わっている i アプリ管理機能（JAM の持つ機能）はエミュレートされません。たとえば、エミュレータで実行できない 1 つの動作に i アプリの更新があります。これは、JAM の i アプリ管理機能がエミュレータに実装されていないためです。テストできない動作はその他に次のものがあります。

- `LastModified`、`TargetDevice`、`LaunchAt` に関して、実際の動作をエミュレートする機能はありません。
- エミュレータでは、`ConfigurationVer` キー（`KvmVer` キー）と CLDC 仮想マシンのバージョン整合性チェックは行われません。
- エミュレータは、開発コンピュータのハードディスク（エミュレータのプロジェクトディレクトリ）にある i アプリ JAR ファイルを読み込んで実行するため、`PackageURL` に指定された値は使用されません。実際に携帯電話に i アプリを配布する際には、`PackageURL` には JAR ファイルをダウンロードするための正しい URL が指定されている必要があります。

実際の携帯電話上でのテストのために i アプリを携帯電話にインストールするには、インターネットに接続された Web サーバーに JAR ファイル、ADF、i アプリダウンロード用 HTML を配置する必要があります。これらの詳細については第 16 章を参照してください。

注意事項：

- DoJa-1.0 プロファイル対応の i アプリ開発環境では、メディアサウンドのデータとして MIDI 形式ファイルを使用する必要があります。このため、DoJa-1.0 プロファイル対応の i アプリ開発環境を使用してメディアサウンドを使用する i アプリを開発する場合、実際の携帯電話を使用したテストを行う際に、MIDI 形式ファイルを MFi 形式ファイルに置き換える必要があります。場合によっては i アプリのソースコード中に指定されたリソースの URL（拡張子）も変更する必要がありますので注意してください。

  なお、DoJa-2.0 プロファイル以降向けの i アプリ開発環境では、メディアサウンドのデータとして MFi 形式ファイルおよび MIDI 形式ファイルを使用することができます。

- i アプリがクライアント/サーバー構成を取っている場合、テストに利用する Web サーバーには、サーバー側アプリケーションも適切に導入されている必要があります。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

第 16 章
i アプリの配布

この章では、i アプリを配布するために Web サーバーをどのように設定する必要があるかについて説明します。

16.1 i アプリ配布のための Web サーバーの構成

前の章で述べたように、i アプリは ADF（.jam ファイル）と JAR ファイル（.jar ファイル）から構成されています。Web サーバーに置かれた ADF をダウンロードするには、i モード HTML の OBJECT タグでこのファイルが参照されていなければなりません。次に、i アプリをダウンロードするための HTML ファイルにおけるタグの記述例を示します。

```
<OBJECT declare id="application.declaration"
data="http://www.nttdocomo.co.jp/java/abc.jam" type="application/x-jam">
<PARAM name=”Param1” value=”i-mode”>
<PARAM name=”Param2” value=”i アプリ”>
</OBJECT>
ダウンロードするには
<A ijam=”#application.declaration” href=”notapplicable.html”>ここ</A>
をクリック。
```

以下に、i アプリのダウンロードのために使用されるタグを解説します。

- A タグおよび ijam 属性

  A タグは、ダウンロード対象の i アプリに対応する OBJECT タグを参照するために使用されます。ユーザーがこの A タグを選択することで、i アプリのダウンロード処理が開始されます。

  A タグの ijam 属性には、OBJECT タグの id 属性に指定された名前を指定します。ijam 属性は i アプリ対応携帯電話向けに新設された属性であり、従来の i モード携帯電話では解釈することができません。A タグに href 属性を併記することで、従来の i モード機種は href 属性を、i アプリ対応機種では ijam 属性を解釈するようになります。また ijam 属性と、ブラウザから i アプリを起動するための ista 属性を同時に有効にすることはできません。ijam 属性と ista 属性を同時に指定した場合、DoJa-2.0 プロファイルに対応した携帯電話では常に ista 属性を解釈し、ijam 属性は無視されます（DoJa-1.0 プロファイル対応の携帯電話では ista 属性を解釈できないため、常に ijam 属性が解釈されます）。

  なお、ijam 属性を持つ A タグでは、必ずペアで href 属性を指定するようにしてください。

  【DoJa-2.0】

  ブラウザからの i アプリ起動および A タグの ista 属性は DoJa-2.0 プロファイルで追加されました。これらの詳細については、第 11 章を参照してください。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

● OBJECT タグ

OBJECT タグは、 i アプリに対応する ADF を参照するために使用されます。OBJECT タグの id 属性には、A タグの ijam 属性で参照される名前（HTML 内で一意）を指定します。また data 属性には ADF の位置を示す URL を指定します。type 属性には data 属性で示されたデータ（この場合 ADF）のコンテンツタイプを指定します。ADF のコンテンツタイプは”application/x-jam”です。

なお、 i アプリの JAR ファイルは、ADF 中に指定された PackageURL キーから参照されます。

OBJECT タグの内部には、下記に示す PARAM タグを記述することができます。

● PARAM タグ

PARAM タグは、 i アプリをダウンロードして初回の実行時(*1)に、IApplication.getParameter ()メソッドにより取得できるパラメータを指定するために使用されます。

PARAM タグは、OBJECT タグ中に最大 16 個含めることができます。また、name 属性の値の長さと value 属性の値の長さの合計は最大 255 バイトに制限されます。name 属性の値および value 属性の値には、それぞれ SJIS で日本語テキストを指定することができます。

(*1) i アプリをダウンロードして初回の実行がアプリケーション連携によるもの（ブラウザ、メーラ、外部機器および i アプリからの起動）である場合、ここで指定された PARAM タグの内容は i アプリに渡されません。初回実行がアプリケーション連携によるものである場合、起動元となるアプリケーションから渡されるパラメータの方が有効となり起動先の i アプリに渡されます。アプリケーション連携の詳細については第 11 章を参照してください。

【DoJa-2.0】

PARAM タグによるパラメータ指定および IApplication.getParameter()メソッドは DoJa-2.0 プロファイルにて追加されました。

i アプリ対応携帯電話は、OBJECT タグの type 属性で ADF ファイルのコンテンツタイプを確認します。したがって Web サーバー側では、ADF の拡張子に対応するコンテンツタイプのコンフィグレーションを行う必要はありません。

【DoJa-2.0】

DoJa-2.0 プロファイルでは、ADF の拡張子に制限はありません。これに対し、DoJa-1.0 プロファイルでは、ADF の拡張子は".jam"である必要があります（ADF の URL の実体が CGI などの場合は、その CGI の URL 末尾を".jam"とします）。

【DoJa-5.0】

DoJa-5.0 プロファイル以降では、3.2 項で解説した規定により 100K バイトを超える JAR ファイルをダウンロードすることができます。ただし携帯電話に 100K バイトを超える JAR ファイルをダウンロードできるようにするためには、 i アプリの配布に使用する Web サーバーがレンジリクエストによるバイト単位の部分的 GET に対応していなければなりません。レンジリクエストに対して部分的 GET の応答（206 Partial content）を返さない Web サーバーからは、100K バイトを超える JAR ファイルをダウンロードすることはできないため注意してください。

特に、JAR ファイルのダウンロードにおいて CGI を利用しているような場合には注意が必要です。

また DoJa-5.0 プロファイル以降、トルカのバージョン 2.0 フォーマットに対応している携帯電話では、トルカ（詳細）の HTML コンテンツ部分に本項で解説したタグを記述することでトルカビューア経由で i アプリをダウンロードすることもできます。なお、この場合 OBJECT タグ data 属性に記載する ADF URL には相対 URL は指定しないようにしてください。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

16.2 サーバー側アプリケーション実行のための Web サーバーの構成

クライアント/サーバー型の i アプリを動作させる場合、サーバー側アプリケーションに適用する技術によっては Web サーバーに特定のコンフィグレーションを行う必要がある場合があります。例えばサーバー側アプリケーションに Java サーブレットを使用する場合、Web サーバー側では Java サーブレットエンジンのインストールとコンフィグレーションを行う必要があります。

使用している Web サーバーのドキュメントおよびサーバー側アプリケーションに適用している技術に関するドキュメントを参照し、Web サーバーに適切なコンフィグレーションを行うようにしてください。

16.3 特定メーカーの i アプリ対応携帯電話へのダウンロード

i アプリオプション機能または i アプリ拡張機能を使用した i アプリや厳密な画面サイズを前提とした i アプリは、特定メーカーの携帯電話でのみ開発者の意図通りに動作します。このような i アプリは、そのターゲットとする機種だけに適切にインストールされることが望まれます。

i アプリのダウンロードを各機種毎に適切にコントロールするために、i アプリ対応携帯電話では次のような機構が利用可能になっています。

1. 携帯電話からの HTTP リクエストには、携帯電話の機種を判別するための `User-Agent` リクエストヘッダが含まれています。Web サーバー側では i アプリのダウンロード用 HTML を表示する際に、`User-Agent` の内容に基づいてユーザーに適切な i アプリを提示するようナビゲートすることができます。
2. 携帯電話側では、ADF の `TargetDevice` キーを参照して、ダウンロードする i アプリがその機種に対応しているかどうかを判定します。特定機種向けの i アプリでは、ADF の `TargetDevice` キーを適切に設定することで、開発者の意図しない機種へのダウンロードを防止することができます。

16.4 ダウンロード即起動 i アプリ

DoJa-3.0 プロファイル以降では、i アプリのダウンロードから起動までをユーザーの 1 クリック操作で行うためのダウンロード即起動機能がサポートされます。ダウンロード即起動 i アプリは一切のユーザー確認確認なしにダウンロードから起動までが行われるかわりに、本来ユーザー確認が必要とされている機能の利用には制限があります。

ダウンロード即起動 i アプリでは、以下の機能に関する API を呼び出すことはできません。これらの API を呼び出すと例外が発生します。

- ADF の `GetSysInfo` キーにより許可される機能（アイコン情報の参照）
- ADF の `GetUTN` キーにより許可される機能（個体識別情報の参照）
- ADF の `AccessUserInfo` キーにより許可される機能（ユーザーデータ領域へのアクセス）
- ADF の `LaunchApp` キーにより許可される機能（ i アプリ連携起動）
- ADF の `UseStorage` キーにより許可される機能（外部メモリ）
- ScratchPad の利用
- ｉアプリからの i アプリ更新機能（JAM）の連携起動

i アプリをダウンロード即起動 i アプリとして配布するには、 i アプリダウンロード用リンク（`A` タグ）の `ijam` 属性のかわりに、`ilet` 属性を指定します。`ijam` 属性と同様に、`ilet` 属性も `href` 属性とセットで使用する必要があります。

以下に、 i アプリをダウンロード即起動 i アプリとして配布するための HTML ファイルの記述例を示します。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

```
<OBJECT declare id="application.declaration"
data="http://www.nttdocomo.co.jp/java/abc.jam" type="application/x-jam">
<PARAM name=”Param1” value=”i-mode”>
<PARAM name=”Param2” value=”iアプリ”>
</OBJECT>
<A ilet=”#application.declaration” href=”notapplicable.html”>ここ</A>
```

をクリックすると、iアプリのダウンロードと起動が一度に行えます。

ダウンロード即起動iアプリにおいても、PARAMタグを使用したパラメータの授受を行うことができます。

【DoJa-5.0】

DoJa-5.0プロファイルでは、iアプリのサイズがJARファイルサイズとScratchPadサイズの合計で1024Kバイトまでに拡張されています。ただしダウンロード即起動iアプリについては、メーカーによっては従来のプロファイルと同様にJARファイルサイズが100Kバイトまで、ScratchPadサイズが400Kバイトまでに制限される場合があります。

注意事項：

- ダウンロード即起動モードでの起動は、アプリケーション連携における、ブラウザからのiアプリ起動と同等の位置付けとなります。このためダウンロード即起動iアプリでは、iアプリダウンロード用HTMLからそのiアプリを連携起動できるよう、ADFの`LaunchByBrowser`キーの設定が適切に行なわれていなければなりません。ADFの`LaunchByBrowser`キーについては、15.5.1項を参照してください。
- ダウンロード即起動iアプリでは、本来iアプリダウンロード後にユーザー確認または禁止設定を行うことのできる機能の利用に制限がありますが、HTTP(S)通信は例外的に使用することができます。ダウンロード即起動iアプリにおいてはHTTP(S)通信の許可確認は、ダウンロード時ではなくダウンロード即起動iアプリから初めてHTTP(S)通信を行おうとした際に行われます。
  許可確認が行われる際、ダウンロード即起動iアプリの実行は中断されます。
- ダウンロード即起動モードで起動されたiアプリは、待ち受けアプリケーションであっても待ち受け起動とはならず通常起動となります。
- ダウンロード即起動モードでの実行が正常に終了すると、携帯電話はユーザーにそのiアプリを保存するかどうかを確認します。保存が選択された場合、ダウンロード即起動iアプリは通常のiアプリとして携帯電話に保存され､その後ADFの記述内容に従ってHTTP(S)通信の許可設定などのユーザー確認が行われます(その時点で、本項に示した機能の利用制限は解除されます）。
  開発者は、ADFの`IletPreserve`キーを使用して、ダウンロード即起動iアプリが携帯電話に保存されないよう設定することができます。ADFの`IletPreserve`キーについては、15.5.1項を参照してください。
  なお､ダウンロード即起動モードで起動されたiアプリがブラウザを連携起動することにより終了する場合、iアプリ保存のためのプロセスは行われず、そのiアプリは保存されません。
- ダウンロード即起動モードで起動されたiアプリでは、`IApplication.getLaunchType()`メソッドで取得される起動種別は`IApplication.LAUNCHED_AS_ILET`となります。また、ダウンロード即起動iアプリ終了後にユーザーがそのiアプリを携帯電話に保存した場合、その後の初回起動が通常起動（iアプリ一覧などのユーザー操作による起動）であれば、起動種別は`IApplication.LAUNCHED_AFTER_DOWNLOAD`となります。
- トラステッドiアプリをダウンロード即起動iアプリとしてダウンロードすることはできません。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

付録 A
DoJa-5.x と DoJa-5.x LE

DoJa-5.x プロファイル世代の FOMA 携帯電話では、90x シリーズ（標準モデル）のラインナップとは別にエントリー向けモデルのシリーズラインナップが用意されます。エントリー向けモデルの FOMA 携帯電話では、DoJa-5.x プロファイルから一部機能を省略した DoJa-5.x LE (Limited Edition) プロファイル対応の i アプリ実行環境が搭載されます。

DoJa-5.x LE プロファイルでは、PDC 携帯電話向け相当のコンテンツを今後のエントリー向け FOMA 携帯電話上でも利用可能とすることを主眼としており、DoJa-4.x プロファイル、および DoJa-5.x プロファイルで追加された機能の一部等に制限があります。

DoJa-5.x プロファイルと DoJa-5.x LE プロファイルの区別も含め、各機種に搭載されている DoJa プロファイルについては別途 NTT ドコモよりアナウンスされます。

本項では、DoJa-5.x プロファイルと DoJa-5.x LE プロファイルの機能差分について解説します。

3D グラフィックス描画機能

i アプリ基本 API の 3D グラフィックス描画機能（`com.nttdocomo.ui.graphics3d` パッケージおよびそのサブパッケージ）は、DoJa-5.x LE プロファイル対応携帯電話ではオプショナルであり、搭載されない機種が存在します。なお、従来より規定されている i アプリ拡張 API の高レベル 3D グラフィックス描画機能（`com.nttdocomo.opt.ui.j3d` パッケージ）は、DoJa-5.x LE プロファイルでも規定されています。

3D サウンド制御機能

3D サウンド制御機能は DoJa-5.x LE プロファイル対応携帯電話ではオプショナルであり、搭載されない機種が存在します。

JAR ファイルサイズ、ScratchPad サイズの制限

DoJa-5.x プロファイルの FOMA 携帯電話では、 i アプリのサイズは JAR ファイルサイズと ScratchPad サイズの合計で 1024K バイトまでと規定されています。これに対し DoJa-5.x LE プロファイルでは、プロファイル内のどの機種でも使用可能な i アプリのサイズは JAR ファイルが 30K バイトまで、ScratchPad サイズが 200K バイトまでに制限されています（機種によっては非 LE プロファイルの規定サイズの範囲内で、より大きなサイズをサポートするものもあります）。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

付録 B
2in1 機能が i アプリに与える影響

904i シリーズ以降の携帯電話では、1 台の携帯電話に 2 つの電話番号、2 つの i モードメールアドレスを割り当てて、ユーザーの利用シーンによって自由に使い分けを行うことができる、2in1 機能と呼ばれる機能を利用できるものがあります(2in1 機能はハイパーマルチナンバー機能、もしくはハイパーマルチ機能と称される場合もあります）。

2in1 機能を利用するためには、ユーザーは NTT ドコモと、通常の回線契約に加え 2in1 機能を利用するための契約を行う必要があります。この契約が行われると、ユーザーは通常の回線契約での電話番号や i モードメールアドレス（A ナンバー/A アドレス）の他にもう一組の電話番号、 i モードメールアドレス（B ナンバー/B アドレス）を利用できるようになります。

2in1 機能は携帯電話の通話・メール機能の動作や、電話帳・発着信履歴などのユーザーデータ管理に強く関わる機能であり、2in1 機能が有効となるよう設定されている携帯電話では一部の i アプリ API に影響を及ぼす場合もあります。本付録では、2in1 機能が i アプリに与える影響について解説します。

なお、携帯電話上で 2in1 機能が有効となるよう設定された状態を活性化状態、無効となるよう設定された状態を非活性化状態と呼びます。また活性化状態には、ユーザーデータの閲覧を制御するためのモードの概念が導入されています。活性化状態中のモードには、以下の 3 つがあります。

A モード	：	A ナンバー/A アドレスに関連するユーザーデータを閲覧できるモード
B モード	：	B ナンバー/B アドレスに関連するユーザーデータを閲覧できるモード
デュアルモード	：	A、B のどちらに関連するユーザーデータも閲覧できるモード（共通モードと呼ばれることもあります）

2in1 活性化 A モードでは、B モードでの発着信履歴や B アドレス宛てのメール、B モード用として属性をつけて保存された電話帳データの閲覧はできません。また 2in1 活性化 B モードでは、A モードでの発着信履歴や A アドレス宛てのメール、A モード用として属性をつけて保存された電話帳データの閲覧はできません。

なお現在の 2in1 サービスでは、B アドレス宛てのメールは通常の i モードメールの形で携帯電話に送られてくるのではなく、Web メール形式で閲覧する形となります。B アドレスにメールが届いたことは、B アドレス宛てとして携帯電話に送付される着信通知メール（ i モードメール）にてユーザーに通知されます。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

また 2in1 活性化 B モードでは、携帯電話から i モードメールおよび SMS の送信は行えません。2in1 活性化 B モードでは、メール送信については閲覧時と同様に Web メール形式で行います。

＜2in1 機能が i アプリに与える影響＞

- 待ち受けアプリケーションについて

  2in1 非活性化時、および 2in1 活性化 A モードでは待ち受けアプリケーションの待ち受け起動がサポートされますが、2in1 活性化 B モードおよび 2in1 活性化デュアルモードでは、待ち受けアプリケーションを待ち受け起動することはできません。

  なお、待ち受けアプリケーションが待ち受け起動されている状態で 2in1 機能の状態やモードの変更を行うと、その待ち受けアプリケーションは一旦終了します（変更後の状態やモードが待ち受け起動をサポートするものだった場合には、待ち受け復帰後に再度待ち受け起動されます）。

- 状態やモードによるユーザデータの参照制限について

  マイプロフィール管理機能の呼び出し（`OwnerProfile` クラス：i アプリオプション API）で参照できるデータは、i アプリが動作している際の状態やモードに応じて変わります。2in1 非活性化状態、2in1 活性化デュアルモード、2in1 活性化 A モードでは A ナンバー/A アドレスに関連付くマイプロフィール情報が参照され、2in1 活性化 B モードでは B ナンバー/B アドレスに関連付くマイプロフィール情報が参照されます。

- 状態やモードによるユーザデータの登録制限について

  i アプリから `PhoneBook.addEntry()` メソッドを使用して電話帳エントリを登録する場合、2in1 非活性化状態、2in1 活性化デュアルモード、2in1 活性化 A モードでは、その電話帳エントリは A モード用の（B モードでは電話帳のデータとして見ることができない）電話帳データとして保存されます。また、2in1 活性化 B モードでは、その電話帳エントリは B モード用の電話帳データとして保存されます。

  なお、上記のように保存された電話帳エントリも、携帯電話ネイティブの設定機能をユーザーが操作することで、対応するモードを変更したり、どのようなモードでも利用可能とするよう設定したりすることができます。

- `IApplication.getSuspendInfo()` メソッドによるサスペンド理由の取得について

  `IApplication.getSuspendInfo()` メソッドを使用することで、直前に発生したサスペンドの理由や、そのサスペンド期間中に発生した事象の情報を取得することができます。サスペンド理由や事象には「通話着信」「メール着信」といった着信系のイベントも含まれますが、2in1 活性化 A モードでは B ナンバー/B アドレスの着信イベント、また 2in1 活性化 B モードでは A ナンバー/A アドレスの着信イベントについて、このメソッドでサスペンド理由などを取得することはできません。

Copyright © 2008 NTT DOCOMO, Inc. All Rights Reserved.

用語集

API	アプリケーションプログラミングインタフェース
Applet	アプレット。Web ページとともにユーザーに送信される小さなプログラム。 i アプリはアプレットではありません。
bps	bits per second。秒当たりのビット数。
CGI	Common Gateway Interface
CHTML	コンパクト HTML
CLDC	Connected Limited Device Configuration
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
IDE	Integrated Development Environment。統合開発環境。
J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition
J2SE	Java 2 Standard Edition
JAM	Java Application Manager
JAR	Java アーカイブ（Java Archive）の略
JDK	Java Development Kit
JNI	Java Native Interface
JVMS	Java Virtual Machine Specification
MIME	Multi-Purpose Internet Mail Extensions。インターネット e メールプロトコルを拡張し、このプロトコルを使って多様なデータファイルをインターネットで送信できるようにしたもの。
Servlet	サーブレット。Web コンテンツを動的に生成するモジュール形式のサーバー側 Java アプリケーションを開発するためのメカニズム。
SSL	Secure Sockets Layer
UI	ユーザーインタフェース
Unicode	文字を表すための 16 ビットの符号化
URL	Uniform Resource Locator。アクセス可能なインターネット上のファイル（資源）のアドレス。