Technology Reports

2008年秋冬モデル搭載アプリケーション機能（2）次世代JAVAアプリ

iアプリのサービス開始から8年が経過した．数多くのサービスやソリューションを生み出したiアプリの，アプリケーションプラットフォームとしての基盤をさらに強化すべく，新たに「Star」と呼ぶJava®*1プラットフォームを開発した．

移動機開発部　大関 江利子　浅井 真生
戸部 章子　土屋 二朗†

1. まえがき

iアプリは，2001年に発売されたmova 503iシリーズにおいて，DoJa（DoCoMo Java）*2と呼ばれるプロファイル*3を搭載し，サービスが開始された．その後も，DoJaプロファイルを進化させ，ゲーム，おサイフケータイ，法人ソリューションなど，さまざまな機能やサービスをiアプリ上で実現してきた．

しかしながら，移動端末の性能向上やサービスの多様化により，DoJaに対する要望も高機能化／多様化し，DoJaの基本設計を根幹から見直すような機能要望や，機能構成の再整理が必要となった．

そこで，アプリケーションプラットフォームとしての基盤をさらに強化すべく，既存コンテンツとの互換性を考慮のうえ，新たに「Star」プラットフォームを開発した．具体的には，アプリケーションのライフサイクルとプログラミングモデルの見直しを行い，iアプリの基底クラス*4を新規に開発した．また，これまでのiアプリの機能に加え，WidgetView（iウィジェット*5），JavaFlash®*6連携，ライブラリ追加機能，マイメニュー登録機能，iアプリオンライン機能，iアプリコール機能を開発した．

本稿では，これらの主要な機能について解説する．なお，iアプリオンライン，iアプリコールについては，[1]を参照されたい．

2. Starプラットフォームの概要

2.1 システム構成

Starプラットフォームのモジュール構成を**図1**に示す．

Starプラットフォームは，フルアプリ実行環境，ミニアプリ実行環境そして両実行環境で動作するiアプリの起動制御などを行うJAM（Java Application Manager）から構成される．2種類の実行環境上では，それぞれ異なる種類のアプリケーションが動作する．

2.2 実行環境の特長

フルアプリ実行環境は，従来iアプリが提供してきたサービスを継承し，かつ今後の機能拡張に長期間耐え得る実行環境として設計された．Javaの標準クラスライブラリであるCLDC（Connected Limited Device Configuration）*7や，DoJaの機能を包含しつつ新規のサービスを実現するためのStarプロファイルから構成される．Starプロファイルは DoJaプロファイルのサービスを継承すべく設計したが，API（Application Program Interface）*8のリファクタリング*9を施しており，バイナリ*10でのDoJaとのAPI互換性はな

† 現在，三菱電機株式会社 インフォメーションシステム事業推進本部

＊1 **Java®**：米国Sun Microsystems, Inc.の登録商標．
＊2 **DoJa**：iアプリが利用する機能群をJavaプログラムの部品群としてまとめたもの．
＊3 **プロファイル**：API（＊8参照）やクラスライブラリ群のこと．
＊4 **基底クラス**：クラス定義を継承する際に，その基になるクラス．クラスとは，オブジェクト指向プログラミングにおける，データやメソッドをまとめた1つの型．
＊5 **iウィジェット**：「iウィジェット＼アイウィジェット」は㈱NTTドコモの登録商標．

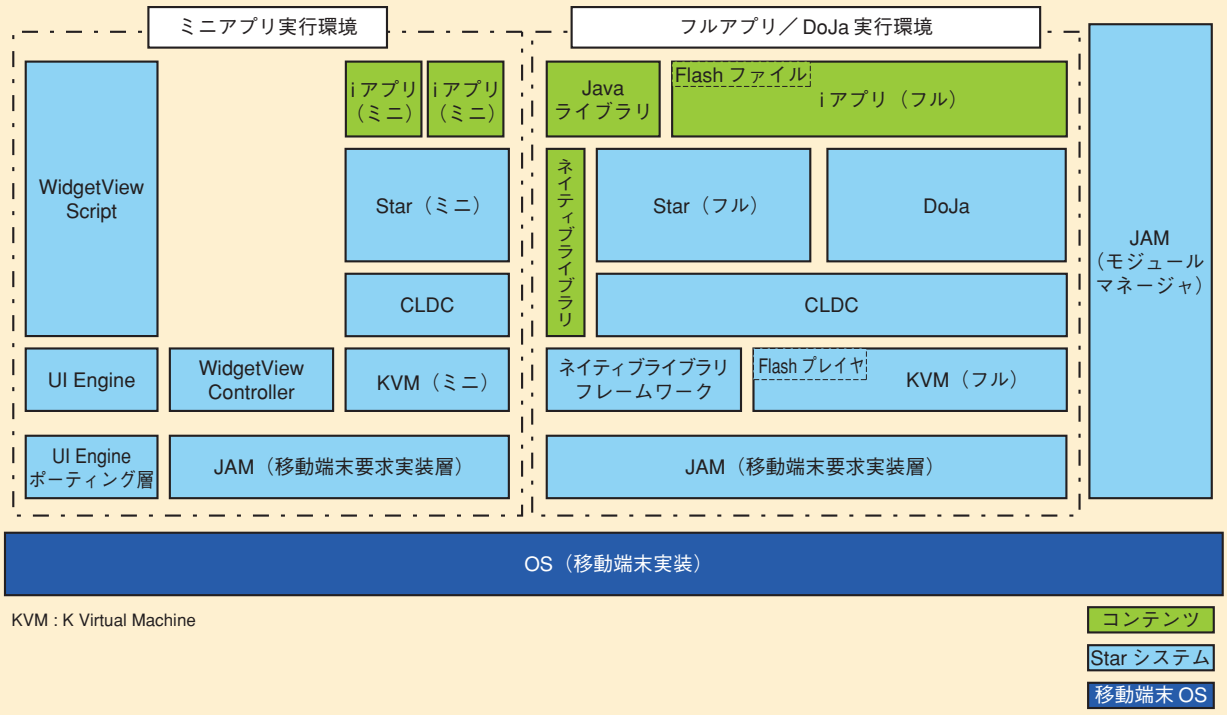

図1　Starプラットフォームモジュール構成

い．

一方，ミニアプリ実行環境は，複数アプリケーションの同時実行を実現するための実行環境であり，WidgetViewと呼ばれる．フルアプリ実行環境との違いは，利用できる機能が限定されている点，VM（Virtual Machine）[*11]が複数アプリケーションを同時に処理することに対応している点である．

2つの実行環境は独立したシステムとして構成し，アプリケーションのメソッド[*12]呼出しによって2つの実行環境を切り替えることを可能とした．これにより，2つの実行環境のそれぞれの特長を活かしたアプリケーションの作成と連携動作の実現を図った．

2.3 アプリケーションライフサイクルモデルの見直し

Starプロファイルの共通アプリケーション基底クラスである「StarApplicationクラス」の実行状態遷移図とDoJaプロファイルにおけるアプリケーション基底クラス「IApplicationクラス」の実行状態遷移図を**図2**に，StarApplicationの各状態の説明を**表1**に示す．図2の矢印に付記されたメソッド名のうち，/で始まるものは，その状態遷移によってコールバックメソッドの呼出しが生じることを表し，/で始まらないメソッドはそのメソッドをiアプリから呼び出すことで状態遷移が発生することを表す．

IApplicationクラスと比較した場合のStarApplicationの特長を3点挙げる．

第1点は，アプリケーション実行時の副状態として「Semi-Active」や「Full-Active」を用意した点である．StarApplicationクラスでは複数アプリの並列実行処理に耐える必要があるため，資源を協調的に利用する「Semi-Active」状態と，資源を優先確保可能な「Full-Active」状態の2段

＊6 **Flash**®：Adobe社が開発した，音声やベクターグラフィックスのアニメーションを組み合わせて作成されたコンテンツ．Flash®は，Adobe Systems Inc.の米国およびその他の国における商標または登録商標．
＊7 **CLDC**：移動端末やPDAなどの小型端末を対象として定義されたJavaのコンフィグレーション．
＊8 **API**：あるプラットフォーム（OSやミドルウェア）向けのソフトウェアを開発する際に使用できる命令や関数を利用するためのプログラム上の手続きを定めた規約の集合．
＊9 **リファクタリング**：冗長性の改善などを目的として，プログラムの振舞いを変えることなくソースコードを変更すること．
＊10 **バイナリ**：実行可能形式のコンピュータプログラムのこと．
＊11 **VM**：Javaプログラムを，異なるプラットフォーム間でも実行可能とするためのソフトウェア．
＊12 **メソッド**：オブジェクト指向プログラミングにおいて，オブジェクトがもっているデータに対する操作．

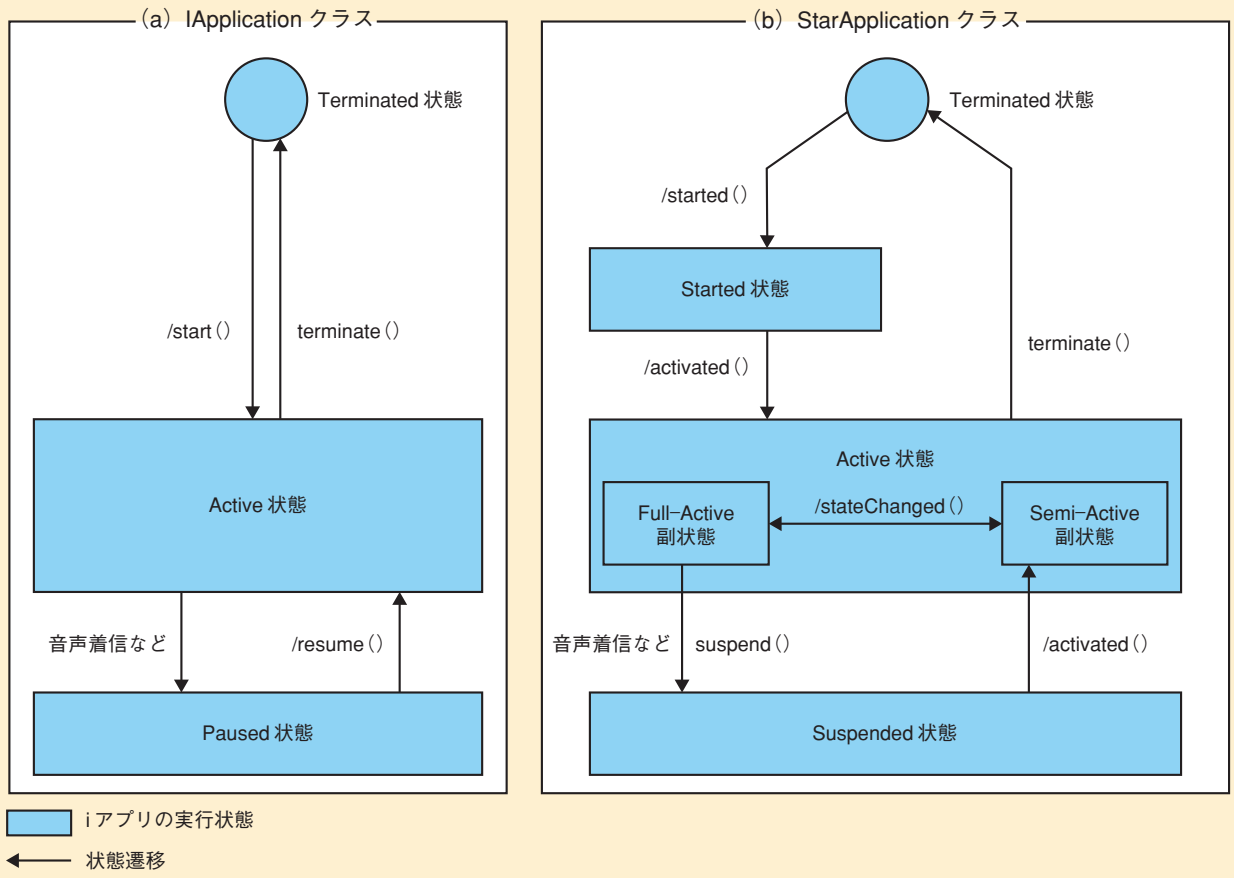

図2　各アプリケーション基底クラスの状態遷移

表1　StarApplication 状態遷移図における各状態の説明

Started状態		・起動直後の状態 ・本状態遷移時にStarApplication.started () がコールバックされる ・完了後，自動的にActive状態に遷移する
Active状態		・実行中状態 ・本状態遷移時にStarApplication.activated()がコールバックされる
	Full-Active	・Active状態の副状態 ・移動端末資源を優先的に利用可能な状態
	Semi-Active	・Active状態の副状態 ・多重実行などの要因で資源を占有できない状態
Suspended状態		・休止状態 ・電話の着信やアプリケーションの休止メソッドにより遷移する
Terminated状態		・終了状態

階を用意し，アプリケーションが自身の実行状態を取得可能とした．

第2点は，プログラム開始直後の状態として「Started」状態を用意した点である．これにより，プログラムの実行時に1度だけ行えばよい初期化処理を分離したプログラミングが可能となった．

第3点はアプリケーションが自ら「Suspended」状態への遷移要求を発行可能とした点である．ユーザ入力待機状態などで積極的に「Suspended」状態に遷移することで，アプリケーション開発者が省電力を意識したプログラミングを可能とした．

3. WidgetView機能

(1)機能概要

複数のミニアプリを同時に実行可

能とするためにWidgetViewと呼ばれるミニアプリ実行環境を開発した．このミニアプリは，フルアプリのサブセットとなっており，プログラムサイズ，利用可能なメモリサイズ，利用可能な機能が，フルアプリに比べて限定されている（**表2**）．WidgetViewは待受画面から専用キーを押下することで実行される．このため，ユーザは，起動したいミニアプリを1つひとつ起動するのではなく，複数のミニアプリ（例えば，ニュースアプリ，天気予報アプリ，株価アプリ，地図アプリなどの生活系ミニアプリ）をワンプッシュで起動できる．また，ミニアプリはWidgetView上でそれぞれ描画領域をもつため，ユーザは複数のミニアプリが提供する情報を同時に得ることができる．

(2)システム構成

WidgetViewのシステム構成を**図3**に示す．WidgetViewは，複数のミニアプリを同時に処理するためのMiniVM，背景画像やミニアプリが描画したフレームなどの各素材に対

表2　フルアプリとミニアプリの違い

	特長	対応機能
フルアプリ	・アプリサイズとデータストレージサイズが合計最大2MB ・同時に実行可能なアプリは1つ ・すべてのStar APIを利用可能	・モバイルFeliCa®※対応機能 ・位置情報取得機能（GPS機能） ・HTTP(S)通信機能 ・Push起動機能 ・TCP/UDP通信機能 ・マイメニュー登録／削除機能 ・ライブラリダウンロード機能 ・Java-Flash連携機能 ・ミニアプリへの連携起動　など
ミニアプリ	・アプリサイズは最大50kB，データストレージサイズは最大200kB ・1画面上に最大8個のアプリが同時に実行可能 ・Star APIを部分的に利用可能 ・ミニアプリ実行時に使用可能なメモリサイズは，フルアプリ実行時に使用可能なメモリサイズの約1/10	・モバイルFeliCa 対応機能 ・位置情報取得機能（GPS機能） ・HTTP(S)通信機能 ・フルアプリへの連携起動　など

※FeliCa®：ソニー㈱の登録商標．

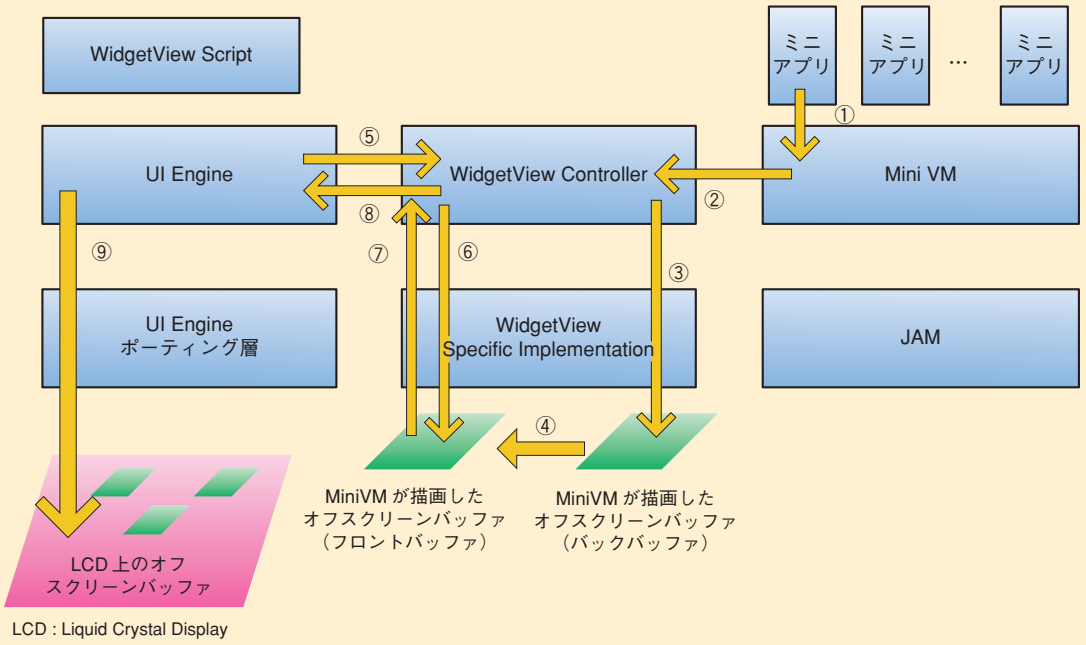

図3　WidgetViewのシステム構成

して，エフェクトなどをかけて1つの画面を作成するためのUI（User Interface） Engine，MiniVMとUI Engineをつなぐためのの3つから構成される．各ミニアプリは，オフスクリーンバッファ[*13]に描画を行い（図3①～③），フロントバッファ[*14]へフリップを行う（図3④）．UI Engineは，ミニアプリのオフスクリーンバッファを取得して（図3⑤～⑧），背景画像などと組み合わせて1つの画面を構成して表示する（図3⑨）．

(3)WidgetViewの表示状態

WidgetViewには3つの表示状態がある（**図4**）．

一覧表示状態では，複数のミニアプリが同時に実行されて1つの画面に縮小されて表示される．ただし，この状態において，ユーザはミニアプリを操作することはできない．待受画面から専用キーを押下した際，WidgetViewはこの状態で表示される．

個別表示状態では，1つのミニアプリのみが実行され（他の実行中のミニアプリはサスペンドする），そのミニアプリのみが画面に表示されている．この状態において，ユーザはミニアプリを操作することが可能である．

ランチャー[*15]表示状態では，ミニアプリのアイコン一覧を表示し，アイコンを選択することによって，ミニアプリを起動することが可能である．この状態においては，起動中のミニアプリはすべてサスペンドしている．

(4)ミニアプリについて

ミニアプリでは表示状態によって利用できる機能が限定される．具体的には，WidgetViewの一覧表示状態においては，個別表示状態よりも使用可能な機能が限定されている．例えば，ブラウザ連携起動や音声発信機能などは，個別表示状態でのみ使用可能な機能である．

4. Java-Flash連携機能

(1)機能概要

iアプリからFlashファイル[*16]を利用できる仕組みを開発した．具体的には，iアプリ上にてFlashファイルを再生することができ，また，Flashファイルからのコマンドをiアプリが処理できる仕組みを開発した．こ

図4　WidgetViewの表示状態

＊13 **オフスクリーンバッファ**：ディスプレイ表示に使われていないメモリ領域で，複数の機器やソフトウェアの間でデータをやり取りするときに，処理速度や転送速度の差を補うためにデータを一時的に保存しておく記憶装置や記憶領域．

＊14 **フロントバッファ**：ディスプレイに表示するためのデータを保持しているメモリ領域のことで，プログラムによって書き込むためのバックバッファへの書込みが完了すると，フロントバッファとバックバッファを切り替え，書き込んだデータを表示する．

＊15 **ランチャー**：あらかじめ登録しておいたファイルやプログラムを一覧表示し，簡単に起動できるようにすること．

＊16 **Flashファイル**：Macromedia社（現Adobe社）が開発した，音声やベクターグラフィックスのアニメーションを組み合わせて作成されたコンテンツ（ファイルの拡張子は.swf）．

れにより，例えば，Flashゲームのホストアプリとしてiアプリを利用したり，iアプリのユーザインタフェースとしてFlashファイルを利用することが可能となった．

なお，本モデルは，インライン再生[*17]には対応しておらず，Flashファイルはiアプリの描画領域全体で再生される．

⑵システム構成

システム構成を**図5**に示す．Flashファイルを実行するためのFlashプレイヤと，iアプリを実行するためのJava VMから構成される．Java VMはFlashプレイヤのホストアプリケーション[*18]としても動作する．

⑶Flashファイル利用方法

iアプリからFlashファイルを利用する際のAPIを新設した．このAPIを通じて，Flashプレイヤを制御したり，Flashプレイヤからのコマンドをiアプリが受け取り，処理を行う．

iアプリとFlashプレイヤ間のシーケンスを**図6**に示す．iアプリからのメソッド呼出しによりFlashプレイヤの生成を行い（図6①），生成したFlashプレイヤにFlashファイルを設定し（図6②），メソッド呼出しにより再生を開始する（図6③）．Flashファイルからは，ActionScriptを利用してiアプリに処理要求を送信することができる（図6④）．iアプリはFlashファイルからの要求をイベント通知として受け取り（図6⑤），iアプリで処理を行い，Flashファイルに処理結果を返す（図6⑥）．iアプリは，メソッド呼出しにより，Flashファイルの再生を停止させることができる（図6⑦）．

また，ユーザ操作に応じてFlash

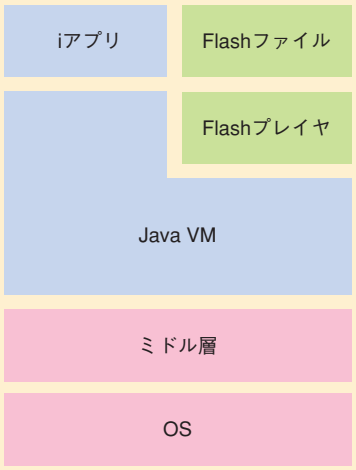

図5　システム構成

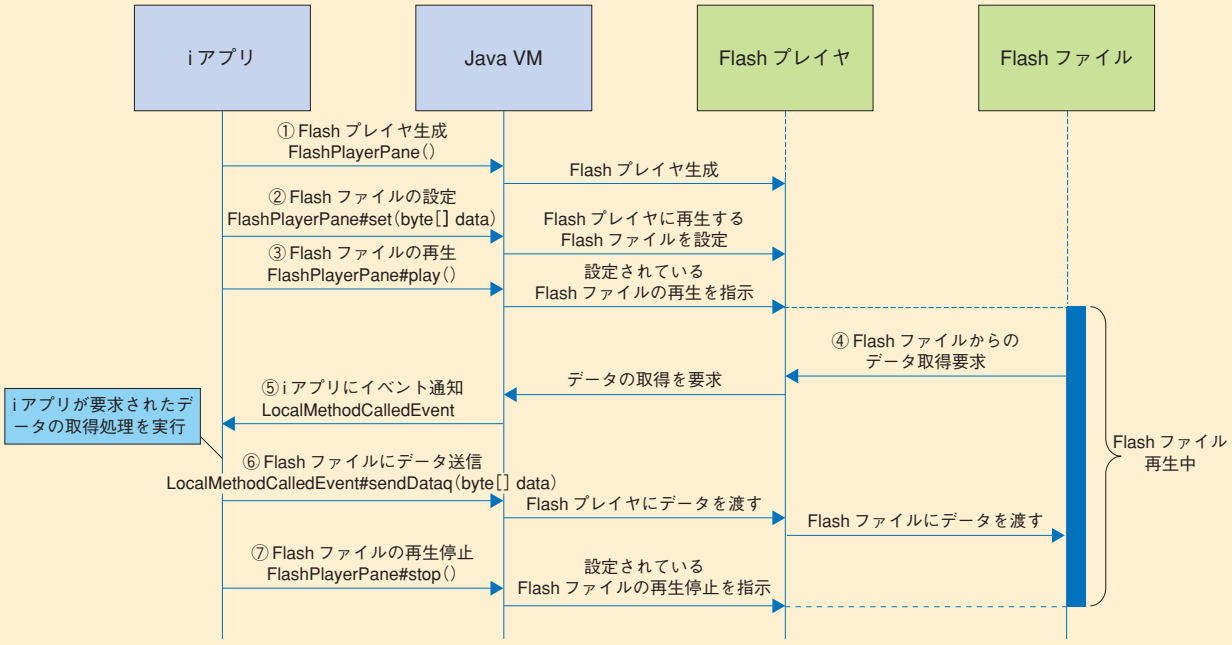

図6　iアプリからの利用方法

＊17 **インライン再生**：再生方法の1つで，他のアプリケーションの描画領域内（HTML内など）の一部にはめ込んで再生をする方法．

＊18 **ホストアプリケーション**：他のアプリケーションに対して，サービスや処理を提供するアプリケーション．

ファイルを再生するインタラクティブ再生に対応しており，キーイベントはFlashファイルに通知される．ただし，ファンクションキーのイベントに関しては，iアプリに通知され，ユーザ操作によってiアプリがFlashファイルの再生状態などを制御できる．

5. ライブラリ追加機能

(1)機能概要

ライブラリ形式のネイティブ機能とそのJava APIを，移動端末にダウンロードし，実行する仕組みを開発した．ライブラリ追加機能は大きく分けて，iアプリから参照するクラスファイルとネイティブ機能[*19]を，iアプリとは別のファイル（ライブラリ）として移動端末にダウンロードする仕組みと，iアプリからライブラリをロードして実行する実行環境で構成される．

ライブラリには，Java言語で記述されるもの（Javaライブラリ）と，C/C++などのJava以外の言語で記述されるもの（ネイティブライブラリ）がある．ライブラリの利用形態としては，Javaライブラリのみを利用する場合と，Javaライブラリからネイティブライブラリを利用する場合の2種がある．なお，ネイティブライブラリを利用するケースはiアプリDXからのみである．

Javaライブラリを利用することにより，表示と処理の実装をiアプリとライブラリに分離して開発することが可能となる．ネイティブライブラリを利用することにより，次のことが可能となる．

- 標準搭載のJavaクラスライブラリから利用できないネイティブ機能の追加（例えば，コード認識機能の新しいコード種別を認識エンジンと一緒に追加する）
- パフォーマンスが重視されるコードをネイティブ言語で実装
- 既存のネイティブ言語で記述されたライブラリをiアプリ向けへ再利用

(2)システム構成

システム構成を**図7**に示す．iアプリとライブラリをダウンロードするためのJAM，iアプリやJavaライブラリを実行するためのJava VM，Javaライブラリからネイティブライブラリをロードして実行するためのネイティブライブラリフレームワークから構成される．

(3)ダウンロード方法

従来のiアプリのダウンロードの仕組みを踏襲し，iアプリと対になるライブラリを順にダウンロードする．具体的には，JAMは取得したiアプリのADF（Application Description File）[*20]/SDF（Security Description File）[*21]の記載に従いライブラリのADF/SDFを取得し，次にライブラリのADF/SDFの記載に従いライブラリを取得し，最後にiアプリを取得する．

(4)ライブラリ利用方法

iアプリはCLDCの標準APIを通じて任意のタイミングでJavaライブラリをロードして，利用することができる．Javaライブラリは新規に用意したネイティブライブラリをロードするためのAPIを通じて，ネイティブライブラリをロードすることができる．Javaライブラリに含まれるクラスで定義されたネイティブメソッドを呼び出すと，ネイティブライブラリフレームワークを介して，そのネイティブメソッドに対応するネイティブ関数を実行できる（**図8**）．ネイティブライブラリフレームワークには，ネイティブライブラリが利用するヒープメモリの確保・解放機

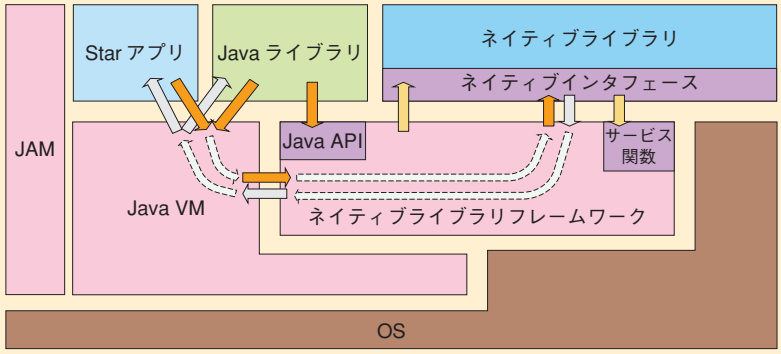

図7　システム構成図

＊19 **ネイティブ機能**：移動端末に搭載されている，iアプリ以外の機能の総称．

＊20 **ADF**：Starアプリ，DoJaアプリの定義，情報を記載するファイル．

＊21 **SDF**：トラステッドアプリで利用される，セキュリティ関連の定義，情報を記載するファイル．トラステッドアプリとは，特定機能の使用を許可されたアプリのこと．

能，iアプリがサスペンドや終了するときに，ネイティブライブラリの中断処理を行うコールバック関数を呼び出す機能，ネイティブライブラリのデバッグのためのログ取得機能など，ネイティブライブラリの実行に必要な機能を実装した．ネイティブライブラリからこれらの機能を利用することにより，着信など移動端末特有の中断契機でネイティブライブラリの処理を終了し，速やかに通話に移行することが可能となる．

6. マイメニュー登録機能

(1)機能概要

iアプリからマイメニュー登録ができるようにCiRCUS（treasure Casket of i-mode service, high Reliability platform for CUStomer）[*22]とのインタフェースを規定し，マイメニュー登録に必要なパラメータをiアプリから送信できる機能を開発した．従来は，i-modeブラウザからのみマイメニュー登録が可能であったため，iアプリからブラウザを連携起動してマイメニュー登録する必要があった．本機能を導入することにより，iアプリに閉じたマイメニュー登録が可能となるため，例えば，お試しのiアプリゲームから直接マイメニュー登録が可能となり，コンテンツ購入の促進が期待できる．

(2)システム構成

マイメニュー登録機能は，iアプリから渡されるパラメータを移動端末システムに受け渡すJava VMや，そのパラメータと移動端末システムから入力されるパスワードなどのパラメータとを連結してCiRCUSに送信し，CiRCUSからの応答を処理したり，表示する移動端末システムのモジュールで構成される．

(3)マイメニュー登録利用方法

iアプリからマイメニュー登録を行うAPIを新規に用意した．処理シーケンスの概略を**図9**に示す．セキュリティの観点から，ユーザが入力したパスワードやCiRCUSからの応答を直接iアプリで扱えないようにするため，一部機能を移動端末システムのモジュールに切り出した．

本APIを通してiアプリから登録／削除パラメータ，単数登録／複数登録を選択し，実行するとJava VMが適切な形式に整形して，移動端末システムのモジュールに受け渡す．このとき，Java VMはサスペンドする．移動端末システムのモジュールは受け渡されたパラメータをCiRCUSに送信し，CiRCUSからの応答に応じてユーザにパスワードの入力を促したり，確認画面の表示を行う．ユーザの行為に従い，移動端末システムのモジュールは，APIで渡

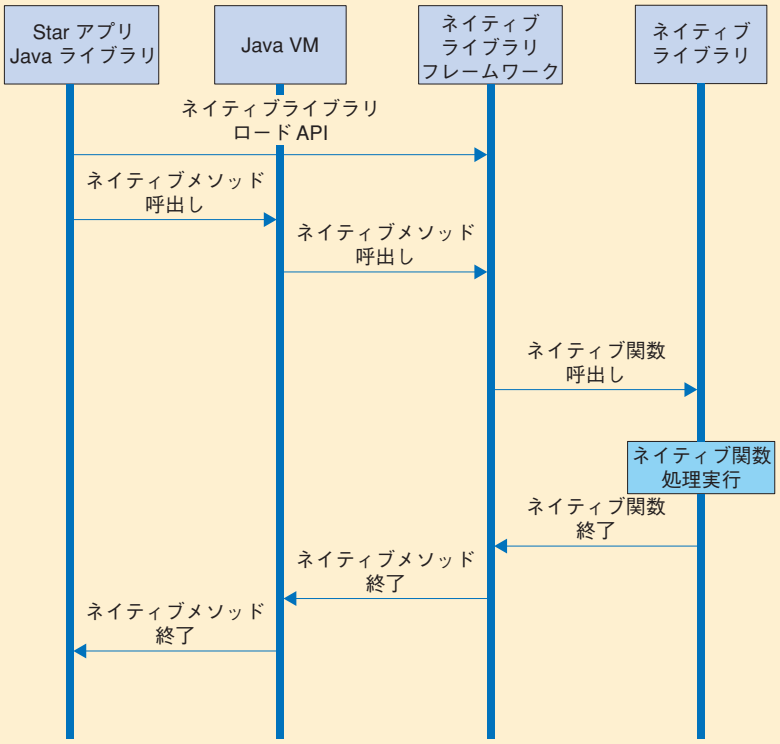

図8　ネイティブライブラリの呼出しシーケンス

＊22 CiRCUS：ドコモのコアネットワークとインターネットを中継する役割をもち，i-modeメール，i-modeメニュー，一般のインターネットへのアクセスなどを提供している装置．

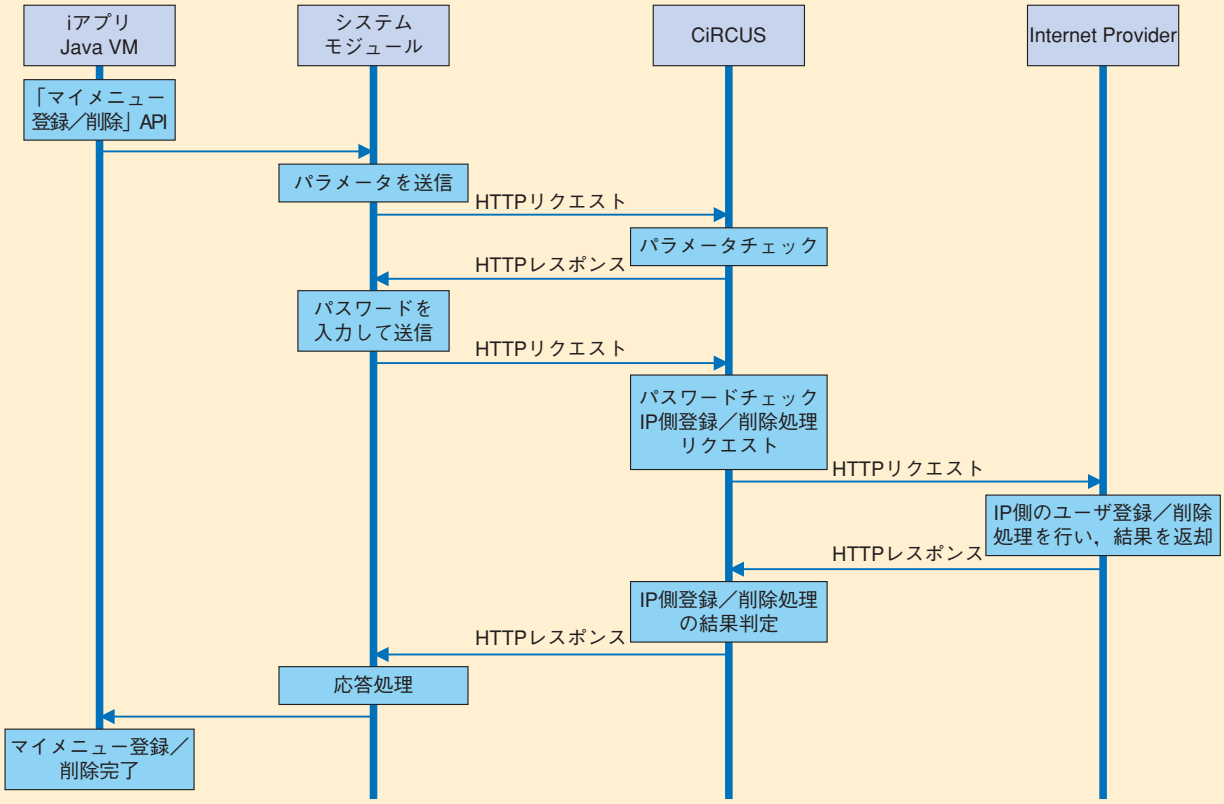

図9　マイメニュー登録の処理シーケンス

されたパラメータと，ユーザ入力パスワードおよびCiRCUSからの応答に付与されたパラメータとを連結して，再度CiRCUSに送信する．その後，CiRCUSからの登録／削除処理結果の応答を処理し，成功／失敗をレジュームしたJava VMを介してiアプリに通知する．

本機能は，ユーザに有料サイトの登録料金が発生する場合があるため，ユーザ保護の観点から，iアプリDXにのみ利用可能とし，一般アプリからの利用は禁止した．

7. あとがき

本稿では，新たに開発したStarプラットフォームの概要および新規機能であるWidgetView，Java-Flash連携，ライブラリ追加，マイメニュー登録の各機能について解説した．

今後も，Starプラットフォームはアプリケーションプラットフォームとしての役割を担い，さらなる高機能化，性能向上，利便性の追求を行っていく．

文　献

[1] 水口，ほか：“移動端末でのリアルタイム通信を目指したiアプリオンライン／iアプリコールのシステム開発，”本誌，Vol.16，No.4，pp.55-62，Jan. 2009.