アプリ連携 共通基盤 SMS Push

Technology Reports

新世代スマートフォン向けサービス特集

Android向けアプリ共通基盤開発

近年，Android™[*1]における市場のニーズが急速に高まっており，その中でフィーチャーフォンでドコモが提供していた機能・サービスに対応してほしいとの声も高まっている．2011-2012年冬春モデルより，Androidにおけるアプリ開発の柔軟性を活かしつつ，フィーチャーフォン同等の機能・サービスを確保するために，「アプリ連携機能」「SMS Push機能」を開発し，Android向けの共通基盤機能として提供した．

移動機開発部 本岡 代介 高橋 悟史
ネットワーク開発部 吉川 実穂 吉見 佐知子

1. まえがき

現在，Androidの普及が急激に進んでおり，ドコモでも端末ラインアップ構成の主軸を，Androidをはじめとしたスマートフォンにシフトしてきている．Androidはオープンソースであるため，自由度があり柔軟性に長けていて，一般のユーザがアプリケーションを開発し，公開することができるプラットフォーム[*2]が構築されている．その中で従来のフィーチャーフォン（iモード端末）の機能をAndroid向けに提供してほしいという声もあり，Androidの利点を活かしつつiモード端末同等の機能・サービスを確保した共通基盤機能を開発した．

その1つは，ドコモのサービスアプリどうしを連携起動させる「アプリ連携機能」である．通常の単体のAndroidアプリケーションと比べ，ドコモ端末にインストールされているアプリケーションどうしがさらに柔軟で強力な連携が可能になる仕組みを導入した．

2つめは「SMS Push機能」の開発である．キャリアのネットワークサービスノードが端末に何かの操作や処理を起動させる手段として，制御用SMS[*3]（SMS Push）があるが，スマートフォンにおいてもよりダイナミックで付加価値の高いサービスを提供することが可能となるよう，SMS Push機能を開発した．

本稿では，その実現方式について解説する．

2. アプリ連携機能

通常の単体のAndroidアプリケーションとは異なって，ドコモ端末にインストールされているアプリケーションは，さまざまなドコモのアプリケーションどうしが協調して動作することを前提に設計されている．Android標準のアプリケーション連携はIntent[*4]を用いて実現されるが，ドコモのサービスアプリどうしではさらに柔軟で強力な連携を導入した．

2.1 Androidの通常の連携

Androidにはintentや Service Binder[*5]，ソケット通信[*6]や共有フ

＊1 Android™：スマートフォンやタブレット向けのオペレーティングシステム，ミドルウェア，主要なアプリケーションからなるソフトウェアプラットフォーム．米国Google, Inc.の商標または登録商標．
＊2 プラットフォーム：アプリケーションを動作させるためのOSや動作環境．

ァイルなどいくつかのアプリ間の連携手法が準備されている．それぞれの手法によるアプリ間連携の特徴を**表1**に示す．

今回注目するのは，このうちService Binderによるアプリ間の連携である．この連携手法はLinux®[*7]のリモートプロシジャーコール（RPC：Remote Procedure Call）[*8]を利用するプロセス間通信の一種である．この仕組みを用いると，アプリケーションの違いを意識することなく，別アプリ（以下，リモートアプリ）のメソッド[*9]を自アプリ（以下，ローカルアプリ）のように呼び出すことが可能となる．

Service Binderのセキュリティは Androidのパーミッション[*10]と署名[*11]によって守られている．Service Binderでメソッドを公開するアプリケーションは，マニフェストファイル[*12]でパーミッションを定義する．このリモートメソッドを呼び出すアプリケーションはこのパーミッションを利用することをマニフェストで宣言する．このようにローカルとリモートが合意したパーミッションを使うことで，ユーザが意図しないアプリケーションの連携を防ぐことができる．また，パーミッションにはAndroidの署名を基にしたレベルを設定することができる．このレベル設定によって，リモート側がアクセスを許可するアプリケーションを指定することが可能となる．パーミッションレベルは署名一致やシステム権限などいくつかのパターンが用意されている．

2.2 通常連携での問題点

通常のAndroidのアプリ連携では，リモートアプリ側がリモートメソッドのアクセスを制限するには，ローカルアプリが同一署名をもっている（またはシステムアプリケーションである）ことが必要となる．ドコモサービスアプリは通常のAndroidアプリケーションと異なり，さまざまなドコモアプリケーションが統一的に連携して，より利便性が高く，直感的な操作を実現している．それらの連携するアプリケーション群はあらゆる状況においてもスムーズに連携する必要があるが，既存の方式ではすべての場合をサポートすることはできない．このような状況でも整合性を保ちながら連携するために，ドコモ独自のアプリ間連携の仕組みを導入する．

2.3 改善されたアプリ間連携

新しいアプリ間連携では，リモートアプリとローカルアプリを分離し，新たな方式によって安全に連携することを実現している．今回開発したアプリ間連携の概要を**図1**に

表1　Androidのアプリ間の連携の特徴

連携方式	特徴
Intent	シンプルな連携方式で，メッセージの通知など一方向的な連携に向いている
Service Binder	独自のAPIを定義でき，より複雑な連携が可能
ソケット通信	独自のプロトコルを作成する必要があり，設計が複雑になる一方，データの受渡しは柔軟に行うことが可能
共有ファイル	単純なデータの共有には向いているが，複雑なタイミングの制御などは難しい

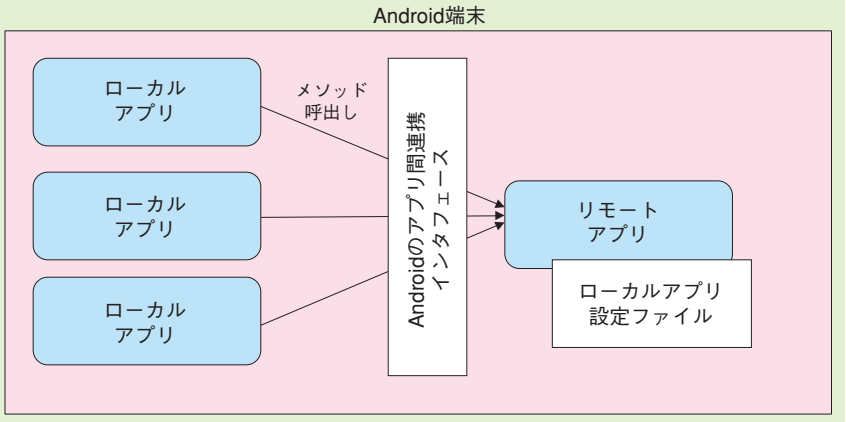

図1　アプリ間連携の概要

＊3　**制御用SMS**：アプリケーションの動作や起動など移動端末の制御に利用されるSMS．

＊4　**Intent**：Android OSが提供するプログラム間のパラメータ交換手段．アプリケーション内のコンポーネント間でのやり取りや，アプリケーションをまたがるやり取りに利用される．

＊5　**Service Binder**：Androidのプロセス間通信の1つ．RPC（＊8参照）をベースにAndroid独自のドライバで実装をしている．

＊6　**ソケット通信**：アプリケーションに対しTCP/IP通信の詳細を隠蔽するインタフェースとなるIPアドレスとポート番号の組．

＊7　**Linux®**：Linus Torvaldsの米国およびその他の国における登録商標あるいは商標．

＊8　**リモートプロシジャーコール（RPC）**：異なるアプリケーション間で関数呼出しを実現する技術．

＊9　**メソッド**：オブジェクト指向プログラミングにおける，オブジェクトがもっている振舞いのこと．

示す．

この改善版のアプリ間連携によって，Androidの既存の枠組みの中で，安全で標準Androidよりも柔軟なアクセス制御を実現することができた．また，既存の開発手法を踏襲しつつドコモ独自部分のみを拡張したことで，開発効率の改善を達成した．その結果より柔軟なアプリケーションどうしの連携を実現し，魅力あるドコモサービスアプリケーションのプラットフォームを構築することができるようになった．

3. SMS Push機能

キャリアのネットワークサービスノードが端末に何かの操作や処理を起動させる手段として，SMS Pushがある．ネットワークサービスノードからSMS Pushを送ることによって，キャリアサービスが能動的にユーザへアクションを促すことができ，よりダイナミックで付加価値の高いサービスを提供することが可能となる．SMS Push機能の全体構成を**図2**に示す．

Android標準のSMS Pushの処理の仕組みは非常にシンプルで，端末がSMS Pushを受信すると端末内の全アプリにIntentをブロードキャストする．任意のアプリケーションでこのintentを受信することができ，アプリケーション開発者は，自由にこのSMS Pushを受信するアプリケーションを作成することができる．

そこで，SMS Pushの基盤開発では，以下の機能を追加し，2011-2012年冬春モデルより実装した．

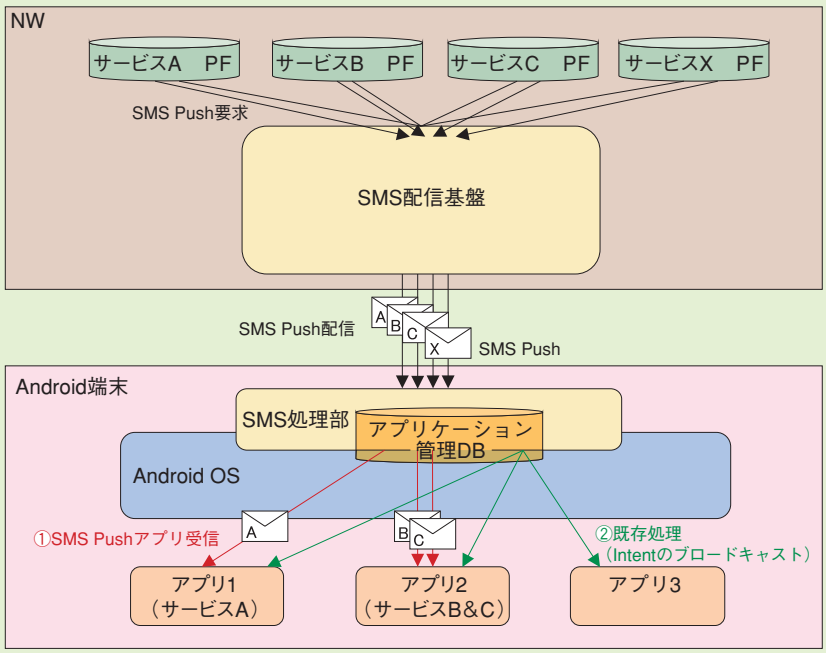

図2　SMS Push機能の構成

(1)端末側のSMS Pushの処理

まず，SMS Pushを受信するアプリケーションは，どのメッセージを受信するかを決め，データベースに登録する．つまり，受信アプリケーションはそれに紐付くSMS Pushメッセージをペアとしてデータベースで管理される．

SMS Pushを受信するとAndroidフレームワーク[*13]でSMS Pushのメッセージを解析し，受信アプリケーション管理データベースからメッセージのあて先を検索する．このメッセージのあて先のアプリケーションがすでに登録されている場合は，そのアプリケーションを起動する（図2①）．あて先アプリケーションが見つからない場合は，既存のAndroidのSMS Pushの処理に戻る（図2②）．これによって，ドコモが提供する高い信頼性を要求するアプリケーションは，処理を優先的に実行することができ，またそれ以外のSMS Pushメッセージは既存Android標準実装に則りメッセージが適切に処理される．

(2)SMS Push受信アプリケーションの変更

SMS Pushの受信アプリケーション管理データベースには管理用API（Application Programming Interface）[*14]が準備されている．ただし，この管理用APIを呼び出すことができるアプリケーションは正規のドコモアプリケーションのみである．

受信アプリケーションの管理デー

＊10 **パーミッション**：Androidのアプリ間連携のための設定．連携するシステムに合わせてパーミッションを定義することで，開発者，利用者双方にアプリケーションがどのように利用されるかを明確にすることができる．

＊11 **署名**：Androidアプリケーションを配布するときに必要な，開発元を証明する電子的な署名．

＊12 **マニフェストファイル**：Androidアプリケーションが利用する機能などを宣言した設定ファイル．すべてのAndroidアプリケーションはそれぞれの用途に合わせたマニフェストファイルを作成する必要がある．

＊13 **Androidフレームワーク**：Androidシステムの本体．Androidアプリケーションはこのフレームワークの上で動作する．

＊14 **API**：OSやミドルウェアなどが提供する機能を，他のソフトウェアが利用するためのインタフェース．

タベースには初期データとして，ドコモサービスアプリケーションが登録されている．将来的にSMS Pushを利用するドコモサービスアプリケーションが増えたり，名称が変わったりした場合は，この管理APIを使ってSMS Push受信アプリデータベースを変更することで，Androidのような開発サイクルが非常に早く，突然大きく変更されるようなプラットフォームにおいても，柔軟に対応することができ，端末リリース後の追加開発や設計へのインパクトを最小限に留めることが可能となる．

⑶SMS配信基盤の拡充

SMS配信において，下記のようなケースが起こり得る課題があり，SMS配信ルートには複数のルートがあるため，開発効率のよい共通的な機能として課題を解決する必要があった．

・ケース1

Pushサーバで，サービスに対応した端末かどうかを判断する端末適合判定をする場合，SMSが着端末まで配信される間に，SIM（Subscriber Identity Module）[*15]差替えにより端末が変更されるケースがある．Pushサーバで端末適合判定をしたにもかかわらず，サービスに適合しない端末でSMS Pushを受信してしまうことになり，誤動作など，想定されない動作を引き起こす可能性がある．

・ケース2

端末情報を取得できないNWの上位に機能分担されるPushサーバの場合は，端末適合判定を行わずSMS Pushを送信する場合がある．この場合，常にサービスに適合しない端末にSMSを着信させてしまうリスクがある．

これらの問題を解決するために，さまざまなSMS配信機能が利用できるSMSC（SMS Center）[*16]を経由するSMS配信基盤において，端末判定機能の整備を行った．今回新たに整備した端末判定機能の概要を**図3**に示す．

ケース1においては，SMS配信基盤の中で，Pushサーバで端末適合判定した際の端末から変更がないか，端末情報の最終チェックを行う必要がある．この機能は，リアルタイム性が求められるため，よりユーザに近い在圏交換機で，「端末適合判定時の端末情報」と「配信直前の最新の端末情報」とを比較し配信判断を行う機能とした（図3①）．

ケース2においては，SMS配信基盤の中で，Pushサーバに代わって端末適合判定を行う必要がある．この機能は，さまざまなSMS Pushルートで使用されることが想定される

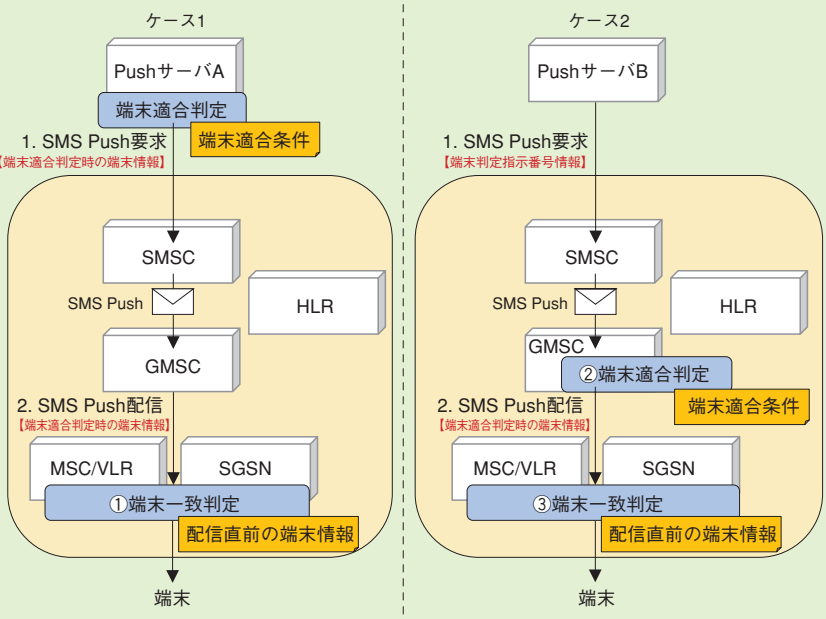

図3　端末判定機能

＊15 **SIM**：携帯電話会社と契約した電話番号などを記録しているICカード．GSMでの加入者識別モジュールをSIMと呼ぶ．

＊16 **SMSC**：3GPPで標準化されているSMSのセンタサーバ．SMSの蓄積，および再送を行う．

ため，機能集約が可能であり海外の配信ルートへも適応ができるGMSCで判定を行うようにした（図3②）．GMSCでは，SMS Push配信信号上の端末適合判定指示番号情報に紐付く端末適合判定条件を参照し，端末適合判定制御を行うことで，サービスの意識をすることなく固有制御ができるようにした．端末適合判定後は，ケース1と同様に，SIM差替えを考慮し，SMS配信基盤で端末適合判定した際の端末から変更がないか，端末情報の最終チェックを行う必要がある（図3③）．本ケースは，将来的に使用が予定されているケースである．

SMS Push機能改善では，AOSP（Android Open Source Project）[*17]で承認されたAndroidフレームワークのコードを使って，キャリアのSMS Pushを用いたより安全なサービスを提供することが可能となった．また，管理APIを効果的に使うことで，端末リリース後のサービス追加なども柔軟に行うことが可能となった．

SMS配信基盤へルートを集約し機能拡充することで，SMS配信基盤を利用する他のSMS Pushにも，共通的に端末判定機能を利用することができ，SMS Pushがサービス対応端末以外に送信されることを防止できるようになった．

4. あとがき

本稿では，Android向けアプリ共通基盤開発として，「アプリ連携機能」「SMS Push機能」について解説した．これらの導入により，Androidでもiモードと同等に柔軟にかつセキュリティを確保しながらサービスを提供できるようになった．Androidはオープンソースたるゆえ，常に次期OSのソースコードおよびSDK（Software Development Kit）[*18]がリリースされたのち，その影響を調査し，対応していく必要がある．

＊17 **AOSP**：Googleが公開しているAndroidのオープンソースプロジェクト．

＊18 **SDK**：アプリケーションを作成するときに必要となる，ドキュメント，ツール，ライブラリ，サンプルプログラムなどからなる開発キット．