S!アプリ開発ガイド

［ 基礎編 ］

Version 1.0.7 / Mar28,2014

ソフトバンク モバイル 株式会社

本書は情報提供を目的として作成されたものです。ソフトバンクモバイル株式会社は本書の記載内容に関して明示的にも、黙示的にも何ら保証するものではありません。

本書に記載されている事柄は、予告なしに変更する可能性があります。

本書の使用、または本書を使用した結果については、ユーザ各位がその責任を負うものとしますのでご了承ください。

■修正履歴

Version	日付	内容
1.0.0	2006/10/1	新規文書
1.0.1	2007/1/26	2.4.2.5. バイブレーション設定 新規追加
1.0.2	2007/6/1	2.2.10 キー同時押下 新規追加
1.0.3	2007/10/5	2.2.1.3 版検査 記載追加 2.2.11 KeyCode 2.2.12 USIMとの紐付け 2.2.13 文字エンコード 2.2.14 ネイティブプレイヤー競合動作 新規追加 2.4.2.3.1 着信時優先動作設定 記載追加 2.4.2.3.3 S!アプリ待受設定中の着信競合動作 新規追加 2.4.2.15 履歴管理 新規追加
1.0.4	2008/4/18	2.2.1.1. ダウンロード MIDP 2.0対応端末のダウンロード画面を追記 2.2.7. 特殊一時停止 新規追加 2.2.10 割り込み 表2.2.9-1. 割込み契機の一覧にS!アプリ起動要求(Bluetooth)を追加 誤記載 修正
1.0.5	2008/7/4	旧3G型(V801SH/V801SA)の記載を削除
1.0.6	2010/4/1	MIDP 1.0対応端末に関する記載を削除 2.4.2.2. S!アプリタイマー起動 項目削除 2.4.2.13. 接続設定 項目削除 誤記修正
1.0.7	2014/03/28	エミュレータに関する記載を削除 図2.3-1. 開発手順 手順削除 表2.3-1. 処理の留意点 項目削除 2.3.4. エミュレータ 項削除

0. イントロダクション

0.1. 目的

本書はS!アプリを作成する際に必要な技術情報を開発者様（以下「CP様」と表記）に提供するものである。

0.2. 前提

本書は以下の技術について熟知していることを前提とする。

- Java™2
- J2ME
- CLDC 1.1 ： Connected Limited Device Configuration 1.1
- MIDP 2.0 ： Mobile Information Device Profile 2.0
- Location API ： Location API for J2ME 1.0
- MMAPI 1.1 ： Mobile Media API 1.1
- WMA 1.1 ： Wireless Messaging API 1.1
- M3G ： Mobile 3D Graphics API for J2ME

その他、弊社提供の各種開発ガイドを熟知していること。

0.3. 参考文献

[CLDC 1.1]
http://www.jcp.org/en/jsr/detail?id=139

[MIDP 2.0]
http://www.jcp.org/en/jsr/detail?id=118

[Location API]
http://www.jcp.org/en/jsr/detail?id=179

[MMAPI 1.1]
http://www.jcp.org/en/jsr/detail?id=135

[WMA 1.1]
http://www.jcp.org/en/jsr/detail?id=120

[M3G]
http://www.jcp.org/en/jsr/detail?id=184

[HTTP]
“Hypertext Transfer Protocol – HTTP/1.1”, IETF, RFC2616, June 1999

[TS26234]
“3GPP TS26.234 Version 5.3.0”, 3GPP

0.4. 本書の構成

本書は以下の構成である。

1章 はじめに：各サービスの呼称について説明する。
2章 基本動作：弊社端末向けS!アプリについて説明する。

1. はじめに

本書では、表のように各サービスを呼称する。

表 1-1. 名称の定義

表記	内容
端末	S!アプリに対応した携帯電話
S!アプリサーバ	S!アプリを格納する弊社が指定する WWW サーバ
S!アプリ	ソフトバンク携帯電話で提供される Java™サービスの名称およびアプリケーションの総称
ネイティブ	S!アプリを使用しない端末の機能
MIDlet	CLDC と MIDP のみを使用して動作するアプリケーション 以降では、各種 JSR および MEXA/JSCL を使用しているものも MIDlet と表記する。
JSCL	J-PHONE Specific Class Library、弊社が定義した拡張ライブラリ
MEXA（メキサ）	Mobile Entertainment eXtention API、JSCL をベースとして弊社が定義した拡張ライブラリ
JSCL-1.2.2 対応端末	System.getProperty(”microedition.profiles”)の取得する値が ”MIDP-1.0 JSCL-1.2.2” である端末
JSCL-1.3.2 対応端末	System.getProperty(”microedition.profiles”)の取得する値が ”MIDP-1.0 JSCL-1.3.2” である端末
MIDP 2.0 対応端末（X）	User-Agent にて取得する値が “Profile/MIDP-2.0 Configuration/CLDC-1.1” および MEXA/JSCL を搭載した端末
MIDP 2.0 対応端末（S）	User-Agent にて取得する値が “Profile/MIDP-2.0 Configuration/CLDC-1.1” である端末

※ 本書は、MIDP2.0 対応端末について説明する。また、以下項目での説明は対応する端末において動作する。詳細については、S!アプリ開発ガイド［端末情報（各種）］および、コンテンツ開発ガイド[端末情報（各種）]を参照のこと。

2. 基本機能

2.1. S!アプリの構成

2.1.1. 構成

S!アプリは 1 個のプログラムと 0 個以上のデータで構成する。

表 2.1.1-1. S!アプリの構成

	構成	形式	内容
S!アプリ	プログラム	必須	S!アプリのプログラムが含まれているファイル
	データ	オプション	プログラムが利用するデータのみが含まれているファイル

2.1.2. プログラム

「プログラム」は「JAR ファイルと JAD ファイルの組」で構成する S!アプリのプログラムである。S!アプリを構成する上で必ず必要である。プログラムは、必ず S!アプリサーバに格納しなければならない。Web サーバには格納できない。

表 2.1.2-1. プログラムの構成

プログラム	JAD（必須）	
	JAR（必須）	Manifest ファイル(必須)
		Class ファイル（≧1、必須）
		Resource ファイル（≧0, オプション）

(ア)「Class ファイル」はパッケージ階層の名前付けに従ったディレクトリ構造を持つように構成する。「Resource ファイル」は S!アプリで利用するリソースを格納したファイルである。「Manifest ファイル」は JAR ファイルについて記載したテキストファイルである。

(イ)「JAR ファイル」は、Manifest ファイル, Resource ファイル, Class ファイルの 3 種類のファイルを JAR アーカイブしたファイルである。ここで必ず、1 個の Manifest ファイルと 1 個以上の Class ファイルをアーカイブしなければならないが、Resource ファイルはオプションである。

(ウ)JAD ファイルと Manifest ファイルとの間で共通の属性については同一の属性値を与えなければならない。

2.1.3. データ

「データ」はプログラムが利用する永続データである。Web サーバと S!アプリサーバに格納可能である。
但し、S!アプリサーバに格納するには「JAR ファイルと JAD ファイルの組」で格納すること。データは S!アプリを構成する上でオプションである。

(ア)　S!アプリサーバに格納する際の構成を以下に示す。

表 2.1.3-1. データの構成

データ	JAD（必須）	
	JAR（必須）	Resource ファイル（≧1，必須）

(イ)　Web サーバに格納する際の構成を以下に示す。

表 2.1.3-2. データの構成

データ	Resource ファイル（≧1，必須）

「Resource ファイル」は S!アプリで利用するリソースを格納したファイルである。

2.1.4. JADファイル

JADファイルは(対応する)JARファイルについての属性情報を記載したテキストファイルである。端末はS!アプリサーバからJARファイル（プログラム）をダウンロードする前に、当該 JAR ファイルが当該端末に適合しているかを判断する。適合している場合には JAR ファイルをダウンロードして端末上で実行する。適合していない場合にはJARファイルをダウンロードしない。

JADファイルには複数個の属性を記載する。個々の属性は属性名と属性値の組である。JADファイルに記載する属性にはMIDlet属性とS!アプリ定義属性がある。MIDlet属性は Java™で予約済みの属性であり、S!アプリ定義属性はS!アプリで任意に定義/利用することができる属性である。MIDlet属性は端末が利用するため、予め決められた規定と異なる用途で利用してはいけない。MIDlet 属性にはまた必須属性とオプション属性がある。必須属性を欠く JAD ファイルでは、対応するプログラムの実行は保証しない。

MIDlet属性は属性名が ”MIDlet-” で始まる属性である。JADファイルで利用できる属性を一覧すると以下のようになる。
S!アプリ定義属性は属性名に任意の文字列を追加した属性である。CP 様にて任意に文字列を追加することは可能だが、その際には「MIDlet-ＸＸＸ」、「MIDxlet-ＸＸＸ」といった文字列を追加記載してはならない。

表 2.1.4-1. JAD属性一覧

MIDlet 属性	必須属性	端末が利用する必須の項目である
	オプション属性	端末が利用するオプション項目である
S!アプリ定義属性		S!アプリで利用する

2つのプログラムでMIDlet-NameおよびMIDlet-Vendorが同一である場合には同一プログラムとみなす。このため、端末上にあるプログラムとMIDlet-Name,MIDlet-Vendor および MIDlet-Version が同一であるようなプログラムを S!アプリサーバから端末にダウンロードすることはできない。MIDlet-NameおよびMIDlet-Vendorが同一でS!アプリサーバにあるプログラムのMIDlet-Versionの値だけ大きい場合には、そのようなプログラムを S!アプリサーバから端末にダウンロードすることはでき、かつ、端末上のプログラムを S!アプリサーバ上のプログラムで上書きすることとなる。

※ ただし、一部の MIDP2.0 対応端末では同一バージョンの S!アプリを上書き保存する。

JAD ファイルは以下の構成でなければならない。

表 2.1.4-2. JADファイルの構成

ファイル名の文字エンコーディング	ファイル名は ISO8859-1(ASCII)のうち、英数字および下線文字のみで表記のこと
ファイル名の書式	FILE_NAME ::= LABEL '.' EXTENSION LABEL ::= ALPHA *ALPHA_NUM EXTENSION ::= "jad" ALPHA_NUM ::= ALPHA | ['0' - '9'] ALPHA ::= ['a' - 'z']|['A' - 'Z']| '_'
ファイル名の拡張子	jad(小文字で表記のこと)
ファイルサイズの制限	6kbytes 未満に収めること
MIME 型の指定	text/vnd.sun.j2me.app-descriptor
文字エンコーディング	UTF-8 で記載のこと

2.1.5. JAR ファイル

プログラムの実体は JAR ファイルに格納する。この JAR ファイルには 1 個以上の class ファイル、0 個以上の Resource ファイル、1 個の Manifest ファイルを JAR アーカイブする。

Manifest ファイルには JAR ファイルの属性情報を記載する。class ファイルはバイトコードを含むファイルである。Resource ファイルには該当 S!アプリで使用するデータを含める。

2.1.5.1. Manifest ファイル

Manifest ファイルはそれを含む JAR ファイルについての属性情報を記載したテキストファイルである。端末はこの情報に基づき、プログラムを識別し、実行するか否かを判定する。Manifest ファイルには JAD ファイルと共通の属性と Manifest ファイルに固有の属性がある。

Manifest ファイルには複数個の属性を記載する。個々の属性は属性名と属性値の組である。必須属性を欠く Manifest ファイルでは、対応するプログラムの実行は保証しない。JAD ファイルと Manifest ファイルとで同一の属性名を有する MIDlet 属性には同一の属性値を与えなければならない。

JAD ファイルと Manifest ファイルとで同一名称を持つ MIDlet 属性のうち、MIDlet-Name, MIDlet-Vendor, MIDlet-Version については JAD ファイルと Manifest ファイルで属性値が異なる場合にはプログラムを起動しない。

MIDP では 1 つの JAR の中に複数個の MIDlet を記載できるので、各々の MIDlet を MIDlet-1,MIDlet-2, MIDlet-3,…で識別して、その名前、アイコン、クラスを記載する。

Manifest ファイルは以下の構成でなければならない。(ファイル拡張子を除き JAD ファイルと同一)

表 2.1.5.1-1. Manifestファイルの構成

ファイル名の文字エンコーディング	ファイル名は ISO8859-1(ASCII)のうち、英数字および下線文字のみで表記のこと
ファイル名の書式	FILE_NAME ::= LABEL '.' EXTENSION LABEL ::= ALPHA *ALPHA_NUM EXTENSION ::= "mf" ALPHA_NUM ::= ALPHA | ['0' - '9'] ALPHA ::= ['a' - 'z']|['A' - 'Z']|'_'
ファイル名の拡張子	mf
ファイルサイズの制限	6kbytes 未満に収めること
文字エンコーディング	UTF-8 で記載のこと

※ JAD ファイルと Manifest ファイルで MIDlet-Name, MIDlet-Vender, MIDlet-Version が同一値でないといけない。同一値でない場合には端末への取り込み、および、端末上での S!アプリの動作は保証しない。

2.1.5.2. Resource ファイル

Resource ファイルにはプログラムで使用する画像(静止画像、動画像)データ、音曲データを載せる。端末上で S!アプリと対応づけるアイコンの画像も Resource ファイルで実現する。アイコンに利用できるデータの例としては、色深度が 8bit/pixel でサイズが 24 x 24 pixel の PNG フォーマットである。Resource ファイルにアイコンを含まない場合、端末では予め決められたアイコン画像を表示する。
尚、アイコンに利用できる画像のサイズは端末実装依存となる。

2.1.5.3. class ファイル

プログラムを実現したバイトコードである。
事前検証(preverify)済である必要がある。

2.1.6. 属性の書式

JAD ファイルおよび Manifest ファイルの属性の書式を BNF で記載する。

```
appldesc:       *attrline
attrline:       attrname ":" WSP attrvalue WSP newline

attname:1*<CTL および separator を除く Unicode 文字(UTF-8)>
attrvalue:      *valuechar | valuechar *(valuechar|WSP) valuechar
valuechar:      <CTL および WSP を除く Unicode 文字(UTF-8)>>

newline:CR LF | LF
CR = <Unicode の改行(0x000D)>
LF = <Unicode のラインフィード(0x000A)>

WSP:            1 * ( SP | HT )
SP = <Unicode の空白(0x0020)>
HT = <Unicode の水平タブ(0x0009)>
CTL = <Unicode で 0x0000～0x0009 の文字>
separator =  "(" | ")" | "<" | ">" | "@" | "," | ";" | ":"
             | "' " | "/" | "[" | "]" | "?" | "=" | "{"
| "}"
             | <"> | SP | HT
```

図 2.1.6-1. MIDlet属性のBNF記法

JAD ファイルおよび Manifest ファイルは UTF-8 で記載する。なお、当該ファイルには BOM(Byte Order Mark)は存在してはならない。(BOM：UTF-8 の場合は、 "0xEF 0xBB 0xBF")

(CR,LF,SP,HT,CTL は Unicode としては 2bytes であるが、UTF-8 表記上は 1byte となる点に注意すること。)

2.2. S!アプリの動作

2.2.1. 基本動作

本章では、S!アプリの動作について説明する。

2.2.1.1. ダウンロード

S!アプリサーバから JAD ファイルを取得すると端末では JAD の内容を検査する。検査の結果が正しい場合には、以下のような確認画面を表示し、利用者に JAR ファイルのダウンロード確認を要求する画面を表示する。

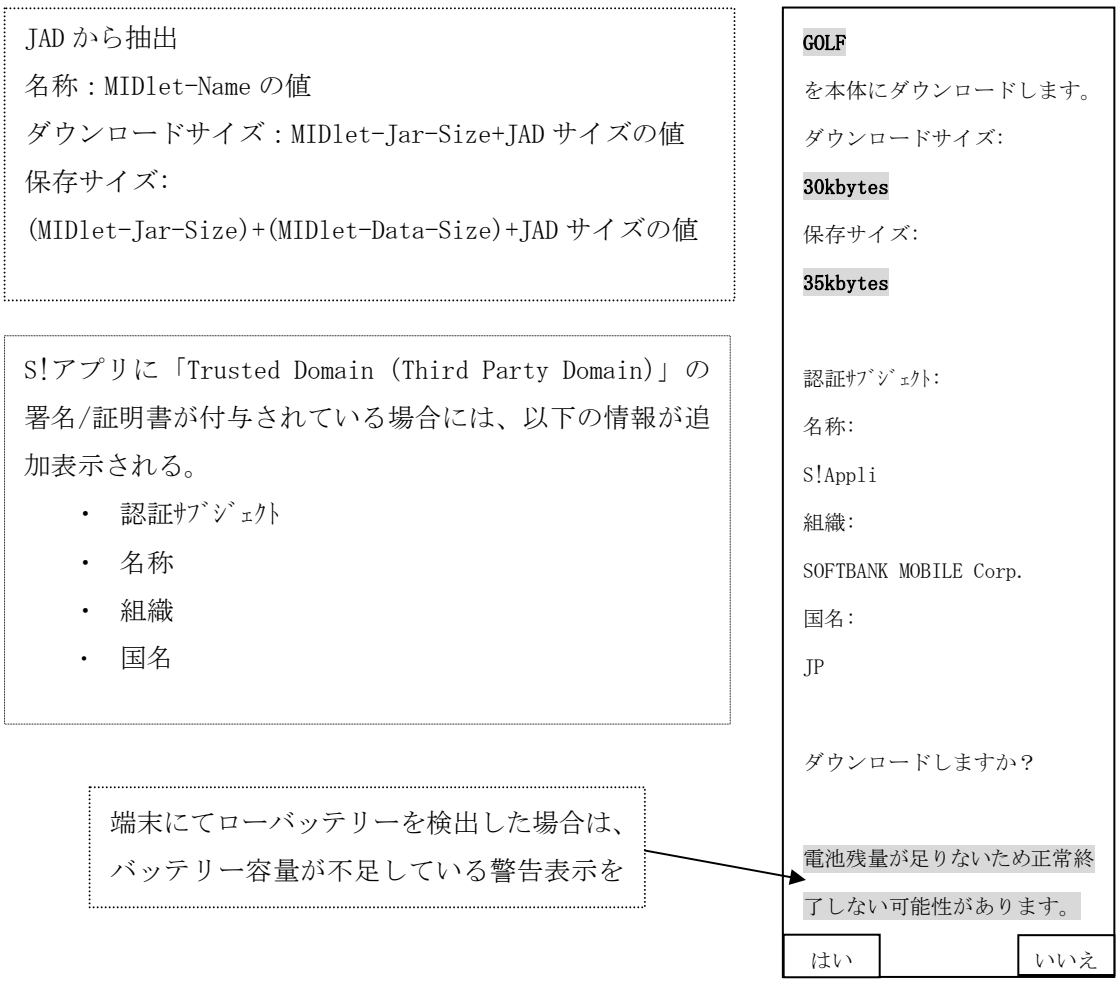

図 2.2.1.1-1. ダウンロード確認画面の設定

「はい」押下：　　JAR ファイルの取得
「いいえ」押下：　該当 S!アプリの取得を行わず、前の画面である選択画面に遷移する。

なお、ダウンロード確認画面の表示方法は端末により異なる。

2.2.1.2.解析

プログラムを実行するまでの間には、「JAD の取得⇒JAR の取得⇒起動」という手順を経る。各々の段階では予め決められた項目についてその適否を検査する。

表 2.2.1.2-1. 起動までの間の検査

検査契機	検査内容
JADダウンロード後	必須 MIDlet 属性の有無を確認する。
	MIDlet 属性の構文、意味を確認する。
JARダウンロード後	バージョンを検査する。(下記参照)
起動時	Manifest ファイル内の必須 MIDlet 属性の有無を確認する。
	Manifest ファイル内の MIDlet 属性の構文､意味を確認する。
	MIDlet-Permission 属性の値と S!アプリ設定を比較する。

上記の検査で必要な項目が存在しなかったり、不正な値、矛盾した値が設定されていた場合には、以降の処理は行わない。即ち、「JAD ダウンロード」後の検査を通過しなければ「JAR ダウンロード」は行わない、「JAR ダウンロード」後の検査を通過しなければ「不揮発性記憶域に蓄積」しない、「起動時」の検査を通過しなければ「起動」しない。
なお、弊社端末で実行する S!アプリは弊社の指定する S!アプリサーバからのみ取得することができる。S!アプリサーバ以外に置いた S!アプリは端末が読み取ることはできない。

2.2.1.3. 版検査

MIDlet-Name 属性および MIDlet-Vendor 属性で同一値の場合には、それらは同一の S!アプリであるものと見なす。同一の S!アプリについては MIDlet-Version 属性により版管理を行う。
従って、S!アプリサーバに対してプログラムの取得を要求した場合に、既に端末上に MIDlet-Name 属性および MIDlet-Vendor 属性について同一値を有するプログラムが蓄積されている場合には、双方の MIDlet-Version 属性の値に応じて以降の動作を決定する。

表 2.2.1.3-1. 版検査に伴うダウンロード制限

バージョン	挙動
(蓄)＜(Ｓ) バージョンアップ	S!アプリサーバ上の S!アプリをダウンロードし、蓄積済み S!アプリに上書きする。
(蓄)＝(Ｓ) 上書き	一部端末にてダウンロード可能。
(蓄)＞(Ｓ) バージョンダウン	一部端末にてダウンロード可能。

[凡例]
(蓄) ：端末上に蓄積済み S!アプリの版
(Ｓ) ：S!アプリサーバ上の S!アプリの版
※　尚、MIDlet-Data-Size 属性値は変更しないことを推奨する。

S!アプリサーバに S!アプリの取得を要求する場合に、既に端末内に蓄積済みの S!アプリと S!アプリサーバ内の S!アプリの MIDlet-Name 属性、MIDlet-Vendor 属性が同一値である場合、該当 S!アプリの更新を上記のとおり行う。この際、RecordStore は、以降のような処理を行う。

◆ S!アプリバージョンアップ時、および上書き時

表 2.2.1.3-2. S!アプリ更新時のRecordStore

	挙動
(蓄)＜(S)	既に端末内に蓄積済みの S!アプリの RecordStore データを継承する。 ダウンロードした S!アプリの RecordStore サイズに変更する。
(蓄)＝(S)	既に端末内に蓄積済みの S!アプリの RecordStore データを継承する。 RecordStore サイズの変更は行なわない。
(蓄)＞(S)	既に端末内に蓄積済みの S!アプリの RecordStore データを継承する。 蓄積済みの RecordStore サイズを保持する。

[凡例]

(蓄) : 端末内に蓄積済み S!アプリの RecordStore のデータサイズ (MIDlet-Data-Size)

(S) : S!アプリサーバ内からダウンロードした S!アプリの RecordStore のデータサイズ (MIDlet-Data-Size)

◆ S!アプリバージョンダウン時

表 2.2.1.3-4. S!アプリ更新時のRecordStore

	挙動
(蓄)＜(S)	既に端末内に蓄積済みの S!アプリの RecordStore データは保障できない。 ダウンロードした S!アプリの RecordStore サイズに変更する。
(蓄)＝(S)	
(蓄)＞(S)	

[凡例]

(蓄) : 端末内に蓄積済み S!アプリの RecordStore のデータサイズ (MIDlet-Data-Size)

(S) : S!アプリサーバ内からダウンロードした S!アプリの RecordStore のデータサイズ (MIDlet-Data-Size)

※ 尚、S!アプリに付与した署名/証明書により、動作は上記限りではない。

※ ダウンロードした S!アプリの RecordStore のデータサイズ (MIDlet-Data-Size)が小さい場合、一部端末ではデータを保障できない。MIDlet-Data-Size は変更しないことを推奨する。

2.2.2. ライフサイクル

CLDC/MIDP で規定されていない端末で固有の振る舞いとして、着呼等のハードウェアによる割込みと割込みからの復帰がある。割り込みに伴いS!アプリは図 2.2.2-1 のように Active 状態と Paused 状態の間を遷移する。ハードウェア割り込みに伴うプログラムの一時停止および再開に伴い下記の点に留意しなければならない。

- 一時停止時には暗黙のうちに MIDlet#pauseApp()を実行する。
- 再開時には暗黙のうちに MIDlet#startApp()を実行する。
- 再開時に制御が戻る先は、一時停止時に制御を手放した場所ではない。
- 一時停止時にはS!アプリの実行状態を保存する。
- 再開時には保存済みのS!アプリ実行状態を復帰する。

※ 割込み時には実行状態を保存はするが、再開時には割り込みの発生した場所へは制御が戻らない点に注意されたい。

なお、MIDlet#pauseApp()を明示的に実行するソフトウェア割り込みの場合にはこの限りではない。CLDC/MIDP の作法に従って制御すること。

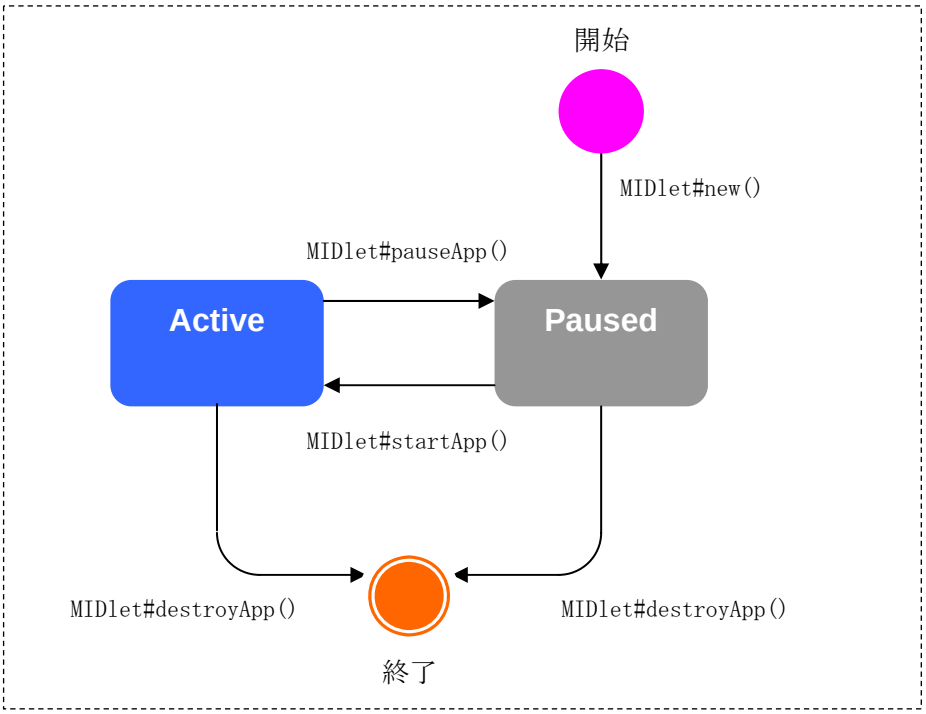

図 2.2.2-1. S!アプリのライフサイクル

図 2.2.2-1 の環境下における動作についての説明は以下の通りである。

- Paused 状態は S!アプリが休止している状態である。休止状態では端末はネイティブの制御下にある。
- Active 状態は S!アプリが Java VM 上にロードされ、S!アプリを実行中の状態である。Active 状態では端末は S!アプリ実行環境の制御下にある。
- 端末が S!アプリサーバに S!アプリ(JAR ファイル)のダウンロードを要求すると、JAR は端末の不揮発性記憶域に蓄積される。
- 不揮発性記憶域に蓄積した S!アプリ(JAR&JAD)は S!アプリライブラリで管理する。端末利用者が S!アプリの実行を指示すると端末は当該コンテンツの JAR を Java VM にロードしてプログラムは Active 状態となる。
- 「待受設定された S!アプリ」と「通常の S!アプリ」とでは同一の S!アプリであっても、状態遷移を引き起こすためのイベントが異なる。

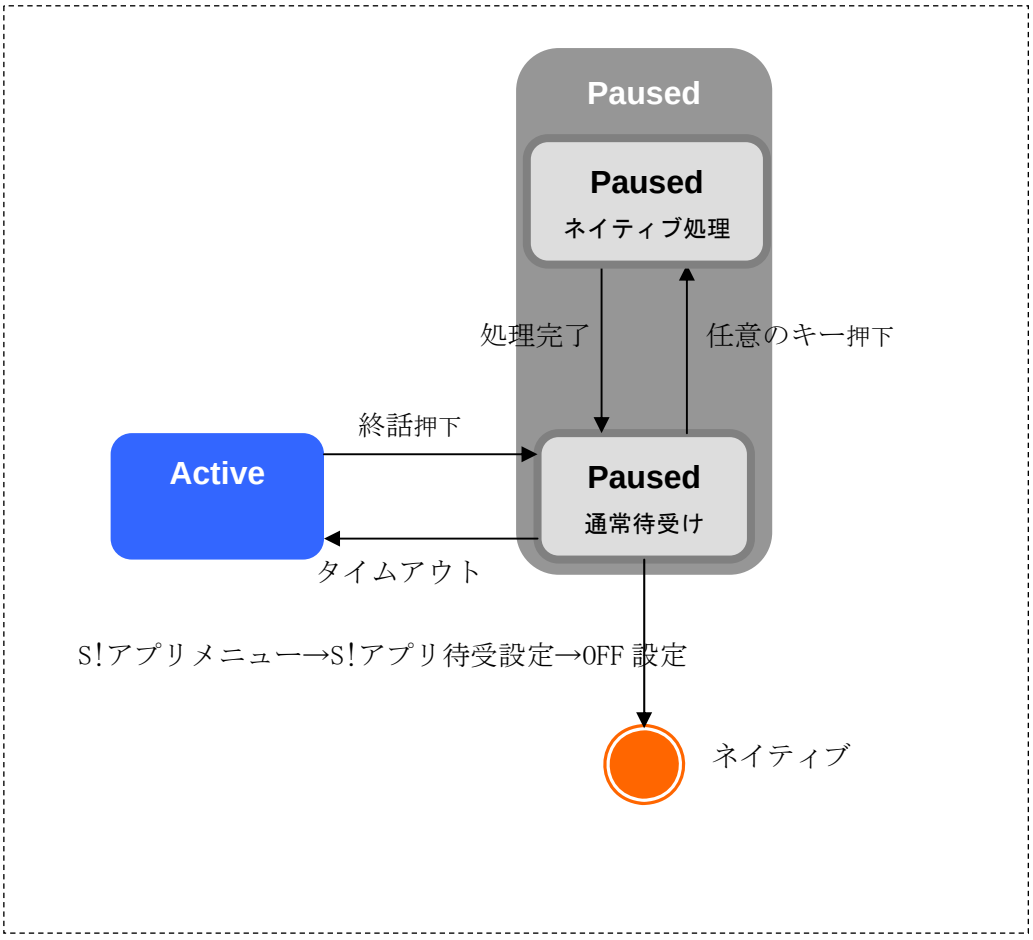

図 2.2.2-2.　待受設定されたS!アプリ

待受設定されたS!アプリではPaused状態にはさらに2つの内部状態(通常待受け、ネイティブ処理)がある。通常待受け状態とは、S!アプリを実行していない通常の待受け画面の状態である。ネイティブ処理状態とは、通話、ウェブ、メール等の従来のサービスを利用している状態である。
待受設定されたS!アプリを終了するには、通常待受け状態から図2.2.2-2.の操作によって、待受設定を解除する。

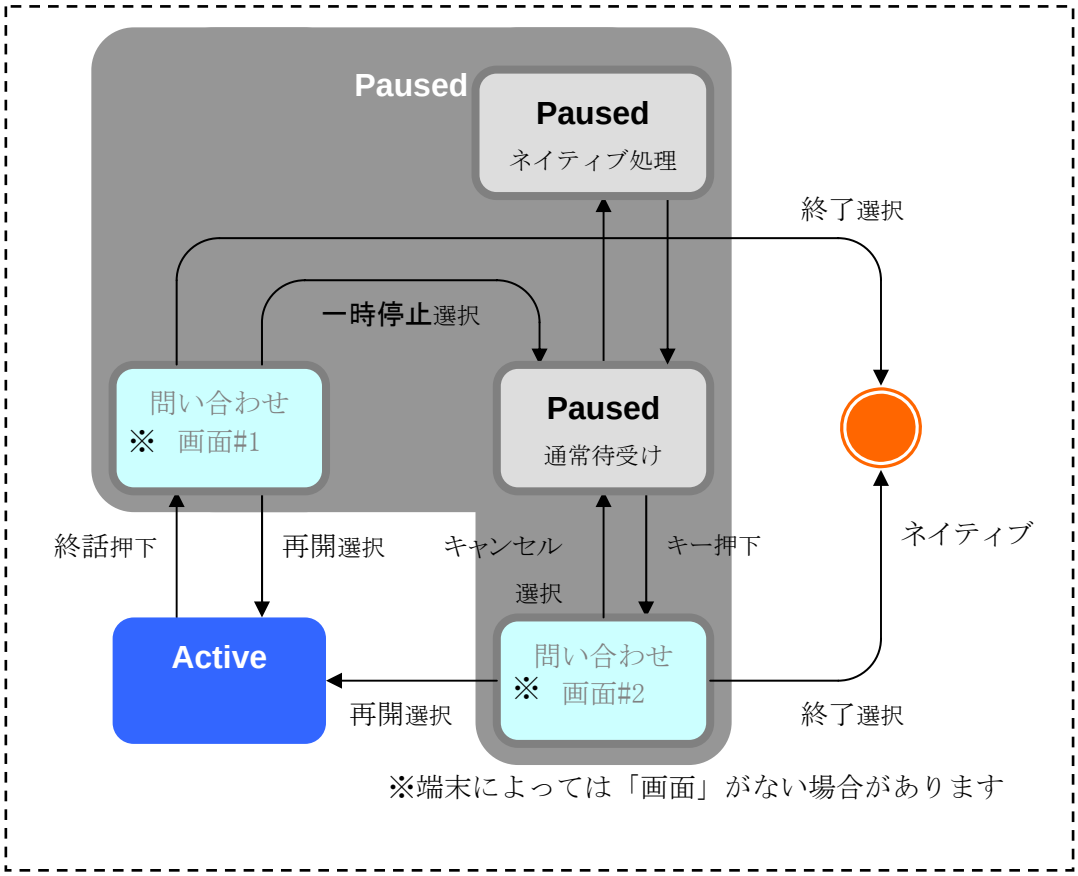

図 2.2.2-3. 通常のS!アプリ

通常の S!アプリ実行では Paused 状態にはさらに 4 つの内部状態(通常待受け、ネイティブ処理、問い合わせ画面#1、問い合わせ画面#2)がある。通常待受け状態およびネイティブ処理状態とは、待受設定された S!アプリで説明した内部状態と同じものである。問い合わせ画面#1, #2 とは、Active と Paused との間での遷移の際に利用者に問い合わせるための画面を表示する状態である。

「待受設定された S!アプリ」と「通常の S!アプリ」のそれぞれについて、上記の状態遷移をふまえ Active 状態において検査を行うこと。

2.2.3. 起動

利用者がS!アプリライブラリから選択して起動する。
MIDP2.0 対応端末においては S!アプリライブラリからの選択以外の起動方法がある。詳細はMIDP2.0対応端末編を確認のこと。
尚、1つのS!アプリ実行中にさらにS!アプリを実行することはできない。

S!アプリ起動時は、
「void Javax.microedition.midlet.MIDlet#startApp()」メソッドを呼び出す。

＜ S!アプリの起動 ＞

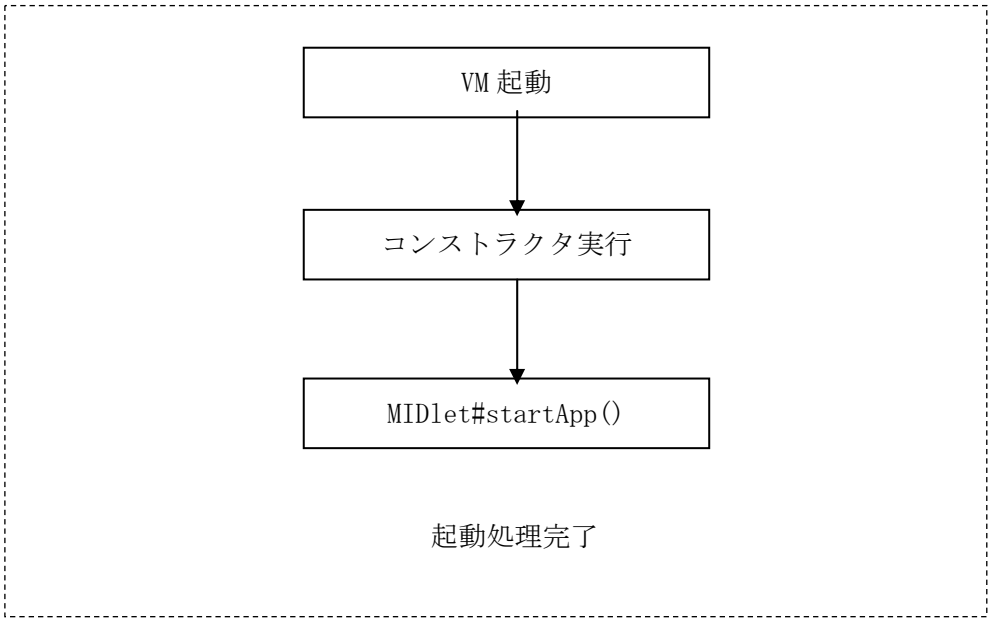

図 2.2.3-1.　S!アプリの起動

注意事項：

- MIDlet#startApp()処理の完了前に一時停止するとS!アプリが異常終了する可能性が高い。一時停止が成功した場合、処理途中から再開される。
- MIDlet#startApp ()処理の完了前に着信及びスケジュールアラーム通知があってもイベントリスナーは呼ばない。
- MIDlet#startApp ()はS!アプリ再開時にも呼ぶ。

2.2.4. 終了

S!アプリからの終了要求は、
「 void javax.microedition.midlet.MIDlet#destroyApp() 」 、 「 void javax.microedition.midlet.MIDlet#NotifyDestroyed()」を呼び出す。

＜ S!アプリの終了（S!アプリから）＞

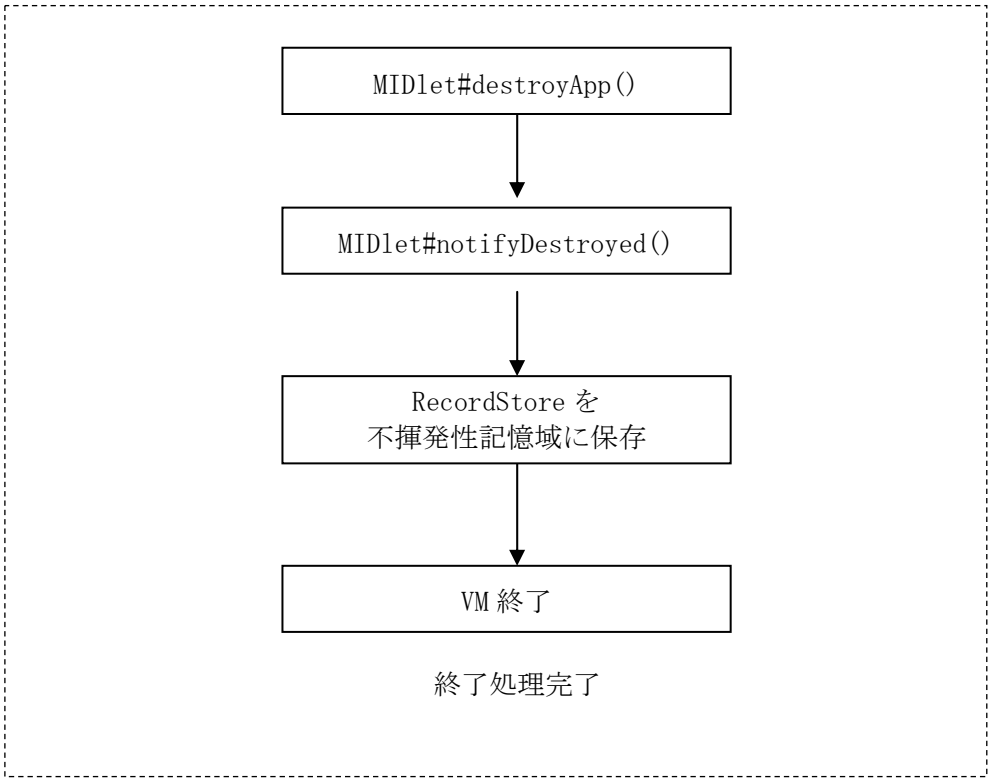

図 2.2.4-1. S!アプリの終了（S!アプリから）

注意事項：

- S!アプリから終了する為には `MIDlet#notifyDestroyed ()`を実行する。

S!アプリ外部からの終了要求は、
「void javax.microedition.midlet.MIDlet#destroyApp()」を呼び出す。

＜ S!アプリの終了（外部から）＞

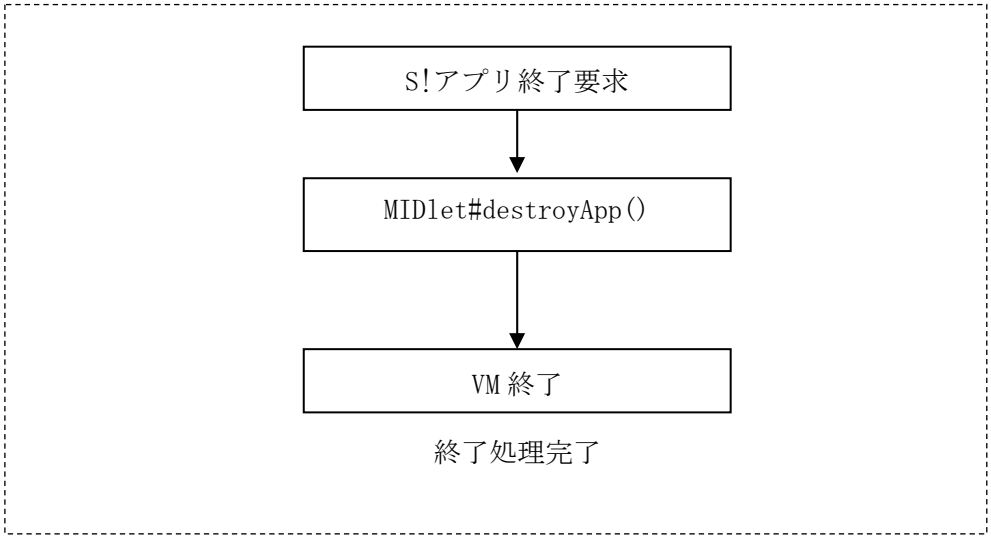

図 2.2.4-2. S!アプリの終了（外部から）

注意事項：

- MIDlet#destroyApp ()はバックグラウンドで行ない、S!アプリ上では何ら副作用も起こさない。
- S!アプリ終了要求があってから、５秒以内に終了処理が完了しない場合は強制終了する。

2.2.5. 強制終了

割込み等により終了処理を要求しても予め決められた時間(≒5sec)内に S!アプリが適切な終了処理を実施できない場合には、ネイティブは Java VM に対して S!アプリの終了を要求する。

強制終了した際には実行中のS!アプリが保持するリソースは保存されない。また、RecordStore に書き込んだ内容は保証しない。

2.2.6. 一時停止

S!アプリ外部からの一時停止の要求は、
「void javax.microedition.midlet.MIDlet#pauseApp()」メソッドを呼び出す。
S!アプリから一時停止するためには、
「void javax.microedition.midlet.MIDlet#pauseApp()
⇒void javax.microedition.midlet.MIDlet#notifyPaused」の順番でメソッドを呼び出す。

一時停止処理を実行すると実行中の S!アプリが保持しているオブジェクトを揮発性記憶域に保持したまま S!アプリの実行を停止し、制御をネイティブに移す。

＜ S!アプリの一時停止（外部から）＞

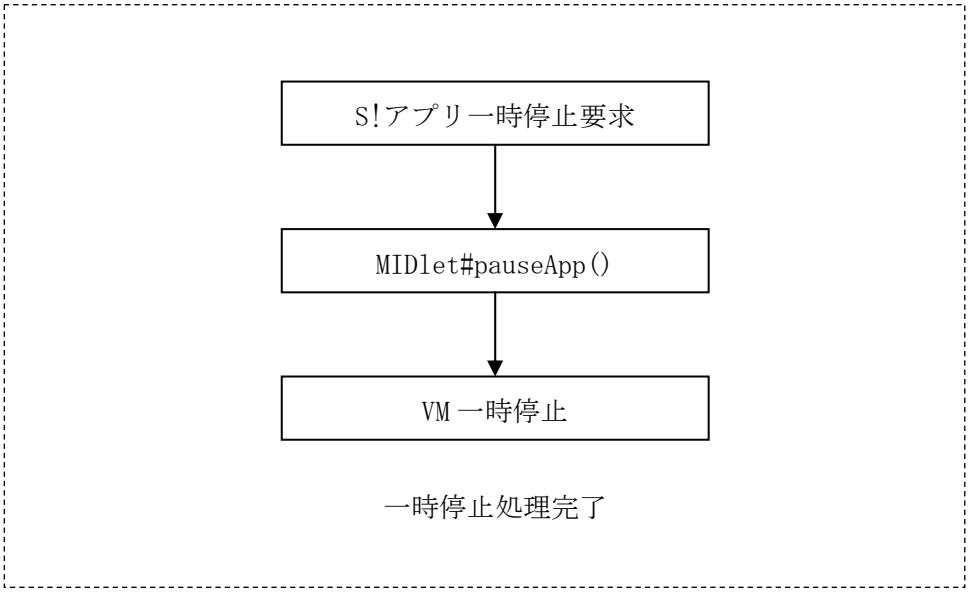

図 2.2.6-1.　S!アプリの一時停止（外部から）

注意事項：

- S!アプリ外部からの一時停止の要求は最も優先度が高い割り込みの為、`MIDlet#pauseApp()`処理が開始すると他のスレッドは一時停止状態になる。
- 一時停止の要求があってから 1 秒以内に一時停止の処理が完了しない場合は、強制終了する。

＜ S!アプリの一時停止（S!アプリから）＞

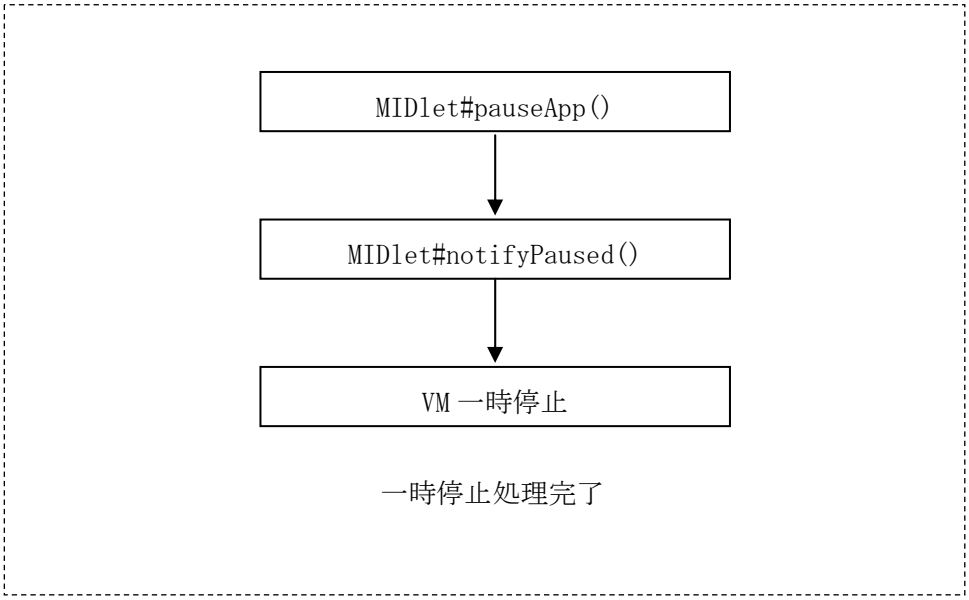

図 2.2.6-2.　S!アプリの一時停止（S!アプリから）

注意事項：

- S!アプリから一時停止する為には MIDlet#pauseApp ()の順で MIDlet#notifyPaused ()を実行する。

2.2.7. 特殊一時停止

S!アプリから以下の端末機能が呼び出された場合、特殊一時停止状態となる。この際にスレッドはブロックされ、ネイティブ機能を起動する。
なお、一時停止処理とは異なり
「void javax.microedition.midlet.MIDlet#pauseApp()」メソッドは呼び出されない。

- 音声発信(MEXA/JSCL)
- カメラ
- バーコード
- ブラウザ起動(MEXA/JSCL)
- アドレス帳登録
- ネイティブメディアプレイヤー起動
- TV 起動

2.2.8. 再開

一時停止により保持したオブジェクトを用いて、S!アプリの実行を再開する。再開は起動時と同じ「void javax.microedition.midlet.MIDlet#startApp()」メソッドを呼び出す。再開に伴うプログラムの実行状態の回復は JavaVM で行うため、S!アプリ上から再開に伴う特殊な処理を記述する必要はない。

＜ S!アプリの再開 ＞

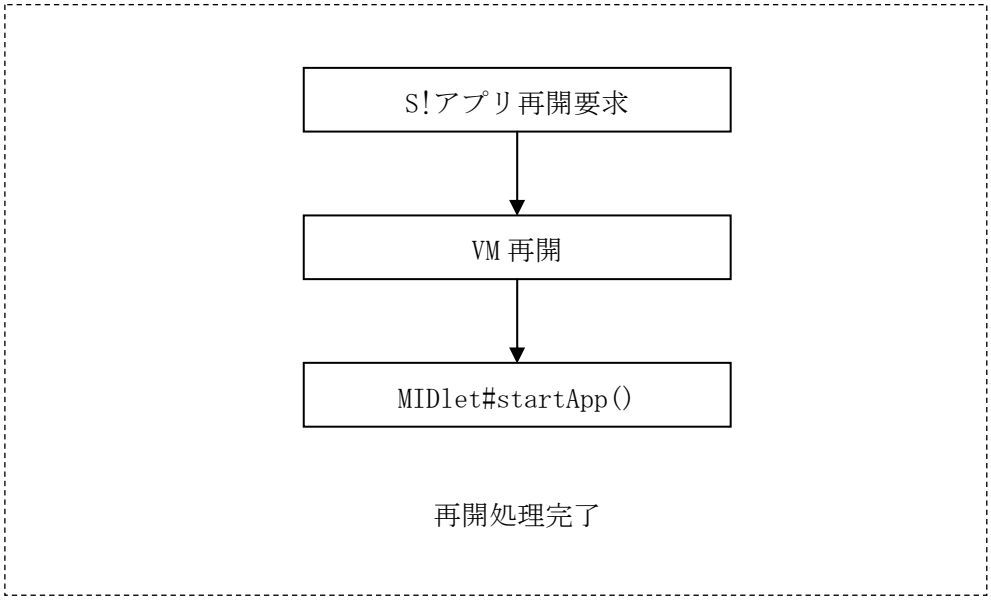

図 2.2.8-1.　S!アプリの再開

注意事項：

- 起動時と同様に、MIDlet#startApp ()の完了前に着信やスケジュールアラーム通知があってもイベントリスナーは呼ばない。
- MIDlet#startApp ()はS!アプリ起動時にも呼ばれる。
- 主スレッド以外のスレッドについては、MIDlet#startApp ()処理後に処理を再開する。

2.2.9. リソースの扱い

一時停止/特殊一時停止および、再開の際、リソースの扱いについて一覧する。

表 2.2.9-1.　リソースの扱い一覧

	一時停止/特殊一時停止	再開
MediaPlayer	MediaPlayer#pause() を実行するのと等価	MediaPlayer#resume() を実行するのと等価
描画デバイス	画像、グラフィックスコンテキストを揮発性記憶域に保存する	画像、グラフィックスコンテキストを再生する
RecordStore	open状態のものについては状態を揮発性記憶域に保存する	open 中のものについてはopen 状態で再開する
UI	インスタンスを揮発性記憶域に保存する	インスタンスを再描画する
タイマー	計時を停止し、時刻印を揮発性記憶域に保存する	保持した時刻印からの 計時を再開する
バイブレータ	バイブレータの動作状態を 揮発性記憶域に保存する	バイブレータの動作状態を 再開する
S!アプリ 点滅制御 (バックライト)	S!アプリの点滅状態を 揮発性記憶域に保存する	S!アプリの点滅状態を 再開する
着信 LED	着信 LED の明滅状態を 揮発性記憶域に保存する	着信 LED の明滅状態を 再開する

※　状態に不整合が発生せぬよう S!アプリ側でも状態を意識した処理を行うことを推奨する。

2.2.10. 割り込み

端末では S!アプリ実行中に割込みが発生した場合には、発生した割込みに応じた処理を実行する。端末で発生する割込みには次の 4 つのカテゴリがある。

表 2.2.10-1. 割込み契機の一覧

利用者割込み(ボタン押下などの利用者操作による)
S!アプリの実行中に電源ボタンや終話ボタン等を押下するなど、利用者操作による割込み
着信割込み
音声着呼
メールなどの着信による割り込み
アラームの発生による割り込み
S!アプリ起動要求(Bluetooth)
減電割込み
端末のバッテリーで残容量が不足した場合に発生する割込み
外部機器信号
端末底面の外部機器端子に接続した機器から発生する割込み

上記の各割り込み契機に対して発生する処理を以下に記載する。

表 2.2.10-2. 割込み時の処理一覧

利用者割込み
S!アプリに一時停止、もしくは終了要求を発行する
着信割込み
着信通知表示：割込みが発生した旨を S!アプリに通知する 着信動作優先：S!アプリに一時停止要求を発行する
減電割込み
稼動中の S!アプリのデータを保存し、終了する
外部機器信号
S!アプリに一時停止、もしくは終了要求を発行する。 (端末メーカーの仕様に依存)

S!アプリの開発にあたり、CP 様には下記の点に留意していただく必要がある。

表 2.2.10-3. S!アプリ上での割込み処理の留意点

利用者割込み、減電割込み
一時停止、再開、終了を行った場合には、S!アプリは MIDP で規定されている 2.2.2 ライフサイクルに従い状態遷移を行う。そのため CP 様は、利用者割込み、減電割込み、が発生した際に実行するメソッドを了解のうえで、S!アプリを作成していただきたい。
着信割込み
「S!アプリ待受設定」をしている時は、着信通知表示となる。利用者が S!アプリを起動していない時に、端末に設定した音およびバイブレーションが鳴動する。

2.2.11. キー同時押下

キーの同時押下について、3 つ同時押下まで可能である。
但し、押下可能なキーは、方向キー（ナビゲーションキー）に加え、数字キー/ソフトキー/センタキーのいづれかとし、それ以外のキーの同時押下は端末実装依存となる。

・方向キー（ナビゲーションキー）1 つ押下の場合、数字キー/ソフトキー/センターキーのいづれか 2 までが可能である。
・方向キー（ナビゲーションキー）2 つ押下の場合、数字キー/ソフトキー/センターキーのいづれか 1 つが可能である。

＊方向キー（ナビゲーションキー）で斜め方向が取得可能な端末におい
斜めキーが 1 つ押下されたこととなる。
（例）右下を押下した場合、1 つのキー（右下キー）としての取得となる。

2.2.12. KeyCode

端末は、以下に示すキーが押下された場合に、Canvas#KeyPressed()、Canvas#KeyReleased()、Canvas#KeyRepeated()に対して、KeyCode を返却する。

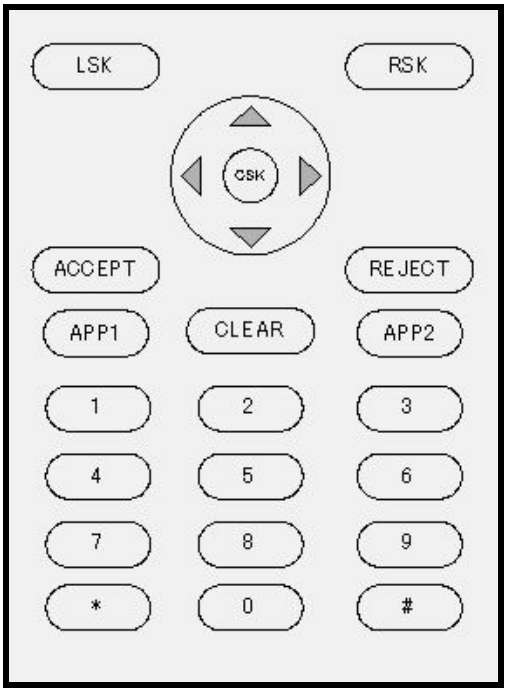

図 2.2.12-1. キーレイアウト 例

表 2.2.12-1. KeyCode一覧

端末入力	KeyCode (Canvas#KeyPressed(), Canvas#KeyReleased(), Canvas#KeyRepeated())
0キー	KEY_NUM0 (48)
1キー	KEY_NUM1 (49)
2キー	KEY_NUM2 (50)
3キー	KEY_NUM3 (51)
4キー	KEY_NUM4 (52)
5キー	KEY_NUM5 (53)
6キー	KEY_NUM6 (54)
7キー	KEY_NUM7 (55)
8キー	KEY_NUM8 (56)
9キー	KEY_NUM9 (57)
＊キー	KEY_STAR (42)
＃キー	KEY_POUND(35)
上キー	-1
下キー	-6
左キー	-2
右キー	-5
センターキー	-20
左ソフトキー	-21
右ソフトキー	-22
通話キー	-10
アプリケーション 1キー	-30 ※
アプリケーション 2キー	-31 ※
クリアキー	-8 ※

※ ただし、KeyCode(値)は端末実装依存となるため注意すること。

2.2.13. USIMとの紐付け

端末はダウンロードしたS!アプリをUSIMと紐付けて管理を行う。このため、ダウンロード時と異なるUSIMにおいては､S!アプリを起動することができない｡また、別USIM(A)でダウンロードしたS!アプリが、既に端末内に存在する場合、USIM(B)を利用した端末において S!アプリの上書きダウンロードはできず、ユーザに対して、その旨を通知する。

2.2.14. 文字エンコード

CLDC の java.lang.String 等で文字エンコードする際のデフォルトエンコードは、MEXA/JSCL を利用した S!アプリ(MIDxlet-API: MEXA or JSCL-X.X.X)と判断した場合は、「SJIS」となる。それ以外の場合は「UTF-8」あるいは「ISO8850-1」となる。エンコード時の入力文字列は以下を対応する。

表 2.2.14-1. エンコード文字列

文字種別	入力文字列
UTF-8	UTF-8
	UTF8
Shift_JIS	SJIS
	SHIFT_JIS
ISO8859-1	ISO-8859-1
	ISO8859_1

※ 尚、対応する文字エンコードについては、端末実装依存となる。

◆ MEXA/JSCL 利用 S!アプリ

・・・MIDlet 属性に「MIDxlet-API: JSCL-1.X.X」あるいは「MIDxlet-API: MEXA」と記載した S!アプリ

デフォルト文字エンコード ：SJIS

◆ JSR 利用 S!アプリ

・・・MIDlet 属性に「MIDxlet-API: JSCL-1.X.X」あるいは「MIDxlet-API: MEXA」と記載のない S!アプリ

デフォルト文字エンコード ：UTF-8 あるいは ISO8859-1

よって、String#getByte()を利用する場合には、String#getByte(“UTF-8”)のように明示的に文字エンコードを指定する必要がある。

2.2.15. ネイティブプレイヤー競合動作

ネイティブプレイヤーを実装している端末における S!アプリとネイティブプレイヤーとのサウンド同時再生可否については、実装依存である。同時再生不可である端末は、終了選択画面を表示した上で S!アプリを終了し、オーディオプレイヤーを再生する。あるいはオーディオプレイヤーを終了し、S!アプリを起動する。

2.3. 開発手順

この章ではプログラムの開発プロセスについて概要を説明する。S!アプリ開発者は Java™2 Standard Edition ソフトウェア開発キット（J2SE™ SDK バージョン 1.3 以上）を利用したことがあり、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する。

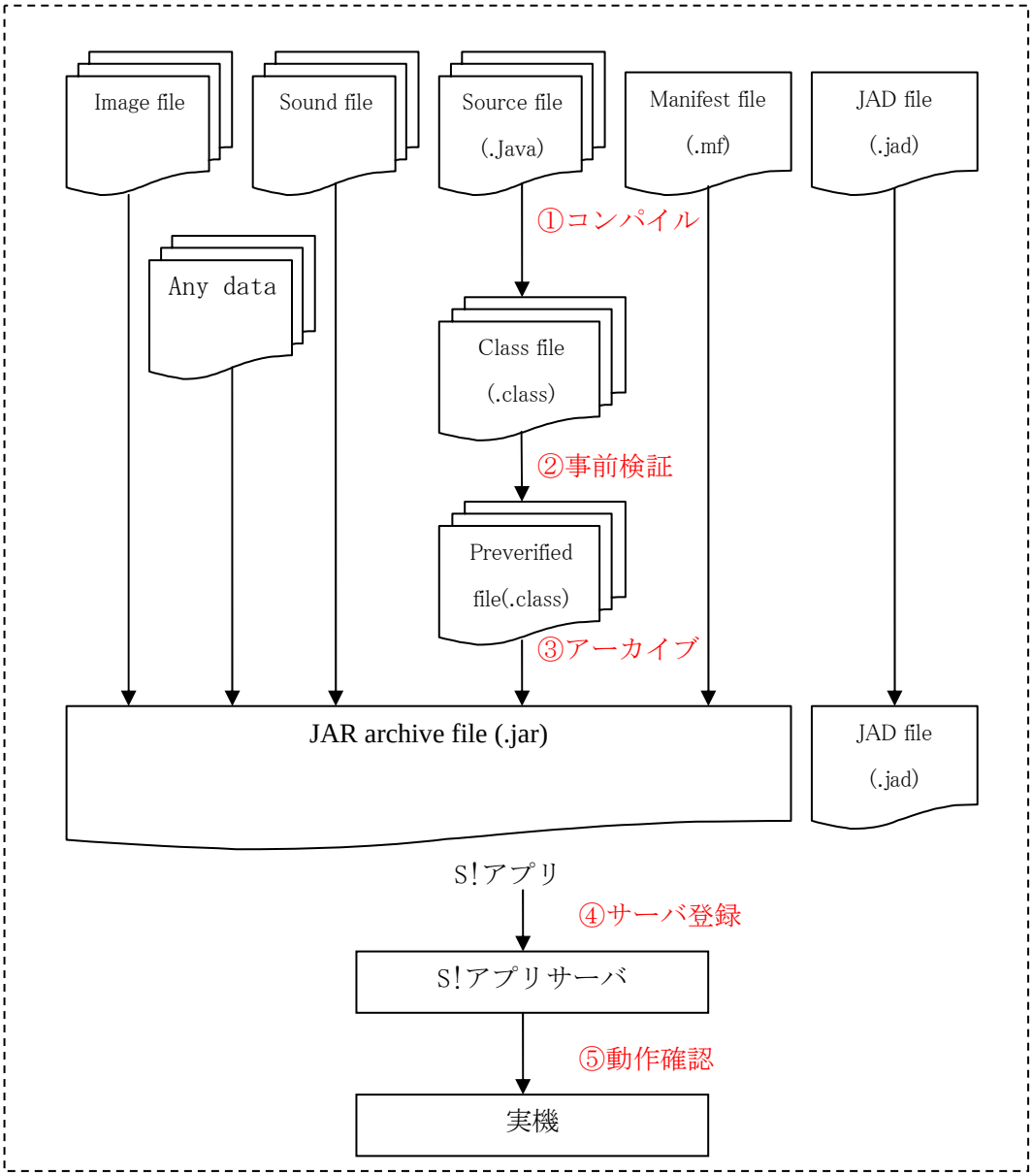

図 2.3-1. 開発手順

以下　図 2.3-1. 開発手順について説明する。

※　コンパイル：ソースファイル(.java)を Java™ コンパイラでコンパイルする。
※　事前検証：コンパイル済み Class ファイル(.class)を事前検証ツールにかける。
※　アーカイブ:事前検証済みClass ファイル(.class)、Manifest ファイル(.mf)、Resource ファイルを JAR アーカイバでアーカイブ(.jar)する。
※　サーバ登録：S!アプリサーバに登録する。
※　動作確認：S!アプリを実機にインストールし、動作確認をする。

上記過程の留意点を記載する。

表 2.3-1. 処理の留意点

処理過程	留意点
①コンパイル	Bootclasspath に弊社が提供するライブラリを指定のこと
②事前検証	米国 Sun Microsystems Inc. が提供する J2ME™ Wireless Toolkit に添付の preverify を利用のこと。
③アーカイブ	メディアプレイヤー, フレーズプレイヤーに渡すリソースデータは弊社指定のフォーマットであること。
④サーバ登録	S!アプリサーバに登録する。
⑤動作確認	実機を用いること。

2.3.1. コンパイル

ソースファイルを Class ファイルにコンパイルする時には、Java™2 SDK の Java™ コンパイラ（Javac）を使用する。
コンパイルするには次のコマンドラインを入力する。

形式

Javac ［ options ］［ sourcefiles ］

引数の順は任意である。

［ options ］　　コマンドラインオプション
［ sourcefiles ］　　コンパイルされる 1 つ以上のソースファイル

標準オプション

-g:none　　デバック情報は生成されない。
-classpath　classpath　　ユーザクラスパスを設定する。
-d　directory　　Class ファイルの出力先ディレクトリを設定する。

クロスコンパイルオプション

-bootclasspath　bootclasspath　　指定された一連のブートクラスに対してクロスコンパイルを行う。

例：コンパイル

JSCL クラスライブラリを C:¥SDK¥lib にインストールした例として以下に示す。
bootclasspath に stubclasses へのパスを指定する。

```
Javac -bootclasspath c:¥SDK¥lib¥stubclasses.zip Any.Java
```

2.3.2. 事前検証

MIDP の事前検証ツールではクラスに対する事前検証を行う。J2SE™ 等ではクラスの検証はプログラム実行時に行えるが、MIDP のようなリソースが限られる環境では実行時の負担を軽減するために事前にクラスの検証を行う。
事前検証するには次のコマンドラインを入力する。

形式

preverify [options] [src_directory]

[options]　　コマンド行オプション
[src_directory]　事前検証されていない Class ファイルが格納されているディレクトリ

オプション

-classpath　classpath　ユーザクラスパスを設定する。
-d　directory　事前検証された Class ファイルの出力先ディレクトリを設定する。

※ -d オプションを省くことによってカレントディレクトリに OUTPUT フォルダが作成される。作成された OUTPUT フォルダの中に事前検証されたクラスが格納される。

例：事前検証

クラスライブラリを C:¥SDK¥lib にインストールした例として以下に示す。
コンパイル生成された Any.class を検証するには、classpath に stubclasses へのパスを指定する。

```
preverify -d test -classpath c:¥SDK¥lib¥stubclasses.zip Any
```

2.3.3. アーカイブ

事前検証済みのすべての Class ファイル・Resource ファイル・Manifest ファイルを JAR ファイルとしてアーカイブするには Java™2 SDK に含まれる JAR ツールを使用する。

① Manifest ファイル作成
Manifest ファイルは JAR ファイルの持つ属性について記述した Unicode(UTF-8)で編集されたテキストファイルである。
作成したファイルの拡張子を.mf にする。

② JAR ファイルの作成
JAR（JavaArchive）は普及している ZIP 形式に基づくファイル形式で、Class ファイル・Resource ファイル・Manifest ファイルを圧縮形式で格納するための標準 Java™アーカイブ機能である。
JAR ファイルではファイルのサイズが圧縮され、これによって端末に必要な S!アプリ保管容量が減り、ダウンロード時間も少なくなる。
JAR ツールの入力ファイルには、次の 3 種類がある。

- Manifest ファイル（オプション）
- 出力先 JAR ファイル
- アーカイブするファイル

JAR を作成するには次のコマンドラインを入力する。

形式

jar [options] [manifest] destination input-file [input-files]

[options]	コマンド行オプション
[manifest]	Manifest ファイル
[input-files]	アーカイブ対象

オプション

c　アーカイブを新規作成する。

m　指定した既存の Manifest ファイルからマニフェスト情報を取り込む。

f　2 番目の引数で、処理される JAR ファイルを指定する。

・「cmf」ではなく「cfm」を指定したとき（「m」オプションと「f」オプションを逆に指定したとき）は、JAR アーカイブの名前のあとに Manifest ファイルを指定する必要がある。

例:アーカイブ

この例では Manifest ファイルと test ディレクトリ上に格納されている Class ファイルと Resource ディレクトリ上のリソースを JAR ファイルに統合する。

```
jar　cmf　manifest.mf　test.jar　-C　test　.　-C resource　.
```

③　JAD ファイルの作成

JAD ファイルは JAR ファイルの持つ属性について記述した Unicode(UTF-8)で編集されたテキストファイルである。

作成したファイルの拡張子を.jad にする。

作成した JAR ファイルサイズを MIDlet-jar-size に入力する。

2.3.4. サーバ登録

S!アプリサーバに登録する。

登録の方法については、各 S!アプリサーバ運営者に確認すること。

2.3.5. 動作確認

実機にて動作確認を行う。

2.4. 端末

2.4.1. S!アプリメニュー

S!アプリメニューでは以下の管理機能を提供し、利用者は端末上の内部メモリおよび外部メモリカード（外部メモリカード対応機種）に蓄積されている S!アプリの管理、S!アプリ実行環境の制御、S!アプリから利用できる(端末上の)デバイスの制御を行うことができる。

各管理機能について説明する。

2.4.1.1. S!アプリライブラリ管理機能

S!アプリライブラリ管理は端末の内部メモリおよび外部メモリカード（外部メモリカード対応端末）に蓄積した S!アプリについて管理する。ダウンロードした S!アプリを一覧表示、選択した S!アプリについて、起動、削除することができる。

2.4.1.2. 実行管理機能

実行管理では「待受時に実行する S!アプリ」および「タイマーで起動する S!アプリ」を設定する。待受時に実行する S!アプリとは、待受画面で S!アプリ待受設定を設定するように、待受時に S!アプリを実行させるものである。タイマーで起動する S!アプリとは、予め指定した時刻に S!アプリを起動するものである。
※　タイマー起動については一部端末での対応となる。

2.4.1.3. 構成管理

端末のデバイスを設定する。本機能では、着信イベント、再生音量、バイブレータ利用の可否、液晶画面のバックライトの照度、ネットワーク接続確認の表示等を設定することができる。

2.4.2. 管理機能

2.4.2.1. S!アプリ待受設定

MIDlet-Resident(MIDxlet-Resident)属性値が ”Y”、 ”S”の S!アプリを待受アプリと表記する。
待受アプリの中からひとつを S!アプリ待受設定することができる。

S!アプリ待受設定で稼動する S!アプリ（待受アプリ）は以下のように振舞う。

表 2.4.2.1-1.　S!アプリ待受設定の振る舞い

項目	説明
割込み処理	割込み処理は着信通知表示に設定される。

S!アプリ待受設定についてはその動作に以下のような特徴がある。

表 2.4.2.1-2.　S!アプリ待受設定の特徴

動作	説明
起動	S!アプリメニューでS!アプリ待受設定したあと待受画面に戻ると起動する。
解除	S!アプリメニューで S!アプリ待受設定を解除する。
一時停止	通常の S!アプリと同様の操作により一時停止する。
	一定時間キー操作がない、もしくは、折畳式端末を閉じた場合には一時停止する。
	継続起動待受アプリは、折畳式端末を閉じた際にも一時停止されない。
再開	割込みで一時停止後、通話やメール処理を終了すると、再開する。
終了	タイマー起動アプリの設定時刻になると、待受アプリを終了して、タイマー起動アプリを実行する。タイマー起動アプリが終了すると、待受アプリを起動する。
	一時停止中にS!アプリライブラリから他のS!アプリの実行を選択すると、待受アプリを終了して選択した S!アプリを実行する。当該 S!アプリが終了すると、待受アプリを起動する。
	一時停止中に S!アプリをダウンロードすると、待受アプリを終了する。ウェブブラウザを終了すると、待受アプリを起動する。
	ヒープの断片化が発生するので、リフレッシュのため 1 日 1 回あるいは一時停止 70 回等のタイミングで自動的に終了/再起動を行う。

※　終了／再起動のタイミングは端末実装依存となる。

◆　起動開始時間設定

待受アプリが未起動／一時停止状態の際に、待受画面に遷移してから待受アプリが起動／再開するまでの時間を設定することが可能である。待受アプリにより一時停止とされた場合には、ユーザーがキー操作を行った後に有効となる。

設定項目は以下指定となる。

- 設定範囲　：　１～１０秒
- デフォルト値　：　３秒

尚、起動開始時間設定の設定時間よりも先に端末が省電力状態に移行した場合には、待受アプリを再開しない。

◆　一時停止移行時間設定

動作中の待受アプリを一時停止するタイマーを設定する。ユーザが最後にキーを押した、または端末が折り畳まれた時点から設定したタイマーの時間後に、待受アプリは一時停止する。設定可能な値は端末実装依存となる

◆　一時停止設定

待受アプリが未起動／一時停止状態の際に、待受アプリが PushRegistry を受信するまで一時停止(常駐)する設定を行うことが可能である。設定は有効／無効とし、有効設定の場合は待受画面に遷移してからPushRegistryを契機に起動／再開する。尚、一時停止設定が有効の場合は起動開始時間を無視する。

2.4.2.2. 着信イベント

2.4.2.2.1. 着信時優先動作設定

着信時優先動作設定では S!アプリの実行中に電話着信などの割込みが発生した場合の端末での割込み処理方法を設定する。着信時優先動作設定では以下のうちいずれか一つを選択する。

1　着信優先動作

2　着信通知表示

それぞれを設定した場合の動作例を説明する。

着信優先動作：
S!アプリには着信のイベントを通知しないため、リスナーインタフェースは実装されていたとしてもコールバックされることはない。端末は S!アプリを強制的に一時停止して割込み処理に制御を移行し、電話着信やメール受信を行う。

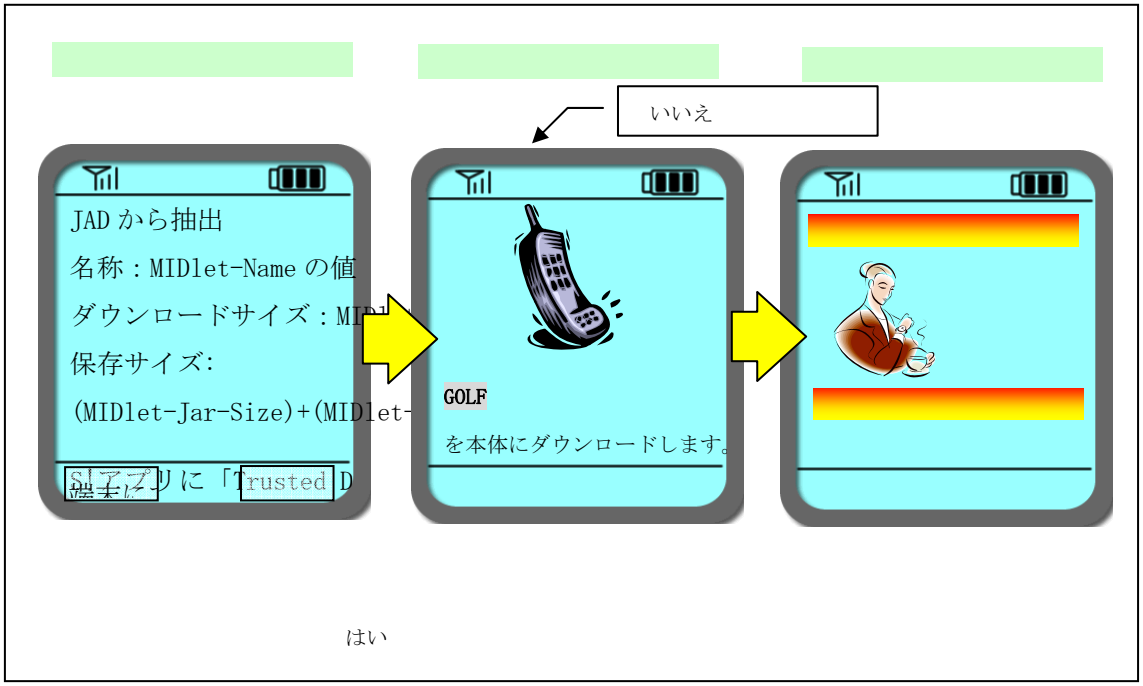

図 2.4.2.2.1-1.　着信優先動作の場合

着信通知表示：

着信した旨を端末上に表示するとともに S!アプリにイベントとして通知する。着信に応ずるか否かは利用者が判断する。S!アプリはイベントの内容(送信者名等)を利用することができる。利用者が着信に応じる場合は、S!アプリの実行を一時停止又は終了した後、割込み処理に制御を移し、電話着信やメール受信を行う。利用者が電話着信に応じない場合は、予め設定された処理(伝言メモ、留守録、応答メッセージ)をバックグラウンドで行う。

優先動作設定を着信種別ごとに設定可能である端末においては、以下のようなデフォルト設定となる。

表　2.4.2.2.1-1.　着信種別毎のデフォルト設定

着信種別	デフォルト設定
音声着信／TV コール／S!一斉トーク／アラーム	着信優先動作
メール／S!アプリ開始要求	着信通知表示

※ メニューに表示される文言は端末実装依存となる。

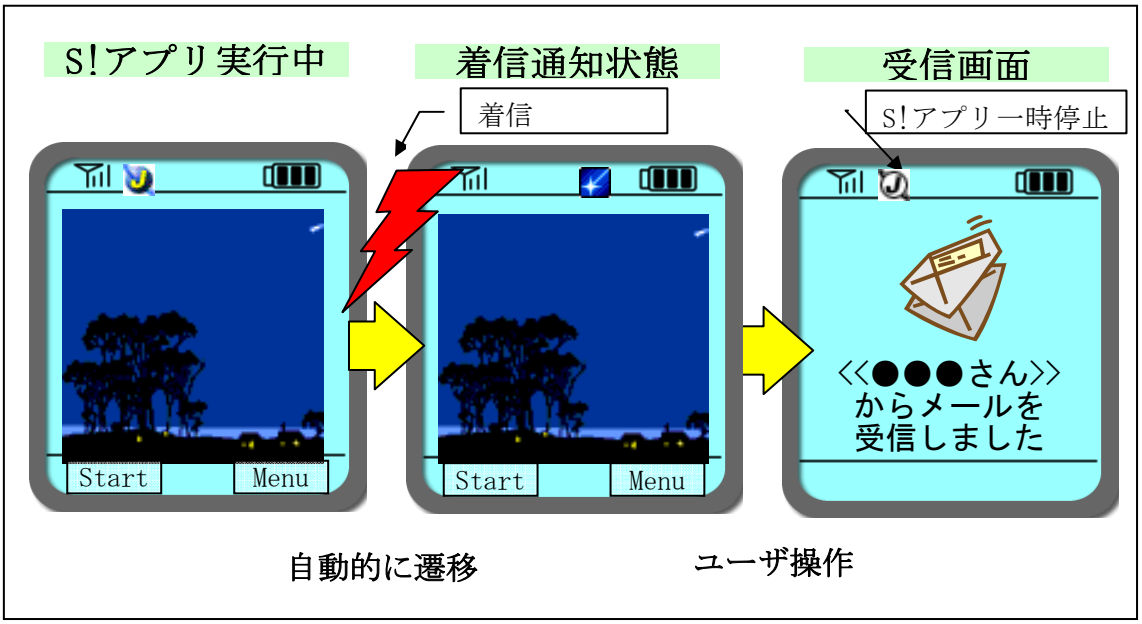

図 2.4.2.2.1-2.　着信通知表示の場合

2.4.2.2.2. ネイティブ機能連携中の着信イベント

以下に各状態中の着信イベントが発生した場合の振る舞いについて説明する。

表 2.4.2.2.2-1. ネイティブ機能連携中の着信イベント

機能	状態	振る舞い
音声発信	ネイティブの発信画面表示	ブロック状態である為、電話着信、メール着信、自動配信サービス着信、ステーション着信、アラーム通知発生は、端末側で着信動作を行い、S!アプリ側に通知されない。待受アプリも同様である。
メール	ネイティブの確認画面表示	
ブラウザ起動	ブラウザ表示	
カメラ バーコード	カメラ、バーコード機能を起動中	
ファイルシステム	ネイティブの確認画面表示	
動画再生	ムービープレイヤー実行中	
赤外線通信	ネイティブの確認画面表示	
	赤外線通信中	オフラインモード中である為、グリーティングやアラーム発生等のイベントに対しての一切の動作は行わず、通信が終了した後に通常の S!アプリ起動中の着信イベント発生と同様の動作を行う。
赤外線リモコン	リモコンデータ送信中	S!アプリが起動中である為、電話着信やメール着信等のイベントが発生した場合、通常の着信通知表示、着信動作優先の動作を行う。
機能遷移	遷移中	一時停止状態である為、電話着信やメール着信等のイベントが発生した場合、通常の一時停止状態中の着信通知表示、着信動作優先の動作を行う。

※ ブロック状態とは、S!アプリの全てのスレッドが停止している状態である。画面上は、一時停止アイコンが表示される。再開時には、一時停止と異なりStartApp()は呼び出されない。

2.4.2.2.3. S!アプリ待受設定中の着信競合動作

S!アプリ待受設定中において、ネイティブ着信音にオーディオファイル([.mp4]等)またはビデオファイル([.3gp]等)が設定された場合の着信時動作は以下となる。

表 2.4.2.2.3-1. 待受設定中の宅診オーディオ競合動作

ファイルタイプ	動作
オーディオファイル	待受アプリを一時停止することなく、バックグラウンドにてオーディオファイルを鳴音する。 ※ 一部端末においては、オーディオプレイヤー起動時に終了選択画面をユーザーに表示した上でS!アプリを終了する
ビデオファイル	待受アプリを一時停止し、ネイティブにてビデオファイルを再生する。着信優先動作と同様の動作を行う。 （イベントをS!アプリに通知することなく一時停止を行う）

2.4.2.3. 音量設定

S!アプリメニューの音量設定では S!アプリで利用できる音量の上限値を定める。S!アプリの再生音量の値は着信音量の設定した値とは独立である。マナーモード設定などにより端末で無音に設定されている場合には、S!アプリからも音は出力できない。

2.4.2.4. バイブレーション設定

S!アプリメニューのバイブレーション設定では S!アプリで利用できるバイブレーション制御方法を定める。但し、マナーモード設定時はマナーモード設定動作を優先とする。尚、デフォルト値は「ON」となる。

ON： バイブレーション制御を有効とする。
OFF： バイブレーション制御を無効とする。

2.4.2.5. S!アプリ点滅制御

S!アプリメニューの「パネル照明」では、液晶のバックライトについてその明滅の制御に加え、S!アプリからの制御の可否を設定することができる。

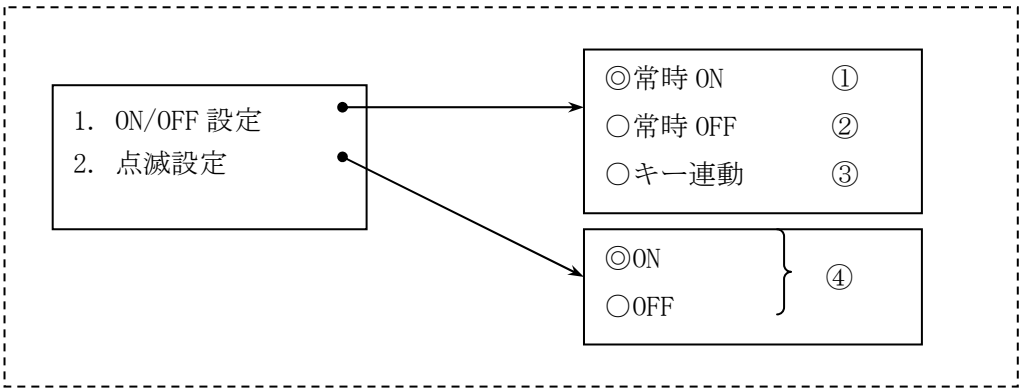

図 2.4.2.5-1.　パネル照明の設定

「1.ON/OFF 設定」では S!アプリ使用時のバックライトを常時点灯、常時消灯、キーと連動、のいずれかから選択できる。「2.点滅設定」では S!アプリの点滅について S!アプリから制御するか否かを選択できる。尚、デフォルト値は「キーと連動」となる。

S!アプリ実行中の明滅状態は、「1. ON/OFF 設定」が①、②である場合には「2. 点滅設定」が ON の場合には S!アプリからの S!アプリ点滅制御が優先する。
但し、「1. ON/OFF 設定」が③の場合であり、かつ、「2. 点滅設定」が ON の場合には、キー押下もしくは S!アプリで点灯指示した場合にバックライトを点灯する。

表 2.4.2.5-1.　S!アプリ点滅制御

		ON/OFF 設定		
		①(常時 ON)	②(常時 OFF)	③(キー連動)
点滅設定	ON	S!アプリからの制御を優先		※
	OFF	常時 ON	常時 OFF	キーに連動

※「点滅設定を ON」かつ「キー連動」を選択した場合には、基本的には S!アプリでの制御を優先する。しかしながら、キー押下による点灯中に S!アプリからの制御が発生した場合には、S!アプリの制御は無視して点灯を続ける。さらにこの際に点灯タイマーが満了した場合の振る舞いは端末の機種に依存する。

2.4.2.6. 背面液晶時間設定

背面液晶で S!アプリを実行中に、ネイティブシステムの時計表示を重ねることができる。

表 2.4.2.6-1.　背面液晶時計設定

設定	内容
ON	ネイティブシステムの時計表示を重ねる。
OFF	S!アプリの実行中は時計表示を行わない

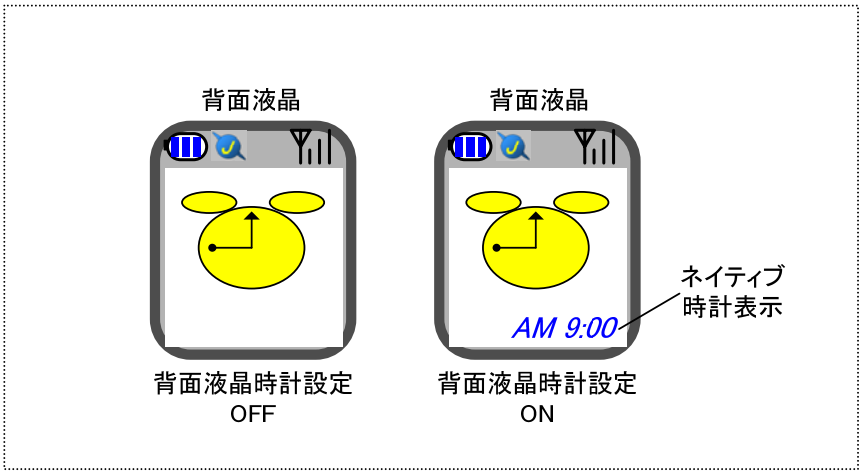

図 2.4.2.6-1.　背面液晶時計表示サンプル

2.4.2.7. マナーモード設定

マナーモード設定時に割り込み設定を着信通知表示に設定した時のバイブレーションと音の鳴動制御を以下に示す。

表 2.4.2.7-1. マナーモード設定

設定	内容
マナーモード設定なし	再生音量で、S!アプリからの鳴動制御に従う
マナーモード ON	マナーモード設定の音量で、S!アプリからの鳴動制御に従う

マナーモード設定時の待受アプリとそれ以外の S!アプリについて以下に示す。

表 2.4.2.7-2. バイブレーション制御

マナーモード設定		待受アプリ	その他 S!アプリ
OFF		S!アプリからのバイブレーション制御に従う	S!アプリからのバイブレーション制御に従う
ON	バイブレーション ON	S!アプリからのバイブレーション制御に従う 但し*着信時はS!アプリからのバイブレーション制御は無視し端末側でバイブレーションする	S!アプリからのバイブレーション制御に従う
	バイブレーション OFF	バイブレーションは振動しない	バイブレーションは振動しない

音声着信時は着信から切断まで､端末側が任意のパターンでバイブレーションし、メール・自動配信サービス着信、ステーション着信及びアラーム満了時は、端末側の設定時間分バイブレーションを振動する。設定時間分動作することが困難な場合は固定時間 動作を行う。これらのイベントで端末がバイブレーション動作している間は、S!アプリ実行環境側からのバイブレーション制御は無視する。呼切断後、または設定時間経過後は、通常通り S!アプリ実行環境側からのバイブレーション制御に従う。

2.4.2.8. サイズ表示について

S!アプリについて端末上で表示されるサイズには 3 種類ある。

表 2.4.2.8-1. サイズ一覧

種類	表示タイミング	意味
ダウンロードサイズ	S!アプリのダウンロード時に表示される。	Jar のサイズ
保存サイズ(S!アプリサイズ)	S!アプリのダウンロード時に表示される。	JAR のサイズ＋MIDlet-Data-Size(RecordStore のサイズ)の値
Property サイズ	S!アプリライブラリ→メニュー→プロパティ→[サイズ]	JAD 、JAR のサイズ＋MIDlet-Data-Size(RecordStore のサイズ)の値

尚、端末内での S!アプリ管理用のデータのサイズは端末メーカーの実装に依存する。

2.4.2.9. 高精細液晶への対応

QVGA 以上の端末は、S!アプリを拡大（整数倍）表示する機能を持つ。
拡大機能を持つ端末では、以下の S!アプリが提供可能である。

- 端末側で拡大表示される S!アプリ
- 高精細液晶サイズの S!アプリ

高精細液晶サイズに対応していない既存 S!アプリを端末側で拡大したい場合

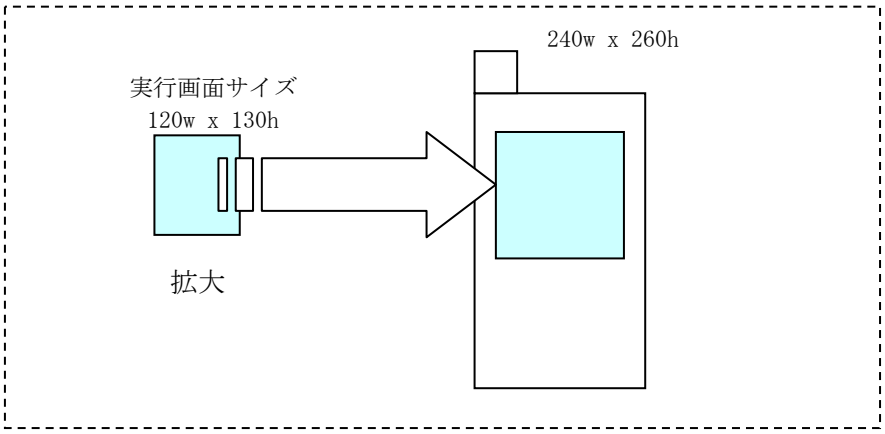

図 2.4.2.9-1. 端末側で拡大表示されるS!アプリ

高精細液晶サイズの S!アプリを作成したい場合

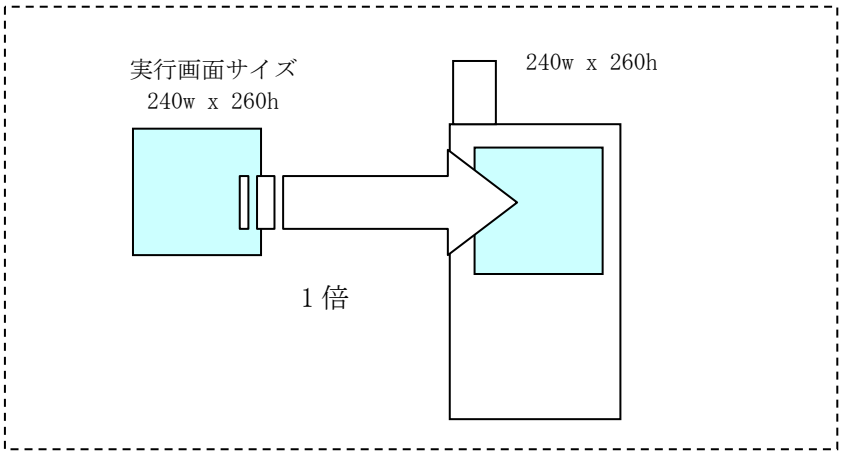

図 2.4.2.9-2. 高精細液晶サイズのS!アプリ

2.4.2.9.1. 拡大

高精細液晶サイズに対応していない既存S!アプリを端末側で拡大するには、120w x 130hの実行画面サイズに対応しているS!アプリを使用すればよい。

2.4.2.9.2. 高精細液晶サイズ

S!アプリ提供者がJADファイルに属性値を記載することによって高精細液晶サイズのS!アプリを提供することが可能である。
詳細については、S!アプリ開発ガイド[MIDP 2.0 対応端末編]2.2.5.21 MIDxlet-ScreenSizeを参照のこと。

◎　高精細液晶サイズ240w x 260h対応のS!アプリ作成例

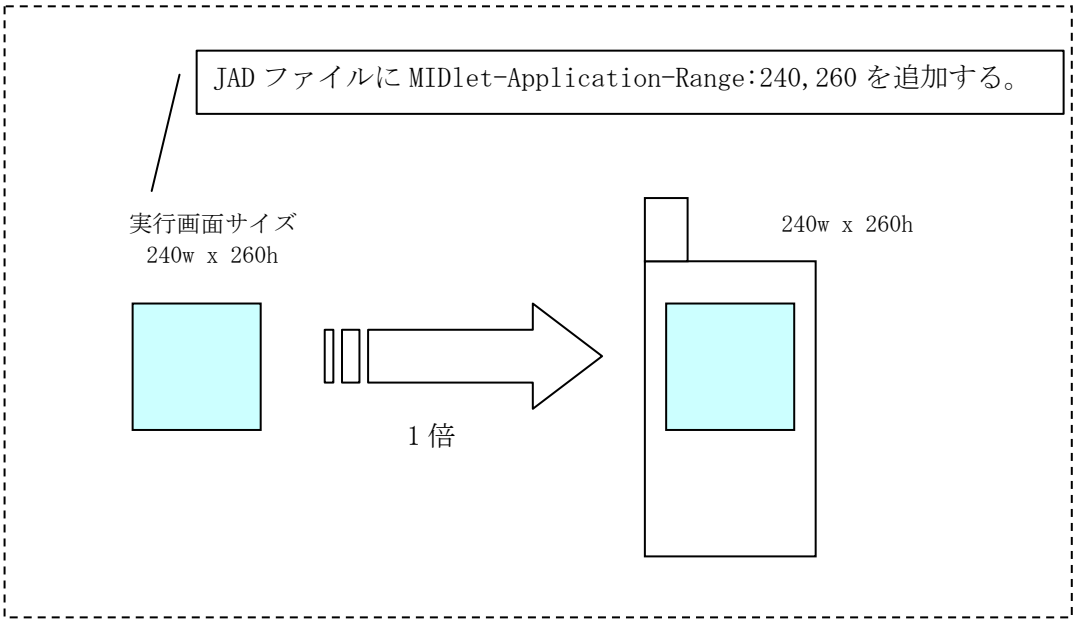

図2.4.2.9.2-1. 高精細液晶サイズ240w x 260hのS!アプリ作成例

2.4.2.10. ダイヤル操作禁止

ダイヤル操作禁止中のS!アプリ動作について以下に示す。

表 2.4.2.10-1. ダイヤル操作禁止中のS!アプリ動作について

S!アプリ種類	動　　作
タイマー起動アプリ	タイマー起動アプリ設定時の S!アプリ一時停止対応可否については、端末メーカーの実装に依存する。
	タイマー起動アプリの一時停止を行う場合、一時停止状態でダイヤル操作禁止となり、ダイヤル操作禁止解除でタイマー起動アプリを再開する。もしくは、ダイヤル操作禁止設定時にタイマー起動アプリを終了する旨を利用者に通知し、終了する。
	タイマー起動アプリ実行中は一時停止を行わないとダイヤル操作禁止設定できない。
	ダイヤル操作禁止中にタイマー起動アプリが起動した場合、キーイベントは S!アプリ実行環境へ通知されない。エニーキー操作により起動中のタイマー起動アプリを終了し、端末は待受状態に遷移し、ダイヤル操作禁止可能となる。
待受アプリ	待受アプリ設定中にダイヤル操作禁止設定を行った場合、その後待受アプリは再開されるが、キーイベントはS!アプリ実行環境に通知しない。
	待受アプリ起動中にダイヤル操作禁止されている場合は、ソフトキー領域等を使用し、「キー操作無効」等の文字列の表示を行う。もしくはエニーキー操作が行われた際に警告画面を表示するなどし、利用者へ通知する。
	待受アプリ起動中にエニーキー操作により、S!アプリ実行環境を一時停止し、ダイヤル操作禁止解除を可能とする。
	待受アプリが一時停止状態にある時、ダイヤル操作禁止解除により端末が待受状態に戻るタイミングで、待受アプリを再開する。
	オートロックでは、オートロック設定後電源 ON 時に待受アプリを起動し、以降の動作は上記のダイヤル操作禁止と同様。
その他S!アプリ	S!アプリ一時停止状態でダイヤル操作禁止となり、ダイヤル操作禁止解除で S!アプリを再開する。もしくは、ダイヤル操作禁止設定時に S!アプリを終了する旨を利用者に通知し、終了する。

2.4.2.11. 誤動作防止

誤動作防止によるキー操作無効中のS!アプリ動作について示す。

表 2.4.2.11-1. 誤動作防止中のS!アプリ動作について

S!アプリ種類	動　　作
タイマー起動アプリ	誤動作防止中にタイマー起動アプリが起動された場合、キーイベントはS!アプリ実行環境には通知しない。エニーキー操作により起動中のタイマー起動アプリを終了し、端末は待受状態へ遷移し、キー操作無効解除を可能とする。
待受アプリ	誤動作防止設定後、待受アプリを再開するが、キーイベントはS!アプリ実行環境に通知しない。
	誤動作防止中は、ソフトキー領域等を使用し、「キー操作無効」などの文字列を表示する、もしくはエニーキー操作が行われた時、警告画面を表示するなどし、利用者に通知する。
	エニーキー操作によりS!アプリ実行環境を一時停止し、誤動作防止の解除を可能とする。
	誤動作防止の解除後、端末が待受状態に戻るタイミングで、待受アプリを再開する。
その他S!アプリ	S!アプリ一時停止状態においてもキー操作無効であり、誤動作防止の解除によりS!アプリを再開する。もしくは、キー操作無効設定時にS!アプリを終了する旨を利用者に通知し、終了する。

2.4.2.12. セキュリティレベル設定

MIDlet-Permissions-Opt および、MIDlet-Permissions 属性値に記載のあるS!アプリのセキュリティレベルを設定する。MIDlet-Permission-Opt、MIDlet-Permissions 属性値に記載がない、または属性の記述がない場合には設定できない。

表 2.4.2.12-2.　セキュリティレベル設定

設定	内容
起動時表示	アプリ起動時に対象機能の利用可否の確認画面を表示する。
毎回表示	対象機能利用毎に利用可否の確認画面を表示する。
表示しない	利用可否の確認画面を表示することなく利用可能とする。
許可しない	対象機能の利用を不可とする。

※　設定可能な項目はセキュリティレベル設定対象となる機能（ネットワークアクセスなど）によって異なる。また、項目名称は端末実装依存となる。

2.4.2.13. 履歴管理

一部機種において､S!アプリの起動要求履歴の管理を行うことができる｡本章では、履歴管理について説明する。

2.4.2.13.1. 履歴表示

S!アプリ起動要求を履歴情報として確認することができる。履歴情報が、最大登録件数に達した場合は、古い履歴から削除を行い、常に最新の情報が更新される。なお、最大登録件数は端末実装依存となる。

① 時刻設定起動履歴管理情報

1 つの S!アプリ起動要求に対して、端末側が管理する履歴管理情報は以下の通りとなる。

表 2.4.2.15.1-1. 時刻設定起動要求履歴管理情報

項目	説明
S!アプリ起動要求受信時間	アラーム発生時間
S!アプリ名	S!アプリ名(MIDlet-Name)
ベンダー名	ベンダー名(MIDlet-Vendor)

② Bluetooth 起動要求履歴管理情報

1 つの Bluetooth 起動要求に対して、端末が管理する履歴管理情報は以下の通りである。

表 2.4.2.15.1-2. Bluetooth 起動要求履歴管理情報

項目	説明
Bluetooth 起動要求受信時間	Bluetooth 起動要求(S!アプリ開始要求)を受信した時間(分単位)
送信者情報	送信者名
S!アプリ名	S!アプリ名(MIDlet-Name)
メッセージ	コメント

※ S!アプリ名を取得できない際は､S!アプリ名を取得できない旨の情報を表示する。
(例：「表示できません」「ー」)

2.4.2.13.2. 履歴情報からのS!アプリ起動

履歴情報画面から当該S!アプリの起動を行うことができる。また、起動対象のS!アプリが一時停止中のS!アプリと同一の場合、一時停止中のS!アプリの再開を行う。なお、起動対象のS!アプリが存在しない場合は、エラー通知を行う。
また、履歴情報からのS!アプリ起動は、通常のS!アプリ起動と同様の扱いとなり、jscl.system.wakeupmodeは通常起動となる。