

DoJa-4.1 API iα ppli Development Kit ユーザーズガイド

平成 18 年 5 月 12 日
株式会社 N T T ドコモ

改版履歴

版数	項目	種別	内容	
1.00	-	新規	-	2005年8月 FOMA 902i向け 制定 <iα ppli Development Kit for DoJa-4.1(FOMA)>
1.01	-	変更	改版内容	JBlend 部分の差し替えに伴う修正
			改版理由	JBlend 改版／不具合修正
1.02	-	変更	改版内容	GET での HTTP 通信時に Content-Length ヘッダが付与されてしまう不具合を修正
			改版理由	不具合修正
1.03	-	変更	改版内容	JBlend 部分の差し替えに伴う修正
			改版理由	JBlend 改版／不具合修正
	6.8	追加	-	CPBindingAuth 署名付与ツールを追加
1.04	-	変更	改版内容	Image#createImage における生成可能サイズ制限(240x240)を撤廃し、ヒープサイズが許容する範囲内であれば Image の生成が可能となるように修正
			改版理由	機能改善
1.05	-	変更	改版内容	・ MessageAgent#delete 実行時に正常にメッセージが削除されない事がある不具合を修正 ・ Graphics2#setRenderMode 使用時に透過 GIF を描画した場合 透過指定が無視される不具合を修正
			改版理由	不具合修正
1.10	-	変更	改版内容	JBlend 部分の差し替えに伴う修正
			改版理由	JBlend 改版／不具合修正
1.11	-	変更	改版内容	サウンド再生時の不具合を修正
			改版理由	不具合修正
2.00	-	変更	-	2006年2月 FOMA 902i S向け 制定
2.00	-	変更	改版内容	・ Sun ONE Studio 4, CE モジュールの削除 ・ NetBeans IDE モジュールの追加
			改版理由	Sun ONE Studio 4, CE は、Sun にてサポート対象外のため、新たに NetBeans IDE に対応
2.01	-	変更	改版内容	OnlineFelica で HTTP 接続エラーになる不具合を修正
			改版理由	不具合修正

目 次

第 1 章. はじめに	8
1.1. iappli Development Kit for DoJa-4.1(FOMA) の動作環境	8
1.1.1. iappliTool について	8
1.2. エミュレータの概要	9
1.2.1. エミュレータの利点	9
1.2.2. 統合開発環境の利用	9
1.2.3. エミュレーションの範囲	10
1.3. エミュレータによる i アプリの開発	13
第 2 章. インストール	15
2.1. インストールする前に	15
2.2. インストール手順	16
2.3. アンインストール手順	24
第 3 章. エミュレータの起動	25
第 4 章. i アプリ	26
4.1. i アプリの作成から起動まで	26
4.2. プロジェクトの新規作成	27
4.2.1. ソースファイルの作成	29
4.2.2. ソースコードの修正	33
4.2.3. ソースエディタの設定	35
4.3. プロジェクトのビルドと通常起動	36
4.3.1. ダウンロード用HTMLファイル	39
4.3.2. i モード携帯電話で i アプリ実行	40
4.4. プロジェクトの読み込み	41
4.5. プロジェクトの設定	43
4.6. ファイル配置カスタマイズ	45
4.7. アプリケーションディスクリプタファイル (ADF) の編集	46
4.7.1. LastModified項目の表記方法	56
4.8. 待ち受け起動	57
4.8.1. 待ち受けアプリケーション	57
4.8.2. 待ち受け起動手順	59
4.9. 受信フォルダ起動	62
4.10. 送信フォルダ起動	63
4.11. 未送信フォルダ起動	64
4.12. 初回起動	65

4.1 3. タイマ起動	66
4.1 4. 外部機器起動	67
4.1 5. 即起動	68
4.1 5.1. 即起動手順	68
4.1 6. メーラ起動	69
4.1 6.1. メーラ起動手順	69
4.1 7. ブラウザ起動	74
4.1 7.1. ブラウザ起動手順	74
4.1 8. コードリーダー起動	79
4.1 8.1. コードリーダー起動手順	79
4.1 8.2. CodeReaderの実行手順	83
4.1 9. 各種起動方法による画面遷移	84
第5章. エミュレータの設定	86
5.1. ネットワークの設定	86
5.1.1. プロキシサーバの設定	88
5.2. トラストド動作の設定	89
5.3. メールアドレスの設定	91
5.4. リモート制御の設定	93
5.5. リモート制御のパラメータ領域の削除	94
5.6. サウンド設定	95
5.7. ヒープサイズの設定	97
5.8. UART用COMポートの設定	98
5.9. 電話帳の編集	99
5.1 0. グループの編集	101
5.1 1. ブックマーク一覧	102
5.1 2. スケジュール一覧	103
5.1 3. 送受信メールの編集	104
5.1 3.1. ショートメールの扱いについて	107
5.1 3.2. メールを送受信エミュレート機能	109
5.1 4. ビルドの設定	110
5.1 4.1. スクリプトの実行	110
5.1 4.2. ターゲットプロファイルの設定	112
5.1 5. ショートカットキーの設定	113
5.1 6. ヘルプの設定	116
5.1 7. エミュレータ性能設定	118
5.1 8. コンパイラ (javac) の選択	120

5.19. 起動時にプロジェクトを読み込む	121
5.20. .jam に関連付ける	122
5.21. コンソールクリア	122
5.22. エミュレーション端末の選択	123
5.23. エミュレーション端末の操作	125
5.23.1. 端末子ウィンドウの操作	129
5.23.2. 3Dサウンド表示	131
第6章. ツール	133
6.1. スクラッチパッド管理ツール	133
6.1.1. スクラッチパッドファイル	133
6.1.2. スクラッチパッド管理ツールの起動	133
6.1.3. スクラッチパッドで J a r ファイルを扱う場合の注意事項	137
6.2. スクラッチパッドを全て初期化	137
6.3. 仮想FeliCaデータファイルビューワ	138
6.3.1. 仮想FeliCaデータファイル	138
6.3.2. 仮想FeliCaデータファイルビューワの起動	138
6.3.3. 仮想FeliCaデータファイルビューワの表示色	140
6.3.4. ブロックの構成	140
6.4. Javadoc 作成ツール	141
6.5. A D F チェックツール	144
6.6. スクリーンショット	146
6.7. i アプリライブラリ	147
6.7.1. i アプリライブラリの作成	147
6.7.2. i アプリライブラリの利用	153
6.8. CPBindingAuth署名付与ツール	154
6.8.1. ドコモが発行したユーザ証明書を使って署名する方法	155
6.8.2. ユーザが発行した証明書を使って署名する方法	156
第7章. エミュレータの終了	157
第8章. i アプリ連携	157
第9章. ファイルの記述方法	159
9.1. メーラ起動ファイルの記述方法	161
9.2. バウンダリデータファイル	161
9.3. メールアドレスファイル	161
9.4. ブラウザ起動HTMLファイルの記述方法	162
9.5. 着信履歴ファイル	162
9.6. 「アプリケーション名.properties」ファイル	163
9.7. メールファイル	164

9.8. 電話帳ファイル	164
9.9. 電話帳グループファイル	165
9.10. マルチメディアファイル	165
9.11. ブックマーク登録ファイル	165
9.12. スケジュール登録ファイル	165
9.13. コード認識結果ファイル	166
9.14. 仮想FeliCaデータファイル	168
9.15. スクラッチパッドデータファイル	169
9.16. LED表示点灯パターン登録ファイル	169
9.17. サウンド設定ファイル	169
9.18. ストレージデバイス ファイルパス	170
9.19. トルカファイル	170
9.20. ファイルの記述具体例	171
第10章. 証明書の管理	172
第11章. OBEXの利用	174
第12章. 再配布不可識別子	174
第13章. NetBeans IDE の利用	175
13.1. はじめに	175
13.1.1. NetBeans IDEについて	175
13.2. 「NetBeans IDE 4.1 モジュール for iappli」 のインストール	175
13.2.1. インストール	175
13.2.2. NetBeans IDE にモジュールを組み込む	176
13.3. 新規アプリケーションの作成	180
13.3.1. プロジェクトの作成	180
13.3.2. ソースファイルの作成	182
13.3.3. アプリケーションディスクリプタファイル (ADF) の設定	184
13.3.4. トラステッド動作の設定	185
13.3.5. アプリケーションの構築と実行	186
13.4. エミュレータの設定	188
13.4.1. デバイスの設定	188
13.4.2. ネットワークの設定	188
13.4.3. NetBeans IDE からモジュールを削除する	189
第14章. Eclipse の利用	190
14.1. はじめに	190
14.1.1. Eclipseについて	190
14.2. 「Eclipseプラグイン for iappli」 のインストール	190
14.2.1. 始める前に	190

1 4.2.2. インストール	191
1 4.3. 新規アプリケーションの作成	192
1 4.3.1. DoJaを利用する前に	192
1 4.3.2. プロジェクトメニューのカスタマイズ	193
1 4.3.3. プロジェクトの作成	194
1 4.3.4. ソースファイルの作成	197
1 4.3.5. プロジェクト設定からのソースファイルの作成	200
1 4.3.6. アプリケーションディスクリプタファイル (ADF) 設定	201
1 4.3.7. トラストド動作の設定	202
1 4.3.8. アプリケーションの構築と実行	203
1 4.3.9. プロジェクトの設定	205
1 4.3.10. リモート制御の設定	206
1 4.3.11. ファイル配置カスタマイズ	207
1 4.3.12. i アプリライブラリの作成	208
1 4.3.13. i アプリライブラリの利用	209
1 4.3.14. デバッグの実行	212
1 4.4. エミュレータの設定	220
1 4.4.1. インストール先の設定	220
1 4.4.2. ヒープサイズの設定	220
1 4.4.3. ターゲットプロファイルの変更	221
1 4.4.4. ネットワークの設定	224
1 4.4.5. メールアドレスの設定	225
1 4.4.6. 電話帳の編集	226
1 4.4.7. グループの編集	228
1 4.4.8. 送受信メールの編集	229
1 4.4.9. スケジュール一覧	231
1 4.4.10. ブックマーク一覧	232
1 4.4.11. サウンド設定	233
1 4.5. 仮想FeliCaデータファイルビューワ	234
1 4.5.1. 仮想FeliCaデータファイルビューワの起動	234
1 4.5.2. 仮想FeliCaデータファイルビューワの表示色	236
1 4.5.3. ブロックの構成	236
1 4.6. スクラッチパッド	237
1 4.6.1. スクラッチパッド管理ツール	237
1 4.6.2. スクラッチパッド管理ツールの起動	237
1 4.6.3. スクラッチパッドを全て初期化する	239
1 4.6.4. スクラッチパッドでJ a r ファイルを扱う場合の注意事項	239
付録.1 エミュレーション端末の設定	240
1. コメント行	240
2. エミュレータウィンドウの設定	240
3. LCD領域の設定	240
4. フォントの設定	241
5. ボタン設定	241
6. ピクチャーライトの設定	241

7. サウンドライブラリの設定.....	242
8. 例（デフォルト）	243
付録.2 端末子ウィンドウの設定	244
1. コメント行.....	244
2. 端末子ウィンドウの設定.....	244
3. ボタン設定.....	244
4. 例（デフォルト）	245

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、NetBeans IDE、Java、J2SE SDK、JDK、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。Windows、DirectX は、米国およびその他の国における米国 Microsoft Corporation の登録商標です。

FeliCa は、ソニー株式会社が開発した非接触 IC カードの技術方式です。FeliCa は、ソニー株式会社の登録商標です。

QR コードは、株式会社デンソーウェーブの登録商標です。

本製品は、株式会社エイチアイの Mascot Capsule Engine Micro3D Edition を搭載しています。Mascot Capsule は、株式会社エイチアイの日本国における登録商標です。

本製品は、株式会社アプリックスの JBlend TM[micro] を搭載しています。JBlend は、株式会社アプリックスの商標または登録商標です。

CPBindingAuth署名付与ツールは、Aicrypto を搭載しています。
Aicrypto :

This product includes software developed by Akira Iwata
Laboratory, Nagoya Institute of Technology in Japan
(<http://mars.elcom.nitech.ac.jp/>).

「i モード」、「i アプリ／アイアプリ」、「i-α ppli」ロゴ、「DoJa」、「i メロディ／アイメロディ」、「FOMA」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。

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 1997-2006 Aplix Corporation. All Rights Reserved.

Copyright 2000-2006 NTT DoCoMo, Inc. 2-11-1 Nagata-cho, Chiyoda-ku, Tokyo, 100-6150, Japan.

第1章. はじめに

本書では、「iα ppli Development Kit for DoJa-4.1(FOMA)」のインストール、設定、および使用方法について説明します。

ここで述べる「iα ppli Development Kit for DoJa-4.1(FOMA)」は、iモード携帯電話で動作する Java アプリケーション（i アプリ）の開発をサポートします。

本書を理解するためには、Java プログラミング、CLDC、および NTT ドコモが提供する DoJa-4.1 API、DoJa-4.1 オプション・拡張 API の知識が必要になります。

i アプリでできることについては、別途、i アプリコンテンツ開発ガイド、もしくは、API リファレンスを参照してください。

1.1. iα ppli Development Kit for DoJa-4.1(FOMA) の動作環境

「iα ppli Development Kit for DoJa-4.1(FOMA)」は、以下の環境での動作を保証します。

- Windows 2000 (SP4) / Windows XP (SP1 | SP2)
- Java^(TM) 2 SDK, Standard Edition 1.4.2_04 (J2SE SDK)
- Java^(TM) 2 Runtime Environment, Standard Edition 1.4.2_04
- DirectX 8.1 以上（サウンドライブラリ利用時）

これらは保証できる範囲であり、動作する環境はこの限りではありません。

1.1.1. iα ppliTool について

「iα ppli Development Kit for DoJa-4.1(FOMA)」をインストールすると「iα ppliTool」という i アプリのエミュレータが実行できるようになります。

「iα ppliTool」（以降、エミュレータと呼びます）の起動方法とその詳細については、「第3章. エミュレータの起動」で説明します。

1.2. エミュレータの概要

エミュレータ (iαppliTool) は、i モード携帯電話でできる様々な動作を PC 上で実現することによって i アプリの開発をサポートします。このエミュレータを使用すれば、i モード携帯電話で動作する i アプリを PC 上で検証することができます。

1.2.1. エミュレータの利点

エミュレータ (iαppliTool) には、次の利点があります：

- 作成した i アプリの動作を検証できます。
- i アプリのライブラリを作成できます（「6.7. i アプリライブラリ」参照）。
- 作成した i アプリがそのまま i モード携帯電話で実行できます。
（Web 上にあれば、簡単にダウンロードして実行できます。）
- 様々な携帯電話の機能をエミュレーションできます。

1.2.2. 統合開発環境の利用

統合開発環境 (IDE) は、アプリケーションの作成に必要な 編集、コンパイル、実行、デバッグ の機能が統合されており、アプリケーションの生産性を向上させることができます。「iαppli Development Kit for DoJa-4.1(FOMA)」では、NetBeans IDE 4.1 および Eclipse 2.1/3.0 で i アプリを開発するために必要な拡張モジュールを提供します。これらの統合開発環境で、i アプリの編集、コンパイル、ビルド、実行、デバッグまでを一貫して行うことができます。

詳細については「第 1 3 章. NetBeans IDE の利用」、「第 1 4 章. Eclipse の利用」を参照してください。

- NetBeans : <http://www.netbeans.org/downloads/> → NetBeans IDE Release archive
(**Mobility Pack は必要ありません。**)
- Eclipse : <http://www.eclipse.org/downloads/>

1.2.3. エミュレーションの範囲

ここでは、エミュレータがエミュレーションできる i モード携帯電話機能の範囲について説明します。

i アプリの動作をテストするときに理解しておく内容ですので、まず i アプリを作成してみたい場合は、読み飛ばしても構いません。

1. アプリケーションマネージャの処理

エミュレータは、Java アプリケーションの実行環境を提供していますが、実際の i モード携帯電話で実行されるアプリケーションマネージャの動作をエミュレートする環境は提供していません。

アプリケーションマネージャに関する動作のうち、i モード携帯電話と異なる動作として次のものがあります：

- アプリケーション、スクラッチパッド、ADF のサイズに制限はありません。
- アプリケーションのダウンロードや更新の機能はありません。ただし、起動種別設定機能を使用することにより `IApplication.LAUNCHED_AFTER_DOWNLOAD` を発生させるテストが可能です。
- ネットワーク接続可否の設定機能はありません。
- タイマ起動を設定する機能はありません。ただし、起動種別設定機能を使用することにより `IApplication.LAUNCHED_FROM_TIMER` を発生させるテストが可能です。
- 実際にメールやメッセージを送受信する事はできません。
- 実際にマルチメディアに保存する事はできません。
- 実際に赤外線リモコンで赤外線を飛ばす事はできません。
- 実際に動画再生する事はできません。
- 実際にカメラ撮影、動画撮影をする事はできません。
- エミュレータは Windows 上で動作するため、実際の i モード携帯電話と多少動作が異なる場合があります。

2. アプリケーションディスクリプタファイル (ADF)

エミュレータでは、アプリケーションディスクリプタファイルにおいて実際に処理される項目と処理されない項目があります。しかし、全ての項目において値の検証をしています。そのため、アプリケーションとして設定する必要のない項目については、値を設定しないことを推奨します。アプリケーションディスクリプタファイルについての詳細は「4.7. アプリケーションディスクリプタファイル (ADF) の編集」を参照してください。

3. 携帯電話固有の機能

エミュレータでは、携帯電話固有機能のうちバックライトやバイブレータの状態表示など一部の機能をエミュレートします。しかし、携帯電話固有機能の多くの機能はエミュレートされません。

携帯電話固有機能に関する動作のうち、i モード携帯電話と異なる動作として次のものがあります：

- バックライトやバイブレータの状態はエミュレータ上のアイコン領域に表示されます。
- 音量やマナーモードを設定する機能はありません。
- 電源の ON/OFF や自動電源 ON/OFF の機能はありません。
- i メロディ、着信メロディ、着信画像を設定する機能はありません。
- ダイヤルロック等、ロック機能や保護機能はありません。
- TV電話機能はありません。

4. 待ち受けアプリケーション

エミュレータでは待ち受けアプリケーションのライフサイクルを実現していますが、待ち受けアプリケーションを登録するための設定機能は提供していません。

待ち受けアプリケーションに関する動作のうち、i モード携帯電話と異なる動作として次のものがあります：

- 待ち受け画面（待ち受け画像など）の設定/表示機能はありません。
- 待ち受けアプリケーションを i モード携帯電話に設定する機能はありません。
- 待ち受けアプリケーションが休眠状態の際に実行される省電力機能はエミュレートされません。

5. ネイティブアプリケーション連携

エミュレータでは、i モード携帯電話に搭載されたネイティブアプリケーションで、ブラウザ、メール、コードリーダー、音声通話の機能連携を提供しますが、外部機器との連携をエミュレートする環境は提供していません。

ネイティブアプリケーションとの連携に関する動作のうち、i モード携帯電話と異なる動作として次のものがあります：

- 外部機器からアプリケーションを起動する機能はありません。ただし、起動種別設定機能を使用することにより IApplication.LAUNCHED_FROM_EXT を発生させるテストが可能です。
- ブラウザから i アプリを起動する際、i モード通信可否の判定は行いません。また、以下の点に注意してください：

- ✧ <A>タグに ista 属性および ijam 属性があっても無視されます。また utn 属性が指定される場合も同様に無視されます。
- ✧ 1 つの HTML 内には、<OBJECT>タグと<A>タグをこの順序で対にして記述してください。
- ✧ 対応関係にある<OBJECT>タグと<A>タグとの間には、他の<OBJECT>タグを書かないでください。
- ✧ 1 つの HTML 内での各<A>タグにおける href 属性値は、異なる文字列として記述してください。
- ✧ <OBJECT>タグでは、data 属性だけが意味を持ち、id、type、declare 属性は意味を持ちません。
- ✧ 最低限、<OBJECT>タグ内においては、data 属性とその値（ADF の URL）、および、<A>タグ内においては、href 属性とその値を記述してください。

- メーラから i アプリを起動する際、以下の点に注意してください：
 - ✧ "--B:A"以降の記述については、規定のフォーマットに従っていない場合は、全て無視します。
 - ✧ メール本文は、HTML フォーマットに対応しています。
- 6. HTTP を使用したアプリケーション
 - エミュレータでは、HTTP1.1 を用いており、pragma: no-cache は、cache-control: no-cache を設定して HTTP 要求を発行します。
 - Connection: keep-alive は、常に設定されて要求を発行します。
- 7. FeliCa を使用したアプリケーション
 - エミュレータでは FeliCa のフリー領域にアクセスする API を利用できる機能を提供します。

1.3. エミュレータによるiアプリの開発

ここでは、エミュレータによる i アプリの開発サイクルについて説明します。

エミュレータを使用した場合、
i アプリの開発サイクルは、次のようなフェーズになります：

1. i アプリのプロジェクト作成
エミュレータで、i アプリのプロジェクトを作成します。
2. i アプリのプログラム (.java) を作成もしくは、修正
エミュレータでは、テンプレートを使って簡単なプログラムを作成できます。
3. i アプリで扱うリソース（画像や音楽、電話帳/メール等）を設定
4. アプリケーションディスクリプタファイル（ADF）を設定
5. プロジェクトのビルド
エミュレータでビルドを実行すると次の作業をまとめておこないます。
自動的に実行されるため、エミュレータを使って i アプリを作成する上では特に意識する必要はありません。
 - コンパイル
J2SE SDK コンパイラを使用して、ソースファイルをクラスファイルにコンパイルします。コンパイラは、CLDC や DoJa API に含まれ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で使用されていないか検証します。
 - バイトコードの検証
事前検証ツールを使用して、Java クラスの事前検証をすると共に、Java クラスの中で浮動小数点演算やファイナライズメソッドが使用されていないかどうかを検査します。
 - パッケージング
J2SE SDK の jar ツールを使用して、事前検証された全てのクラスファイルとリソースファイルを JAR 形式にパッケージングします。
 - アプリケーションディスクリプタファイル（ADF）の作成
アプリケーションディスクリプタファイルを作成します。これは、i アプリの情報を記述するテキストファイルです。
 - ダウンロード用 HTML の作成
作成した i アプリをダウンロードするためのもっともシンプルな HTML を作成します。
6. エミュレータで実行
エミュレータ上で i アプリを実行します。
もし、i アプリの動作に問題があれば、2. からやり直します。
7. i モード携帯電話で実行
作成した i アプリ一式を HTTP サーバ上に配置後、i モード携帯電話で i アプリをダウンロードして実行します。

上述の開発サイクルを図 1-1 に示します。

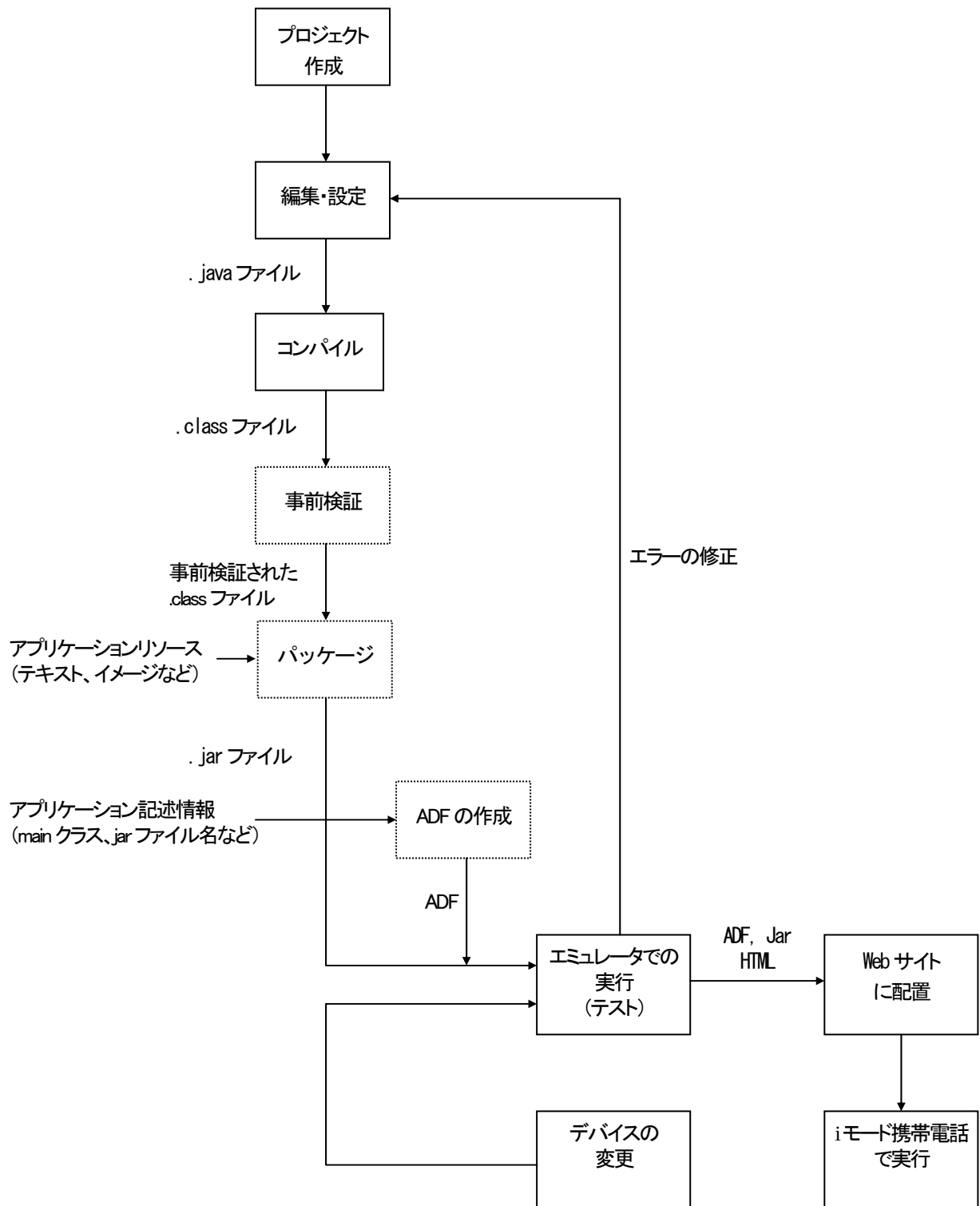


図 1-1 アプリケーション開発サイクル

第2章. インストール

2.1. インストールする前に

「iα ppli Development Kit for DoJa-4.1(FOMA)」をインストールする前に次のことを確認してください。

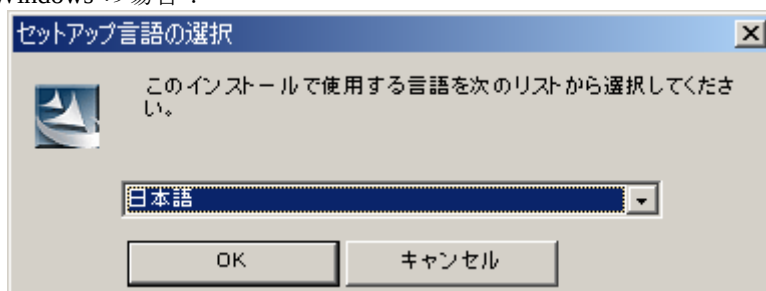
- ① 「iα ppli Development Kit for DoJa-4.1(FOMA)」を動作させるには、**J2SE SDK (JDK)** がインストールされている必要があります。もし、J2SE SDK がインストールされていない場合は、Sun のサイトから J2SE SDK を入手してインストールしてください。
- ② エミュレータでは、より携帯端末に近いサウンドを再生するために **DirectX (Direct Sound) 8.1 以上** を使っています。ただし、エミュレータのサウンド設定が『ビジュアルモード』に設定されている場合は使われません。もし、DirectXが入っていない場合は、インストール時にサウンド設定が『ビジュアルモード』に変更されます。携帯端末に近い『通常再生モード』でサウンドを鳴動させたい場合は、事前に DirectX を入手してインストールしてください。
 - ・ DirectX : <http://www.microsoft.com/japan/windows/directx/>
- ③ 「iα ppli Development Kit for DoJa-4.1(FOMA)」の以前のバージョンが既にインストールされている場合には、これをアンインストールしてください。アンインストールについては、「2.3. アンインストール手順」を参照してください。
- ④ Eclipse 統合開発環境を使う場合で、他のプロファイル (DoJa-4.0 や DoJa-3.5) と共存させたい時は、あらかじめ別の Eclipse 環境を用意しておいてください。
(例 : C:\Eclipse_DoJa-4.1)
- ⑤ NetBeans 統合開発環境を使う場合、JDK5.0 にバンドルされているもの(「NetBeans IDE + JDK 5.0」)がありますが、**JDK5.0 の環境では i アプリのコンパイルが正常に実行されません**。これは、i アプリのコンパイル時に 事前検証 (つまり CLDC) が JDK 5.0 に対応していないためです。
この場合、別途、JDK1.4.2 をインストールして、環境変数の JAVA_HOME は JDK1.4.2 を参照するようにしてください。

2.2. インストール手順

「iα ppli Development Kit for DoJa-4.1(FOMA)」をインストールするには：

1. 「DISK1」ディレクトリの下にある「setup.exe」をダブルクリックします。
2. 最初にインストールする言語を聞いてきますので言語（日本語）を選択します。
英語版の Windows の場合、英語で表示されます。

日本語版 Windows の場合：



英語版 Windows の場合：

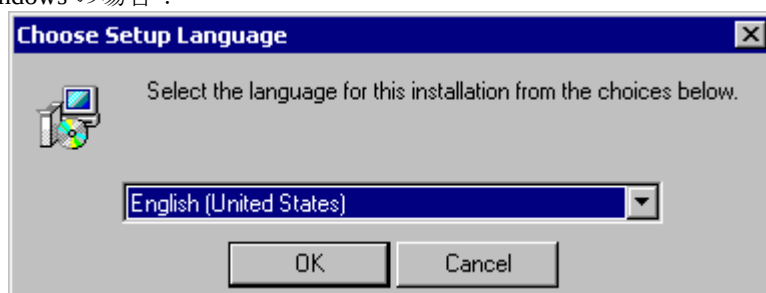


図 2-1 セットアップ言語の選択画面

3. ウィザードに従い「次へ」を押します。

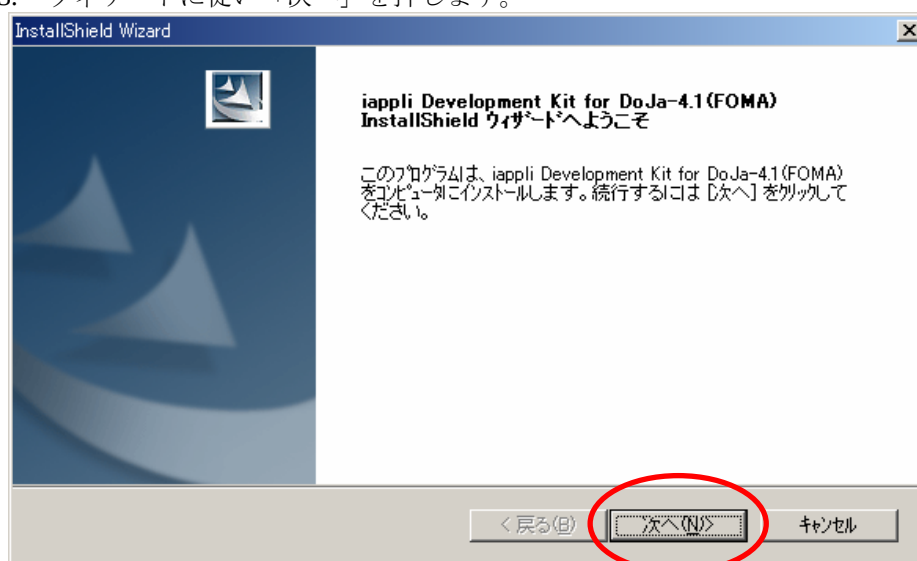


図 2-2 ようこそ画面

4. 「使用許諾契約」が表示されるので、よく読んでから「はい」を押します。

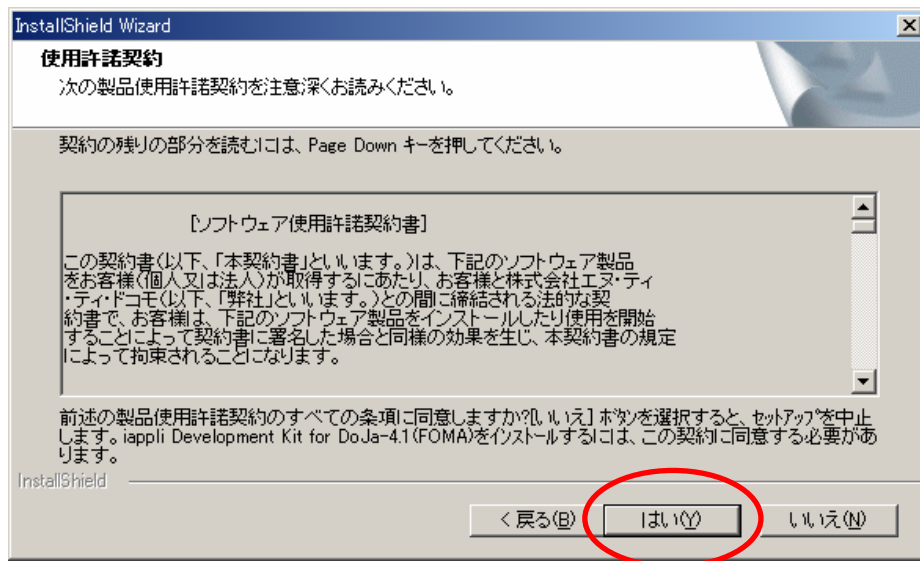


図 2-3 使用許諾契約画面

5. 「Binary Code License Agreement」が表示されるので、「はい」を押します。

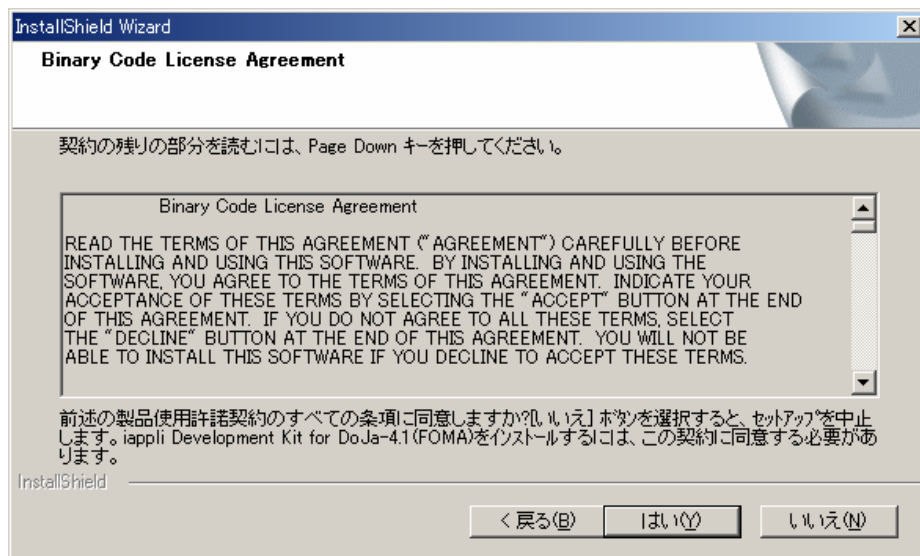


図 2-4 「Binary Code License Agreement」

6. インストール先のフォルダを指定^[1]して、「次へ」を押します。

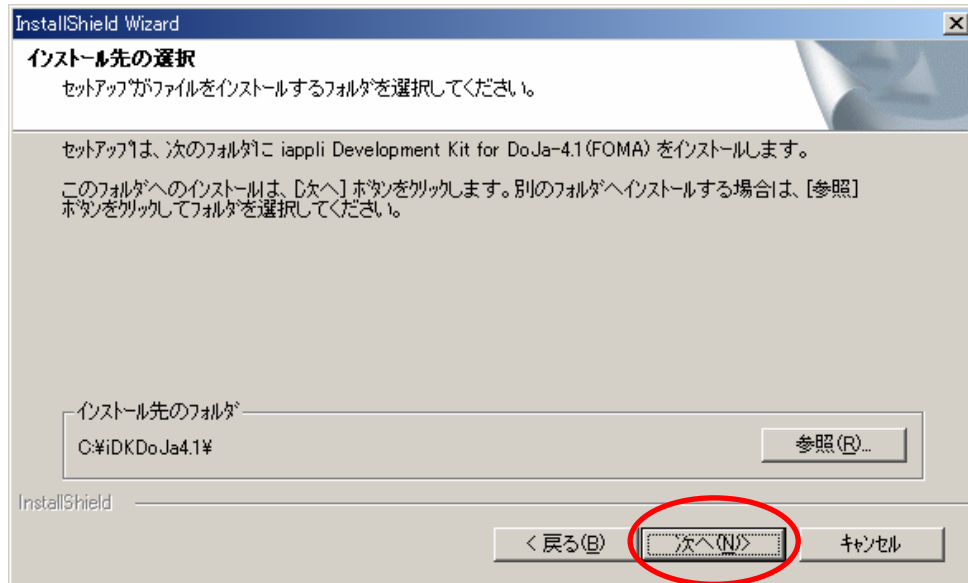


図 2-5 インストール先の選択画面

7. セットアップの種類を選択して、「次へ」を押します。

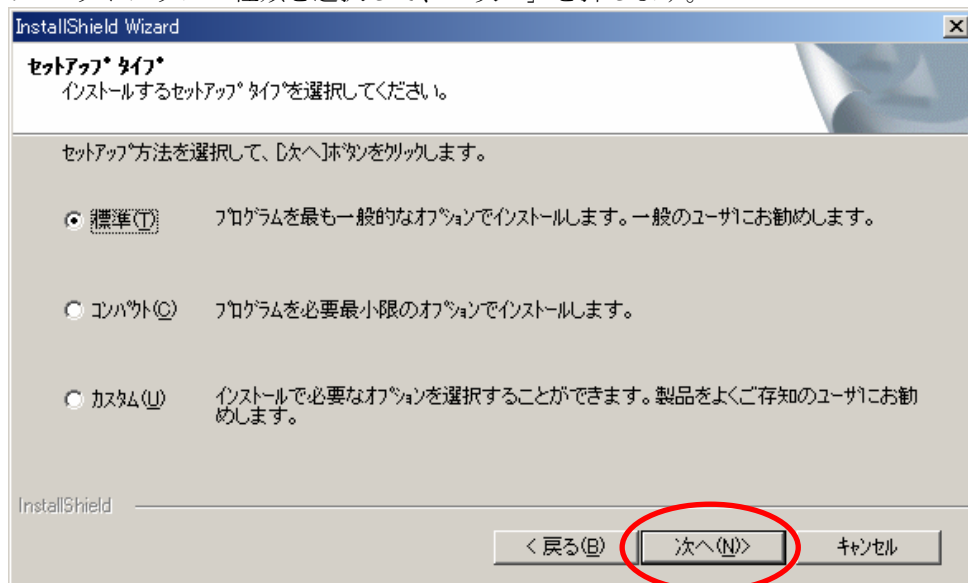


図 2-6 セットアップタイプの選択画面

- ・標準 : プログラムとサンプルアプリをインストールします。
- ・コンパクト : プログラムのみをインストールします。
- ・カスタム : 何をインストールするかを選択することができます。
もし、NetBeans IDE 4.1 や Eclipse を使ってアプリケーションを作成したい場合には、こちらを選択してください。

^[1] インストールのディレクトリを指定する際「スペース（空白）」及び「2 バイト文字（全角文字）」を含まないようにしてください。これらの文字が含まれているとディレクトリを正しく扱うことができません。

8. 手順7で、「カスタム」以外を選択した場合は、手順11に進んでください。
「カスタム」の場合、インストールしたい機能を選択して、「次へ」を押します。

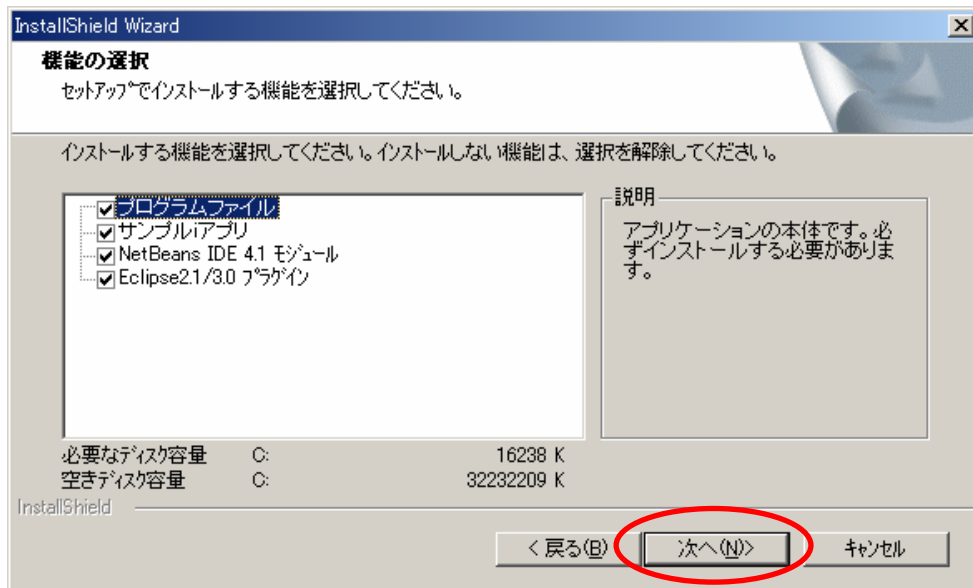


図 2-7 機能の選択画面

- 「NetBeans IDE 4.1 モジュール」を選択すると、NetBeans IDE 4.1 を使って i アプリを開発する事ができます。
ここでインストールされた **NetBeans IDE 4.1 モジュール**は、自動的に組み込まれませんので、**NetBeans IDE** の機能「**アップデートセンター**」にて組み込んでください（「第13章. **NetBeans IDE** の利用」参照）。
 - 「Eclipse2.1/3.0 プラグイン」を選択すると、Eclipse2.1 もしくは、Eclipse3.0 を使って i アプリを開発する事ができます。
ただし、あらかじめ Eclipse がインストールされている必要があります。
注意：DoJa-4.1 以前のプラグインがインストールされている場合に、
DoJa-4.1 以前と共存させたい場合は、別の Eclipse 環境を用意してください。
9. 手順8で「NetBeans IDE 4.1 モジュール」を選択した場合、注意事項が表示されますので、OKを押します。

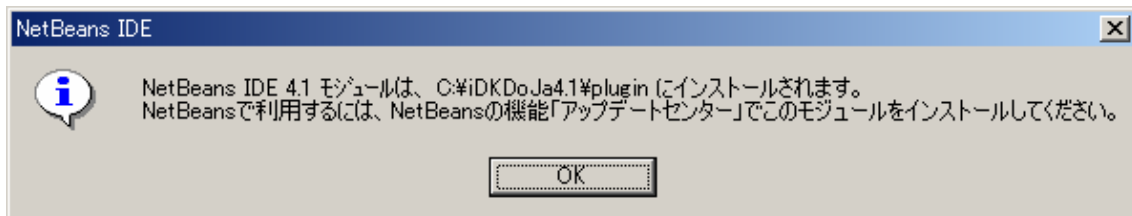


図 2-8 NetBeans IDE モジュールの注意事項

10. 手順8で「Eclipse2.1/3.0 プラグイン」を選択した場合、Eclipseのインストール先を聞いてきますので、インストール先を指定してください。

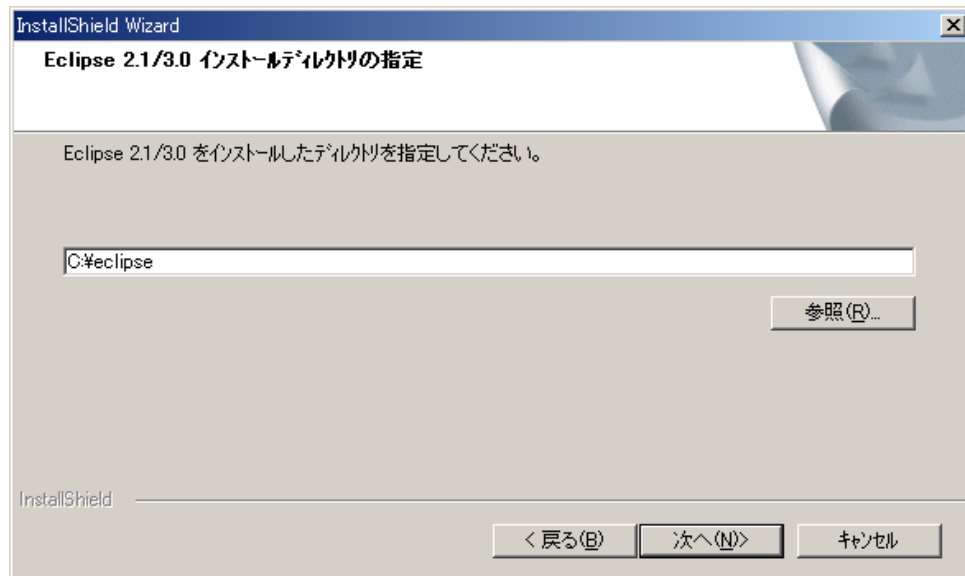


図 2-9 Eclipse プラグインインストール先設定画面

11. インストールされるファイルの確認画面が表示されるので「次へ」を押します。

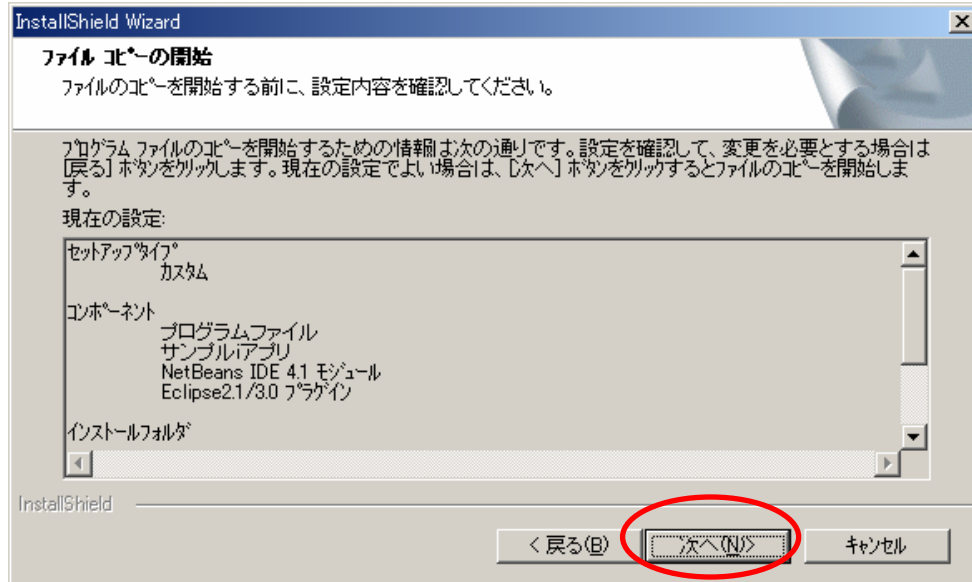


図 2-10 ファイルのコピー開始画面

12. Windows にプロキシサーバ情報が設定されている場合、エミュレータのネットワーク設定にプロキシサーバ情報を登録するかどうかの確認画面が表示されますので、プロキシサーバ情報を設定してもよろしければ、「はい」を押します。

Windows のプロキシサーバの情報を利用している場合：

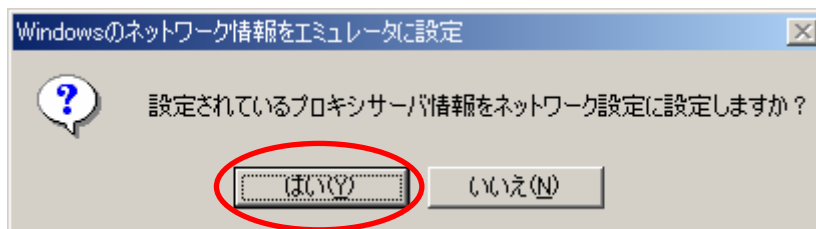


図 2-1 1 プロキシ情報の設定画面

Windows のプロキシサーバの情報を利用していない場合：

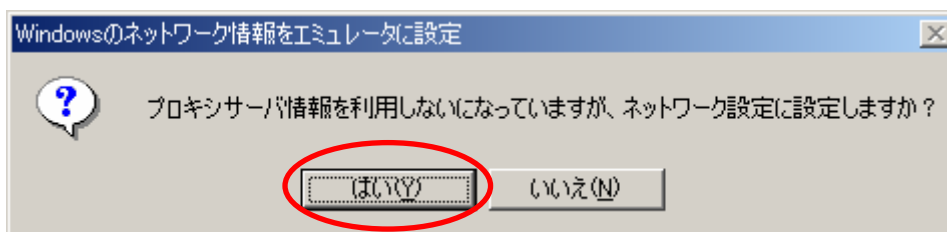


図 2-1 2 プロキシ情報の設定画面

13. i アプリ（.jam ファイル）をエミュレータに関連付けるかどうかを問い合わせるので、i アプリ（.jam ファイル）からエミュレータが起動できるようにしたい場合は、「はい」を押します。
関連付けをしない場合は、エミュレータを起動して i アプリを読み込むという手順になります。
拡張子の関連付けは、後からでも変更できます（「5.2 0. .jam に関連付ける」参照）。

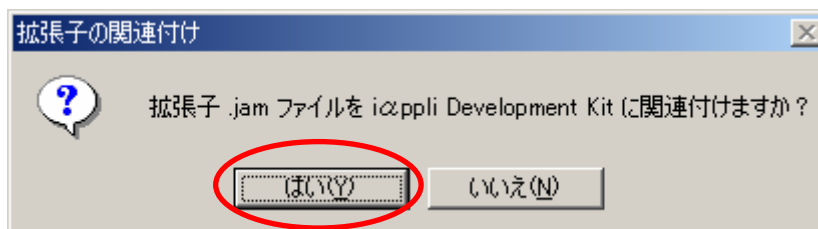


図 2-1 3 .jam ファイルの関連付け設定画面

14. インストールが実行され、完了画面が表示されたら「完了」を押します。

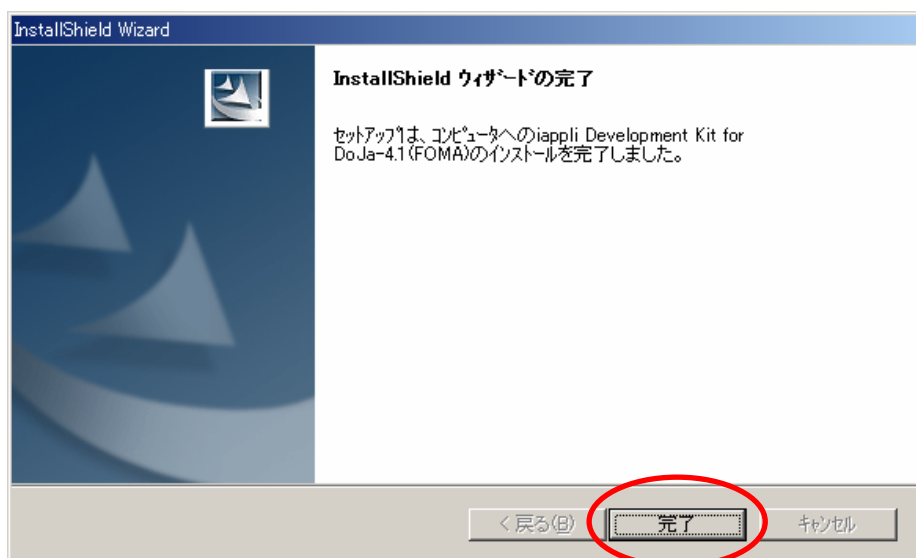


図 2-1 4 インストール完了画面

15. 日本語版Windowsで「iα ppli Development Kit for DoJa-4.1(FOMA)」をインストールした場合、i モード携帯電話と同じ「絵文字」を登録するための設定画面が表示されます。Windowsの外字領域（何も使われていないところ）に絵文字を設定してもよろしければ、そのまま「設定」ボタンを押します（図 2-1 5）。
もし、後から登録したい場合は、「iα ppli Development Kit for DoJa-4.1(FOMA)」をインストールしたディレクトリの「gaiji」ディレクトリにある「ChangeEUDC.exe」を実行してください。

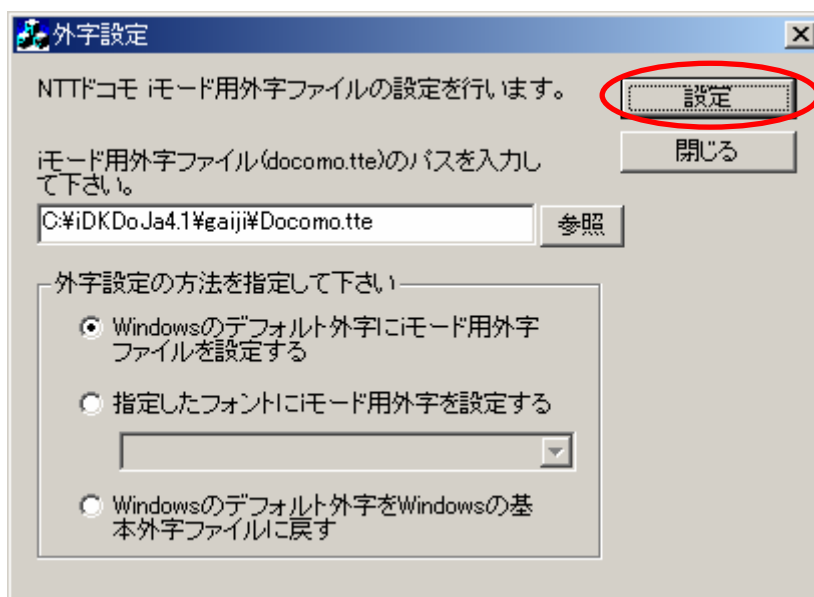


図 2-1 5 「ChangeEUDC.exe」の実行（外字設定画面）

16. 設定終了の確認画面が表示されたら、「OK」を押します（図 2-1 6）。

17. 外字設定画面の「閉じる」を押します（図 2-1 5）。



図 2-1 6 絵文字（外字）設定終了の確認

一度、インストールした絵文字（外字）をアンインストールするには、図 2-1 5 の「外字設定の方法」の一番下「Windowsのデフォルト外字をWindowsの基本外字ファイルに戻す」を選択して「設定」ボタンを押します。

18. i アプリオプションAPIと呼ばれる各メーカー機種固有のAPIを使う場合は、各メーカーのサイトからライブラリ（DLL）を取得します。

取得したライブラリは、「C:\iDKDoJa4.1」にインストールした場合、「C:\iDKDoJa4.1\bin」の下に置くだけで、i アプリオプションAPIを利用したアプリケーションを作成してエミュレーションできるようになります。

ライブラリに関する詳しい情報は、各メーカーのi アプリ関連サイトをご参照ください。各メーカーのi アプリ関連サイトへは、NTTドコモのページからたどることができます。

19. ヘルプからAPIリファレンスを参照したい場合は、別途、APIリファレンスをhttp://www.nttdocomo.co.jp/p_s/imode/make/ から取得し、インストールしたディレクトリのdocの下に置いてください。
（インストールしたディレクトリが「C:\iDKDoJa4.1」の場合、「C:\iDKDoJa4.1\doc」の下です。）

本書では、

「iα ppli Development Kit for DoJa-4.1(FOMA)」を「C:\iDKDoJa4.1」にインストールした場合として説明します。

ワンポイント

インストール後、「iα ppli Development Kit for DoJa-4.1(FOMA)」のヘルプメニューから操作ガイド（本書：ユーザーズガイド）とリリースノート参照できるようになります（「5.1 6. ヘルプの設定」参照）。
もし、インストール後、操作ガイドおよびリリースノートが参照できない場合は、インストールしたディレクトリの直下にこれらのファイル（操作ガイド、リリースノート）を置いてください。

2.3. アンインストール手順

「iαppli Development Kit for DoJa-4.1(FOMA)」をアンインストールするには：

1. Windowsの『スタート』→『プログラム』→『iαppli Development Kit for DoJa-4.1(FOMA)』→『Remove iαppli Development Kit for DoJa-4.1(FOMA)』を実行します（図 2-17）。
（「アプリケーションの追加と削除」から削除しても構いません。）

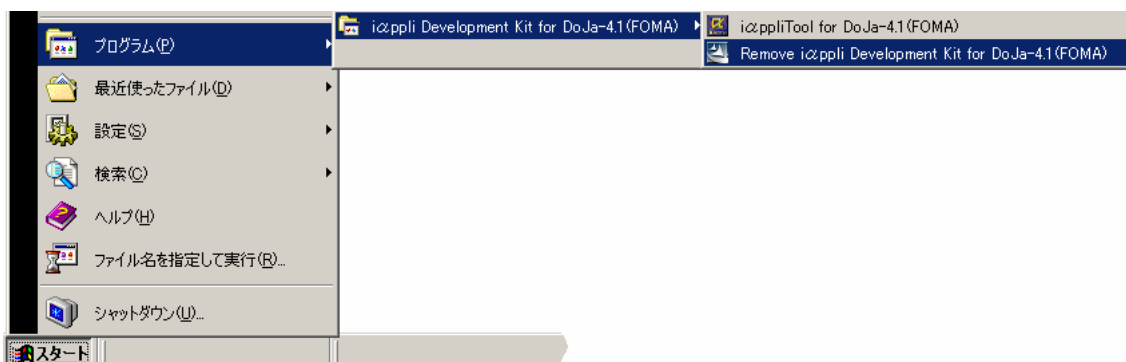


図 2-17 アンインストールの実行

2. 削除の確認画面が表示されるので、「OK」を押します（図 2-18）。

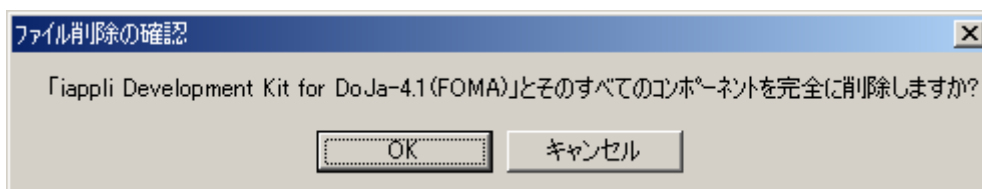


図 2-18 ファイル削除の確認画面

3. アンインストールを実行してもインストール先に作成された i アプリやデータは、そのまま残ります。
もし、それらも不要の場合は、インストール先のディレクトリ
「C:\iDKDoJa4.1」を直接削除してください。

第3章. エミュレータの起動

エミュレータ (iα ppliTool) を起動するには、Windowsの『スタート』→『プログラム』→『iα ppli Development Kit for DoJa-4.1(FOMA)』→『iα ppliTool for DoJa-4.1(FOMA)』を実行します。^[2]

エミュレータ (iα ppliTool) が起動されると、図 3-2、図 3-1 に示す最初の画面が表示されます。

図 3-2 は、「エミュレータ (iα ppliTool)」画面であり、i アプリのビルドや実行、携帯端末の設定などが行えます。

図 3-1 は、「エミュレーション端末」画面であり、ここで i アプリを実行します。この画面がエミュレータの待ち受け画面となります。

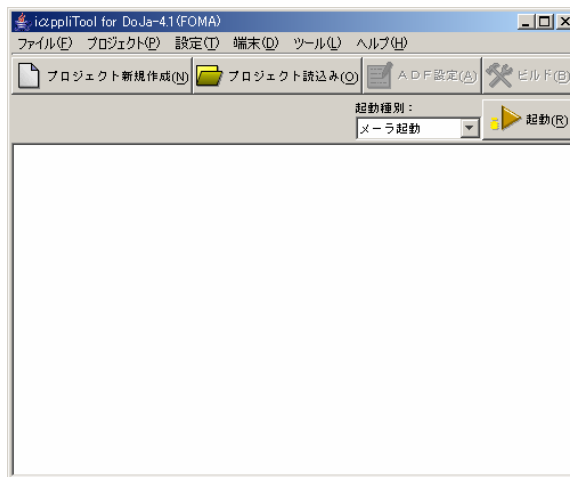


図 3-2 エミュレータ (iα ppliTool) 画面



図 3-1 エミュレーション端末

^[2] エミュレータでプロジェクトのビルド（「4.3. プロジェクトのビルドと通常起動」参照）を実行したとき、エミュレータが J2SE SDK を見つけられない場合があります。この場合、「環境変数 JDK_HOME に J2SE SDK をインストールしたディレクトリを設定してください」というメッセージが表示されますので、まず環境変数 JDK_HOME に J2SE SDK をインストールしたディレクトリが正しく設定されているかを確認してください。

第4章. iアプリ

i アプリは、i モード携帯電話の Java 環境で動作するアプリケーションです。

i アプリを i モード携帯電話で動作させるには、最低でも次のものが必要になります：

- i アプリの Jar ファイル
- アプリケーションディスクリプタファイル (ADF)
- i アプリをダウンロードするための HTML ファイル
(詳しくは、別途、「i アプリコンテンツ開発ガイド」を参照してください。)

これらは、すべてエミュレータで簡単に作成することができます。

この章では、i アプリの作成から実際に動かすところまでを説明します。

4.1. iアプリの作成から起動まで

エミュレータでは、i アプリを「プロジェクト」という単位で作成します。「プロジェクト」とは、1つの i アプリに対するファイルの集合です。このプロジェクトには、ソースコードから最終的な JAR ファイルやアプリケーションディスクリプタファイル (ADF)、画像、音楽のリソースに至る全てのファイルが含まれています。

エミュレータでは、i アプリの作成から起動までを次の手順で行います：

- ① 新しいプロジェクトの作成（「4.2. プロジェクトの新規作成」を参照）もしくは、既存のプロジェクトの読み込み（「4.4. プロジェクトの読み込み」を参照）
- ② プロジェクトの設定（「4.5. プロジェクトの設定」を参照）
- ③ i アプリの情報が記述された アプリケーションディスクリプタファイル (ADF) の設定（「4.7. アプリケーションディスクリプタファイル (ADF) の編集」を参照）
- ④ プロジェクトのビルドと通常起動（「4.3. プロジェクトのビルドと通常起動」を参照）

i アプリでは、スクラッチパッドという永続的な記憶領域を確保する事ができます。

このスクラッチパッドに i アプリで動作させた情報を保存しておくことで、次回、起動した時にその情報を復元させることができます。

スクラッチパッドは、各 i アプリのアプリケーションディスクリプタファイル (ADF) の SPSIZE を指定すると作成されます。

ワンポイント

i アプリの Jar ファイルは、ダウンロードが可能なサイズが限られています。
そのため、あまり大きな Jar ファイルは、エミュレータで実行できていても i モード携帯電話にダウンロードすることができません。

このような場合、i アプリで扱うリソースは、HTTP 経由で取得してスクラッチパッドに保存することで Jar ファイルに含めないようにします。

エミュレータに搭載されたサンプルでも、この手法を用いています。

4.2. プロジェクトの新規作成

新しいプロジェクトを作成するには：

1. メニューから「プロジェクト」→「新規作成...」を選択するか、「プロジェクト新規作成」ボタンを押します。

「プロジェクト新規作成画面」が表示されます（図 4-1）。



図 4-1 プロジェクト新規作成画面

2. プロジェクトの格納先（C:\iDKDoJa4.1\apps）を変更する場合は、「参照...」ボタンを押して格納先のディレクトリを選択します。
ただし、メーラ起動、ブラウザ起動、コードリーダー起動、i アプリ連携をおこなう場合は、デフォルトの格納先（「C:\iDKDoJa4.1\apps」）を変更しないでください。
3. 新しいプロジェクト名を入力し、「作成」ボタンを押します。^[3]

メイン画面のタイトルバーにプロジェクト名が表示されます（図 4-2）。

プロジェクトのソースコード（.java ファイル）とリソースファイル（アイコンなどのイメージファイルなど）をどこに置くべきか、また、スクラッチパッドの格納場所を示すメッセージが、コンソールに次の形式で表示されます：

ソースコードの場所: “C:\iDKDoJa4.1\apps\<プロジェクト名>\src”
リソースファイルの場所: “C:\iDKDoJa4.1\apps\<プロジェクト名>\res”
スクラッチパッドの場所: “C:\iDKDoJa4.1\apps\<プロジェクト名>\sp”

^[3] 新規プロジェクトを作成した際に、エミュレータは以下のディレクトリとファイルを作成します：

ソースコードの格納場所: C:\iDKDoJa4.1\apps\<プロジェクト名>\src
リソースファイルの格納場所: C:\iDKDoJa4.1\apps\<プロジェクト名>\res
スクラッチパッドの格納場所: C:\iDKDoJa4.1\apps\<プロジェクト名>\sp
JAR、ADF、HTML の格納場所: C:\iDKDoJa4.1\apps\<プロジェクト名>\bin

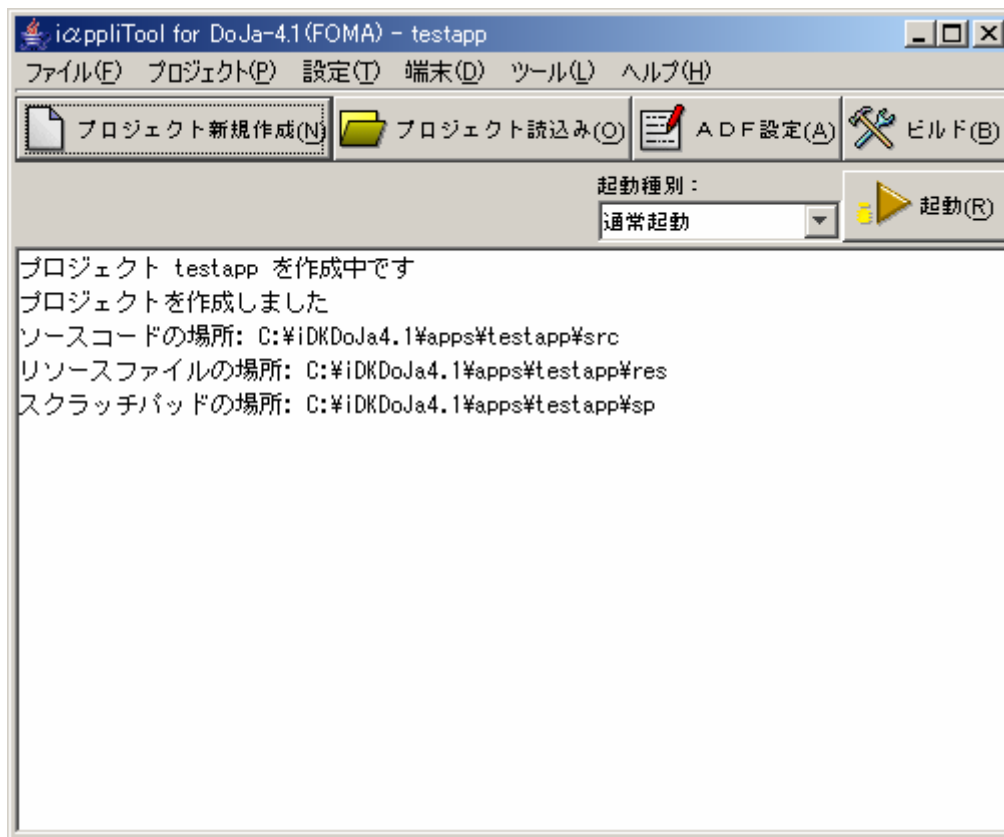


図 4-2 新規プロジェクト作成後のメイン画面

4.2.1. ソースファイルの作成

ソースファイル（ソースコードを記述したファイル）を作成するには：

1. メニューから「プロジェクト」→「ソースファイルの新規作成...」を選択します（図 4-3）。

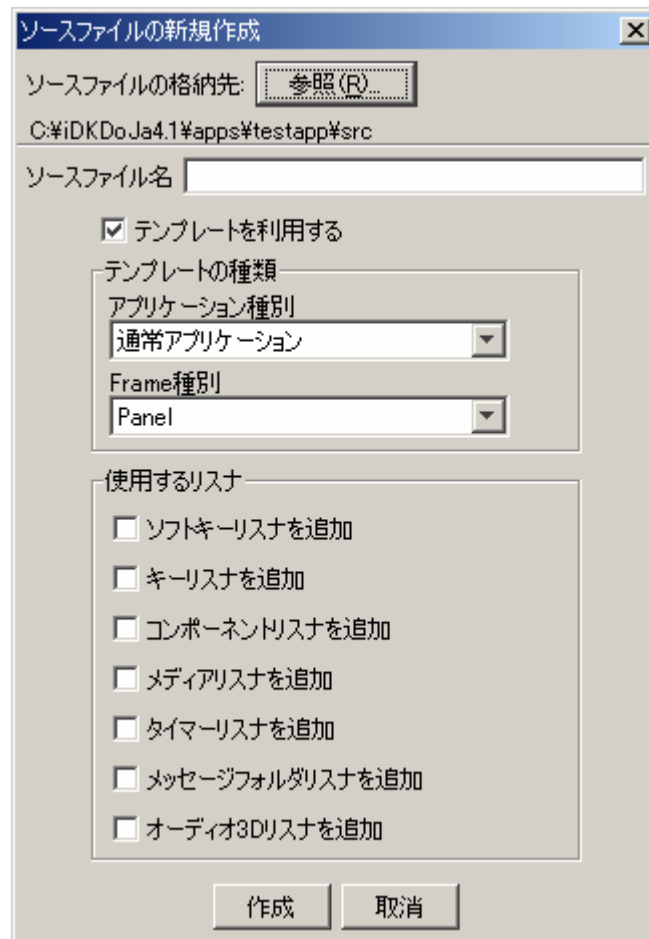


図 4-3 ソースファイルの新規作成画面

2. ソースファイル名を入力します。
3. 「テンプレートを利用する」にチェックを付けると、テンプレートのソースコードが書かれたソースファイルが作成されます。

テンプレートを利用する場合：

- (1) テンプレートの種類「アプリケーション種別」で、
- 「通常アプリケーション (IApplication)」
 - 「待ち受けアプリケーション (MApplication)」
 - 「なし」
- のいずれかを選択します。^[4]

「なし」は、Canvas か Panel だけのクラスを作成する場合に選択します。
待ち受けアプリケーションに関しては、後ほど「4.8. 待ち受け起動」で説明します。

- (2) テンプレートの種類「Frame 種別」で、
- 「Panel」 (MainPanel というクラスで作成されます。)
 - 「Canvas」 (MainCanvas というクラスで作成されます。)
- のいずれかを選択します。



Panel は DoJa の高レベル A P I 用、Canvas は DoJa の低レベル A P I 用です。
ボタンやラベル、テキスト入力には、Panel。
画像や点/線の描画には、Canvas を用いるとよいでしょう。

^[4] テンプレートは、以下のファイルに格納されています。

- Panel 用通常アプリケーション：
C:\idKDoJa4.1\lib\template\IApplicationPanel.txt
- Panel 用待ち受けアプリケーション：
C:\idKDoJa4.1\lib\template\MApplicationPanel.txt
- Canvas 用通常アプリケーション：
C:\idKDoJa4.1\lib\template\IApplicationCanvas.txt
- Canvas 用待ち受けアプリケーション：
C:\idKDoJa4.1\lib\template\MApplicationCanvas.txt
- Panel のみ：
C:\idKDoJa4.1\lib\template\Panel.txt
- Canvas のみ：
C:\idKDoJa4.1\lib\template\Canvas.txt
- 各リスナの設定ファイル (埋め込みの定義)：
C:\idKDoJa4.1\lib\template\listener\xxxListener.conf

テンプレートファイル中で変換される変数は以下のとおりです。

変数名	変換内容
{CLASSNAME}	クラス名
{DATE}	作成日付
%IMPORT%	import 文の埋め込み
%IMPLEMENTS%	implements 文の埋め込み
%INIT_PROC%	コンストラクタの埋め込み
%ABSTRACT_METHOD%	リスナのメソッドの埋め込み
%FIELD%	フィールドの埋め込み

% で囲まれた埋め込み変数は変更しないでください。

(3) 使用するリスナで、使いたいリスナにチェックを付けます。

- ソフトキーリスナ . . . ソフトキーを扱う場合
- キーリスナ . . . キーを扱う場合
- コンポーネントリスナ . . . ボタンなどのコンポーネントを扱う場合
- メディアリスナ . . . 動画像や音のメディアデータを扱う場合
- タイマリスナ . . . タイマを扱う場合
- メッセージフォルダリスナ . . . メッセージフォルダを扱う場合
- オーディオ3Dリスナ . . . 3D音源を扱う場合

4. 「作成」ボタンを押します。

5. i アプリで参照するリソースファイル（画像や音楽など）があれば、それらを置いておきます。

i アプリのJarに含めてしまうもの：

- ・ リソースファイル：**C:\iDKDoJa4.1\apps\<プロジェクト名>\res** の下
あらかじめ i モード携帯電話上に保存されているもの：
- ・ マルチメディアファイル：**C:\iDKDoJa4.1\lib\multimedia** の下
(詳しくは、「9.10. マルチメディアファイル」参照)

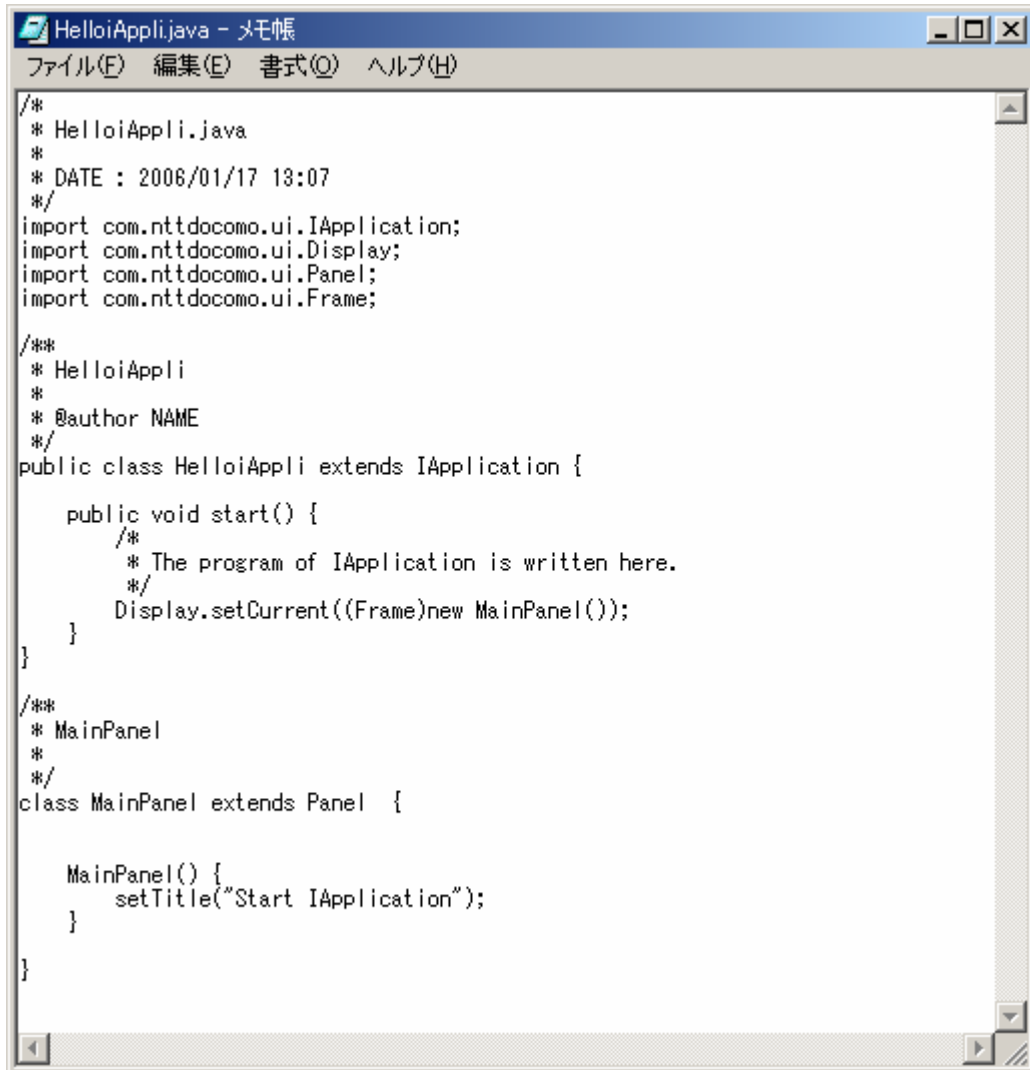
6. 任意のエディタでソースファイルが開かれるので、ソースコードを作成してください。
(任意のエディタについては、「4.2.3. ソースエディタの設定」参照)

(例)

図 4-3 でソースファイル名に "HelloiAppli" と入力し、テンプレートに「通常アプリケーション」の「Panel」を選択して作成すると、図 4-4 のようにファイルが開かれます。

このソースコードは、

C:\iDKDoJa4.1\apps\<プロジェクト名>\src\HelloiAppli.java
で作成されています。



```
/*
 * HelloiAppli.java
 *
 * DATE : 2006/01/17 13:07
 */
import com.nttdocomo.ui.IApplication;
import com.nttdocomo.ui.Display;
import com.nttdocomo.ui.Panel;
import com.nttdocomo.ui.Frame;

/**
 * HelloiAppli
 *
 * @author NAME
 */
public class HelloiAppli extends IApplication {
    public void start() {
        /*
         * The program of IApplication is written here.
         */
        Display.setCurrent((Frame)new MainPanel());
    }
}

/**
 * MainPanel
 *
 */
class MainPanel extends Panel {
    MainPanel() {
        setTitle("Start IApplication");
    }
}
```

図 4-4 (例) HelloiAppli ソースコード

※1

ソースコードを記述するには、D o J a の A P I を理解する必要があります。
ソースコードを作成するために必要なD o J a の A P I リファレンスは、
NTTドコモの次のURLから入手することができます:

<http://www.nttdocomo.co.jp/p s/imode/make/>

4.2.2. ソースコードの修正

ソースコードを修正するには、

1. メニューから「プロジェクト」→「ソースファイルを開く...」を選択します。
ソースファイルは、直接、他のエディタで開いて変更しても構いません。
最終的に src の下に置かれたソースコードがコンパイルされます（「4.3. プロジェクトのビルドと通常起動」参照）。

ソースファイルの選択画面が表示されます（図 4-5）。

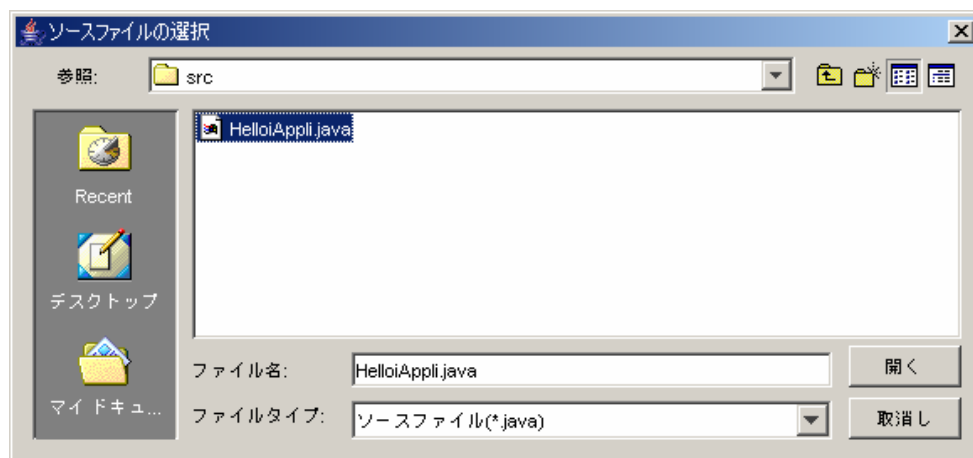


図 4-5 ソースファイルの選択画面

2. 修正したいソースファイルを選択して、「開く」を押します。
3. 任意のエディタでソースファイルが開かれるので、ソースコードを作成してください（任意のエディタについては、「4.2.3. ソースエディタの設定」参照）。

4.2.1. ソースファイルの作成の 図 4-4（例）HelloiAppli ソースコードで作成したソースコードの場合、図 4-6 のように開かれます。



```
/*
 * HelloiAppli.java
 *
 * DATE : 2006/01/17 13:07
 */
import com.nttdocomo.ui.IApplication;
import com.nttdocomo.ui.Display;
import com.nttdocomo.ui.Panel;
import com.nttdocomo.ui.Frame;

/**
 * HelloiAppli
 *
 * @author NAME
 */
public class HelloiAppli extends IApplication {
    public void start() {
        /*
         * The program of IApplication is written here.
         */
        Display.setCurrent((Frame)new MainPanel());
    }
}

/**
 * MainPanel
 *
 */
class MainPanel extends Panel {

    MainPanel() {
        setTitle("Start IApplication");
    }
}
```

図 4-6 エディタで開かれたソースコード

4.2.3. ソースエディタの設定

ソースコードを編集するためのエディタは、初期値で「NOTEPAD.EXE」になっています。ソースコードを編集するためのエディタを変更したい場合は、

1. メニューから「設定」→「ソースエディタの設定...」を選択します。（図 4-7）

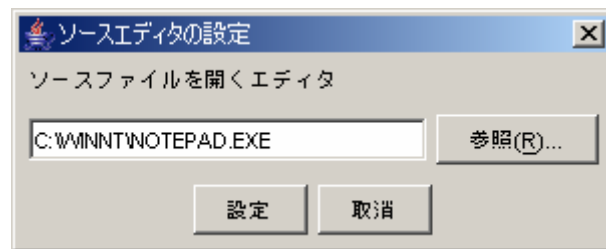


図 4-7 ソースエディタの設定画面

2. 「ソースファイルを開くエディタ」に任意のエディタの起動プログラムを指定するか「参照...」を押してファイル選択ダイアログで選択します。
3. 「設定」を押します。
ソースファイルがここで指定されたエディタで開かれるようになります。

4.3. プロジェクトのビルドと通常起動

プロジェクトのビルドと通常起動を行うには：

1. メニューから「プロジェクト」→「ビルド」を選択するか、「ビルド」ボタンを押します（図 4-8）。

ソースコードのコンパイルと事前検証を行い、JARファイル、アプリケーションディスクリプタファイルおよびダウンロード用のHTMLファイルを作成します。^[5]

このとき、アプリケーションの実行に必要なADFのAppNameとAppClassが未設定だった場合、ソースファイルから最初に見つかったIApplicationの実装クラス名が設定されます。

アプリケーションディスクリプタファイル（ADF）については、「4.7. アプリケーションディスクリプタファイル（ADF）の編集」で説明します。

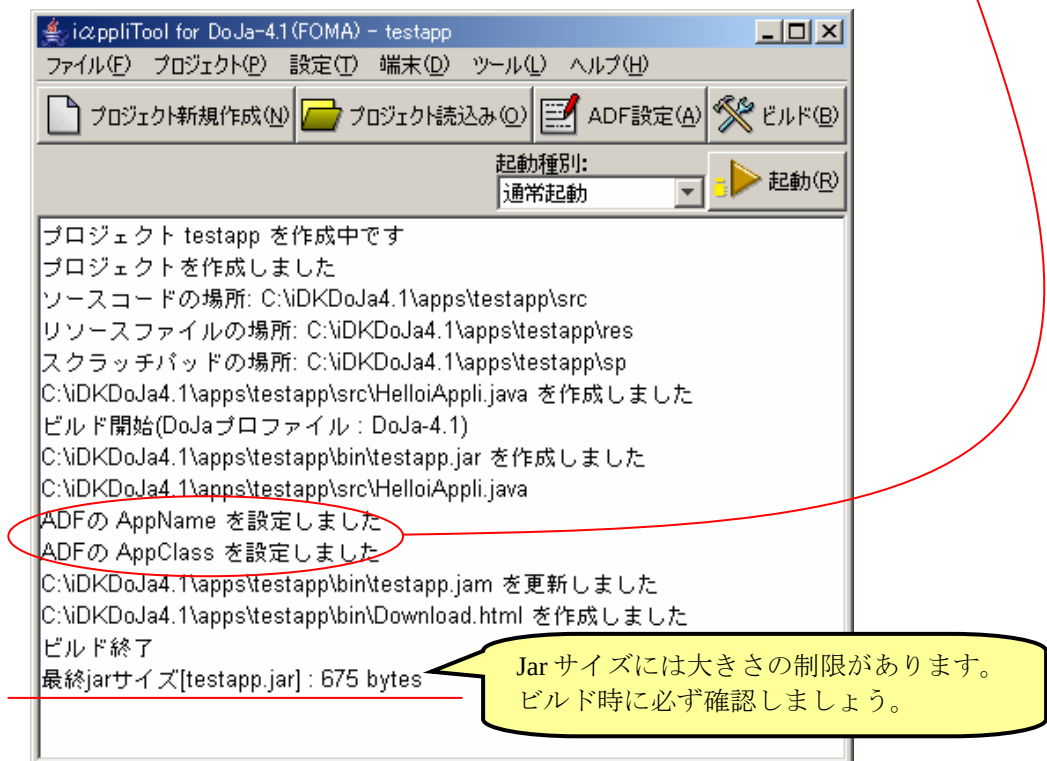


図 4-8 プロジェクトのビルド

2. メニューから「プロジェクト」→「ADF設定...」を選択するか、「ADF設定」ボタンを押します（図 4-9）。
アプリケーションに必要なADF項目を設定し、「設定」ボタンを押します（「4.7. アプリケーションディスクリプタファイル（ADF）の編集」参照）。

^[5] 正確には、既に存在する<プロジェクト名>.jam という ADF の内容を更新します（「AppSize」と「LastModified」属性が設定されます）。また、「ビルド」時に「環境変数 JDK_HOME に J2SE SDK をインストールしたディレクトリを設定してください」というメッセージが出力された場合、環境変数 JDK_HOME に J2SE SDK をインストールしたディレクトリを設定した後、改めて「iαppliTool」を起動してください。

ADF設定

AppName HelloiAppli

AppVer

PackageURL testapp.jar

AppSize 675

ConfigurationVer

ProfileVer

SPsize

AppClass HelloiAppli

AppParam

LastModified Thu, 19 Jan 2006 11:15:19

UseNetwork ☐ http

TargetDevice

LaunchAt

MyConcierge ☐ yes

UseTelephone ☐ call

UseBrowser ☐ launch

LaunchByMail

設定 取消

図 4-9 ADF の設定

(図 4-4 の例の場合は、AppName と AppClass が "HelloiAppli" になっています。
この例では特に変更する必要はありません。)

3. メニューから「プロジェクト」→「起動...」を選択するか、「起動」ボタンを押します。

ビルドした i アプリが起動されます (図 4-10)。

ただし、エミュレータが以下の状態の時には i アプリを起動できません：

- メーラ (「4.16. メーラ起動」を参照) が起動している
- ブラウザ (「4.17. ブラウザ起動」を参照) が起動している
- コードリーダー (「4.18. コードリーダー起動」を参照) が起動している

「通常起動」の起動タイプは、「LAUNCHED_FROM_MENU」です。
デフォルトの起動方法は「通常起動」ですが、その他にもいくつかの起動方法があります。



図 4-10 iアプリ 起動

4.3.1. ダウンロード用 HTML ファイル

i アプリをビルドすると、JARファイル、アプリケーションディスクリプタファイルと同じ場所に、i アプリをダウンロードするためのHTMLファイル（Download.html）が作成されます（図 4-1 1）。

ただし、このファイルを直接書き換えた場合は、次回から再作成されなくなります。

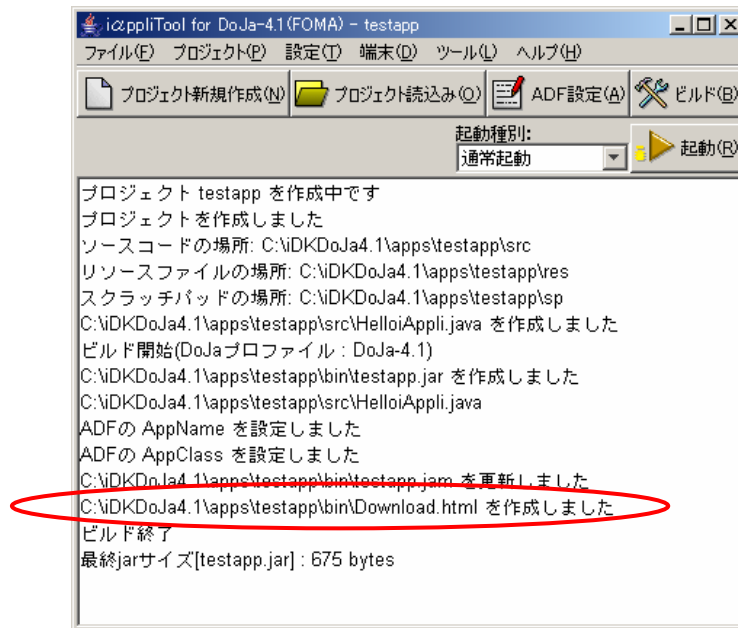


図 4-1 1 ダウンロード HTML ファイル作成

このHTMLファイルは、テンプレートから作成されています。
テンプレートとなるファイルは次の場所に置いてあります：

C:\iDKDoJa4.1\lib\template\Download.txt

また、テンプレートファイル中には、以下の定数を指定できます。

定数	内容
{PROJECTNAME}	プロジェクト名です。
{PACKAGEURL}	ADFのPackageURLの値と同じパスの jam ファイルの位置です（4.7 アプリケーションディスクリプタファイル（ADF）の編集 参照）。PackageURLが設定されていない場合、プロジェクト名.jam になります。
{ATTRIBUTE}	起動種別により、ijam、ilet、ista のいずれかになります。 ijam … 即起動、ブラウザ起動以外の場合 ilet … 即起動の場合 ista … ブラウザ起動の場合

HTMLファイルのタグ記述については、「9.4. ブラウザ起動HTMLファイルの記述方法」を参照してください。

4.3.2. iモード携帯電話でiアプリ実行

もっとも簡単に、作成した i アプリを i モード携帯電話で実行する例として：

1. アプリケーションディスクリプタファイル（ADF）の「PackageURL」を、i アプリ（Jar）を配置する HTTP サーバの URL に変更します。
2. 「ビルド」を実行して、Jar、ADF、ダウンロード用 HTML を作り直します。
もし、ダウンロード用 HTML に手を加えている場合は再作成されませんので、直接、ダウンロード用 HTML 中の URL を修正してください。
3. 次のファイルを HTTP サーバ上に配置します。
 - C:\iDKDoJa4.1\apps\<プロジェクト名>\bin\<プロジェクト名>.jar
 - C:\iDKDoJa4.1\apps\<プロジェクト名>\bin\<プロジェクト名>.jam
 - C:\iDKDoJa4.1\apps\<プロジェクト名>\bin\Download.html
4. i モード携帯電話のブラウザから Download.html を開きます。
5. 「Download」リンクをクリックして、i アプリをダウンロードします。
6. i アプリを実行します。

ワンポイント

i アプリを i モード携帯電話で動作させるとき、ダウンロードした機種を特定させたい場合があります。

i モード携帯電話の機種を特定させるには、次の 2 つの方法があります。

1 つは、`java.lang.System.getProperty("microedition.platform")` を使って i アプリが取得する方法（エミュレータでは、ネットワーク設定の **UserAgent** に設定された **"DOJA_INET_CLIENT"** の部分を返します）、もう 1 つは、HTTP 通信時に i モードサーバから送出される User-Agent ヘッダを見る方法です。
（詳しくは、「i アプリコンテンツ開発ガイド」を参考にしてください。）

4.4. プロジェクトの読み込み

既存プロジェクトを読み込むには：

1. メニューから「プロジェクト」→「読み込み...」を選択するか、「プロジェクト読み込み」ボタンを押します。

「プロジェクト読み込み」画面が表示されます（図 4-1 2）。^[6]

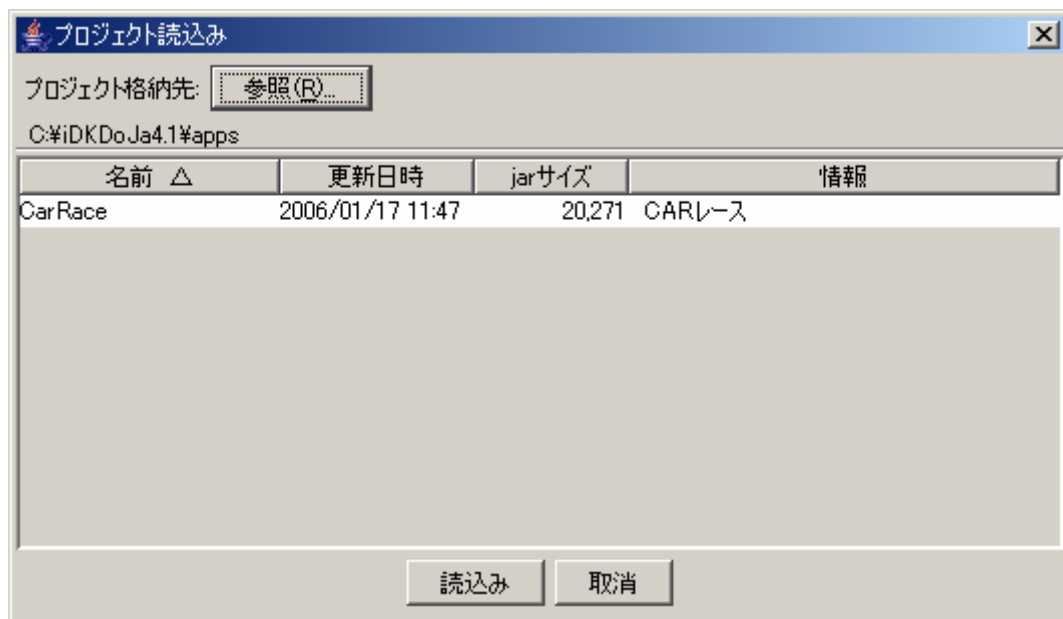


図 4-1 2 プロジェクト読み込み画面

2. プロジェクトの格納先（C:\iDKDoJa4.1\apps）を変更する場合は、「参照...」ボタンを押して格納先のディレクトリを選択します。
一覧のタイトル「名前」、「更新日時」をクリックすると並び替えて表示されます。

^[6] プロジェクト一覧の「情報」欄は、デフォルトでADFの AppName の値が表示されます。（ADFについては、「4.7. アプリケーションディスクリプタファイル（ADF）の編集」参照）

この情報を任意の説明に変更したい場合は、規定クラスのドキュメンテーションコメントに「@info」で付与します。（ただし、画面には最大 128 バイトまでしか表示されませんので、i アプリの概要が分かる程度の短い文章で記述しましょう。）

<例>

```
/**
 * @info HelloiAppli のメインクラスです。
 *
 * @author NAME
 */
public class HelloiAppli extends IApplication {
    :
```

3. プロジェクト名を選択し「読み込み」ボタンを押します。

メイン画面のタイトルバーにプロジェクト名が表示されます（図 4-1 3）。

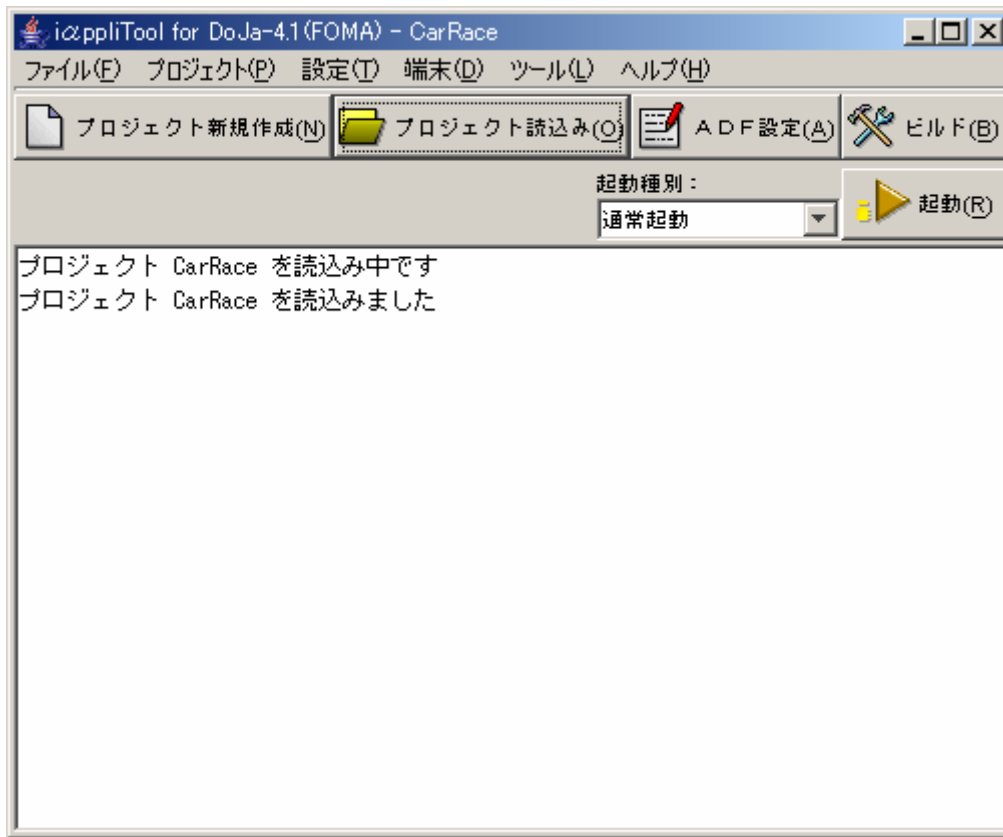


図 4-1 3 既存プロジェクト読み込み後のメイン画面

4. 「起動」ボタンを押すと読み込んだプロジェクトの i アプリが実行されます。

4.5. プロジェクトの設定

i モード携帯電話に合わせたプロジェクトの設定を行います。

プロジェクトの設定をするには：

1. メニューから「プロジェクト」→「プロジェクト設定...」を選択します（図 4-14）。

プロジェクト プロパティ

最大件数設定

最大送信メール件数: 50 最大受信メール件数: 200

最大電話帳登録件数: 10 最大電話帳グループ登録件数: 10

メディアリソースプロパティ設定

ビデオトラック数: 1 オーディオトラック数: 1

テキストトラック数: 1

☐ 拡張HTTPヘッダの関連付け

X-Dcm-Move: 0000

アプリケーション端末移動情報設定

☐ アプリ移動情報フラグ ☐ 異なる端末間の移動

フォントベースライン設定:

カメラデバイス数: 1

3Dサウンドリソース数: 4

自動起動時刻登録設定

ONETIME YYYY: 1999 Dec SUN DD: 9 HH: 9 MM: 9

Onetime 1999 December 09 09:09

Every Monday 02:02

null

null

設定 取消

自動起動時刻設定

自動起動時刻一覧

図 4-14 プロジェクトの設定

2. 必要な設定項目を設定します。
3. 「設定」ボタンを押すと、変更が有効になります。
各設定項目は、表 4-1 に示すとおりです。

表 4-1 プロジェクトの設定項目

設定項目	説明	デフォルト
最大送信メール件数	送信/未送信メールの最大保存件数を設定します。	50
最大受信メール件数	受信メールの最大保存件数を設定します。	200
最大電話帳登録件数	電話帳エントリの最大保存件数を設定します。	10
最大電話帳グループ登録件数	電話帳グループの最大保存件数を設定します。	10
ビデオトラック数	MediaImage#MP4_AUDIOTRACKのプロパティで取得できる Video トラック数を設定します。	1
オーディオトラック数	MediaImage#MP4_AUDIOTRACKのプロパティで取得できる Audio トラック数を設定します。	1
テキストトラック数	MediaImage#MP4_AUDIOTRACKのプロパティで取得できる Text トラック数を設定します。	1
拡張 HTTP ヘッダの関連付け	メディアに設定する HTTP のヘッダ値 "X-Dcm-Move" で、「0000」か「0001」を設定します。	設定なし
アプリ移動情報フラグ	IApplication#isMoved()で取得できる状態を設定します。	チェックなし (false)
異なる端末間の移動	IApplication#isMovedFromOtherTerminal()で取得できる状態を設定します。	チェックなし (false)
フォントベースライン設定	文字のベースラインを変更する場合に設定します。	なし
カメラデバイス数	カメラの数を設定します。 Camera#getCamera() のカメラ ID 0, 1, 2... の順で対応します。	1
3Dサウンドリソース数	Audio3D#getResources() で取得できる 3Dサウンドのリソース数を設定します。	4
自動起動時刻設定 自動起動時刻一覧	自動起動時刻を設定します。 設定されていない初期状態は、「null」と表示されます。 ※ 値を変更して別の項目に移動すると一覧に反映されます。	Onetime 1999 January 09 09:09 Every Monday 02:02 Every 01 of month 12:00 Every July 01 15:30

詳細については、「9.6. 「アプリケーション名.properties」 ファイル」を参照してください。

4.6. ファイル配置カスタマイズ

ビルドされた i アプリを自動的に別の場所に配置したい場合、配置先を変更する事ができます。

i アプリの配置を設定するには：

1. メニューから「プロジェクト」→「ファイル配置カスタマイズ(D)...」を選択します（図 4-15）。

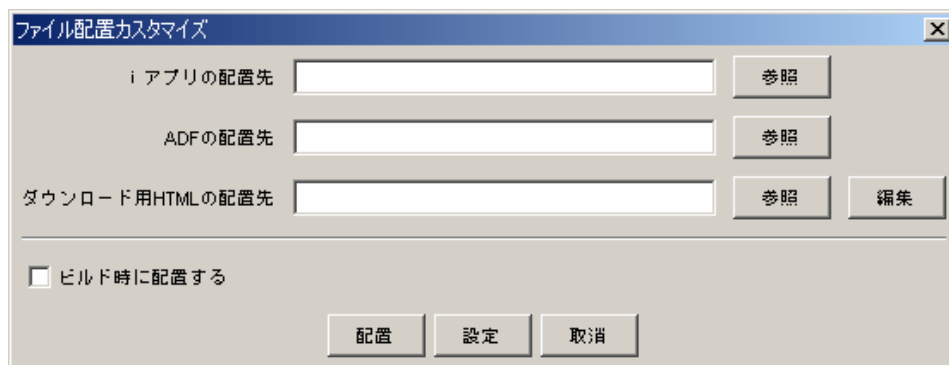


図 4-15 ファイル配置カスタマイズ画面

2. 必要な設定項目を設定します。
3. 「設定」ボタンを押すと、変更が有効になります。
各設定項目は、表 4-2 に示すとおりです。
4. 「配置」ボタンを押すと、それぞれのファイルを配置します。
5. 「ビルド時に配置する」にチェックをつけると、ビルドしたときに毎回配置されます。
これにより、「iα ppli Development Kit for DoJa-4.1(FOMA)」で作成された i アプリが、そのまま本番環境にも反映されている状態となります。

表 4-2 ファイル配置カスタマイズの設定項目

設定項目	説明	デフォルト
i アプリの配置先	i アプリ (xxx.jar) の配置先を設定します。 ファイル名まで指定するとそのファイル名に変更されます。	なし
ADF の配置先	ADF (xxx.jam) の配置先を設定します。 ファイル名まで指定するとそのファイル名に変更されます。	なし
ダウンロード用HTMLの配置先	ダウンロード用HTMLの配置先を設定します。 ファイル名まで指定するとそのファイル名に変更されます。	なし

4.7. アプリケーションディスクリプタファイル(ADF)の編集

i モード携帯電話では、Java Application Manager (JAM) と呼ばれるコンポーネントで i アプリの実行を管理しています。そのため、i アプリの JAR ファイルとは別に i アプリの情報を JAM に渡すためのファイル (アプリケーションディスクリプタファイル) が必要になります。

アプリケーションディスクリプタファイルを編集するには：

1. メニューから「プロジェクト」→「ADF 設定...」を選択するか、「ADF 設定」ボタンを押します。

「ADF設定」画面が表示されます (図 4-16)。

ADF設定

AppName: CARレース

AppVer:

PackageURL: http://localhost/CarRace/bin/CarRace.jar

AppConfiguration: [必須入力項目]
i アプリの格納場所を示すURL
プロジェクト作成時に自動設定

ProfileVer:

SPsize: 20,30700

AppClass: CarRace

AppParam:

LastModified: Thu, 27 Oct 2005 18:44:55

UseNetwork: ☒ http

TargetDevice:

LaunchAt:

MyConcierge: ☐ yes

UseTelephone: ☐ call

UseBrowser: ☐ launch

LaunchByMail:

設定 取消

各項目にマウスポインタを合わせるとツールチップで設定する内容が表示されます。

図 4-16 ADF 設定画面

2. 必要な設定項目を入力します。
3. 「設定」ボタンを押すと、変更が有効となります。
各設定項目は、表 4-3 に示すとおりです。

表 4-3 アプリケーションディスクリプトファイル(ADF)の設定項目

設定項目	説明	種別
AppName	i アプリの名称（最大 16 バイト）。設定する文字列は、SJIS を使用します。 プロジェクトのビルド時に未設定であれば、IApplication が実装されたクラス名が設定されます。 (ただし、IApplication がソースコード中に複数ある場合、最初に見つかったクラスになります。)	必須
AppVer	i アプリのバージョン（最大 10 バイト）。この項目は、エミュレータでは使用しないので設定しなくても問題ありません。	オプション
PackageURL	i アプリの格納場所を表す URL（最大 255 バイト）。設定する文字列は、ASCII 形式とします。ただし、サーバの IP アドレスを直接指定された場合は、アプリケーションの起動を禁止します。 この項目に相対URLを設定した場合、「5.1. ネットワークの設定」で設定した「ADFのURL」フィールドがベースのURLとなります。 i アプリはこの PackageURL のホストとポートのみ通信可能です。 新規プロジェクトを作成したときには、i アプリの Jar ファイル名になっているので、適切な URL (i アプリのダウンロード先) に変更してください。	必須
AppSize	アプリケーションサイズ（バイト単位）。 実際の i アプリのサイズ（Jar）と違っていたり、0 または負数が指定されたりした場合は、アプリケーションの起動を禁止します。 プロジェクトのビルド時に生成された Jar ファイルのサイズが設定されます。	必須
ConfigurationVer (KvmVer)	コンフィグレーション(KVM)のバージョン。 KvmVer と ConfigurationVer はどちらも同じ意味とします。 ただし、両方指定された場合は、アプリケーションの起動を禁止します。 設定する文字列は、“CLDC-1.0” (DoJa-1.0, 2.0, 2.1, 2.2, 3.0, 3.5, 4.0, 4.1) または、“CLDC-1.1” (DoJa-4.0, DoJa-4.1) とし、それ以外を設定した場合は、i アプリの起動を禁止します。 但し、大文字・小文字の区別はしません。 この項目は、エミュレータでは使用しないので設定しなくても問題ありません。	オプション

ProfileVer	プロファイルのバージョン。 設定する文字列は、以下のとおりで、それ以外を設定した場合は、アプリケーションの起動を禁止します。但し、大文字・小文字の区別はしません。 この項目は、エミュレータでは使用しないので設定しなくても問題ありません。 ・"DoJa-1.0" (503i シリーズ対応) ・"DoJa-2.0" (504i シリーズ対応) ・"DoJa-2.1" (FOMA2051 シリーズ対応) ・"DoJa-2.2" (FOMA2102 シリーズ対応) ・"DoJa-3.0" (505i、506i シリーズ対応) ・"DoJa-3.5" (900i シリーズ対応) ・"DoJa-4.0" (901i シリーズ対応) ・"DoJa-4.1" (902i シリーズ対応)	オプション
SPSize	スクラッチパッドのサイズ (バイト単位)。 複数のスクラッチパッドは、カンマ(,)で区切って指定します。(最大16個) エミュレータ上で指定できるサイズに制限はありませんが、実際の端末では、機種により合計したサイズに制限があります。この項目を設定しない場合のデフォルト値は0バイトとなります。 また、負数、整数以外が設定された場合は、アプリケーションの起動を禁止します。 (例: 16, 1024) スクラッチパッドは、アプリケーションディスクリプタファイルの編集画面 (図 4-16) で「設定」を押すと再作成(自動拡張)されます。このとき既に記憶されているファイル中のデータはサイズの範囲内でそのまま残ります。	オプション
AppClass	i アプリのメインクラス名 (最大 255 バイト)。 このクラス名は、com.nttdocomo.ui.IApplication のサブクラスの名前でなければなりません。 設定する文字列は、ASCII 形式とし、255 バイトを超えて設定した場合は、アプリケーションの起動を禁止します。 プロジェクトのビルド時に未設定であれば、IApplication が実装されたクラス名が設定されます。 (ただし、IApplication がソースコード中に複数ある場合、最初に見つかったクラスになります。)	必須
AppParam	メインクラスに渡すパラメータ (最大 255 バイト)。 パラメータは、スペースで区切って設定します。アプリケーションは、com.nttdocomo.ui.IApplication.getArgs() メソッドを使用してこのパラメータを取得します。 このメソッドは、この項目が設定されていないと null を返します。設定できる文字列は ASCII 形式とし、255 バイト以内ならパラメータの個数に制限はありません。 しかし、255 バイトを超えて設定した場合は、アプリケーションの起動を禁止します。	オプション

LastModified	i アプリの最終更新時刻。設定する文字列の形式については「4.7.1. LastModified項目の表記方法」を参照。この項目は、ビルドのJARファイル生成時に自動的に設定されます。 プロジェクトのビルド時にビルドした時間が設定されます。	必須
UseNetwork	i アプリがネットワーク通信を使用することを表すフラグ。 有効な値は "http" (ASCII 形式とし、大文字・小文字の区別はしません) のみとし、"http" 以外の文字を設定した場合は、アプリケーションの起動を禁止します。 また、この項目に何も設定しなかった場合は、ネットワーク接続できない i アプリとみなされますので、ネットワークを接続しようとする例外が発生します。	オプション
TargetDevice	i アプリが動作する機種名 (user agent に記載されているものと同等で、最大 128 バイト)。 この項目は、特定の機種に対してのみ i アプリのダウンロードを許可したいときに設定します。 設定する文字列は、ASCII 形式とし、大文字・小文字の区別はしません。カンマ(,)で区切られた文字列を 1 機種とします。 複数の機種をターゲットとしている場合、"X2005i, Y2005i, Z2005i" (PDC(MOVA)の場合、"X506i, Y506i, Z506i") のように設定します。 また、この項目に何も設定しなかった場合は、全ての機種をターゲットとしているとみなされます。 シリーズの後に "i", "is" 等が付与されている場合も同一シリーズとみなされてダウンロードできます。 同一シリーズ全機種をターゲットとする場合は、"all2005" のようにシリーズ番号の前に "all" を付加します。 この項目は、エミュレータでは使用しないので設定しなくても問題ありません。	オプション
LaunchAt	起動時間間隔指定。 i アプリを一定時間ごとに自動起動させるときに設定します。 形式: I h ("h" は起動間隔 (時間 単位)。I と h の間には 1 つのスペースを入れる。例: "I 6" ... 6 時間毎に起動) "h" に 0 が設定された場合は、無効なものとみなします。 "h" の有効桁数は 2 桁とします。 また、設定する文字列は、ASCII 形式とし、それ以外が設定された場合は、アプリケーションの起動を禁止します。但し、大文字・小文字の区別はしません。	オプション

MyConcierge	待ち受けアプリケーションかどうかを表すフラグ。 この項目は、エミュレータ上で待ち受け起動可能であるかどうかの判定に使用します（「4.8.待ち受け起動」参照）。 有効な値は "yes"（ASCII 形式とし、大文字・小文字の区別はしません）のみとし、"yes" 以外の文字を設定した場合は、アプリケーションの起動を禁止します。 また、この項目に何も設定しない場合は、待ち受け起動しない i アプリであるとみなし、エミュレータ上で待ち受け起動を禁止します。	オプション
UseTelephone	電話の音声通話機能を利用するかどうかを表すフラグ。 有効な値は "call"（ASCII 形式とし、大文字・小文字の区別はしません）のみとし、"call" 以外の文字を設定した場合は、アプリケーションの起動を禁止します。 また、この項目に何も設定しない場合は、電話の音声通話機能を利用しないアプリケーションであるとみなします。	オプション
UseBrowser	i アプリからブラウザ機能を利用するかどうかを表すフラグ。 有効な値は "launch"（ASCII 形式とし、大文字・小文字の区別はしません）のみとし、"launch" 以外の文字を設定した場合は、アプリケーションの起動を禁止します。 また、この項目に何も設定しない場合は、ブラウザ機能を利用しないとみなしますので、アプリケーションからブラウザ機能を使用すると例外が発生します。	オプション
LaunchByMail	i アプリの起動を許可するメールアドレス（最大 50 バイト）。 この項目に "any" を設定した場合は、任意のメールアドレスからのアプリケーションの起動を許可します。 "any" 以外のメールアドレスを設定した場合は、アプリケーションを起動しようとしているメールアドレスの From: アドレスと、この項目で設定した値が完全に後方一致するときのみ起動を許可します。 それ以外はアプリケーションの起動を禁止します。 また、この項目に複数のメールアドレスを設定した場合は、先頭の 1 つのメールアドレスのみ有効とします。 設定する文字列は、ASCII 形式とします。	オプション
LaunchByBrowser	i アプリの起動を許可する URL（最大 255 バイト）。 この項目に "any" を設定した場合は、任意の URL からのアプリケーションの起動を許可します。 "any" 以外の URL を設定した場合は、アプリケーションを起動しようとしている URL と、この項目で設定した値が完全に前方一致するときのみ起動を許可します。 また、この項目に複数の URL を設定した場合は、先頭の 1 つの URL のみ有効とします。 なお、先頭から空白、タブまでを 1 つの URL とみなします。 設定する文字列は、ASCII 形式とします。	オプション

AllowPushBy	i アプリのプッシュ起動を許可する外部インタフェース（最大 255 バイト）。 この項目に "Irda:<起動コマンド>" を設定した場合は、赤外線から、"Code:<起動コマンド>" を設定した場合は、コードリーダーから i アプリの起動を許可します（<起動コマンド>は、i アプリの起動チェックの際に使用し、1 文字以上の文字列とします）。 "Irda:"、"Code:"以外の文字を指定した場合は、アプリケーションの起動を禁止します。 また、複数の外部インタフェースを設定する場合は、空白、タブで機器を区切ります。 この項目に複数の外部インタフェースを設定した場合は、先頭の 1 つの外部インタフェースのみ有効とします。	オプション
AppTrace	i アプリのトレース機能を利用するかどうかを表すフラグ。 有効な値は "on"（ASCII 形式とし、大文字・小文字の区別はしません）のみとし、"on"以外の文字を設定した場合は、アプリケーションの起動を禁止します。 また、この項目に何も設定しなかった場合は、i アプリのトレース機能を使用しないものとみなします。 この項目は、エミュレータでは使用しないので設定しなくても問題ありません。	オプション
DrawArea	i アプリが使用する描画領域。 この項目に以下の形式に従った値を設定した場合： 形式：（横のピクセル数）x（縦のピクセル数） 96 <= （横のピクセル数） <= 端末の最大横ピクセル数 72 <= （縦のピクセル数） <= 端末の最大縦ピクセル数 i アプリ起動時の画面を、端末の描画領域の中央に設定します。（x は小文字の x (ASCII 0x78) です） その際、描画領域周囲の表示色は実装依存とし、i アプリから描画を行うことはできません。 上記の形式（設定可能なピクセル数は正の整数のみ）以外の値を設定した場合は、アプリケーションの起動を禁止します。 また、この項目に何も設定しなかった場合は、通常通りの描画とします。	オプション

GetSysInfo	メール、メッセージアイコン、圏外／圏内情報、電池残量、マナーモード設定情報取得可否を表すフラグ。 有効な値は "yes" (ASCII 形式とし、大文字・小文字の区別はしません) のみとし、"yes" 以外の文字を設定した場合は、アプリケーションの起動を禁止します。 また、この項目に何も設定しない場合は、メール、メッセージアイコン、圏外／圏内情報、電池残量、マナーモード設定情報を取得しないものとみなしますので、i アプリからこれらを取得しようとする場合例外が発生します。	オプション
TrustedAPID	NTTドコモで付与するトラステッドの管理ID。 この項目に11桁の数字以外が設定された場合は、アプリケーションの起動を禁止します。 この項目が設定された場合は、i アプリはトラステッド (i アプリ d x) であるとみなします。 <u>トラステッド動作設定のいずれかを "Yes" にした場合は、この項目に "00000000000" などの11桁の数字文字列を設定してください。</u>	オプション
TrustedLmd	トラステッドの最終更新日時	オプション
LaunchByApp	ランチャモードによる他のiアプリからの起動禁止を表すフラグ。 有効な値は "deny" (ASCII 形式とし、大文字・小文字の区別はしません) のみとし、"deny" 以外の文字を設定した場合は、アプリケーションの起動を禁止します。 この項目が設定されている場合は、ランチャモードにより他のiアプリから連携起動をしようとする場合例外が発生します。	オプション
MessageCode	NTTドコモで付与するメッセージiアプリの管理ID。 この項目に10桁のASCII形式の文字列以外が設定された場合は、アプリケーションの起動を禁止します。 また、この項目が設定されている場合は、メッセージiアプリであるとみなします。この項目は、メールフォルダや送受信メールの管理IDになります。 メッセージiアプリである場合は、この項目を設定し、トラステッド動作設定→メッセージiアプリ機能の許可を Yes、TrustedAPIDに "0000000000" などの10桁の数字文字列を設定してください。	オプション
AccessUserInfo	電話帳、スケジュール、画像、ブックマーク登録の可否を表すフラグ。 有効な値は "yes" (ASCII 形式とし、大文字・小文字の区別はしません) のみとし、"yes" 以外の文字を設定した場合は、アプリケーションの起動を禁止します。 また、この項目に何も設定しない場合は、電話帳登録、スケジュール登録、マルチメディアへの画像保存／参照、ブックマーク登録／参照はしないものとみなしますので、アプリケーションからこれらの機能を実行しようとする場合例外が発生します。	オプション

GetUtn	端末及びU I Mの固体識別番号の利用を表す文字列。この項目に設定された固体識別番号をアプリケーションで取得することができます。 U I Mの固体識別番号を取得したい場合は、 "userid" 端末の固体識別番号を取得したい場合は、 "terminalid" 両方を取得したい場合は、 "userid,terminalid" もしくは、 "terminalid,userid" を指定します。	オプション
IletPreserve	ダウンロード即起動アプリの保存不可を表すフラグ。 有効な値は "deny"（ASCII 形式とし、大文字・小文字の区別はしません）のみとし、"deny"以外の文字を設定した場合は、アプリケーションの起動を禁止します。 この項目が設定されている場合は、即起動で起動されたアプリを保存しようとする例外が発生します。	オプション
LaunchApp	アプリ連携機能の使用可否を表すフラグ。 有効な値は "yes"（ASCII 形式とし、大文字・小文字の区別はしません）のみとし、"yes"以外の文字を設定した場合は、アプリケーションの起動を禁止します。 また、この項目に何も設定しない場合は、アプリケーションの連携起動はしないものとみなし、アプリケーションの連携起動をしようとする例外が発生します。	オプション
RemoteControl	i アプリがリモート制御アプリケーションかどうかを表します。 リモート制御アプリケーションの時は、"yes"を設定します。	オプション
FeliCaCPID	コンテンツプロバイダの識別子。 FeliCaCPIDボタンを押し、ユニークなコードを付与します（図 4-17 参照）。 フリー領域用 API を使用する場合は、この項目を設定します。 32 桁の 16 進数文字列を設定してください。 ※ 仮想FeliCaデータファイルを作成しても、フリー領域の CPID Data Blockには自動的に反映されません。仮想FeliCaデータファイルビューワを利用して手動にて設定する必要があります（「6.3. 仮想FeliCaデータファイルビューワ」を参照）。	フリー領域用 API を使用する場合は必須
UseStorage	外部メモリアクセス機能利用可否を表すフラグ。 有効な値は "ext"（ASCII 形式とし、大文字・小文字の区別はしません）のみとし、"ext"以外の文字を設定した場合は、アプリケーションの起動を禁止します。	オプション

CPBindingAuth	CP の署名及び証明書 ダウンロードしようとしている i アプリが SD-Binding 機能を利用する場合に、コンテンツプロバイダの署名と証明書を示す値を設定します。 CPBindingAuth は、FOMA のユーザ証明書でも設定することもできます。FOMA のユーザ証明書でこのキーを設定するには、「6.8. CPBindingAuth 署名付与ツール」を参照してください。	オプション
FeliCaPIN	対応するフリー領域に設定する PIN 情報。 フリー領域用 API を使用する場合は、この項目を設定してください。 100000 以上 8 桁以内の 10 進数文字列（100000 ～ 99999999）を設定してください。 ※ 仮想 FeliCa データファイルを作成しても、フリー領域の PIN Data Block に自動的に反映されません。仮想 FeliCa データファイルビューワを利用して手動にて設定する必要があります（「6.3. 仮想 FeliCa データファイルビューワ」を参照）。	フリー領域用 API を使用する場合は必須
AppIcon	FeliCa のアプリケーションのアイコンとして使用する画像ファイル名。（最大 255 バイト） 複数の画像ファイルを設定する場合は、カンマ（,）で区切って設定します。 設定できる文字列は ASCII 形式とし、ディレクトリ階層（¥ や /）が指定された場合、255 バイトを超えて設定した場合は、アプリケーションの起動を禁止します。	オプション
PackageHashCode	Jar ファイルの Hash 値。 PackageURL で指定された Jar の Hash 値（SHA-1）を Base64 エンコードした値を設定します。 SD-Binding 機能を利用する際に CPBindingAuth キーを設定する必要がある場合は、本キーも共に設定する必要があります。 PackageHashCode は、CPBindingAuth を設定するツールで追加する事ができます。CPBindingAuth の設定については、「6.8. CPBindingAuth 署名付与ツール」を参照してください。	オプション
CPName	CP 名を表す文字列。（最大 16 バイト） この文字列は、ストレージデバイス上に CP バインドで生成したフォルダの情報表示に利用します。 設定できる文字列は ASCII 形式とし、16bytes を超えた文字列が設定されている場合は、各携帯端末に依存します。	オプション

このアプリケーションディスクリプタファイル（ADF）は、
「C:\iDKDoJa4.1\apps\<プロジェクト名>\bin」
に「アプリケーション名.jam」というファイルで作成されます。

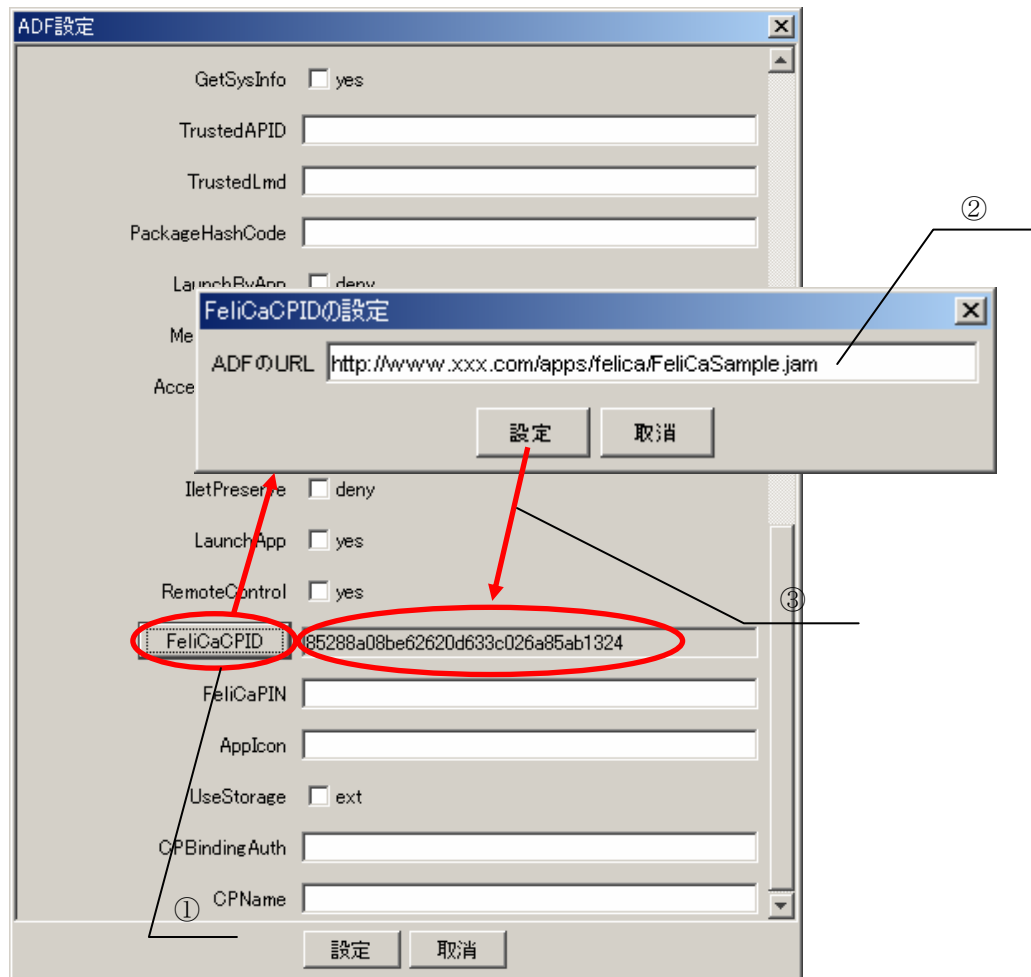


図 4-17 FeliCaCPID 設定画面

- ① ADF設定画面から「FeliCaCPID」ボタンを押すと、FeliCaCPID設定画面が表示されます（図 4-17）。
- ② FeliCaCPID 設定画面の「ADF の URL」に ADF ファイルをダウンロードする URL をスキーム(http://)から絶対パスで指定します。
- ③ FeliCaCPID 設定画面の「設定」ボタンを押します。ADF の URL をハッシュした値が ADF 設定画面に表示されます。
ADF の URL は「アプリケーション.properties」ファイルに保持され次回から ADF の URL に表示されます。
- ④ 「取消」ボタンを押した場合、FeliCaCPID は付与されず、ADF 設定画面に戻ります。

ここで付与した FeliCaCPID を削除したい場合は、直接、アプリケーションディスクリプタファイルから削除してください。

4.7.1. LastModified 項目の表記方法

ADF の LastModified 項目の表記方法を下記のとおりに規定します。

- 表記法は、RFC2822 の「3.3. Date and Time Specification」に準拠します。ただし、「4.3. Obsolete Date and Time」に記述されている内容には対応しておりません。以下に規定した項目については、これに従うものとします。
- 日付の表記は、ゼロサプレスの有無に関わらず可能とします。
例:
Sun, 1 Apr 2001 19:05:50 +0900 /* OK (ゼロサプレス無) */
Sun, 01 Apr 2001 19:05:50 +0900 /* OK (ゼロサプレス有) */
- 時、分、秒の表記は、ゼロサプレスの有無に関わらず可能とします。
例:
Sun, 01 Apr 2001 19: 5:50 +0900 /* OK (ゼロサプレス無) */
Sun, 01 Apr 2001 19:05:50 +0900 /* OK (ゼロサプレス有) */
- 曜日名の表記は、“Mon”、“Tue”、“Wed”、“Thu”、“Fri”、“Sat”、“Sun”とします。大文字・小文字の区別はしません。これに合致しない場合は、不正とみなします。
例: MON, mon, MoN, mOn /* OK */
- 月名の表記は、“Jan”、“Feb”、“Mar”、“Apr”、“May”、“Jun”、“Jul”、“Aug”、“Sep”、“Oct”、“Nov”、“Dec”とします。大文字・小文字の区別はしません。これに合致しない場合は、不正とみなします。
例: JAN, jan, JaN, jAn /* OK */
- 日付と時刻の有効範囲を閏年まで含めてチェックします。但し、日付と曜日の整合性はチェックしません。
例:
Sun, 01 Apr 2001 19:05:50 +0900 /* OK */
Tue, 01 Apr 2001 19:05:50 +0900 /* OK (曜日の整合性チェックなし) */
Sun, 01 Apr 2001 25:05:50 +0900 /* NG (時刻が不正) */
Sun, 31 Apr 2001 19:05:50 +0900 /* NG (4/31 は存在しない) */
Sun, 29 Feb 2001 19:05:50 +0900 /* NG (2001/2/29 は存在しない) */
Sun, 29 Feb 2004 19:05:50 +0900 /* OK (2004/2/29 は存在する) */
- タイムゾーンの表記は省略可能とし、省略された場合は+0900 と解釈します。また、タイムゾーンのアルファベット表記 (“GMT”、“JST”) も可能とします。

4.8. 待ち受け起動

「待ち受け起動」とは、i アプリを「待ち受けアプリケーション」として起動することです。「待ち受け起動」の起動タイプは、「LAUNCHED_AS_CONCIERGE」です。

4.8.1. 待ち受けアプリケーション

「待ち受けアプリケーション」とは、待ち受け画面になることができる i アプリのことで、com.nttdocomo.ui.MApplication クラスを継承している必要があります。表 4-4 に「待ち受け画面」と「待ち受けアプリケーション」との比較を示します。

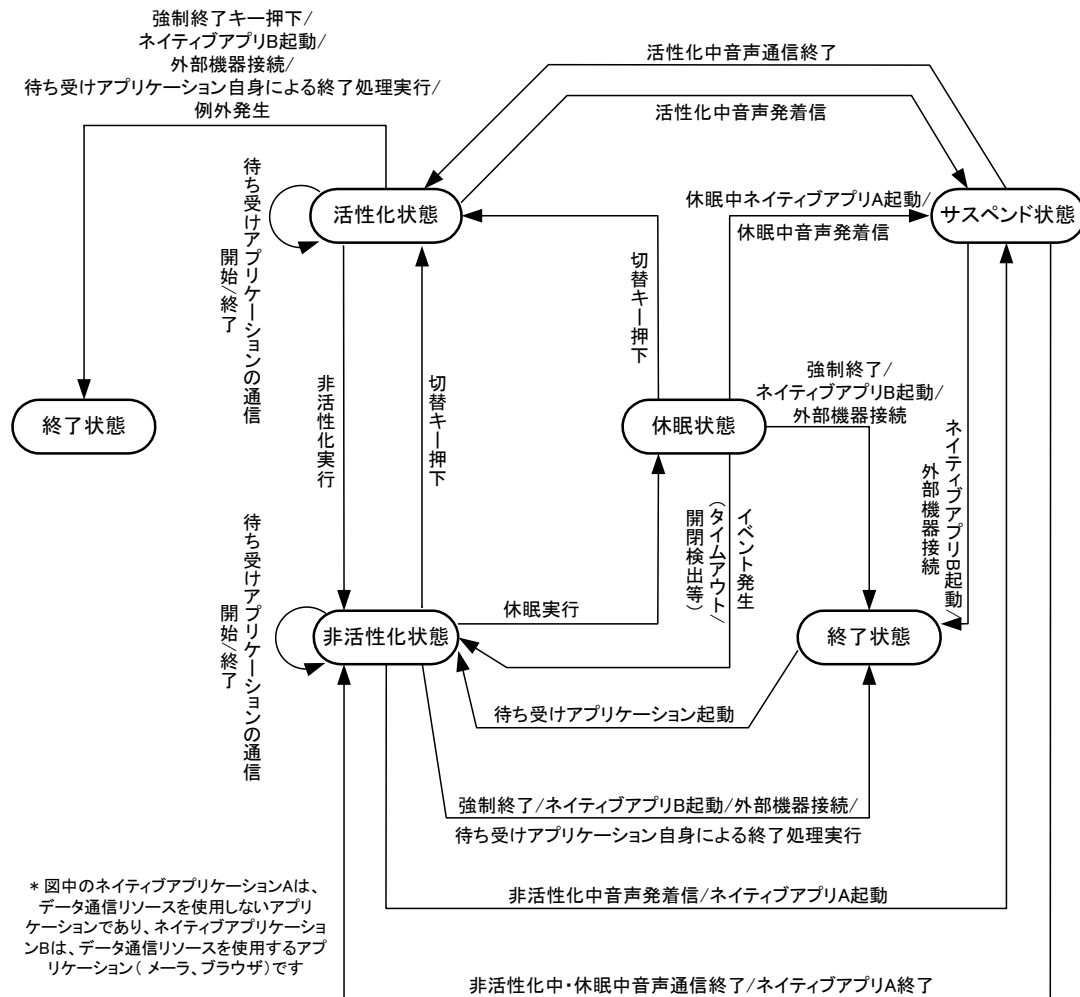
表 4-4 「待ち受け画面」と「待ち受けアプリケーション」の比較

待ち受け画面設定	表示	動作
待ち受け画像 (GIF・JPEG)	静止画またはアニメ画像	キー押下、開閉に合わせて アニメ表示
待ち受けアプリケーション	待ち受けアプリケーションに よる描画	特定キー押下でキーイベン トを有効化

待ち受けアプリケーションは、通常のアプリケーションと同様にキー入力を受け取る状態（活性化状態）と、待ち受け画面と同様に画面の表示は行いがキー入力は受け取らない状態（非活性化状態）及び、一時停止した状態（休眠状態あるいはサスペンド状態）の3つの状態を持っています。待ち受けアプリケーションが実行中であっても、非活性化状態及び休眠状態では端末のネイティブ機能を優先します。

待ち受けアプリケーションが非活性化状態及び休眠状態では、キーイベントの受信とソフトキーの表示は行えず、全てネイティブアプリケーション側で処理します。一方、活性化状態ではキーイベントの受信とソフトキーの表示は、待ち受けアプリケーションで行います。

図 4-18 に待ち受けアプリケーションの状態遷移を示します。



切替キー：待ち受けボタン

図 4-18 待ち受けアプリケーションの状態遷移

待ち受けアプリケーションのソースファイルは、「プロジェクト」→「ソースファイルの新規作成...」で、テンプレートを「待ち受けアプリケーション」にすると、簡単に作成することができます（「4.2.1 ソースファイルの作成」参照）。

4.8.2. 待ち受け起動手順

エミュレータで待ち受けアプリケーションを起動するには：

1. メイン画面のプルダウンから「待ち受け起動」を選択します（図 4-19）。

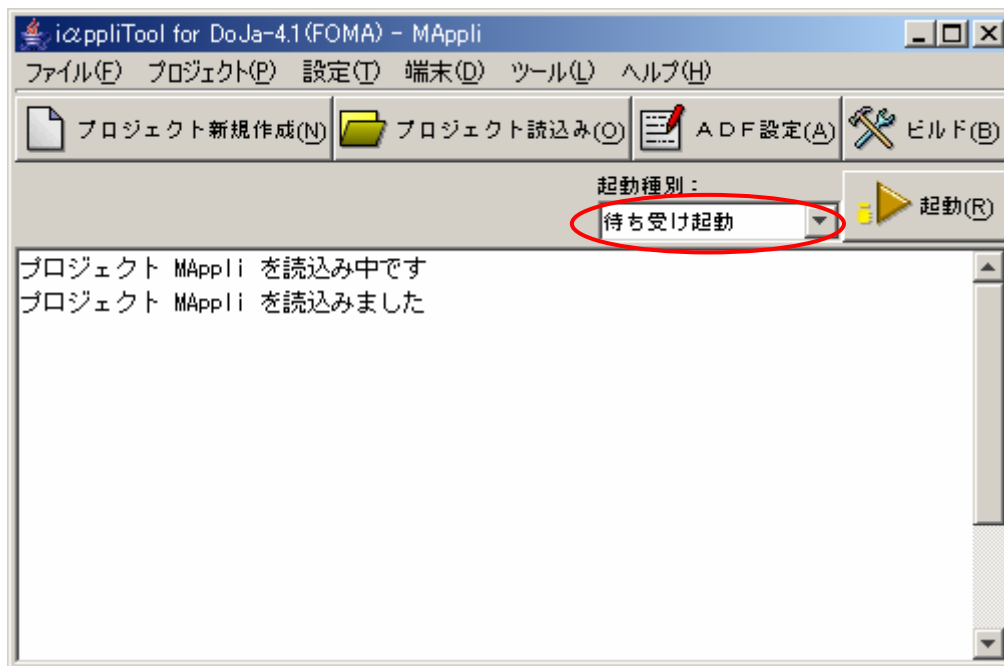


図 4-19 待ち受け起動準備

2. メニューから「プロジェクト」→「起動...」を選択するか、「起動」ボタンを押します。
i アプリを待ち受けアプリケーションとして起動するためには、以下の条件を満たしている必要があります：

- i アプリを待ち受けアプリケーションとして作成していること（com.nttdocomo.ui.MApplication クラスを継承していること）。
- ADFの項目で「MyConcierge」に「Yes」が設定されていること（「4.7.アプリケーションディスクリプタファイル（ADF）の編集」を参照）。
- メーラ（「4.16.メーラ起動」を参照）、ブラウザ（「4.17.ブラウザ起動」を参照）が起動していないこと。

com.nttdocomo.ui.MApplicationクラスの継承の有無とMyConciergeの設定の有無との関係を表 4-5 に示します。

表 4-5 MApplication クラス継承の有無と MyConcierge 設定の有無の関係

	MyConcierge 設定有	MyConcierge 設定無
MApplication 継承有	起動	エラー
MApplication 継承無	エラー	エラー

上記の条件を満たしていない場合には、i アプリの起動を禁止するダイアログあるいはコンソールにエラーメッセージが表示されます。

上記の条件を満たしている場合には、i アプリを待ち受けアプリケーションとして起動します（図 4-20）。



図 4-20 待ち受けアプリケーション起動

3. 待ち受けアプリケーションが「非活性化状態」あるいは「休眠状態」にあるときには、以下のアプリケーションが起動可能です：
 - 「待ち受け起動」以外の別の i アプリ
 - メーラ（「4.16. メーラ起動」を参照）
 - ブラウザ（「4.17. ブラウザ起動」を参照）
 - コードリーダー（「4.18. コードリーダー起動」を参照）

また、上記のアプリケーションが終了すると自動的に待ち受けアプリケーションが「非活性化状態」で再起動されます。

以下（図 4-2 1）は、待ち受けアプリケーションのソースコードのサンプルです。
（このソースコードの場合は、
C:\iDKDoJa4.1\apps\<プロジェクト名>\src\MAppli.java を作成します。）

```

/**
 * 待ち受けアプリケーション
 */
import com.nttdocomo.ui.*;

public class MAppli extends MApplication implements SoftKeyListener {
    Panel        p;
    MediaImage    image;
    ImageLabel    imageLb;
    Label         lb;
    boolean       flag = false;

    public void start() {
        try {
            // イメージとラベルの表示
            p = new Panel();
            lb = new Label("[待受] を押してください。");
            imageLb = new ImageLabel();
            image = MediaManager.getImage("resource:///sample1.gif");
            image.use();
            imageLb.setImage(image.getImage());
            HTMLLayout lm = new HTMLLayout();
            p.setLayoutManager(lm);
            lm.begin(HTMLLayout.CENTER);
            p.add(imageLb);
            lm.br();
            p.add(lb);
            lm.end();
            p.setSoftKeyListener(this);
        } catch (Exception e) {
        }
        Display.setCurrent(p);
    }

    /**
     * システムイベント
     */
    public void processSystemEvent(int type, int param) {
        if (type == MODE_CHANGED_EVENT) {
            p.setSoftLabel(Frame.SOFT_KEY_1, "戻る");
            lb.setText("活性化状態");
            flag = false;
        }
        // フリップ変更イベント
        else if (type == FOLD_CHANGED_EVENT) {
            if (param == 0) { // 閉じる
                if (flag) {
                    lb.setText("休眠状態");
                    sleep(); // 休眠状態へ（省電力モードにする）
                    lb.setText("非活性化状態");
                }
            }
        }
    }

    /**
     * ソフトキーイベント
     */
    public void softKeyPressed(int no) {
    }
    public void softKeyReleased(int no) {
        // ソフトキー 1 押下
        if (no == Frame.SOFT_KEY_1) {
            p.setSoftLabel(Frame.SOFT_KEY_1, "");
            lb.setText("非活性化状態");
            deactivate(); // 非活性化状態
            flag = true;
        }
    }
}

```

図 4-2 1 （例）待ち受けアプリケーションソースコード

（lib\Multimedia\sample1.gif を apps\<プロジェクト名>\res にコピーします。）

4.9. 受信フォルダ起動

受信フォルダ起動とは、com.nttdocomo.ui.IApplication クラスを継承している i アプリをメールの受信フォルダから起動する方法で「通常起動」とは以下の点が異なります：

- 「受信フォルダ起動」の起動タイプは「LAUNCHED_MSG_RECEIVED」です。
- ADFの設定項目「MessageCode」、「TrustedAPID」に適切な値を設定し、「トラステッド動作設定」→「メッセージ i アプリ機能の許可」を yes にした場合のみ起動可能となります（「4.7. アプリケーションディスクリプタファイル（ADF）の編集」参照）。

実際の端末において、「通常起動」とは通常のメニューからの起動であり、「受信フォルダ起動」とは受信メールボックスのメールから起動した場合に相当するものです。

受信フォルダ起動するには、プロジェクトを読み込み、ビルド、ADFの設定を行った後、図 4-2 2 に示すように、メイン画面のプルダウンから「受信フォルダ起動」を選択し、「起動」ボタンを押します。

また、起動の際の制約などに関しては「通常起動」の場合と同様です（「4.3. プロジェクトのビルドと通常起動」参照）。

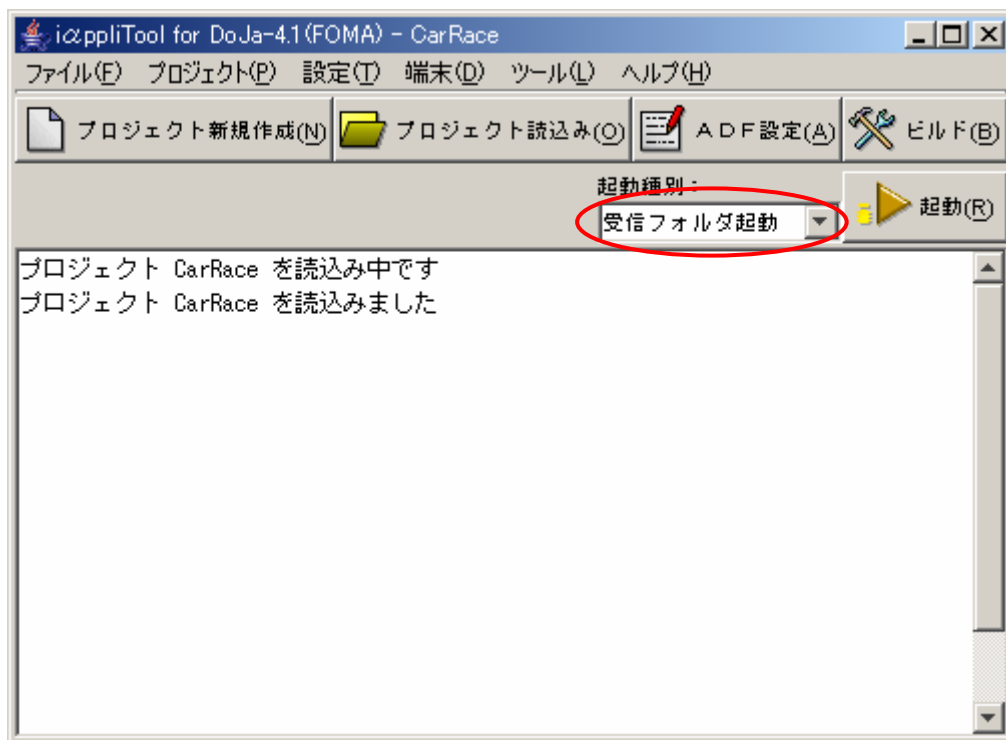


図 4-2 2 受信フォルダ起動準備

4.10. 送信フォルダ起動

送信フォルダ起動とは、com.nttdocomo.ui.IApplication クラスを継承している i アプリをメールの送信フォルダから起動する方法で「通常起動」とは以下の点が異なります：

- 「送信フォルダ起動」の起動タイプは「LAUNCHED_MSG_SENT」です。
- ADFの設定項目「MessageCode」、「TrustedAPID」に適切な値を設定し、「トラステッド動作設定」→「メッセージ i アプリ機能の許可」を yes にした場合のみ起動可能となります（「4.7. アプリケーションディスクリプタファイル（ADF）の編集」参照）。

実際の端末において、「通常起動」とは通常のメニューからの起動であり、「送信フォルダ起動」とは送信メールボックスのメールから起動した場合に相当するものです。

送信フォルダ起動するには、プロジェクトを読み込み、ビルド、ADFの設定を行った後、図 4-23 に示すように、メイン画面のプルダウンから「送信フォルダ起動」を選択し、「起動」ボタンを押します。

また、起動の際の制約などに関しては「通常起動」の場合と同様です（「4.3. プロジェクトのビルドと通常起動」参照）。

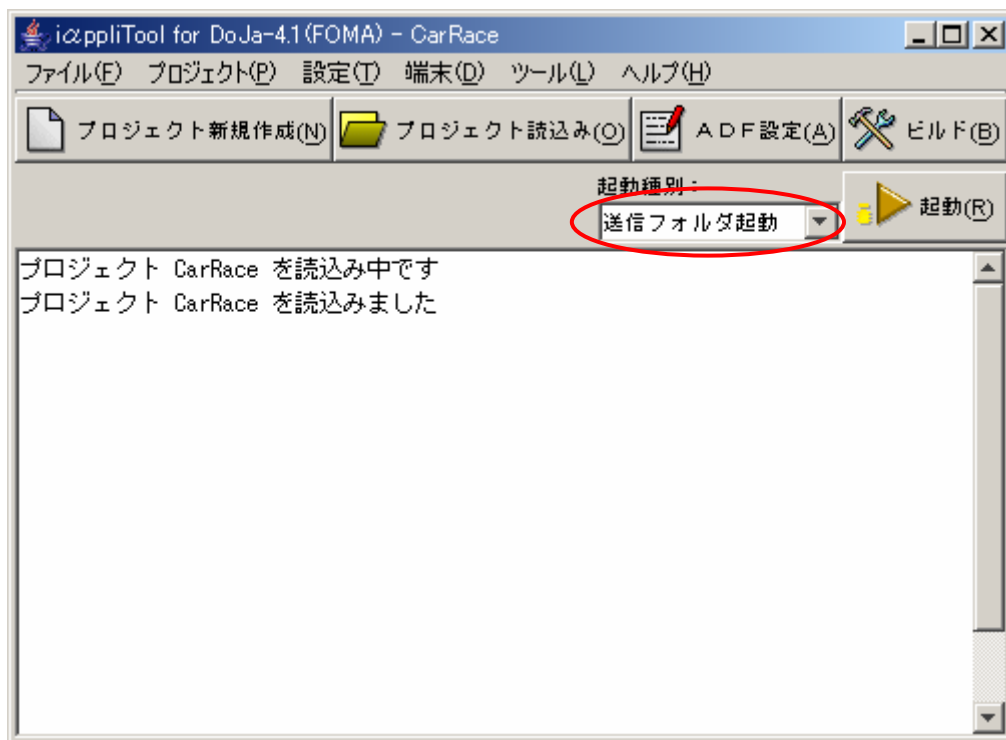


図 4-23 送信フォルダ起動準備

4.11. 未送信フォルダ起動

未送信フォルダ起動とは、com.nttdocomo.ui.IApplication クラスを継承している i アプリをメールの未送信フォルダから起動するで「通常起動」とは以下の点が異なります：

- 「未送信フォルダ起動」の起動タイプは「LAUNCHED_MSG_UNSENT」です。
- ADFの設定項目「MessageCode」、「TrustedAPID」に適切な値を設定し、「トラステッド動作設定」→「メッセージ i アプリ機能の許可」を yes にした場合のみ起動可能となります（「4.7. アプリケーションディスクリプタファイル（ADF）の編集」参照）。

実際の端末において、「通常起動」とは通常のメニューからの起動であり、「未送信フォルダ起動」とは未送信メールボックスのメールから起動した場合に相当するものです。

未送信フォルダ起動するには、プロジェクトを読み込み、ビルド、ADFの設定を行った後、図 4-2 4 に示すように、メイン画面のプルダウンから「未送信フォルダ起動」を選択し、「起動」ボタンを押します。

また、起動の際の制約などに関しては「通常起動」の場合と同様です（「4.3. プロジェクトのビルドと通常起動」参照）。

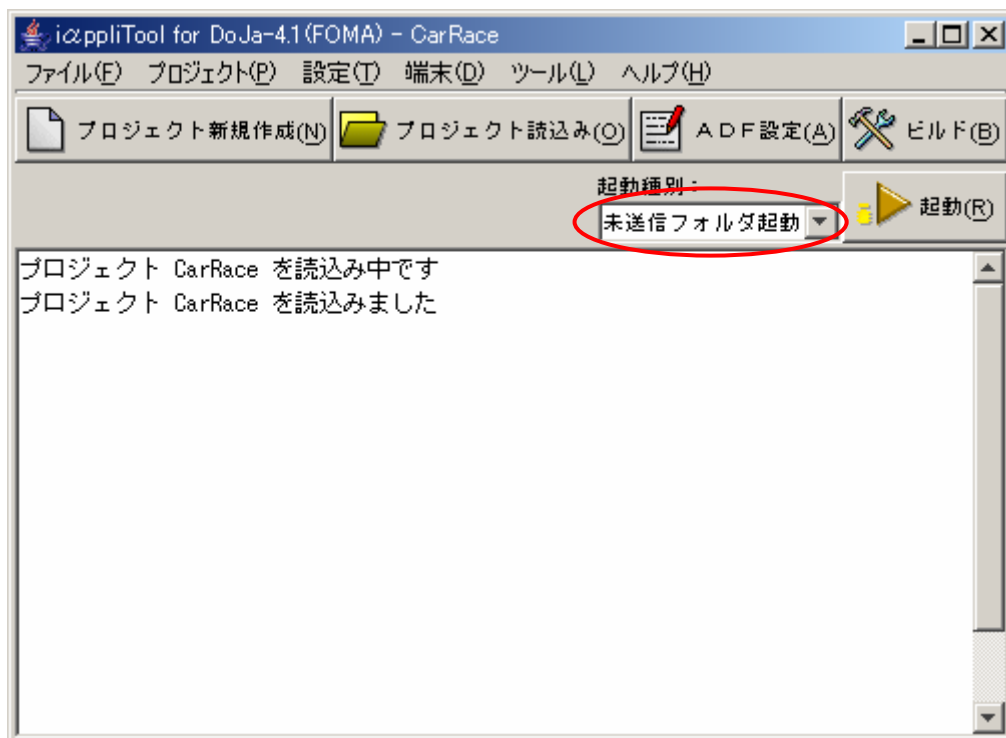


図 4-2 4 未送信フォルダ起動準備

4.12. 初回起動

初回起動とは、`com.nttdocomo.ui.IApplication` クラスを継承している i アプリを始めて起動する方法で「通常起動」とは以下の点が異なります：

- 「初回起動」の起動タイプは「LAUNCHED_AFTER_DOWNLOAD」です。

実際の端末において、「通常起動」とは通常のメニューからの起動であり、「初回起動」とはダウンロード直後（通常起動の1回目）の起動に相当するものです。

初回起動するには、プロジェクトを読み込み、ビルド、ADFの設定を行った後、図 4-2 5 に示すように、メイン画面のプルダウンから「初回起動」を選択し、「起動」ボタンを押します。

また、起動の際の制約などに関しては「通常起動」の場合と同様です（「4.3. プロジェクトのビルドと通常起動」参照）。

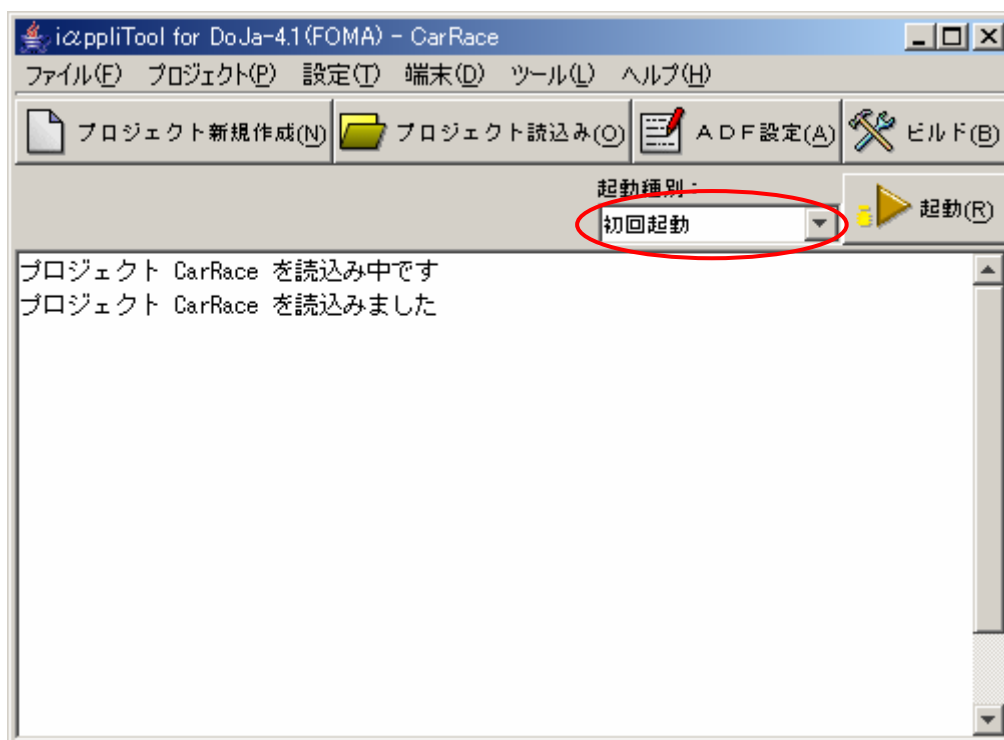


図 4-2 5 初回起動準備

4.13. タイマ起動

タイマ起動とは、com.nttdocomo.ui.IApplication クラスを継承している i アプリをタイマによって起動する方法で「通常起動」とは以下の点が異なります：

- 「タイマ起動」の起動タイプは「LAUNCHED_FROM_TIMER」です。
- ADFの設定項目「LaunchAt」に適切な値を設定した場合のみ起動可能となります（「4.7. アプリケーションディスクリプタファイル（ADF）の編集」参照）。

実際の端末において、「通常起動」とは通常のメニューからの起動であり、「タイマ起動」とはタイマエージェントにより指定された時刻の起動に相当するものです。

タイマ起動するには、プロジェクトを読み込み、ビルド、ADFの設定を行った後、図 4-2 6 に示すように、メイン画面のプルダウンから「タイマ起動」を選択し、「起動」ボタンを押します。

また、起動の際の制約などに関しては「通常起動」の場合と同様です（「4.3. プロジェクトのビルドと通常起動」参照）。

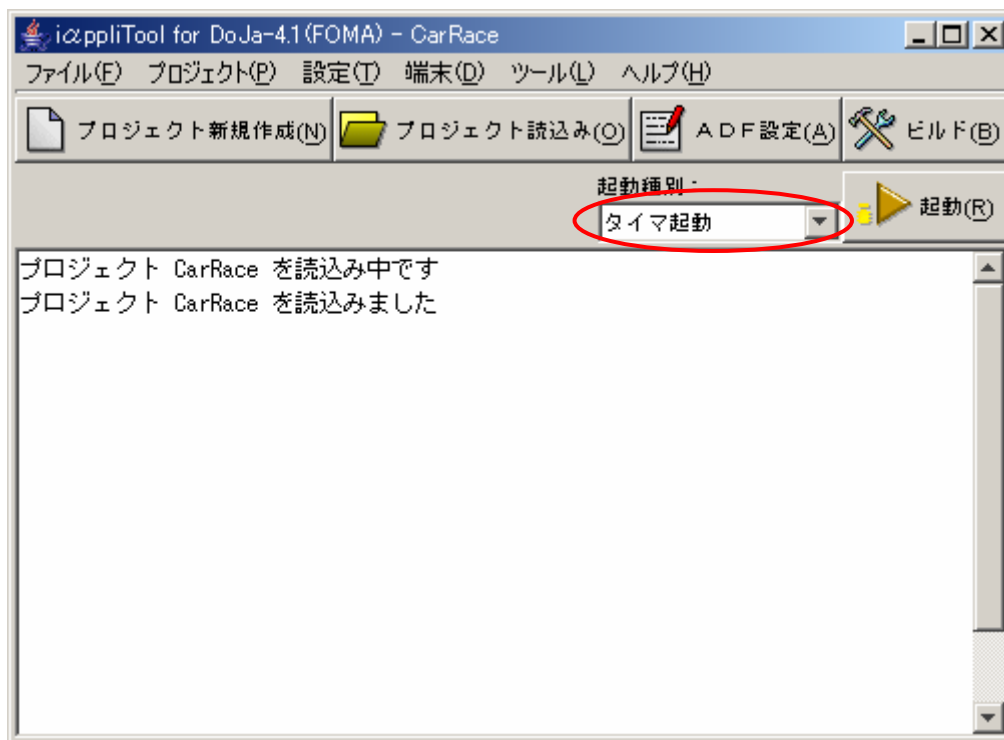


図 4-2 6 タイマ起動準備

4.14. 外部機器起動

外部機器起動とは、com.nttdocomo.ui.IApplication クラスを継承している i アプリを外部機器から起動する方法で「通常起動」とは以下の点が異なります：

- 「外部機器起動」の起動タイプは「LAUNCHED_FROM_EXT」です。
- ADFの設定項目「AllowPushBy」に適切な値を設定した場合のみ起動可能となります（「4.7. アプリケーションディスクリプタファイル（ADF）の編集」参照）。

実際の端末において、「通常起動」とは通常のメニューからの起動であり、「外部機器起動」とは外部インタフェースのトリガによる起動に相当するものです。

外部機器起動するには、プロジェクトを読み込み、ビルド、ADFの設定を行った後、図 4-2 7 に示すように、メイン画面のプルダウンから「外部機器起動」を選択し、「起動」ボタンを押します。

また、起動の際の制約などに関しては「通常起動」の場合と同様です（「4.3. プロジェクトのビルドと通常起動」参照）。

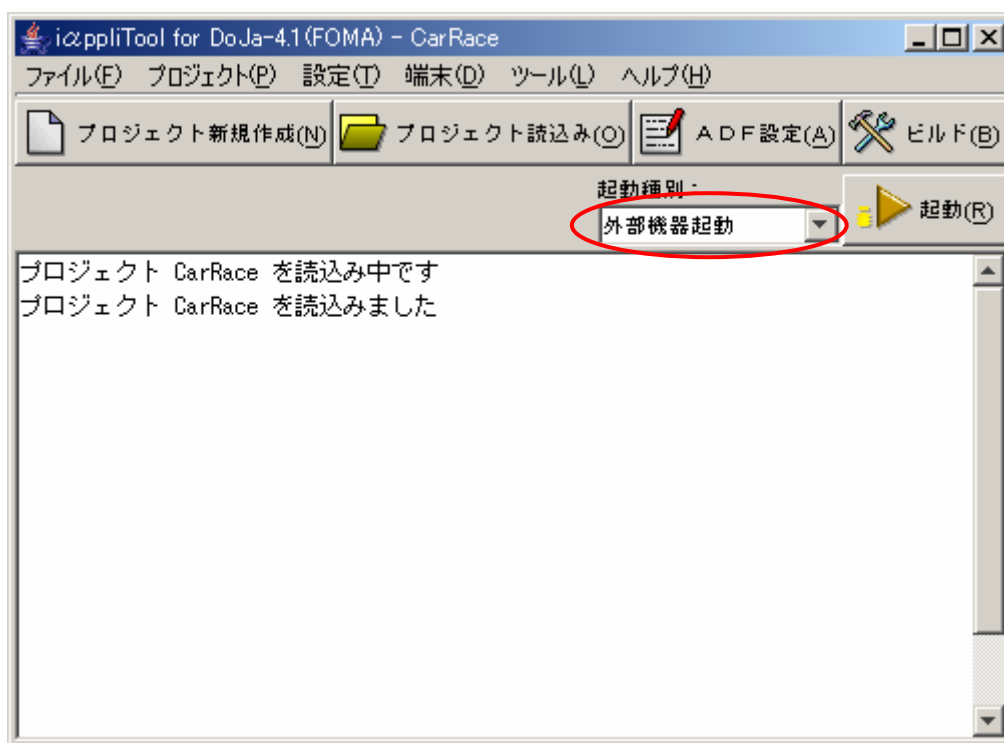


図 4-2 7 外部機器起動準備

4.15. 即起動

「即起動」とは、ダウンロード後に i アプリを即起動することです。

「即起動」の起動タイプは、「LAUNCHED_AS_ILET」です。

「即起動」で起動された i アプリは、以下の機能を利用することができません：

- A D F の"GetSysInfo" で許可される機能
- A D F の"GetUtn" で許可される機能
- A D F の"AccessUserInfo" で許可される機能
- A D F の"LaunchApp" で許可される機能
- A D F の"TrustedAPID" が指定される機能
(トラステッド動作設定で "Yes" に設定する機能)
- スクラッチパッドの利用
- i アプリのバージョンアップ
- i アプリの保存

上記の条件であった場合には、例外が発生します。

4.15.1. 即起動手順

1. メイン画面のプルダウンから「即起動」を選択します（図 4-28）。

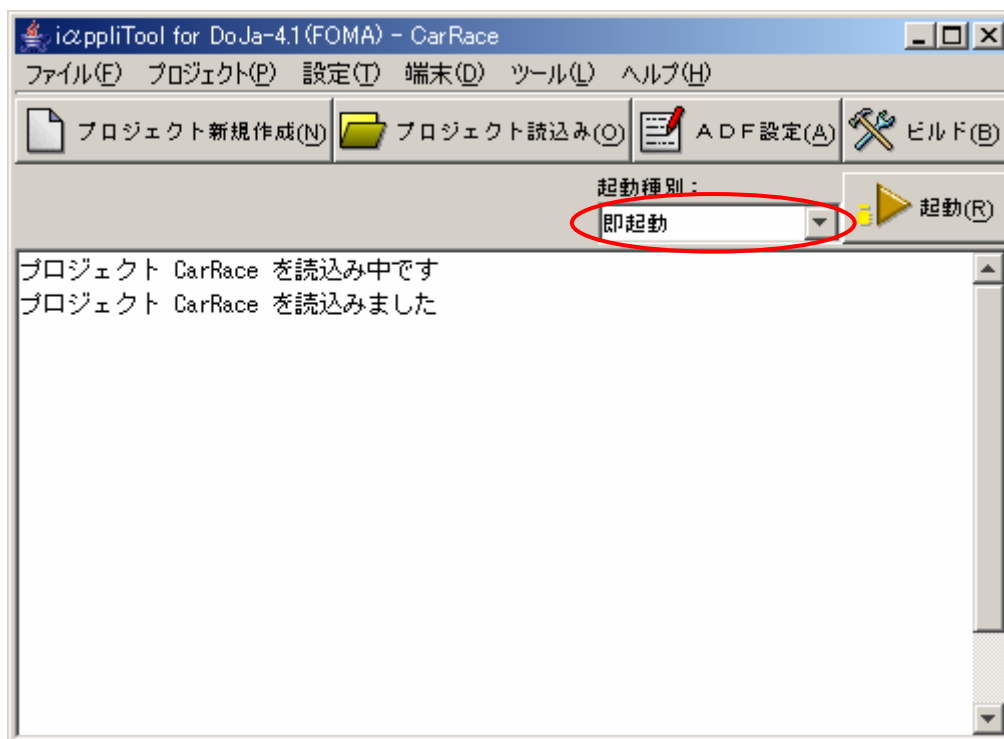


図 4-28 即起動準備

2. 「起動」ボタンを押します。

4.16. メーラ起動

「メーラ起動」とは、i アプリをメーラから起動することです。

「メーラ起動」の起動タイプは、「LAUNCHED_FROM_MAILER」です。

4.16.1. メーラ起動手順

エミュレータで i アプリをメーラから起動するには：

1. メイン画面のプルダウンから「メーラ起動」を選択します（図 4-29）。

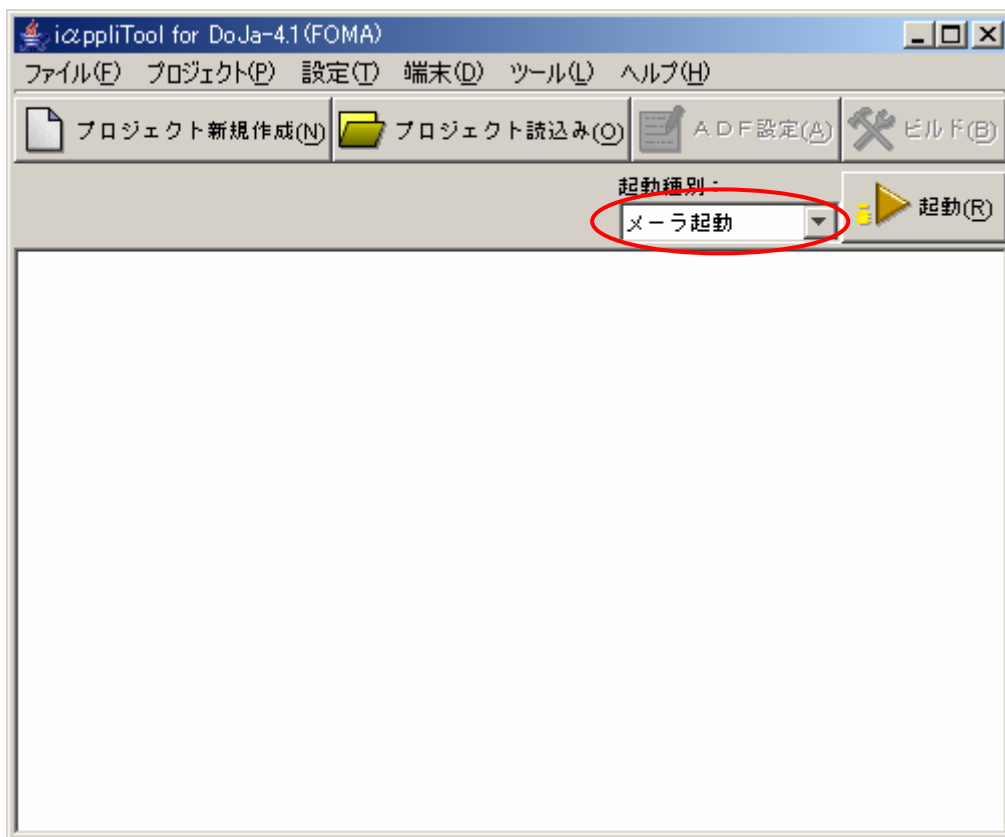


図 4-29 メーラ起動準備

2. メーラ起動ファイルを編集して起動するアプリを追記します（「9.1. メーラ起動ファイル」を参照）。

3. 「起動」ボタンを押します。

メーラ画面が表示されます（図 4-30）。このとき、エミュレータが以下の状態の時にはメーラ画面は表示されません:

- 既にメーラが起動している。
- ブラウザ（「4.17. ブラウザ起動」を参照）が起動している。
- コードリーダー（「4.18. コードリーダー起動」を参照）が起動している。

メーラ画面の上方には、受信メールの一覧が表示されます（メーラ起動ファイル *.mail（*は0、1、2、・・・）のヘッダ部分が表示されます）。

受信メールの一覧からメールを選択すると、メーラ画面の下方に選択したメールの内容が表示されます。

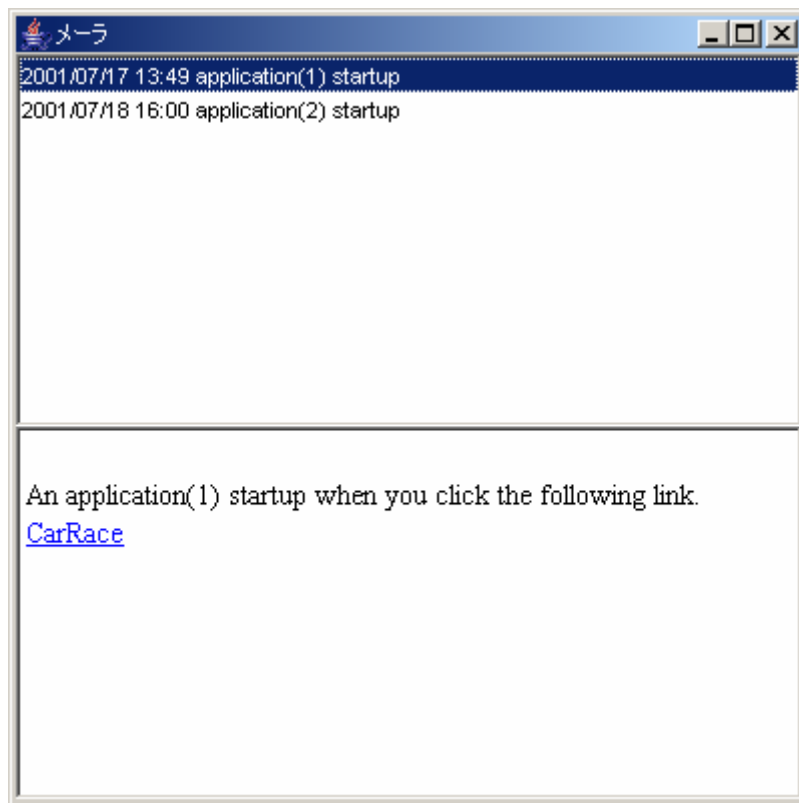


図 4-30 メーラ画面

4. メーラ画面の下方に表示されているリンク（青い文字）をクリックします。

i アプリをメーラから起動するためには、以下の条件を満たしている必要があります:

- ADFの項目で「LaunchByMail」に「any」あるいは起動を許可するメールアドレスを設定し（「4.7. アプリケーションディスクリプタファイル（ADF）の編集」を参照）、そのメールアドレスがメールのFromと一致していること。
- メーラ起動ファイルに起動する i アプリが正しく記述されていること（「9.1. メーラ起動ファイル」を参照）。
- i アプリがデフォルトの格納先（「C:\iDKDoJa4.1\apps」）で作成されていること。
これは、エミュレータでのURLの参照先がデフォルトの格納先であるためです。

上記の条件を満たしていない場合には、i アプリの起動を禁止するダイアログあるいはコンソールにエラーメッセージが表示されます。

上記の条件を満たしている場合には、i アプリをメーラ起動するための確認ダイアログが表示され（図 4-3 1）、「はい」ボタンを押すと i アプリが起動します（図 4-3 2）。

メーラから待ち受けアプリケーションを起動した場合、待ち受けアプリケーションとしては動作せず通常のアプリケーションとして動作します。

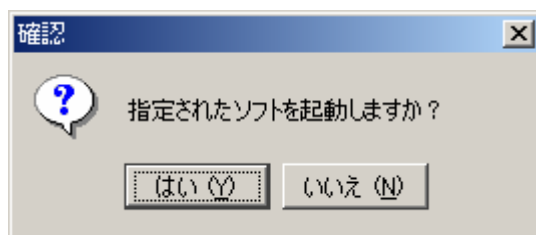


図 4-3 1 起動確認ダイアログ



図 4-3 2 メーラ起動された i アプリ

5. メーラ起動された i アプリの終了方法により、その後、自動起動する i アプリが異なります：
 - メーラ起動された i アプリを正常終了（i アプリが `IApplication.terminate()` メソッドを呼び出して終了した場合）

再びメーラ画面が表示されます。メーラ画面を終了する（画面の右上の「×」を押す）と、もし、メーラを起動する際に待ち受けアプリケーションが動作していたのであればそれを起動し、そうでなければ通常の待ち受け画面に戻ります。

- メーラ起動された i アプリを強制終了（端末の「終話ボタン」を押した場合）

メーラ画面は表示されず、もし、メーラを起動する際に待ち受けアプリケーションが動作していたのであればそれを起動し、そうでなければ通常の待ち受け画面に戻ります。

4.17. ブラウザ起動

「ブラウザ起動」とは、i アプリをブラウザから起動することです。
「ブラウザ起動」の起動タイプは、「LAUNCHED_FROM_BROWSER」です。

4.17.1. ブラウザ起動手順

エミュレータで i アプリをブラウザから起動するには：

1. メイン画面のプルダウンから「ブラウザ起動」を選択します（図 4-3 3）。

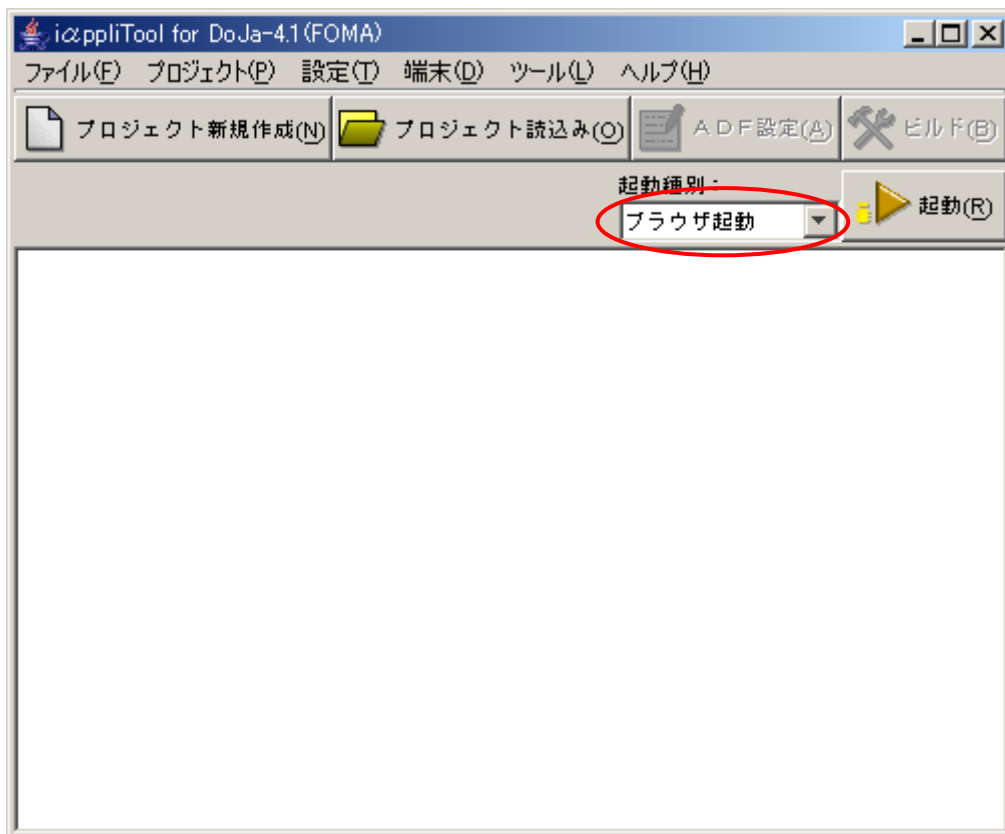


図 4-3 3 ブラウザ起動準備

2. 「起動」ボタンを押します。

ブラウザ画面が表示されます（図 4-3 4）。このとき、エミュレータが以下の状態の時にはブラウザ画面は表示されません:

- 既にブラウザが起動している。
- メーラ（「4.1 6 メーラ起動」を参照）が起動している。
- コードリーダー（「4.1 8 コードリーダー起動」を参照）が起動している。

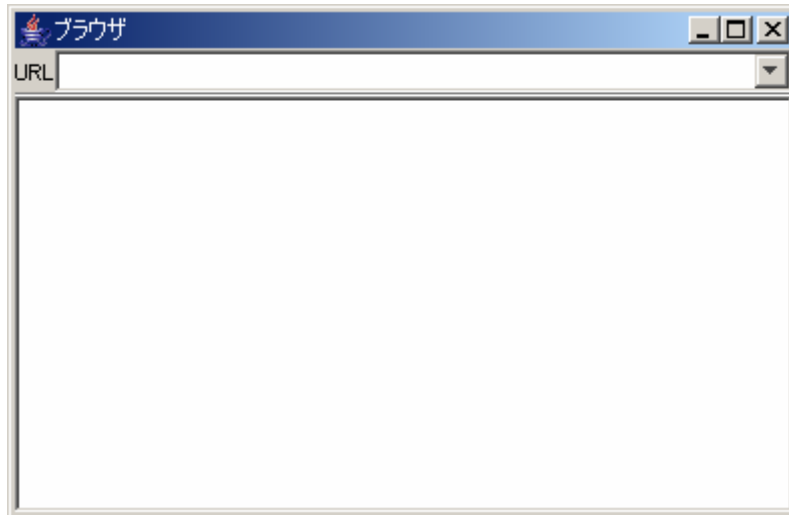


図 4-3 4 ブラウザ画面

3. ブラウザに表示するHTMLファイルを編集して起動するアプリを追記します（「9. 4. ブラウザ起動HTMLファイルの記述方法」参照）。
4. 図 4-3 4 の上方のURLフィールドに3. で編集したHTMLのURLを入力し、リターンキーを押します（図 4-3 5）。



図 4-3 5 URL の表示

- 図の URL は、同一マシン(localhost)に Web サーバ（Apache 等）を導入し、
C:/iDKDoJa4.1/lib/sample.html を用いた場合です。
もし、Webサーバを利用しない場合は、「file:」で指定できます。
その場合は、URLに「**file:C:/iDKDoJa4.1/lib/sample.html**」と指定
します。

5. 図 4-35 のリンク（青い文字）をクリックします。

i アプリをブラウザから起動するためには、以下の条件を満たしている必要があります：

- ADFの項目で「LaunchByBrowser」に「any」あるいは起動を許可するURLが
設定されていること（「4.7.アプリケーションディスクリプタファイル
（ADF）の編集」を参照）。
- HTMLファイルに起動するアプリが正しく指定されていること（「9.4.
ブラウザ起動HTMLファイルの記述方法」参照）。
- i アプリが「プロジェクト読み込み画面」で指定したプロジェクト格納先に作
成されていること。必ずしもデフォルトの格納先（「C:/iDKDoJa4.1/apps」）
に作成されている必要はありません。

上記の条件を満たしていない場合には、i アプリの起動を禁止するダイアログあるいは
コンソールにエラーメッセージが表示されます。

上記の条件を満たしている場合には、i アプリをブラウザ起動するための確認ダイア
ログが表示され（図 4-36）、「はい」ボタンを押すと i アプリが起動します（図
4-37）。

ブラウザから待ち受けアプリケーションを起動した場合、待ち受けアプリケーション
としては動作せずに通常のアプリケーションとして動作します。



図 4-36 起動確認ダイアログ



図 4-3 7 ブラウザ起動された i アプリ

6. ブラウザ起動された i アプリの終了方法により、その後、自動起動する i アプリが異なります:
 - ブラウザ起動された i アプリを正常終了（i アプリが `IApplication.terminate()` メソッドを呼び出して終了した場合）

再びブラウザ画面が表示されます。ブラウザ画面を終了する（画面の右上の「×」を押す）と、もし、ブラウザを起動する際に待ち受けアプリケーションが動作していたのであればそれを起動し、そうでなければ通常の待ち受け画面に戻ります。

- ブラウザ起動された i アプリを強制終了（端末の「終話ボタン」を押した場合）

ブラウザ画面は表示されず、もし、ブラウザを起動する際に待ち受けアプリケーションが動作していたのであればそれを起動し、そうでなければ通常の待ち受け画面に戻ります。

4.18. コードリーダー起動

「コードリーダー起動」とは、i アプリをコードリーダーから起動することです。
「コードリーダー起動」の起動タイプは、「LAUNCHED_FROM_EXT」です。

4.18.1. コードリーダー起動手順

エミュレータで i アプリをコードリーダーから起動するには：

1. メイン画面のプルダウンから「コードリーダー起動」を選択します（図 4-38）。

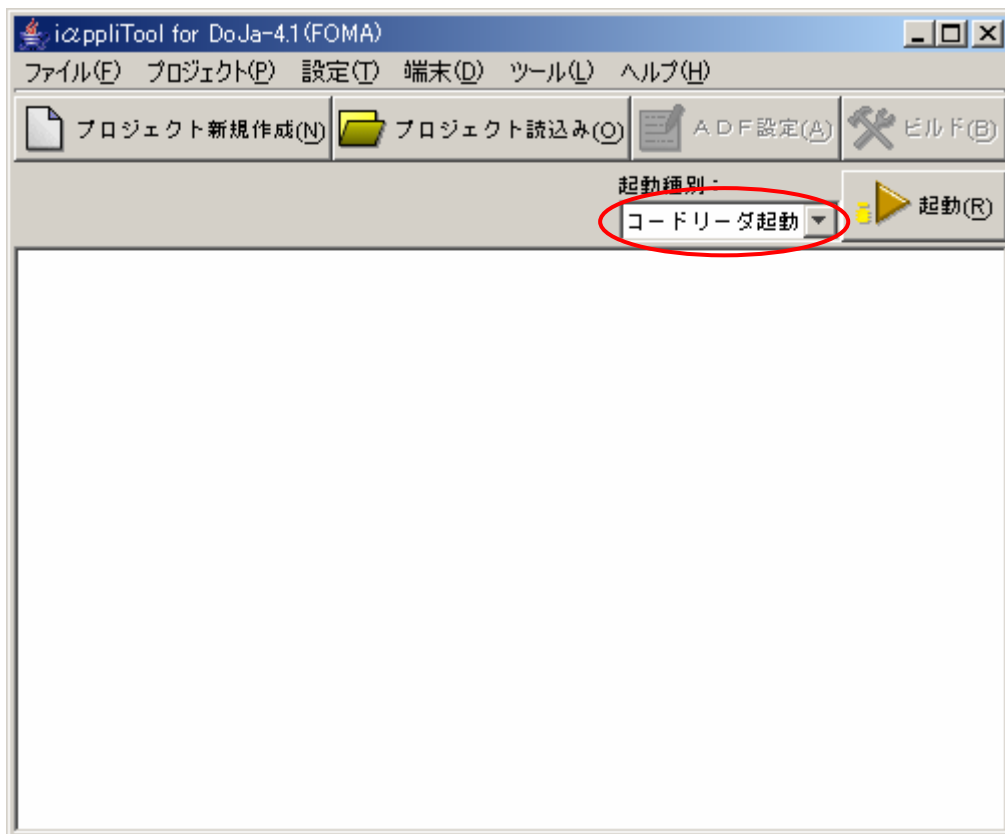


図 4-38 コードリーダー起動準備

2. コード認識結果ファイルを編集して起動するアプリを追記します（「9.13. コード認識結果ファイル」参照）。

3. 「起動」ボタンを押します。

コード認識結果ファイル読み込み画面が表示されます（図 4-3 9）。
このとき、エミュレータが以下の状態の時にはコード認識結果ファイル読み込み画面は表示されません：

- 既にコードリーダーが起動している。
- メーラ（「4.1 6. メーラ起動」を参照）が起動している。
- ブラウザ（「4.1 7. ブラウザ起動」を参照）が起動している。

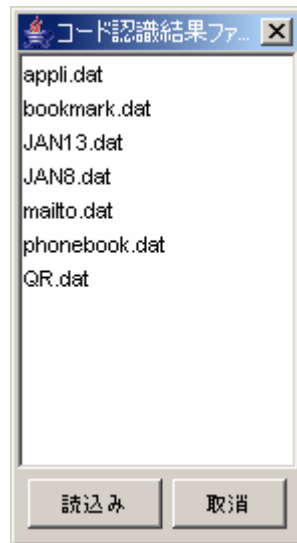


図 4-3 9 コード認識結果ファイル読み込み画面

4. 「図 4-3 9」から認識したいコード認識ファイルを選択して「読み込み」ボタンを押します。

i アプリをコードリーダーから起動するためには、以下の条件を満たしている必要があります：

- ADFの項目で「AllowPushBy」に「Code:起動コマンド」が設定されていること（「4.7. アプリケーションディスクリプタファイル（ADF）の編集」参照）。
- コード認識結果ファイルのCMD:に起動する i アプリが正しく記述されていること（「9.1 3. コード認識結果ファイル」参照）。
- i アプリがデフォルトの格納先（「C:\iDKDoJa4.1\apps」）で作成されていること。
これは、エミュレータでのURLの参照先がデフォルトの格納先であるためです。

上記の条件を満たしていない場合には、i アプリの起動を禁止するダイアログあるいはコンソールにエラーメッセージが表示されます。

上記の条件を満たしている場合には、コード認識表示画面が表示されます（図 4-4 0）。

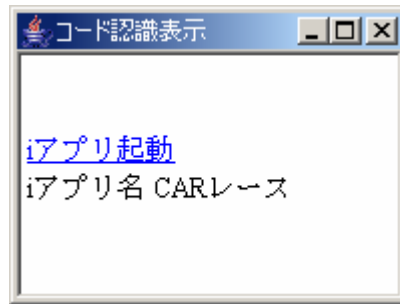


図 4-4 0 コード認識表示画面

5. 図 4-4 0 のリンク（青い文字）をクリックします。

i アプリをコードリーダーから起動するための確認ダイアログが表示され「はい」ボタンを押すと i アプリが起動します（図 4-4 1）。



図 4-4 1 コードリーダー起動された i アプリ

コードリーダーから待ち受けアプリケーションを起動した場合、待ち受けアプリケーションとしては動作せずに通常のアプリケーションとして動作します。

4.18.2. CodeReader の実行手順

「iα ppli Development Kit for DoJa-4.1(FOMA)」で、アプリケーションから CodeReader#read() を実行した場合は、次の手順に従います。

1. CodeReader#read() を実行すると、コード認識結果ファイルを指定するダイアログが表示されます（図 4-4 2）。

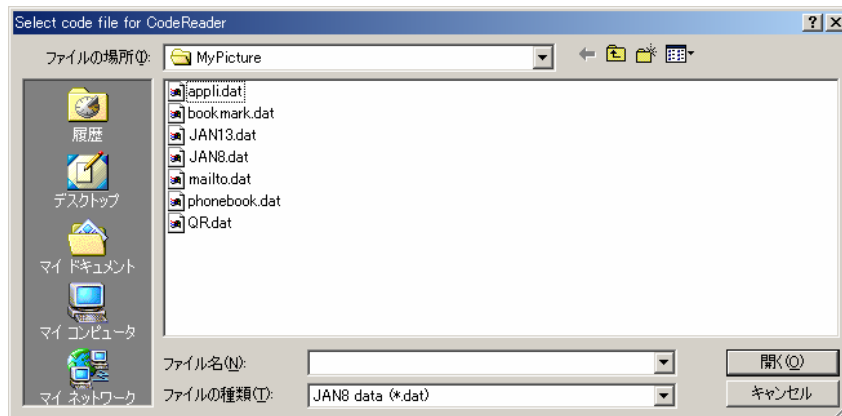


図 4-4 2 コード認識結果ファイル 指定ダイアログ

2. 読み込むコード認識結果ファイルを選択し、「ファイルの種類」から読み込むデータに該当する種類を選択して、「開く」を押します。
3. 読み込んだデータやコード種別が CodeReader に返ります。

4.19. 各種起動方法による画面遷移

これまで「iα ppliTool」での i アプリの起動方法には、以下のような方法があると説明してきました：

- 通常起動
- 待ち受け起動
- 初回起動
- タイマ起動
- 外部機器起動
- 即起動
- 受信フォルダ起動
- 送信フォルダ起動
- 未送信フォルダ起動
- メーラ起動
- ブラウザ起動
- コードリーダー起動

ここで、これらを組み合わせた場合の画面遷移を図 4-4 3 に示します。

「iα ppliTool」実行中、画面に表示されている i アプリは、常に 1 つだけです。

また、それぞれ以下に示す条件で起動する事が可能です：

- メーラが起動していない
- ブラウザが起動していない
- コードリーダーが起動していない

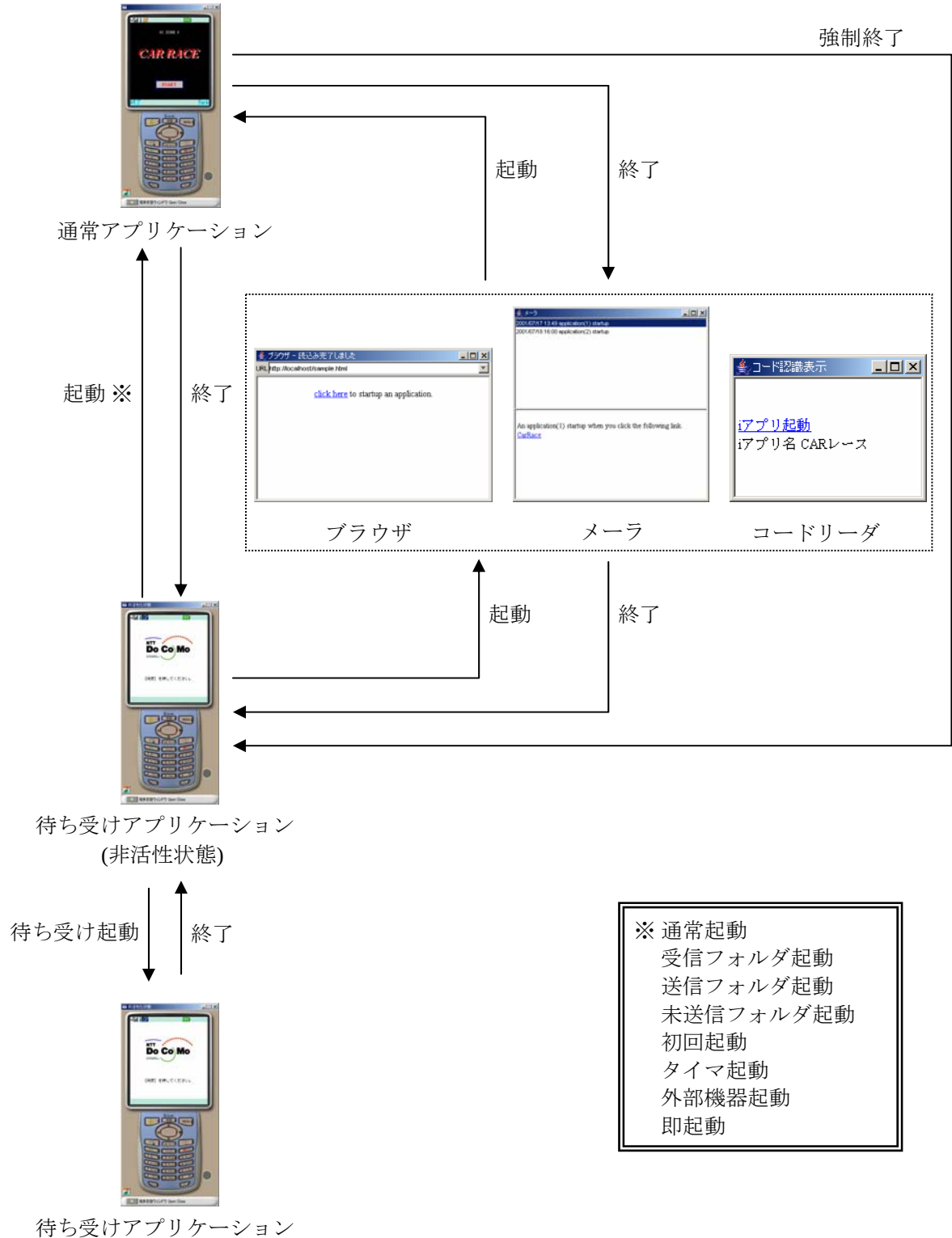


図 4-4 3 「iα ppliTool」における画面遷移

第5章. エミュレータの設定

この章では、エミュレータのいろいろな設定について説明します。

5.1. ネットワークの設定

HTTP通信および赤外線通信のネットワークの設定をします。

ネットワークの設定をするには、

1. メニューから「設定」→「アプリケーション動作環境設定」→「ネットワーク設定」タブを選択します（図 5-1）。

図 5-1 ネットワークの設定画面

2. 各設定項目を設定します。

各設定項目は表 5-1 に示すとおりです。

表 5-1 エミュレータの設定項目

設定項目		説明	デフォルト
ADF の URL		i アプリの ADF が存在する URL。 PackageURL タグに相対 URL が指定された場合、本 URL がベースの URL となります。i アプリはこの PackageURL のホストとポートのみ通信可能です。 このフィールドに指定する URL は、ADF の PackageURL タグのフォーマットと同様です。 また、このフィールドは、アプリケーションごとに設定可能なので、「iα ppliTool」でプロジェクトが読み込まれている場合に設定可能です。さらに、このフィールドに設定した URL は、ファイルに保存されるので、次回「iα ppliTool」起動時に自動的に設定されます。	なし
UserAgent		HTTP の「UserAgent」ヘッダ。 このフィールドに設定した「UserAgent」ヘッダは、レジストリに保存されるので、次回「iα ppliTool」起動時に自動的に設定されます。	DoCoMo/2.0 DOJA_INET_CLIENT(c100;TJ)
HTTP 設定	プロキシ ホスト	i アプリ及び「iα ppliTool」のブラウザが Web サーバとの通信に使用する HTTP プロキシサーバのホスト名。「5.1.1. プロキシサーバの設定」を参照。 また、このフィールドに設定したホスト名は、レジストリに保存されるので、次回「iα ppliTool」起動時に自動的に設定されます。	なし
	プロキシ ポート	i アプリおよび「iα ppliTool」のブラウザが Web サーバとの通信に使用する HTTP プロキシサーバのポート番号。「5.1.1. プロキシサーバの設定」を参照。 また、このフィールドに設定したポート番号は、レジストリに保存されるので、次回「iα ppliTool」起動時に自動的に設定されます。	なし

HTTPS 設定	プロキシ ホスト	i アプリがWebサーバとの通信に使用するHTTPSプロキシサーバのホスト名。「5.1.1.1. プロキシサーバの設定」を参照。 また、このフィールドに設定したホスト名は、レジストリに保存されるので、次回「iアプリTool」起動時に自動的に設定されます。	なし
	プロキシ ポート	i アプリがWebサーバとの通信に使用するHTTPSプロキシサーバのポート番号。「5.1.1.1. プロキシサーバの設定」を参照。 また、このフィールドに設定したポート番号は、レジストリに保存されるので、次回「iアプリTool」起動時に自動的に設定されます。	なし
OBEX 設定	ホスト	i アプリがOBEXを使用して通信する際のIPアドレス。OBEX over TCPを用いる場合に、接続先ノードのIPアドレスを指定します。指定がない場合は、127.0.0.1（ローカルホスト）にコネクションを確立する試みをします。なお、本設定は、ObexClientConnectionクラスのconnectメソッドで意味をなします。「第11章. OBEXの利用」を参照。 また、このフィールドに設定したホスト名は、レジストリに保存されるので、次回「iアプリTool」起動時に自動的に設定されます。	なし

5.1.1.1. プロキシサーバの設定

エミュレータのもとで動作する i アプリからプロキシサーバ経由で Web サーバに接続する必要がある場合には、エミュレータの設定でプロキシサーバの情報を設定する必要があります。i アプリは、HTTP か HTTPS を使用して Web サーバに接続しますが、同じ i アプリで両方のプロトコルを使用することはできません。

HTTPとHTTPSのホストとポートは、図 5-1 ネットワークの設定画面で設定します。

なお、HTTPSの際に使用する証明書の管理については「第10章. 証明書の管理」を参照してください。

5.2. トラステッド動作の設定

トラステッド動作は、i アプリがトラステッド動作機能を利用する**トラステッドi アプリ（i アプリDX）**である場合に設定します。

トラステッドi アプリでは、セキュリティ上、通常のi アプリで禁じられている以下のような機能を実現することができます。

- 携帯電話本体に登録されたデータ（電話帳、発着信履歴等）との連携
- i アプリダウンロード元以外のサーバとの通信
- i モードメールとの連携
- FeliCa を使った機能（一部を除く）

i モードメニューに掲載されていないコンテンツ提供者は、トラステッドi アプリ（i アプリDX）機能を利用したi アプリコンテンツを配信することは出来ません。

トラステッド動作を設定するには：

1. メニューから「設定」→「トラステッド動作設定...」を選択します。
「トラステッド動作設定画面」が表示されます（図 5-2）。

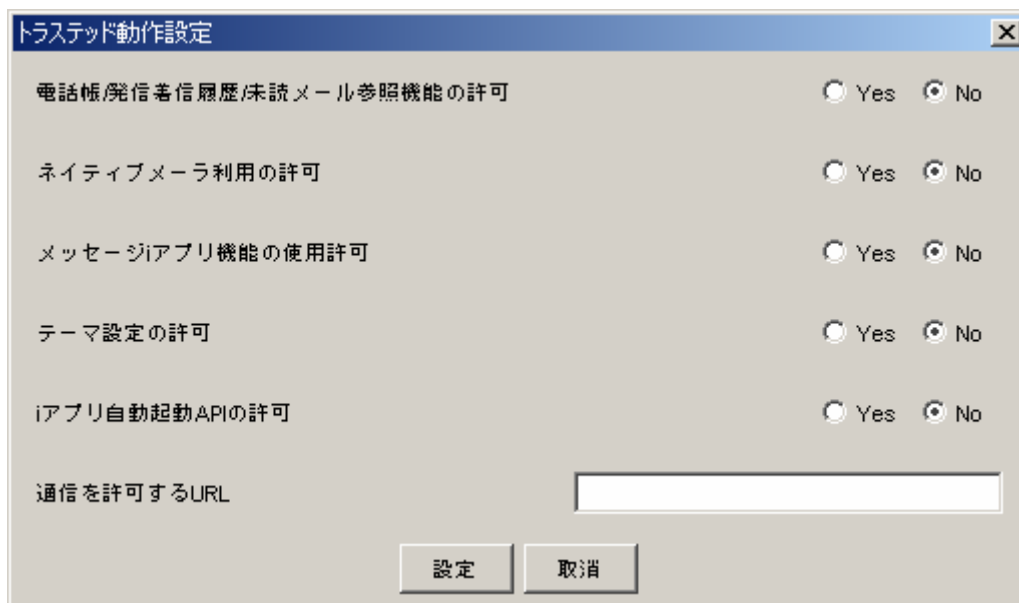


図 5-2 トラステッド動作設定画面

2. 利用したい機能を Yes に設定します。
「設定」ボタンを押すことにより変更が有効となります。

各設定項目は、表 5-2 に示すとおりです。

表 5-2 トラストッド動作設定の設定項目

設定項目	説明
電話帳/発着信履歴/未読メール参照機能の許可	電話帳の参照、電話発着信履歴の参照、メール受信履歴の参照 をする A P I を利用する場合に yes に変更します。 (関連: PhoneBook, PhoneBookGroup, PhoneBookParam, CallRecord)
ネイティブメーラ利用の許可	ネイティブメーラのメールメニューを利用する場合に yes に変更します。 (関連: Mail, MailAgent, MailDraft)
メッセージ i アプリ機能の使用許可	メッセージ i アプリの A P I を利用する場合に yes に変更します。 (関連: Message, MessageAgent, MessageDigest, MessageReceive, MessageSent)
テーマ設定の許可	着信メロディの保存と設定、発着信画像および待ち受け画面画像の保存と設定 をする A P I を利用する場合に yes に変更します。 (関連: SetPhoneTheme, SoundStore)
i アプリ自動起動 A P I の許可	アプリケーションの自動起動をする A P I を利用する場合に yes に変更します。 (関連: IApplication#setLaunchTime)
通信を許可する URL	ダウンロード元以外に i アプリが通信を許可する URL を設定します。(最大 255 バイト) ディレクトリの指定は無効で、設定する文字列は、ASCII 形式とし、255 バイトを超えて設定した場合は、i アプリの起動を禁止します。 (エミュレータでは設定しなくてもエラーになりません。)

ワンポイント

A P I リファレンスにて「トラストッド i アプリ」と指定のある A P I、もしくは、「**SecurityException** - パーミッションとして○○○の参照が許可されているが、i アプリ個別のユーザ設定により許可されない場合に発生します。」という例外が発生する A P I を利用する場合には、この動作を設定する必要があります。

A P I リファレンスは、NTTドコモの次の URL から入手することができます:
http://www.nttdocomo.co.jp/p_s/imode/make/

5.3. メールアドレスの設定

端末に登録された自分のメールアドレスを設定します。

メールアドレスの設定をするには：

1. メニューから「設定」→「アプリケーション動作環境設定」→「メールアドレス設定」タブを選択します（図 5-3）。

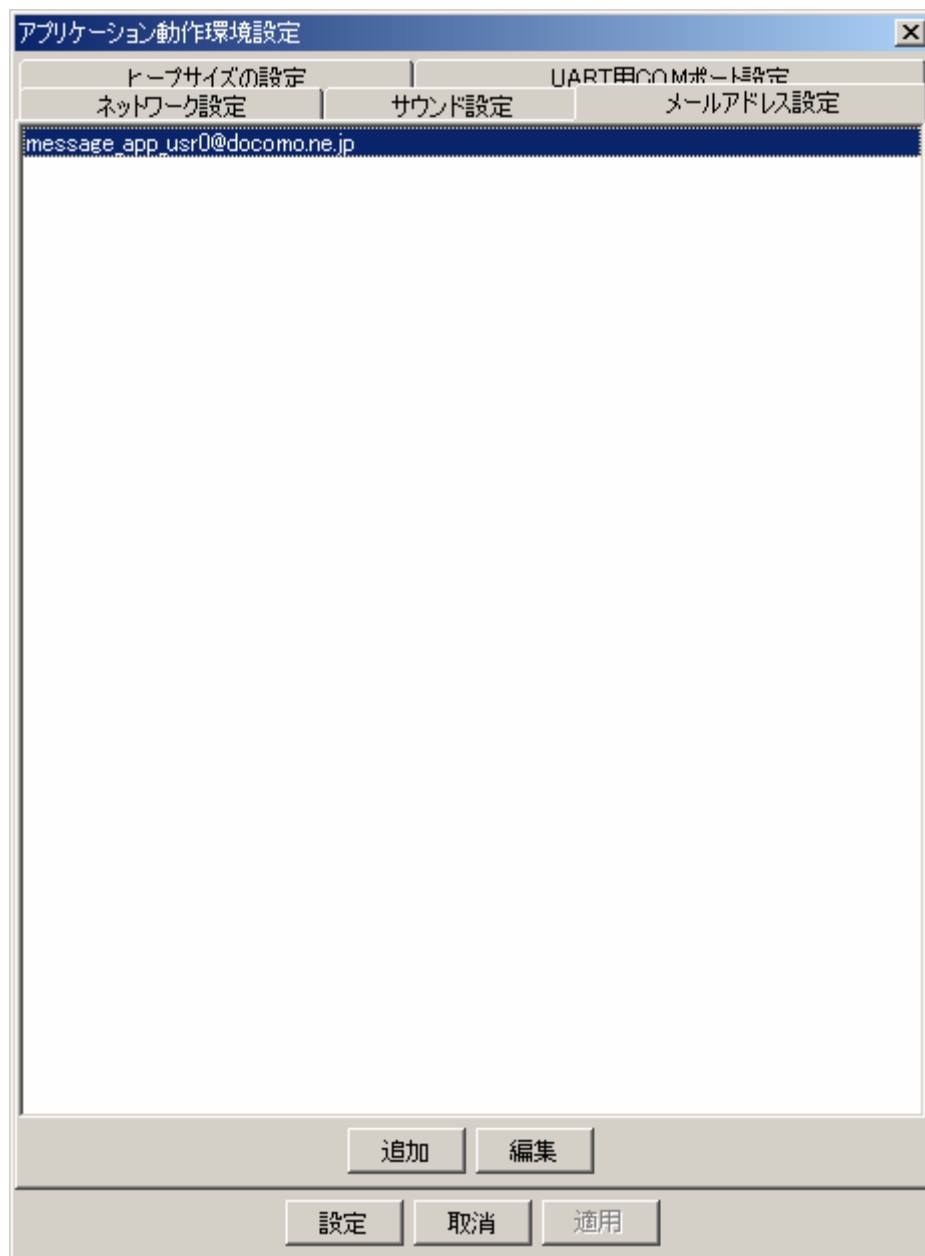


図 5-3 メールアドレスの設定画面

2. メールアドレスを新たに追加する場合は「追加」、変更する場合はメールアドレスを選択して「編集」を押します（図 5-4）。



図 5-4 メールアドレス追加画面

3. メールアドレスを入力して「設定」ボタンを押します。
メールアドレス設定画面の一覧にメールアドレスが追加されます。
4. 移動機に登録するメールアドレスを選択して「設定」ボタンを押します。
デフォルトでは、「message_app_usr0@docomo.ne.jp」が選択されています。
(実際の移動機に登録できるメールアドレスは1つだけです。)

ワンポイント

エミュレータ特有の機能で、ここで作成したメールアドレスに i アプリから送信すると、そのメールアドレスを選択している i アプリでメールを受信することができます（受信ボックスに格納されています）。

これにより、エミュレータでも i アプリ同士で相互にメールをやり取りすることができます。（「5.13.2.メールの送受信エミュレート機能」を参照）

5.4. リモート制御の設定

プッシュ機能により、サーバから i アプリに渡すパラメータを設定します。

(関連 A P I : PushManager)

パラメータの設定をするには：

1. メニューから「設定」→「リモート制御設定...」を選択します（図 5-5）。

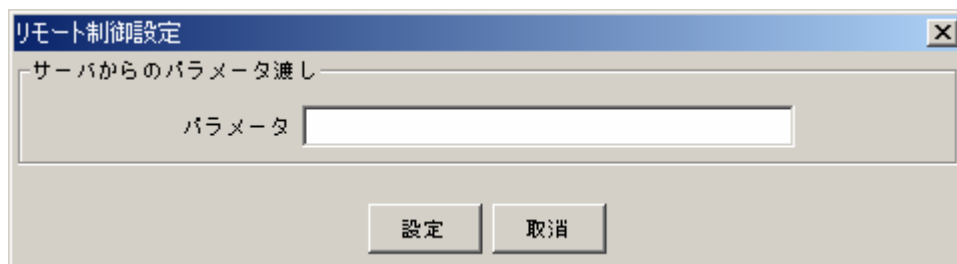


図 5-5 リモート制御パラメータの設定画面

2. パラメータを入力して「設定」ボタンを押します。
「アプリケーション名.properties」ファイルにパラメータが追加されます。

設定したパラメータを i アプリから取得するには：

1. i アプリの起動中に端末ウィンドウを右クリックします。右クリックメニューから「パラメータ受信」を選択すると、パラメータ受信のイベントが発生し、i アプリにてパラメータを取得できます（図 5-6）。

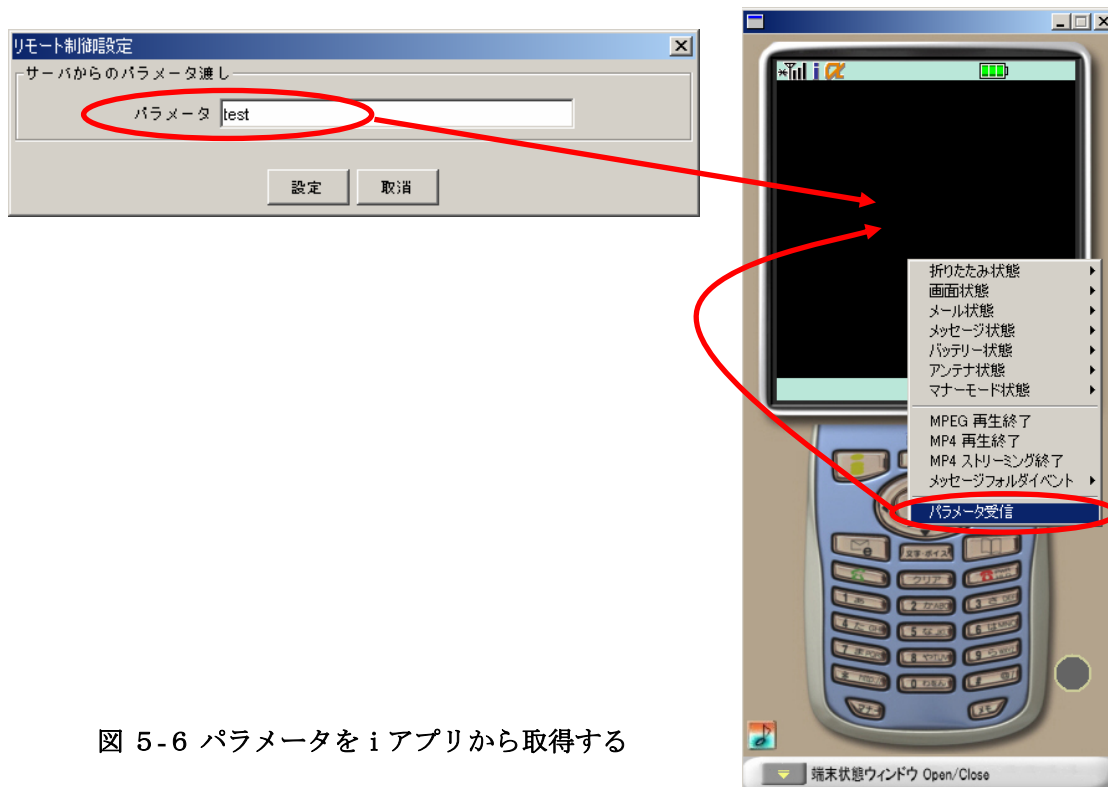


図 5-6 パラメータを i アプリから取得する

5.5. リモート制御のパラメータ領域の削除

「5.4. リモート制御の設定」で設定したパラメータを削除します。

リモート制御のパラメータ領域の削除をするには：

1. メニューから「設定」→「リモート制御のパラメータ領域削除...」を選択すると「アプリケーション名.properties」ファイルからパラメータが削除されます（図5-7）。
これにより、「一度もパラメータを受信していない状態」を試す事ができます。

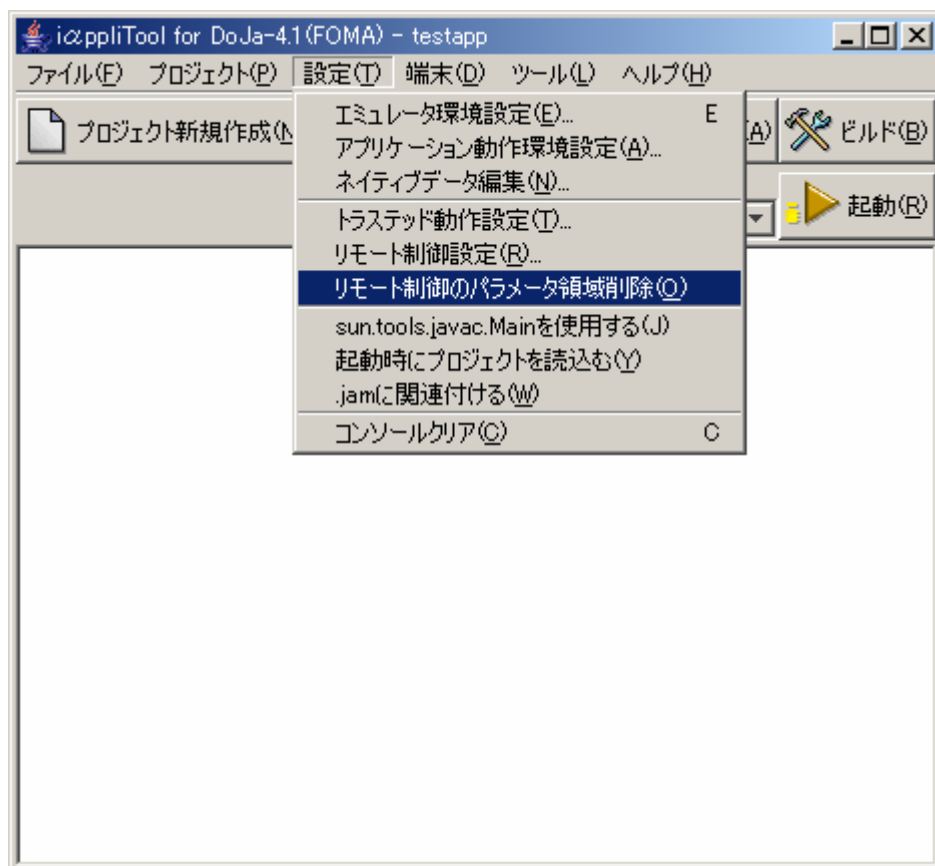


図 5-7 リモート制御パラメータ領域削除

5.6. サウンド設定

サウンドの再生モードの切り替えと、i アプリから利用するMF i のサウンドデータで、UCS 音色など通常のP C上では鳴動できない音の代替音を設定します。

サウンドの再生モードが、「通常再生モード」の場合、**DirectX** を必要とします。もし、**DirectX** が入っていない状態で「通常再生モード」に設定した場合は、サウンドを鳴動させるとエラーになります。

DirectX は、次のサイトから入手できます：

<http://www.microsoft.com/japan/windows/directx/>

サウンドデータの設定をするには：

1. メニューから「設定」→「アプリケーション動作環境設定」→「サウンド設定」タブを選択します（図 5-8）。

アプリケーション動作環境設定

トープサイズの設定 | HART用COMポート設定
 ネットワーク設定 | サウンド設定 | メールアドレス設定

再生モード選択
 再生モード: 通常再生モード

デバイス設定
 デバイス: プライマリ サウンド ドライバ

代替音設定
 チャンネル: チャンネル 1
 楽器グループ: 未設定
 楽器: 未設定

設定状況:

チャンネル	楽器グループ	楽器
1	未設定	未設定
2	未設定	未設定
3	未設定	未設定
4	未設定	未設定
5	未設定	未設定
6	未設定	未設定
7	未設定	未設定
8	未設定	未設定
9	未設定	未設定
10	未設定	未設定
11	未設定	未設定
12	未設定	未設定
13	未設定	未設定
14	未設定	未設定
15	未設定	未設定
16	未設定	未設定

設定 取消 適用

図 5-8 サウンド設定画面

2. 各設定項目を設定します。
各設定項目は、表 5-3 に示すとおりです。

表 5-3 サウンド設定の設定項目

設定項目	説明
再生モード選択	通常再生モードとビジュアルモードを選択できます。 ■通常再生モード(※)： DirectX を使って P C 上で実際に 3 D サウンドを再生することができます。ただし、お使いの P C のサウンドカードが 3 D サウンドに対応している必要があります。 ■ビジュアルモード： DirectX を使わない従来の再生モードです。 代替音や 3 D サウンドは鳴らせませんが、3 D サウンドを視覚的に表示する事ができます（5. 2 3. 2. 3 D サウンド表示参照）。
デバイス設定	通常再生モードのみ有効です。 P C に組み込まれているサウンドカードのサウンドデバイスを選択します。ここで選択したデバイスで再生されます。
代替音設定	通常再生モードのみ有効です。 一覧から設定したいチャンネルを選択し、「チャンネル」、「楽器グループ」、「楽器」を選択します（図 5-9）。 M F i 中の該当チャンネルで、P C 上で再生できない音色があった場合、ここで指定した音を使って再生されます。

「通常再生モード」では、サウンドライブラリが使われています。サウンドライブラリの音のタイミングなどを細かく調整したい場合は、「付録.1 エミュレーション端末の設定」の「サウンドライブラリの設定」を参考に skin.conf を書き換えてください。

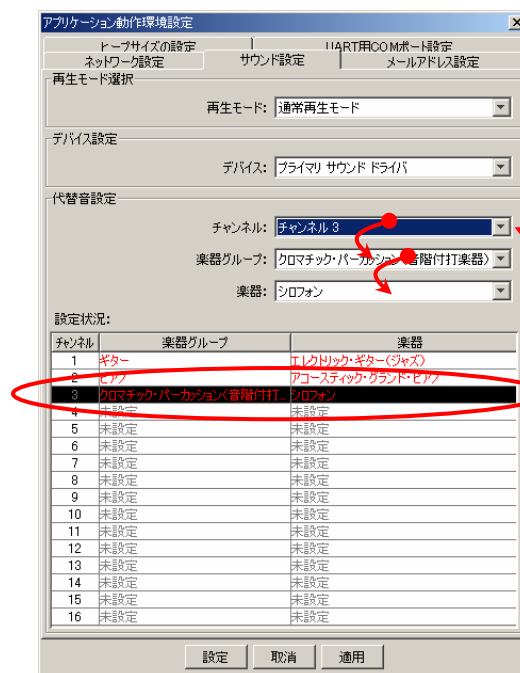


図 5-9 代替音設定

5.7. ヒープサイズの設定

ヒープサイズとは、i モード携帯電話上で動作する Java が扱えるメモリの大きさです。エミュレータでは、デフォルトで 8000Kbyte のヒープ領域が使えるように設定されています。しかし、機種の違いによって、ヒープ領域がこれよりも少ないか多いことがあります。その場合、エミュレータで動いた i アプリが i モード携帯電話で動かないことになります。そこで、次の方法により i アプリが扱えるヒープサイズを変更することができます。（実際のヒープサイズについては、各機種の説明書を参照してください。）

ヒープサイズを設定するには：

1. メニューから「設定」→「アプリケーション動作環境設定」→「ヒープサイズの設定」を選択します（図 5-10）。

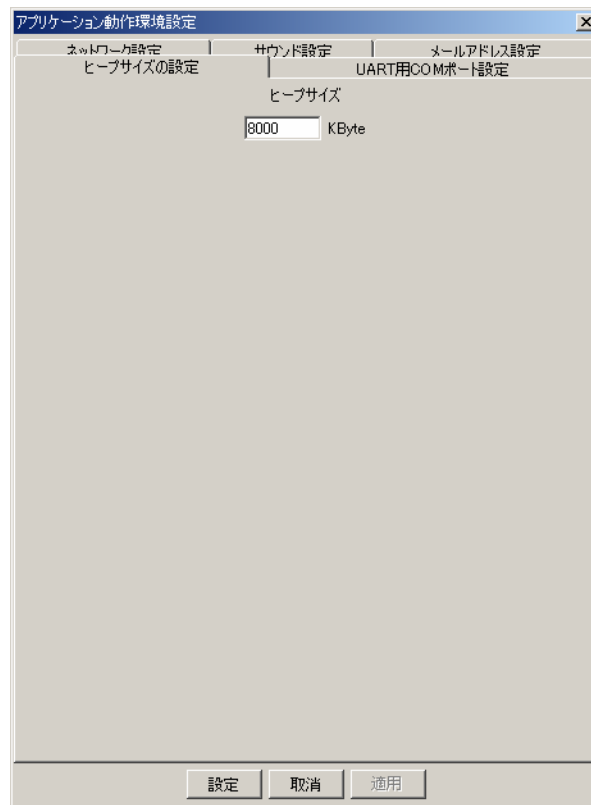


図 5-10 ヒープサイズの設定

2. 1000 ～ 32767 KByte の範囲で、ヒープサイズを入力して「設定」ボタンを押します。

💡 ワンポイント

ヒープサイズは、標準の JavaVM でも使われています。
そのため、あまり小さな値を設定すると、i アプリの起動ができなかったり、動作が不安定になったりする場合があります。
とくに必要ではない限り、この値を変更することはおすすめできません。

5.8. UART用COMポートの設定

エミュレータでは、UARTで実際にPCのシリアルポートから通信をすることができます。

UART で用いるシリアル回線のCOMポートを設定するには：

1. メニューから「設定」→「アプリケーション動作環境設定」→「UART用COMポート設定」タブを選択します（図 5-1 1）

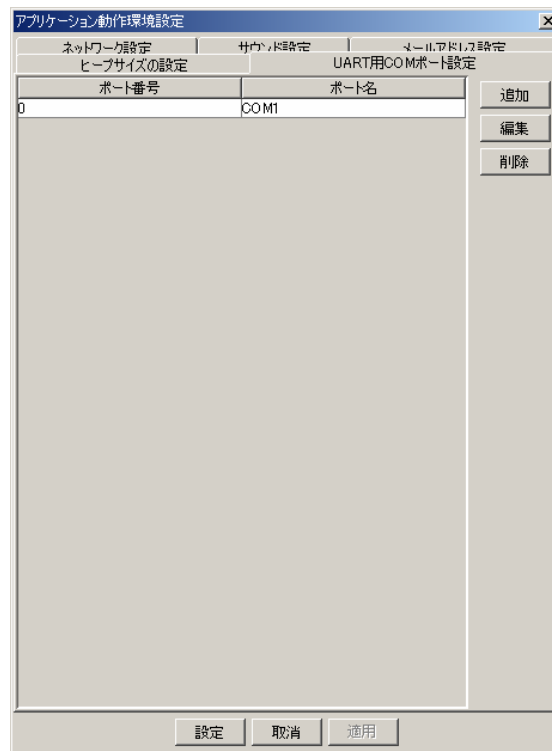


図 5-1 1 UART用COMポート設定画面

2. COMポートを追加したい場合は、「追加」を押します。
UART用COMポート編集画面が表示されるので、ポート番号とポート名を選択して「設定」を押します（図 5-1 2）。



図 5-1 2 COMポート編集画面

3. COMポートを変更したい場合は、一覧からCOMポートを選択して「編集」、削除したい場合は「削除」を押します。

5.9. 電話帳の編集

i アプリから利用する電話帳の編集をします。

電話帳の編集をするには：

1. メニューから「設定」→「ネイティブ編集」→「電話帳編集」タブを選択します（図 5-1 3）。

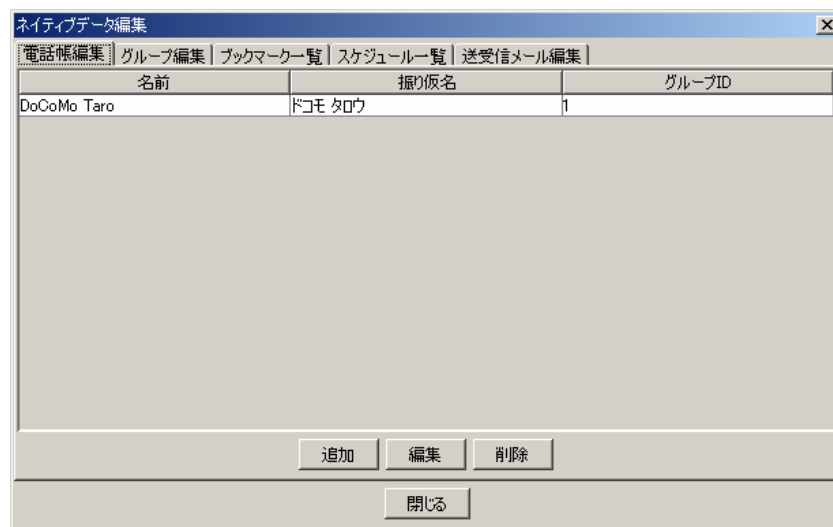


図 5-1 3 電話帳エントリ一覧画面

2. 電話帳エントリを追加したい場合は、「追加」ボタンを押します。
電話帳エントリを変更したい場合は、一覧から変更したい電話帳エントリを選択して、「編集」ボタンを押します（図 5-1 4）。

図 5-1 4 電話帳エントリ登録画面

3. 各設定項目を設定します。

各設定項目は表 5-4 に示すとおりです。

表 5-4 電話帳エントリの設定項目

設定項目	説明	デフォルト
名前	名前。	なし
振り仮名	振り仮名。	なし
電話番号	電話番号を最大 3 件まで設定します。 電話番号は、ハイフン(-)を除いて指定してください。(例：09011112222)	なし
メールアドレス	メールアドレスを最大 3 件まで設定します。	なし
グループ I D	グループ作成時に自動的に付与された I D 番号。 右の「…」ボタンを押すと、グループ一覧画面 (図 5-1 6) で編集したグループのファイルを選択するダイアログ (図 5-1 5) が表示されますので、この電話帳エントリを登録したいグループのファイルを選択して「開く」を押してください。	なし

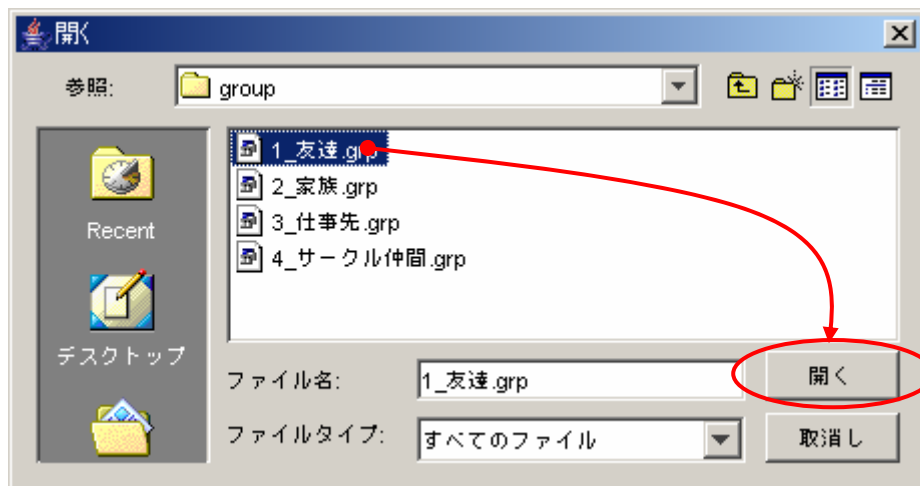


図 5-1 5 グループの選択ダイアログ

ここで作成した電話帳エントリは、
lib\phonebook\エントリ ID_名前.tel というファイルで作成されます。

4. 電話帳エントリを削除したい場合は、一覧から削除したい電話帳エントリを選択して、「削除」ボタンを押します。

5.10. グループの編集

i アプリから利用するグループの編集をします。

グループの編集をするには：

1. メニューから「設定」→「ネイティブ編集」→「グループ編集」タブを選択します（図 5-16）。

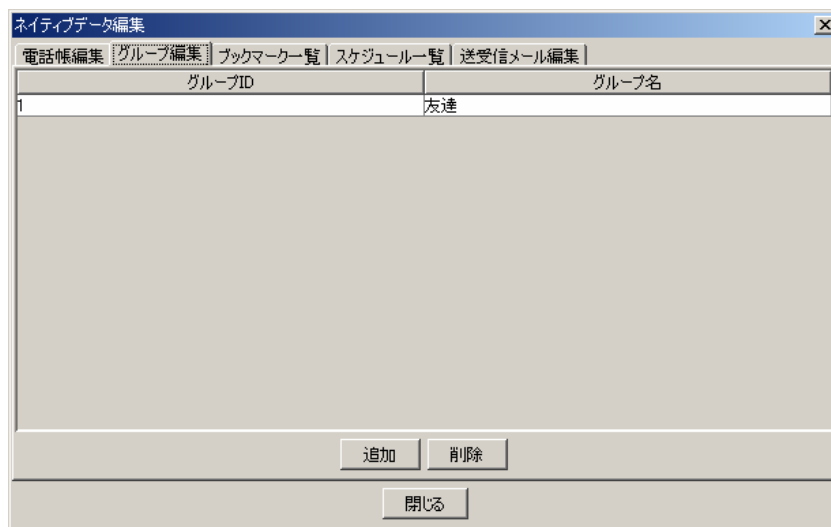


図 5-16 グループ一覧画面

2. グループを追加したい場合は、「追加」ボタンを押します（図 5-17）。

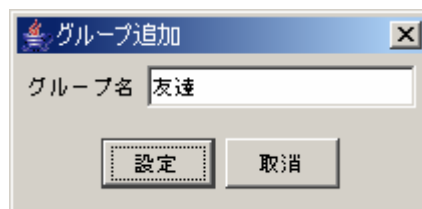


図 5-17 グループ追加画面

3. グループ名を入力して「設定」ボタンを押します。

ここで作成したグループは、
`lib\group\グループ ID_グループ名.grp` というファイルで作成されます。

4. グループを削除したい場合は、一覧から削除したいグループエントリを選択して、「削除」ボタンを押します。

5.11. ブックマーク一覧

i アプリから扱えるブックマークは、個人情報のセキュリティ上、設定することしかできません。

ブックマーク一覧では、アプリケーションから `Bookmark#addEntry( )` で作成されたブックマークを一覧で確認することができます。また、アプリケーションで作成したブックマークを削除することができます。

ブックマークの一覧を編集するには：

1. メニューから「設定」→「ネイティブ編集」→[ブックマーク一覧] タブを選択します（図 5-18）。

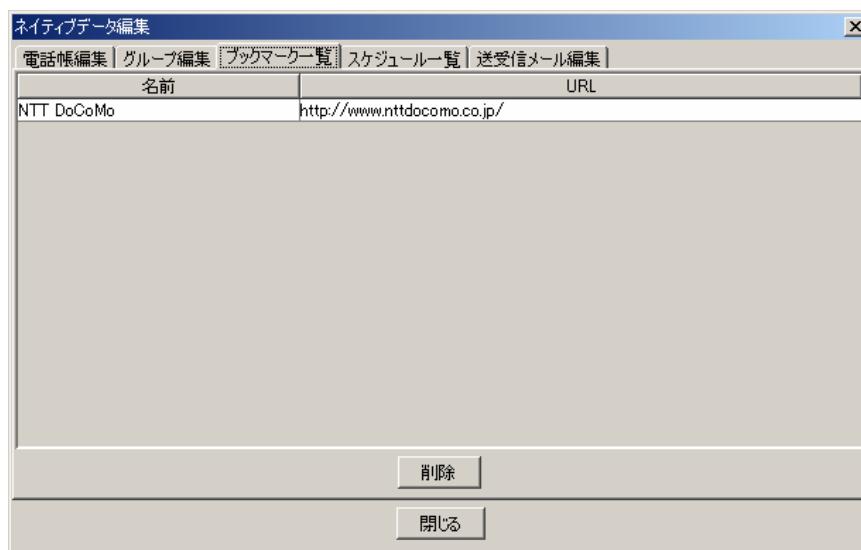


図 5-18 ブックマーク一覧

2. i アプリから作成されたブックマークの一覧が表示されるので、不要なブックマークを選択して「削除」ボタンを押します。
3. 「閉じる」ボタンを押すと、ブックマーク一覧が終了します。

5.12. スケジュール一覧

i アプリから扱えるスケジュールは、個人情報のセキュリティ上、設定することしかできません。

スケジュール一覧では、アプリケーションから `Schedule#addEntry( )` で設定されたスケジュールを一覧で確認することができます。また、i アプリで作成したスケジュールを削除することができます。

スケジュールの一覧を表示するには：

1. メニューから「設定」→「ネイティブ編集」→「スケジュール一覧」タブを選択します（図 5-19）。

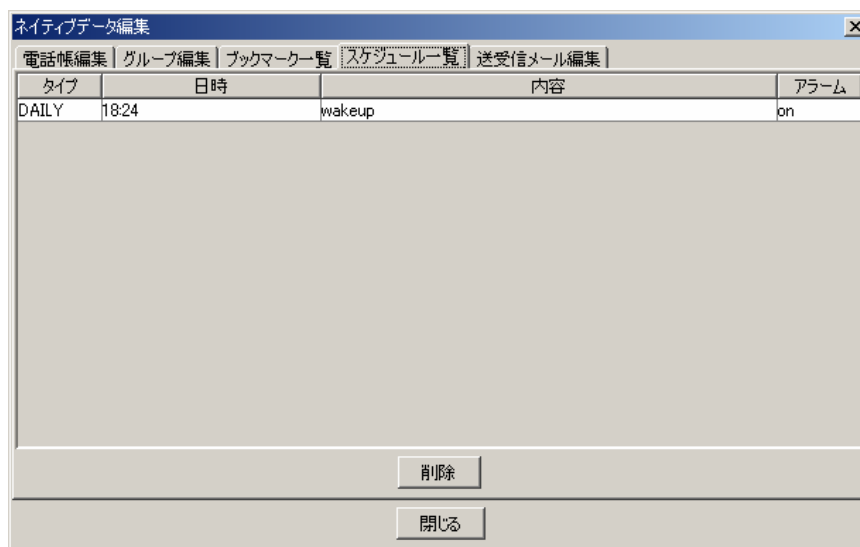


図 5-19 スケジュール一覧

2. i アプリから作成されたスケジュールの一覧が表示されるので、不要なスケジュールを選択して「削除」ボタンを押します。
3. 「閉じる」ボタンを押すと、スケジュール一覧が終了します。

5.13. 送受信メールの編集

i アプリから利用する送受信メールの編集をします。

送受信メールの編集をするには：

1. メニューから「設定」→「ネイティブ編集」→「送受信メール編集」タブを選択します（図 5-20）。

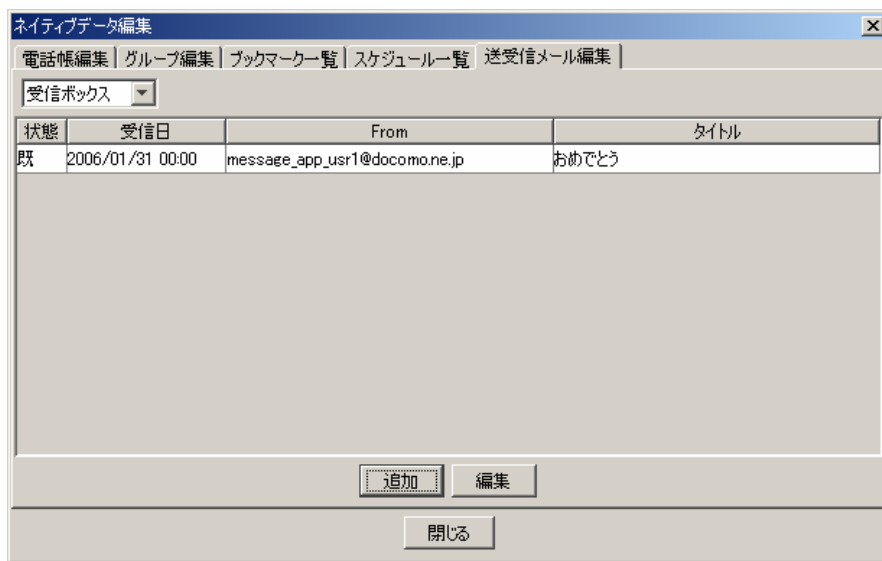


図 5-20 送受信メール一覧画面

2. 編集したいメールボックスを選択すると、一覧が切り替わります（図 5-20）。メールボックスの種類は以下の3つから選べます。
 - 受信ボックス
受信したメールを保存します。
 - 送信ボックス
送信したメールを保存します。
 - 未送信ボックス
送信前（編集途中）、または、送信に失敗したメールを保存します。
3. メールを追加したい場合は、「追加」ボタンを押します。
メールを変更したい場合は、一覧から変更したいメールを選択して、「編集」ボタンを押します（図 5-21、図 5-22）。

メール編集(受信ボックス)

To: message_app_usr0@docomo.ne.jp

From: message_app_usr1@docomo.ne.jp

タイトル: おめでとう

受信日: 2005 年 06 月 30 日 8 時 15 分

MessageCode: 0000000000

ネイティブメール: ☐

状態: ☒ 既読 ☐ 未読

本文

お誕生日おめでとう

設定 取消

図 5-2 1 メール編集 (受信ボックス) 画面

メール編集(送信ボックス)

To: message_app_usr1@docomo.ne.jp

From: message_app_usr0@docomo.ne.jp

タイトル: Re:おめでとう

送信日: 2005 年 06 月 30 日 8 時 20 分

MessageCode: 0000000000

ネイティブメール: ☐

宛先状態: ☒ XString ☐ 通常文字列

本文

ありがとう

設定 取消

図 5-2 2 メール編集 (送信ボックス) 画面

4. 各設定項目を入力して「設定」を押します。

各設定項目は表 5-5 に示すとおりです。

表 5-5 送受信メールの設定項目

設定項目	説明	デフォルト
T o	受信ボックスの場合は受信者、送信ボックスおよび未送信ボックスの場合は受信者のアドレス。 複数指定する場合は、カンマ(,)で区切って指定します。	なし
F r o m	受信ボックスの場合は送信者、送信ボックスおよび未送信ボックスの場合は送信者のアドレス。 複数指定する場合は、カンマ(,)で区切って指定します。	なし
タイトル	メールのタイトル。	なし
受信日	メールを受信した日時。	なし
送信日	メールを送信した日時。	なし
MessageCode	A D F に設定した MessageCode。	なし
ネイティブメール	ネイティブメール（端末が管理するメール）の場合は、チェックをつけます。 チェックをつけた場合、このメールは、メッセージ i アプリのメールボックスに移動されたネイティブメールとなります。 ネイティブメールには、バウンダリ (--B:P 以降) は付与されません。	チェックなし
状態	既読か未読の状態。 既に読まれたメールは「既読」、まだ読まれていないメールは「未読」にします。	既読
宛先状態	宛先の完全カプセル化文字列 (A P I から取得できない文字列) の指定。 宛先を完全カプセル化文字列にする場合は XString にします。 電話帳エントリから取得したメールアドレスを指定した場合、および、受信ボックスから返信するメールの場合は、XString になります。	XString
本文	メールの本文。	なし

ここで作成したメールは、それぞれ以下のファイルで作成されます。

受信ボックス： lib\mail\received\メッセージコード\受信日_通番.mail
 送信ボックス： lib\mail\sent\メッセージコード\送信日_通番.mail
 未送信ボックス： lib\mail\unsent\メッセージコード\送信日_通番.mail

もし、メールを削除したい場合は、このファイルを直接削除してください。
メールファイルの形式については、「9.7. メールファイル」を参照してください。

また、メッセージコードは、ADFの MessageCode の値ですが、このキー項目が設定されていない場合は、“0000000000”であるとみなしてメールボックスとメールを作成します。
そのため、ADFの MessageCode を設定する前に作成したメールをメッセージ i アプリで使いたい場合には、ADFの MessageCode を“0000000000”に設定してください。
(「4.7. アプリケーションディスクリプタファイル (ADF) の編集」を参照)

メール中には、i アプリ固有のバイナリデータが付与されます。
メールに付与されるバイナリデータは、以下のファイルに保存されています。

バウンダリ部のバイナリデータ : lib\mail\boundary.data

バイナリデータを変更したい場合は、このファイルの内容を変更してください。
ただし、このバイナリデータは、BASE 64 でエンコードされた値でなくてはなりません。

また、各メールの状態は、以下のファイルに保存されています。

状態 : lib\mail\receive\メッセージコード\mail.state
宛先状態 : lib\mail\sent\メッセージコード\xstring.state
lib\mail\unsent\メッセージコード\xstring.state
メールアドレス : lib\mail\myAddress

これらのファイルは、特に修正する必要はありません。

5.13.1. ショートメールの扱いについて

ショートメールをメールボックスに作成する場合は、メールの追加で、それぞれ次のように指定します。(図 5-23、図 5-24)

- 受信ボックス
From : iMS
Subject : 電話番号(ショートメール)
ネイティブメール
- 送信ボックス
To : iMS
Subject : 電話番号(ショートメール)
ネイティブメール
- 未送信ボックス
To : iMS
Subject : 電話番号(ショートメール)
ネイティブメール

メール編集(受信ボックス)

To: message_app_usr0@docomo.ne.jp

From: IMS

タイトル: 090XXXXXXXX(ショートメール)

受信日: 2005 年 06 月 30 日 09 時 59 分

MessageCode: 0000000000

ネイティブメール: ☐

状態: ☒ 既読 ☐ 未読

本文

おはようございます

設定 取消

図 5-2 3 受信ボックスのショートメール作成

メール編集(送信ボックス)

To: IMS

From: message_app_usr0@docomo.ne.jp

タイトル: 090XXXXXXXX(ショートメール)

送信日: 2005 年 06 月 30 日 10 時 0 分

MessageCode: 0000000000

ネイティブメール: ☐

宛先状態: ☒ XString ☐ 通常文字列

本文

こんにちは

設定 取消

図 5-2 4 送信ボックス／未送信ボックスのショートメール作成

5.13.2. メールの送受信エミュレート機能

メールを扱う i アプリは、メールの送受信をエミュレートする事ができます。

メールの送受信をエミュレートするには：

1. メールアドレスの設定画面に送受信するメールアドレスを設定しておきます。
(「5.3. メールアドレスの設定」を参照)
2. エミュレートする i アプリの MessageCode を同一の値にします。
(「4.7. アプリケーションディスクリプタファイル (ADF) の編集」を参照)
※ 異なる MessageCode (i アプリ) 同士でのメールはやり取りできません。
3. i アプリからメールを送信します。
(MessageAgent#send() を実行します。)
4. 送信したメールが、宛先に指定したメールアドレスに該当する受信ボックスに受信メールで格納されます。
ただし、1. で宛先に指定したメールアドレスを設定した i アプリが存在しない、もしくは、MessageCode が異なる場合、受信メールは作成されません (図 5-25)。

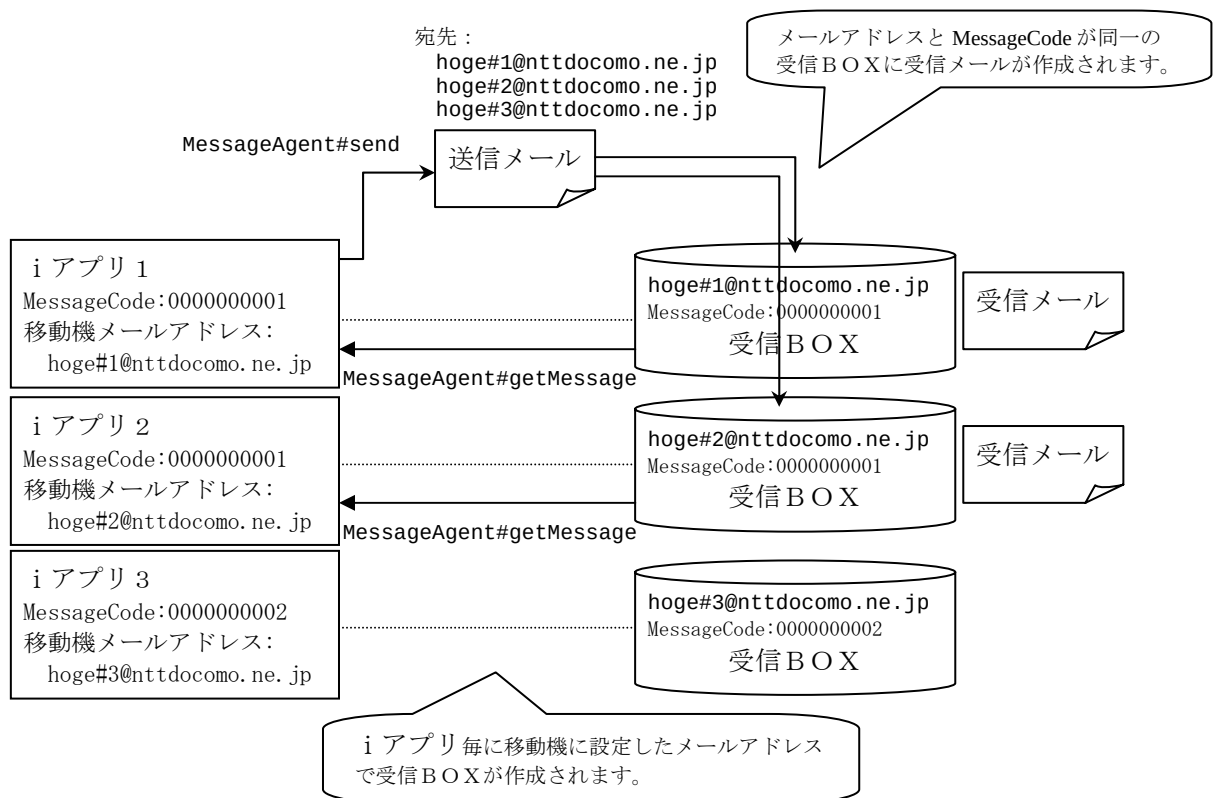


図 5-25 i アプリによるメールの送受信

5.14. ビルドの設定

5.14.1. スクリプトの実行

i アプリをビルドする際に、リバースエンジニアリングなどで最適化をする事によって、i アプリの Jar ファイルのサイズをコンパクトにしたい場合があります。

エミュレータでは、これを支援する機能として、ビルドの実行前か実行後に i アプリ (Jar) が作成されるパスでスクリプトコマンドを実行できるようにしています (図 5-26)。

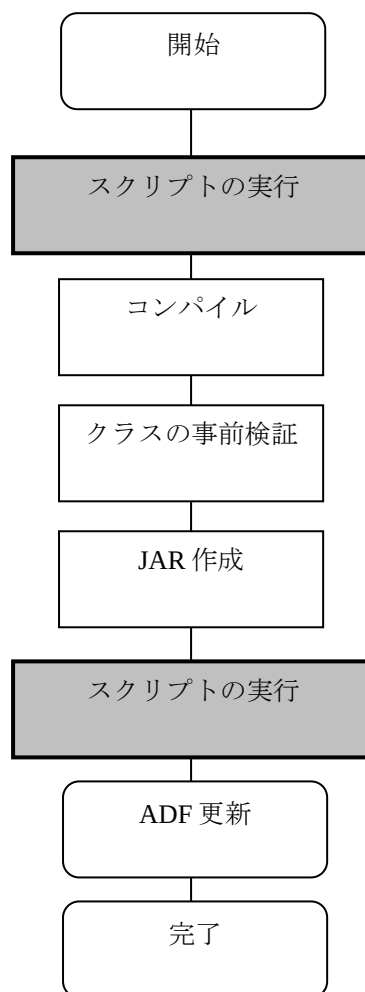


図 5-26 ビルド時のスクリプトコマンド実行の流れ

ビルドの設定でスクリプトを実行できるようにするには：

1. メニューから「設定」→「エミュレータ環境設定」→「ビルドの設定」タブを選択します（図 5-27）。

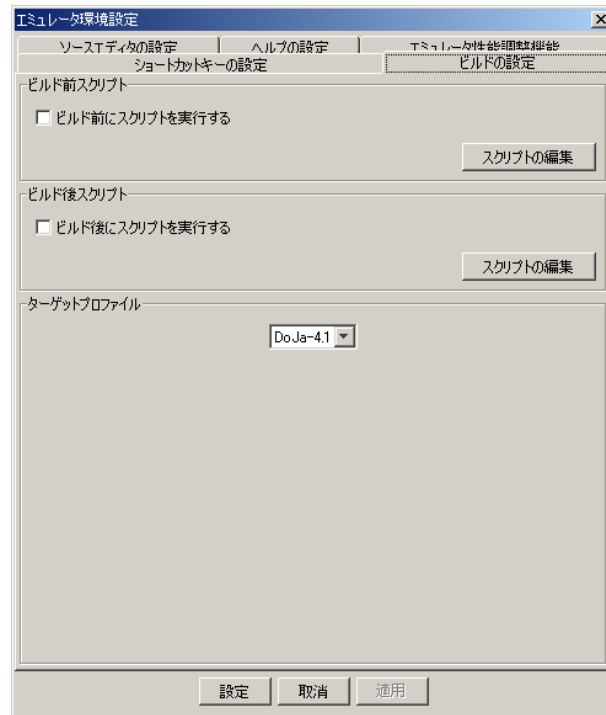


図 5-27 ビルドの設定 - スクリプトの実行

2. ビルドの実行前にスクリプトを実行したい場合は、「ビルド前にスクリプトを実行する」にチェックを付けます。
ビルドの実行後にスクリプトを実行したい場合は、「ビルド後にスクリプトを実行する」にチェックを付けます。
3. 「スクリプトの編集」を押すとエディタでスクリプトファイルが開かれるので、ここに DOS のバッチファイル形式で起動したいコマンドや必要な環境変数を記述します。
4. 「設定」を押します。

スクリプトの内容には、次の環境変数が使えます：

環境変数	内容
PROJECT_NAME	ビルドするプロジェクト名
JAR_PATH	Jar コマンド (i アプリの Jar ファイルを操作する際に使用する)
APP_CLASS	start() メソッドが存在するクラス名

スクリプトの終了は、次のように指定してください。

■ 正常終了の場合

exit 0

■ 異常終了の場合 (メッセージを表示してビルドを中断します。)

exit 1 (0 以外の値)

スクリプトは、以下のファイルで作成されます：

- ビルド前スクリプト：
C:\iDKDoJa4.1\lib\optimize_script1.bat
- ビルド後スクリプト：
C:\iDKDoJa4.1\lib\optimize_script2.bat

5.14.2. ターゲットプロファイルの設定

i アプリをビルドする時の DoJa のプロファイルを設定します。
デフォルトは、最新の「DoJa-4.1」になっています。

ビルド時の DoJa のプロファイルを設定するには：

1. メニューから「設定」→「エミュレータ環境設定」→「ビルドの設定」タブを選択します（図 5-28）。

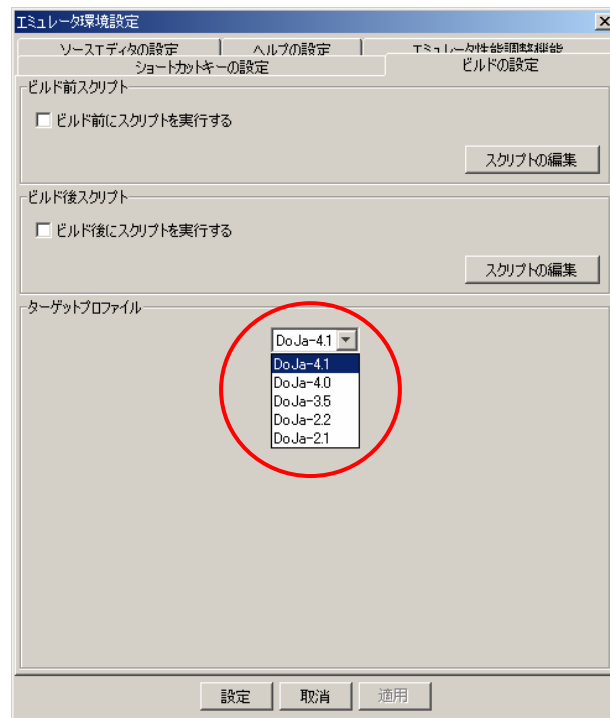


図 5-28 ビルドの設定 - プロファイルの変更

2. ターゲットにしたい DoJa のプロファイルを選択します。
3. 「設定」を押します。
これにより、ビルド時にターゲットプロファイルに含まれないAPIを用いていた場合は、コンパイルエラーになります。

【注意】

ここでプロファイルを切り替えても i アプリの実行ではチェックされません。
この設定は、i アプリをビルドする際に、選択したターゲットプロファイルを使います。

5.15. ショートカットキーの設定

エミュレータで扱うショートカットキーを設定します。

エミュレータでは、インストールされた状態で、あらかじめ以下のショートカットキーが設定されています：

■エミュレータ (iα ppliTool)

コンソールクリア：	C
プロジェクト新規作成：	N
プロジェクト読込み：	O
ソースファイルを開く：	F
ADF 設定：	A
ビルド：	B
起動：	R

■エミュレーション端末

数字ボタン：	0 ～ 9
# ボタン：	#
* ボタン：	*
決定ボタン：	Enter (↵)
方向ボタン：	Up (↑) , Down (↓) , Left (←) , Right (→)
待受ボタン：	D
ソフトキー 1：	A (テキスト編集中は無効)
ソフトキー 2：	S (テキスト編集中は無効)

※ エミュレーション端末のショートカットキーを変更することはできません。

エミュレータのショートカットキーを設定するには：

1. メニューから「設定」→「エミュレータ環境設定」→「ショートカットキーの設定」タブを選択します（図 5-29）。

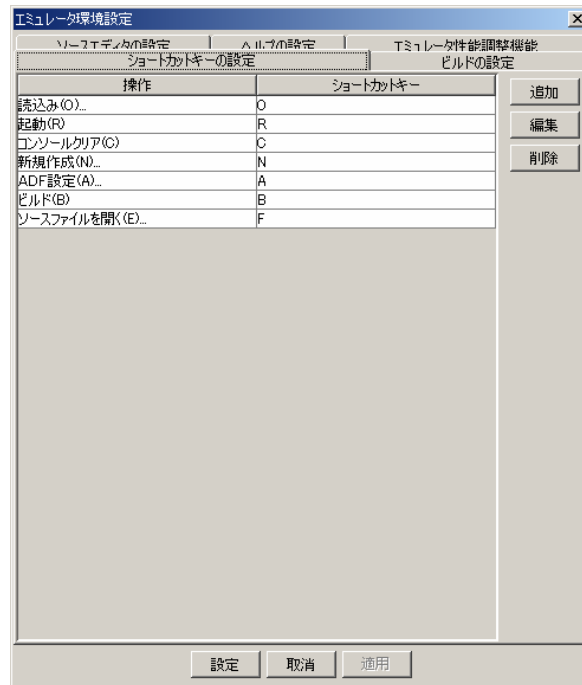


図 5-29 ショートカットキーの設定

2. ショートカットキーを追加する場合は、「追加」ボタンを押します（図 5-30）。ショートカットキーを変更する場合は、変更したいショートカットキーを選択して「編集」ボタンを押します。

※ 同じ操作に複数のショートカットキーを割り当てることはできません。

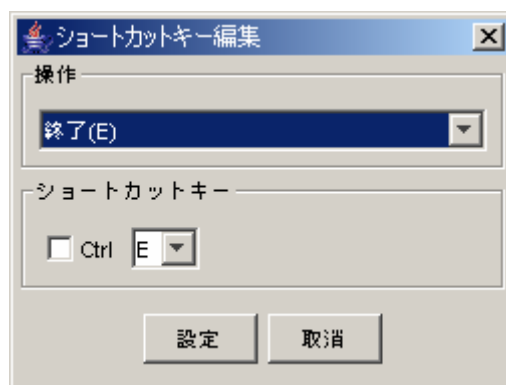


図 5-30 ショートカットキーの編集

3. 各設定項目を入力して「設定」ボタンを押します。

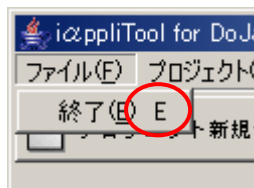
各設定項目は表 5-6 に示すとおりです。

表 5-6 ショートカットキーの設定項目

設定項目	説明	デフォルト
操作	ショートカットキーを設定したい操作を選択します。 既に設定済みの同じ操作を指定することはできません。	C: コンソールクリア N: プロジェクト新規作成 O: プロジェクト読み込み F: ソースファイルを開く
ショートカットキー	設定するショートカットキーを選択します。Ctrl にチェックを付けるとショートカットは、Ctrl + キー になります。	A: A D F 設定 B: ビルド R: 起動

4. ショートカットキーの一覧に設定したショートカットが追加されます。

ショートカットキーを設定するとiα ppliToolのメニューに設定したショートカットキーが表示されます。(図 5-3 1)



エミュレータ上で「e」または「E」を押すとエミュレータが終了します。

図 5-3 1 終了メニューに「E」のショートカットキーが設定された状態

5.16. ヘルプの設定

エミュレータのヘルプメニューからAPIリファレンス、操作ガイド（本書）およびリリースノートを参照する設定をします。

ヘルプの設定をするには：

1. メニューから「設定」→「エミュレータ環境設定」→「ヘルプの設定」を選択します（図 5-3 2）。

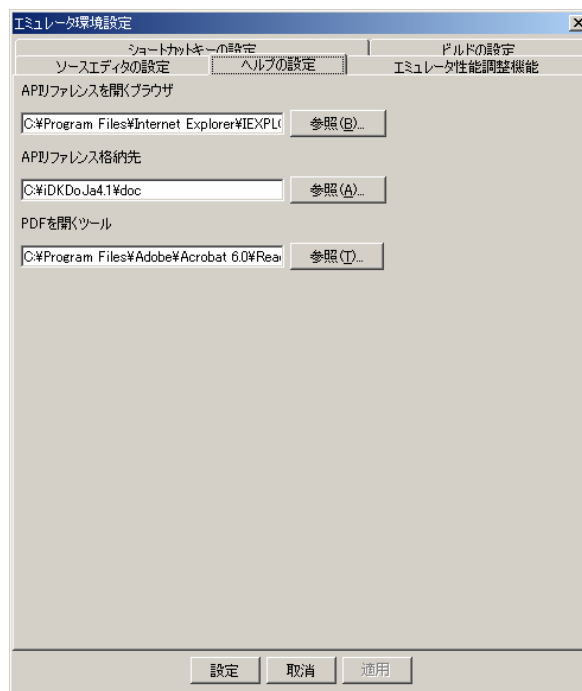


図 5-3 2 ヘルプの設定画面

2. 各設定項目を入力して「設定」ボタンを押します。
各設定項目は表 5-7 に示すとおりです。

表 5-7 ヘルプの設定項目

設定項目	説明	デフォルト
API リファレンスを開く ブラウザ	API リファレンスを参照するためのブラウザを設定します。	インストールされた Internet Explorer もしくは、 Netscape
API リファレンス格納先 (※)	API リファレンスが格納されているディレクトリを設定します。 ■API リファレンス： API リファレンス格納先¥index.html を開きます。	インストールディレクトリ¥doc
PDF を開くツール	PDF ファイルを参照するための i アプリを設定します。 ■操作ガイド： インストールディレクトリ¥～DevGuide～.pdf ■リリースノート： インストールディレクトリ¥～ReleaseNote～.pdf を開きます。	インストールされた Acrobat Reader

※ 別途 http://www.nttdocomo.co.jp/p_s/imode/make/ から入手してください。

5.17. エミュレータ性能設定

エミュレータの性能を変えることで、エミュレータを携帯端末の性能に近づけることができます。

エミュレータの性能を設定するには：

1. メニューから「設定」→「エミュレータ環境設定」→「エミュレータ性能設定」を選択します（図 5-3 3）。

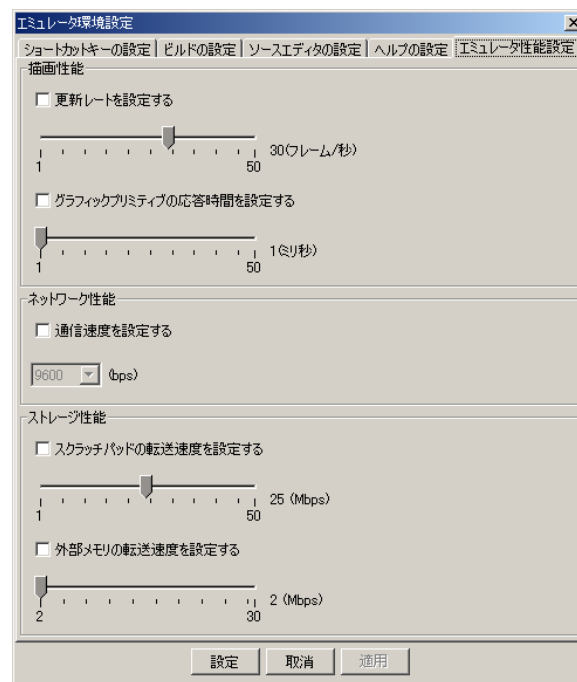


図 5-3 3 エミュレータ性能設定画面

2. 各設定項目を設定して「設定」ボタンを押します。
各設定項目は、設定したい項目にチェックを付けるとスライドバーで変更できるようになっています。
チェックを外すと性能の設定は無効になります。

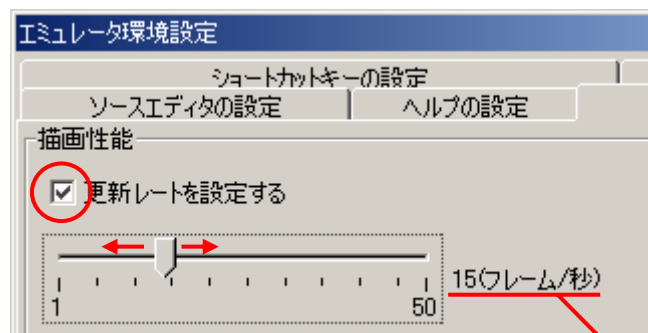


図 5-3 4 性能値の変更

現在の値

各設定項目は表 5-8 に示すとおりです。

表 5-8 エミュレータ性能の設定項目

性能項目	設定項目	説明	デフォルト
描画性能	更新レートを設定する	Graphics のフレームレートを 1 ～ 5 0 fs の範囲で変更します。 未設定の場合、77 フレーム/秒。	30 フレーム/秒
	グラフィックプリミティブの応答時間を設定する	Graphics の各プリミティブ描画の待ち時間を 1 ～ 5 0 ms の範囲で設定します。 未設定の場合、0 ミリ秒。	1 ミリ秒
ネットワーク性能	通信速度を設定する	HTTP 通信の待ち時間を 1 ～ 5 0 bps の範囲で設定します。 未設定の場合、デバイス依存。	9600 bps
ストレージ性能	スクラッチパッドの転送速度を設定する	スクラッチパッドの読み書きにかかる待ち時間を 1 ～ 5 0 Mbps の範囲で設定します。 未設定の場合、PC 性能に依存。	25 Mbps
	外部メモリの転送速度を設定する	外部メモリの読み書きにかかる待ち時間を 2 ～ 3 0 Mbps の範囲で設定します。(最小は 2 Mbps) 未設定の場合、PC 性能に依存。	2 Mbps

5.18. コンパイラ(javac)の選択

メニューから「設定」→「sun.tools.javac.Mainを使用する」を選択することにより、コンパイラ (javac) を選択することができます (図 5-35)。ここで選択されたコンパイラ (javac) は、プロジェクトのビルド時 (「4.3. プロジェクトのビルドと通常起動」を参照) に使用されるコンパイラ (javac) です。

デフォルトでは「com.sun.tools.javac.Main」を使用しますが、「sun.tools.javac.Mainを使用する」チェックボックスをチェックすることにより「sun.tools.javac.Main」を使用します。

また、選択されたコンパイラ (javac) の情報は保存されるので、次回「iαppliTool」起動時に自動的に設定されます。

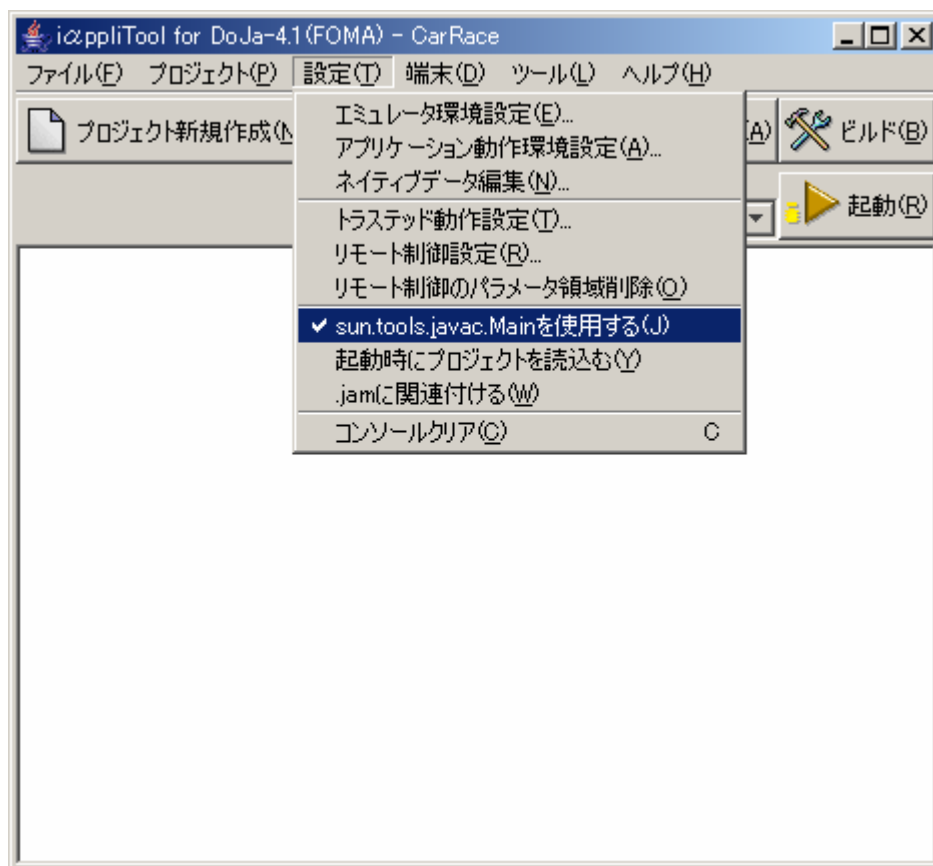


図 5-35 コンパイラ (javac) の選択

5.19. 起動時にプロジェクトを読み込む

エミュレータが起動した時に、前回の終了時に開いていたプロジェクトを開く事ができます。これによりプロジェクトを開きなおよす手間が省けます。

プロジェクトが開いた状態でエミュレータを起動するには、メニューから「設定」→「起動時にプロジェクトを読み込む」にチェックを付けます（図 5-3 6）。

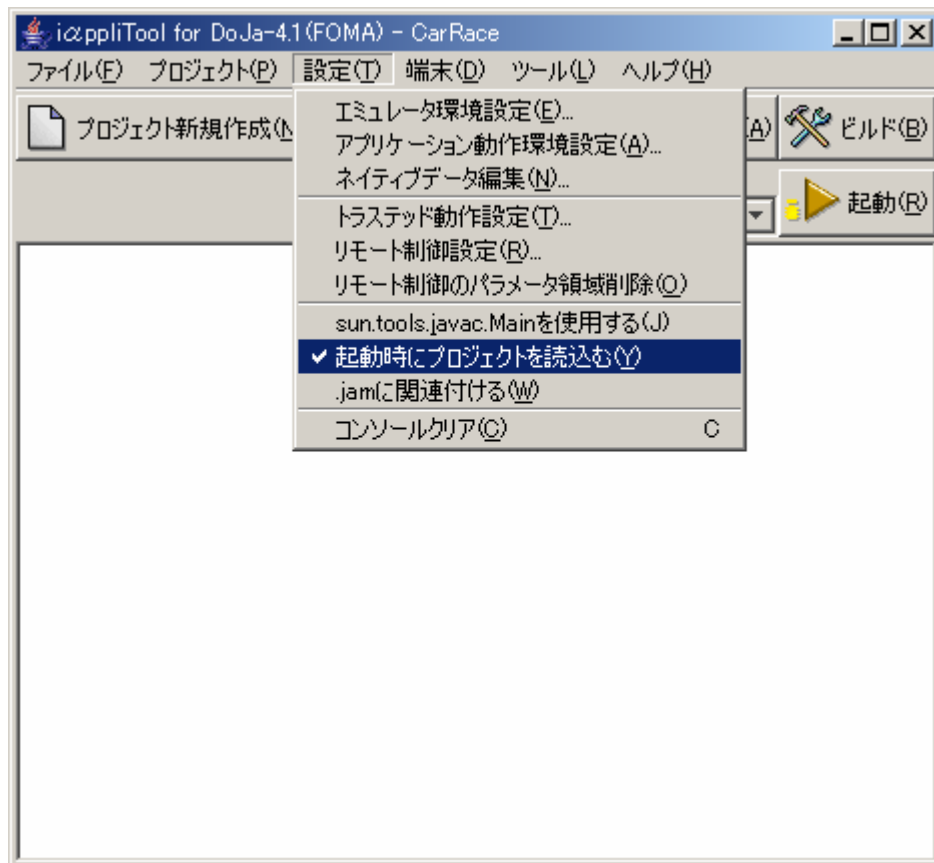


図 5-3 6 起動時にプロジェクトを読み込む

5.20. jam に関連付ける

ADF (.jam ファイル) をエミュレータに関連付けします。
 これにより、ADF ファイルをクリックするだけで、エミュレータが起動し、i アプリが実行されます。
 この機能は、インストール時も設定するかどうかを問い合わせています。

ADF (.jam ファイル) をエミュレータに関連付けるには、メニューから「設定」→「jam に関連付ける」にチェックを付けます (図 5-37)。

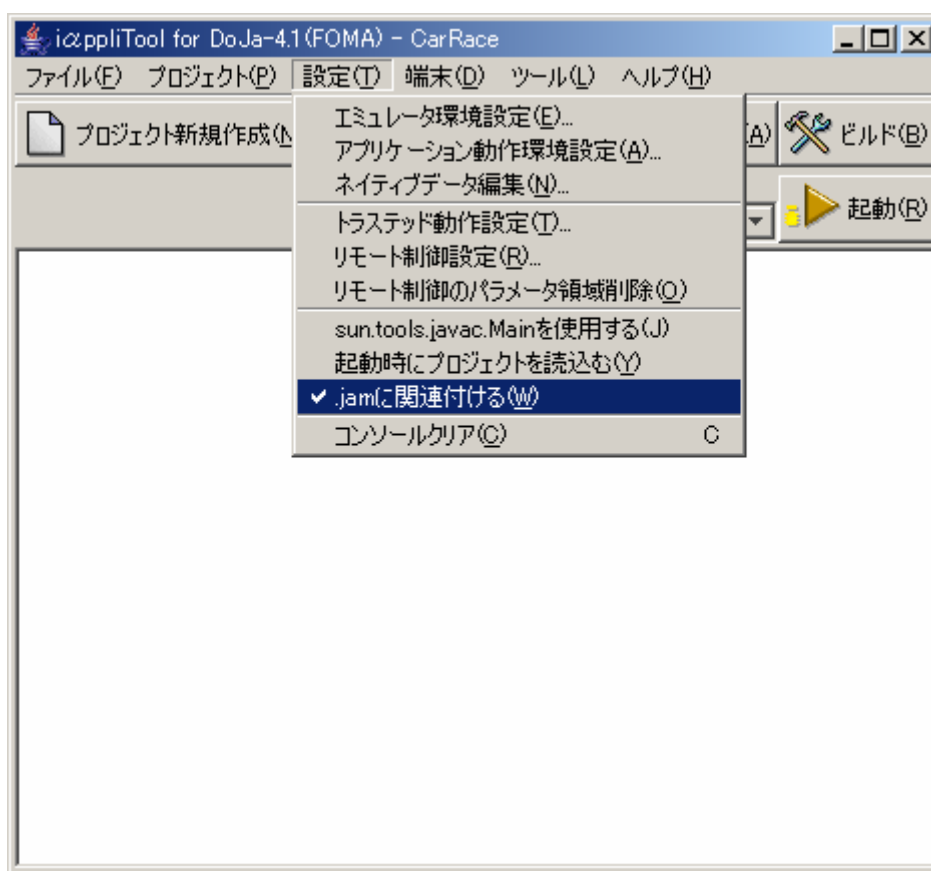


図 5-37 .jam に関連付ける

5.21. コンソールクリア

起動中の i アプリからの標準出力・エラー出力は、コンソールに表示されます^[7]。
 メニューから「設定」→「コンソールのクリア」を選択すると、コンソールに表示されているメッセージが消去されます。

^[7] ただし、絵文字は「・」、JIS X 0208 で定義されている漢字以外の漢字は「?」で表示されます。

5.22. エミュレーション端末の選択

i アプリが動作するエミュレーション端末を変更することができます。

i アプリが動作するエミュレーション端末を変更するには：

1. メニューから「端末」を選択し、その中からエミュレートする端末をチェックします（図 5-38）。

このメニューには、ビルドした i アプリに対して使用できる端末が表示されます。なお、ここで選択した端末は、レジストリに保存されるので、次回「iαppliTool」を起動したとき最初に選択されている端末として使用されます。

また、各端末のイメージを図 5-39 に示します。

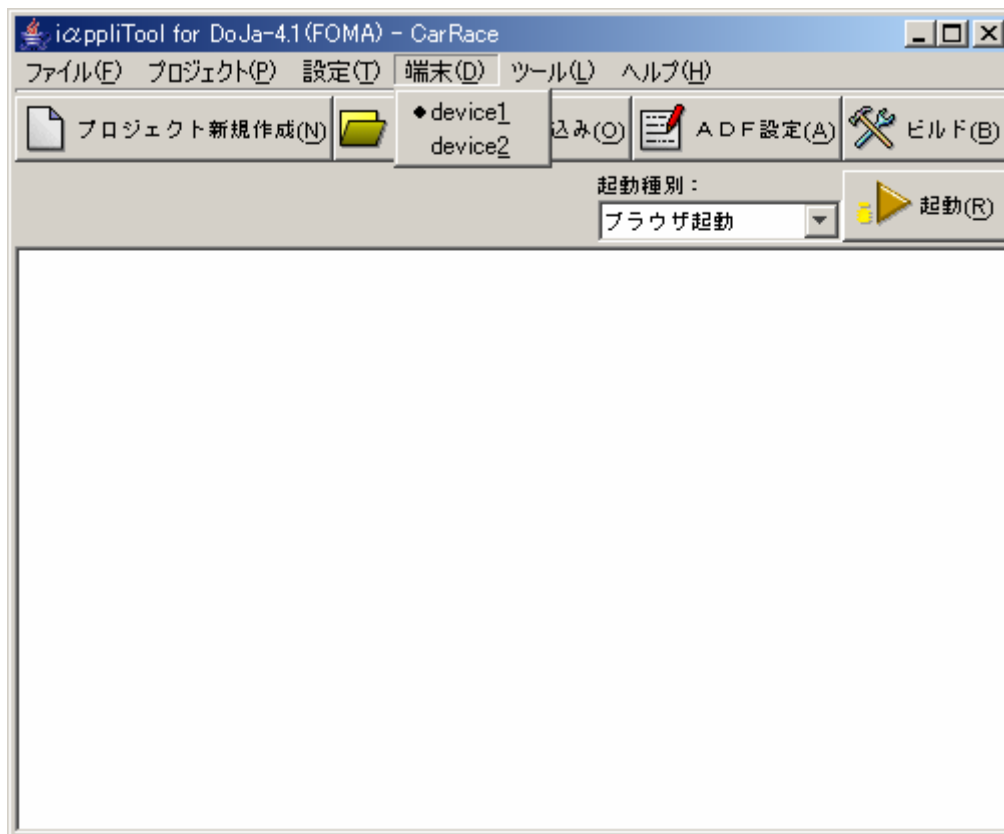


図 5-38 端末の選択



図 5-3 9 エミュレーション端末 (左から「device1」、「device2」)

💡 ワンポイント

エミュレーション端末は、自由にカスタマイズすることが可能です。
エミュレーション端末のカスタマイズについては、「付録.1 エミュレーション端末の設定」を参照してください。

5.23. エミュレーション端末の操作

次にエミュレーション端末の基本操作を説明します（図 5-40）。
ここで説明している基本操作はどのエミュレーション端末でも同様です。

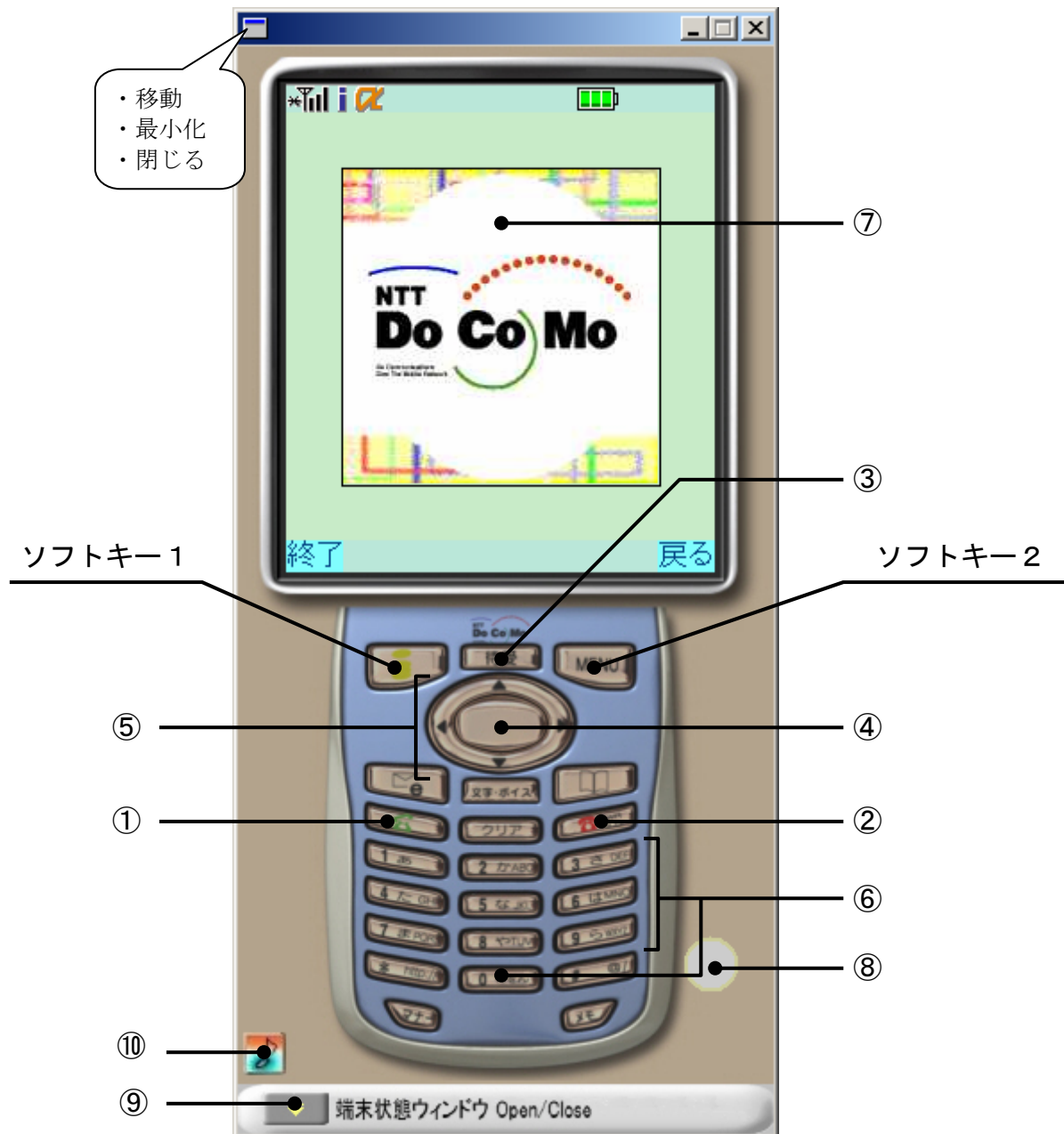


図 5-40 エミュレーション端末

①. 発信ボタン

- 通常のアプリケーションおよび待ち受けアプリケーション動作中に押すことにより、通常の通話を開始した状態になり、i アプリがサスペンドします。

②. 終話ボタン

- ①の発信ボタンを押した状態（通話中）のときに押すことにより、通話が終了し、サスペンドしていた i アプリがレジュームします。
- 上記の状態以外のときに押すことにより、通常のアプリケーションおよび待ち受けアプリケーションを強制終了させます。

③. 待受ボタン

- 待ち受けアプリケーションが、「休眠状態」あるいは「非活性化状態」のときに押すことにより「活性化状態」になります（「4.8.1. 待ち受けアプリケーション」参照）。このボタンは、キーボードの「D」キーに対応しています。

④. 決定ボタン

- このボタンは、キーボードの「Enter（リターン）」キーに対応しています。

⑤. 方向ボタン（上下左右ボタン）

- このボタンは、キーボードの「上下左右（↑↓←→）」キーにそれぞれ対応しています。

⑥. 数字ボタン（0～9 ボタン）

- このボタンは、キーボードの「0～9」キーにそれぞれ対応しています。

⑦. LCD

- i アプリ起動中に LCD 上でマウスを右クリックすることで、エミュレーション端末の状態を変更する事ができます。また、エミュレーション端末に無いイベントを発生する事ができます。

ただし、API から状態を変更した場合は、チェックのついた属性とは異なる場合があります。

状態とイベントには、それぞれ以下に示す属性があります：

■端末状態

- 「折りたたみ状態」
 - 開く
 - 閉じる
- 「画面状態」
 - 可視
 - 不可視
- 「メール状態」
 - メールなし
 - 未読メールあり
 - センターに未読メールあり

- 「メッセージ状態」
 - メッセージなし
 - 未読メッセージあり
 - センターに未読メッセージあり
- 「バッテリー状態」
 - バッテリー充電中
 - バッテリーMAX
 - バッテリーその他
- 「アンテナ状態」
 - アンテナ圏外
 - アンテナ圏内
- 「マナーモード状態」
 - マナーモードOFF
 - マナーモードON

■ 端末ウィンドウ以外で発生できるイベント

- 「MPEG 再生終了」
 - 「MP4 再生終了」
 - 「MP4 ストリーミング終了」
 - 「メッセージフォルダイベント」
 - 「受信フォルダ」
 - 「送信フォルダ」
 - 「未送信フォルダ」
 - 「パラメータ受信」
- 上記の状態を i アプリから取得するには、**PhoneSystem.getAttribute(int attr)**メソッドを使用します（詳細は、APIドキュメント参照）。なお、「メール状態」、「メッセージ状態」、「バッテリー状態」、「アンテナ状態」、および「マナーモード状態」をこのメソッドで取得する場合、ADFの設定項目「GetSysInfo」を設定しておく必要があります（「4.7. アプリケーションディスクリプタファイル（ADF）の編集」を参照）。

⑧. ピクチャライト

i アプリオプションAPI **PictureLight** を使った場合に発光します。

⑨. 端末子ウィンドウ表示

ここをクリックすると、端末の状態や、端末ウィンドウにないイベントを発生させる子ウィンドウを表示します。
もう一度クリックすると子ウィンドウは閉じられます。
端末子ウィンドウについては、「5.2 3.1. 端末子ウィンドウの操作」で説明します。

⑩. 3Dサウンド表示

3Dサウンドを演奏している時に視覚的に演奏している位置を表示させます。

■ ソフトキー1（左側）、ソフトキー2（右側）

キーボードの「A」「S」キーにそれぞれ対応しています（ただし、テキスト編集中は無効です）。

5.23.1. 端末子ウィンドウの操作

端末子ウィンドウは、エミュレーション端末の端末子ウィンドウ表示を押すと表示されます（5.23. エミュレーション端末の操作 – ⑨ 参照）。

図 5-4 1 を用いて端末子ウィンドウの基本操作を説明します。

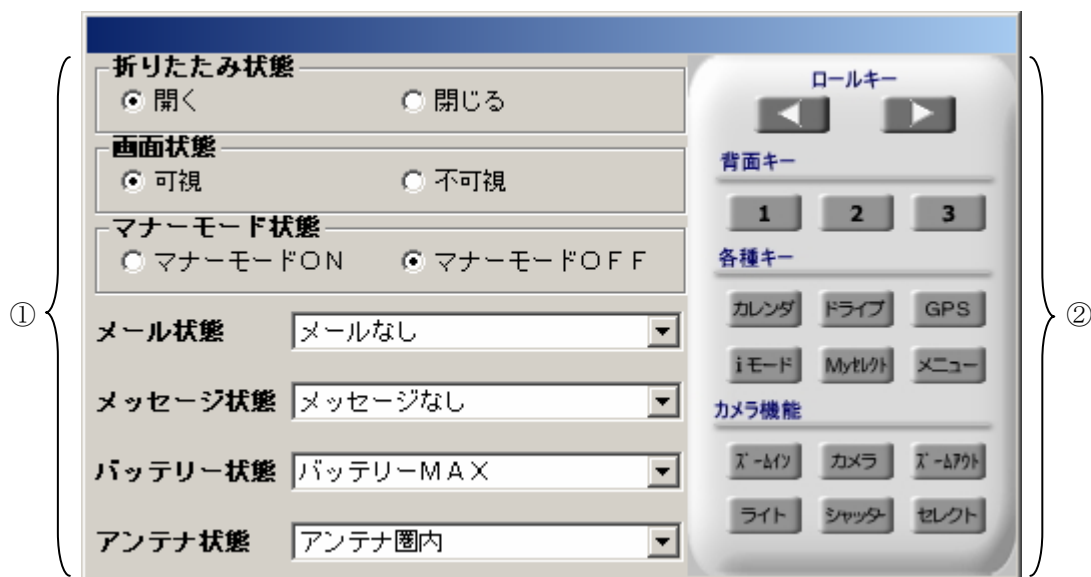


図 5-4 1 子ウィンドウ

- ① エミュレーション端末の状態を変更する事ができます。
この状態は、エミュレーション端末のLCD上でも同様に変更する事ができます。
状態を変更するには、それぞれの状態を選択します。また、i アプリ実行前に状態を設定した場合、その状態がエミュレーション端末に反映されます。

- 「折りたたみ状態」
 - 開く
 - 閉じる
- 「画面状態」
 - 可視
 - 不可視
- 「メール状態」
 - メールなし
 - 未読メールあり
 - センターに未読メールあり
- 「メッセージ状態」
 - メッセージなし
 - 未読メッセージあり
 - センターに未読メッセージあり

- 「バッテリー状態」
 - バッテリー充電中
 - バッテリーMAX
 - バッテリーその他
- 「アンテナ状態」
 - アンテナ圏外
 - アンテナ圏内
- 「マナーモード状態」
 - マナーモードOFF
 - マナーモードON

② エミュレーション端末ウィンドウにないキーのイベントを発生させます。
エミュレーション端末ウィンドウにないキーのイベントは以下のとおりです。

- 「ロールキー」
 - 右回転
 - 左回転
- 「背面キー」
 - 背面キー1
 - 背面キー2
 - 背面キー3
- 「各種キー」
 - カレンダーキー
 - ドライブモードキー
 - GPS
 - iモードキー
 - Myセレクトキー
 - メニューキー
- 「カメラ機能」
 - ズームイン
 - カメラキー
 - ズームアウト
 - ライトフラッシュ
 - シャッター
 - カメラセレクトキー

5.23.2. 3Dサウンド表示

3Dサウンド表示ウィンドウは、エミュレーション端末の3Dサウンド表示を押すと表示されます（5.23. エミュレーション端末の操作 – ⑩ 参照）。

3Dサウンドの表示は、サウンド設定が「ビジュアルモード」に設定されている必要があります（「5.6. サウンド設定」 参照）。「通常再生モード」では表示されませんのでご注意ください。

図 5-4 2 を用いて3Dサウンド表示ウィンドウの基本操作を説明します。



図 5-4 2 3Dサウンド表示

3Dサウンド表示ウィンドウでは、上、正面、側面の三面図で3Dの座標位置を表します。

- ① 定位の指定
「極座標」か「デカルト座標」を選択します。
- ② 距離の拡大
描画する半径の大きさを指定します。
(最大で半径 12.75m、最小で半径 1.27m で表示させることができます。)

③ 音源の移動

現在再生中の音源の移動位置が表示されます。

複数再生している場合は、複数の音源が表示されます。

定位が移動開始する始点、及び定位が移動完了する終点は点線の円で描画されます。

現在の定位は、塗りつぶされた円で表示され、移動の軌跡は点線で描画されます。

④ 仮想音源（情報）

仮想音源の定位情報を表示します。①の「定位の指定」により選択された座標系で、現在の定位情報を表示します。

「発音」の表示は MFi に埋め込まれたオーディオデータが再生中の場合は ♪ が表示されます。

停止中の場合は何も表示されません（アプリから 3D 環境を制御する場合、「発音」は何も表示されません）。

第6章. ツール

この章では、エミュレータで扱える便利なツールについて説明します。

6.1. スクラッチパッド管理ツール

スクラッチパッド管理ツールは、i アプリのスクラッチパッドファイルを 16 進数でダンプして編集するためのツールです。

スクラッチパッド管理ツールを使用すれば、スクラッチパッドのデータを作成したり、i アプリからスクラッチパッドにデータが適切に書き込まれているかを確認したりすることができます。

6.1.1. スクラッチパッドファイル

スクラッチパッドは、実際の端末では不揮発性の記憶領域ですが、エミュレータでは 1 つのファイルです。スクラッチパッドには、文字列以外に i アプリで使用可能な基本タイプのデータを格納できるため、通常テキストビューアなどで表示することができません。

エミュレータは、スクラッチパッドファイルを以下のディレクトリに格納します:

“C:\iDKDoJa4.1\apps\<プロジェクト名>\sp“

スクラッチパッドファイルは <プロジェクト名>.sp です。

スクラッチパッドの指定方法については、ADFのSPSize（「4.7. アプリケーションディスクリプタファイル（ADF）の編集」）を参照してください。

6.1.2. スクラッチパッド管理ツールの起動

スクラッチパッド管理ツールを起動するには、

1. メニューから「ツール」→「スクラッチパッド...」を選択します。

「スクラッチパッド管理ツール」画面が表示されます（図 6-1）。



図 6-1 スクラッチパッド管理ツール

2. 現在、読み込まれているプロジェクトの最初のスクラッチパッドファイルの内容が、16進数で表示されます。
3. 値を変更したい場合は、変更したい所の値を選択して直接変更します。



4. 「前へ」 ボタンを押すと、1つ前のスクラッチパッドの内容が表示されます。
5. 「次へ」 ボタンを押すと、1つ先のスクラッチパッドの内容が表示されます。
6. プルダウンからスクラッチパッドの連番を選択すると、そのスクラッチパッドの内容が表示されます (図 6-2) 。



図 6-2 スクラッチパッドの選択

7. 「適用」ボタンを押すと、表示されているスクラッチパッドの変更がスクラッチパッドファイルに書き出されます。
8. 「取込」ボタンを押すと、カーソルの位置から外部のファイルを取り込むことができます。
「取込」ボタンを押すと、「ファイル選択」ダイアログが表示されるので、取り込みたいファイルを選択し、「開く」ボタンを押します。

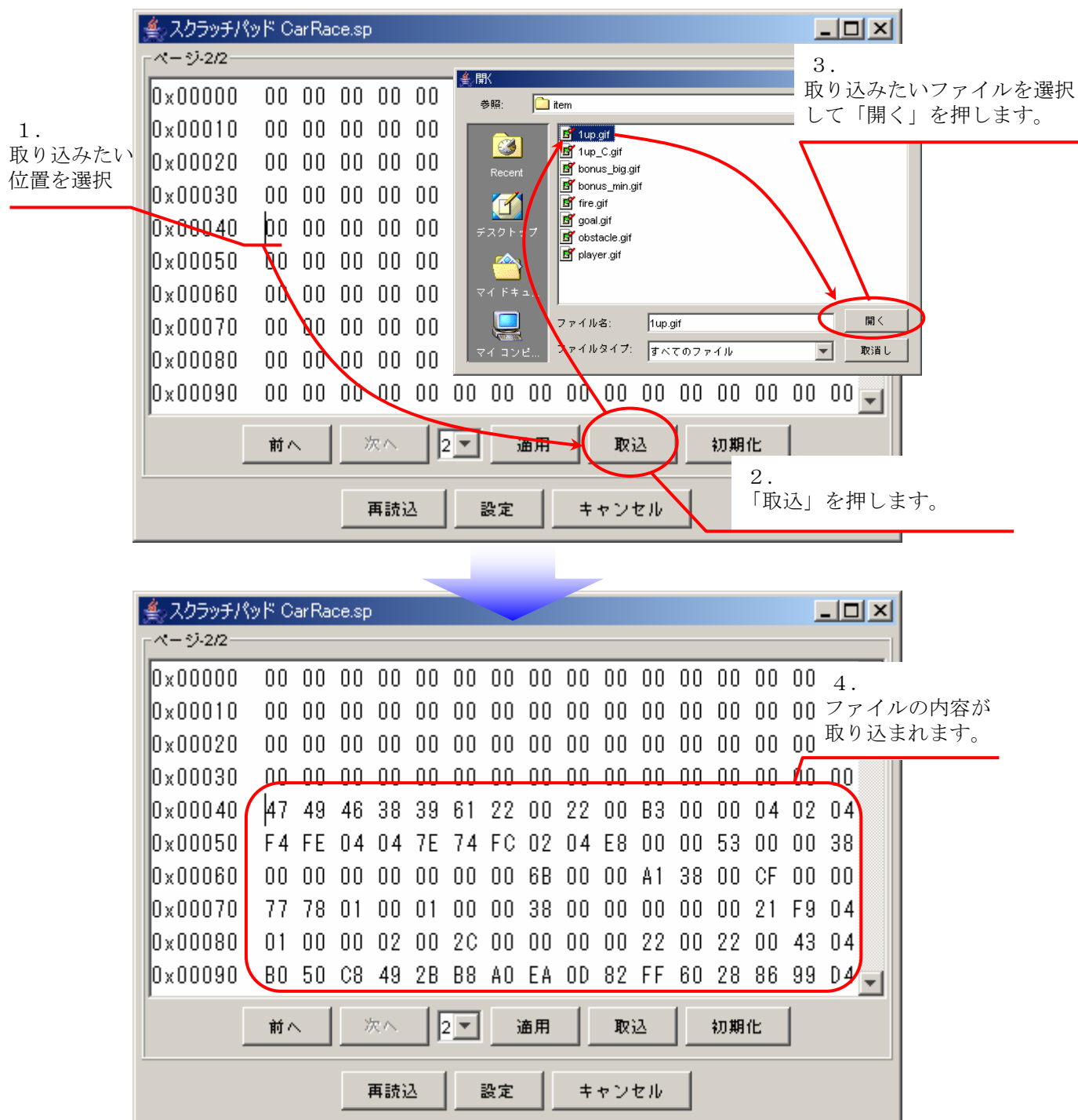


図 6-3 スクラッチパッドの置換処理

これにより、これまで他のバイナリエディタや HTTP サーバを経由してアプリケーションで取り込まないと作れなかったスクラッチパッドファイルをあらかじめ作成しておくことができます。

【注意】

スクラッチパッドのサイズは変わりませんので、取り込むファイルのサイズがスクラッチパッドのサイズを超えた場合は、ファイルの途中で切られてしまいます。その場合、全て取り込めなかった旨が警告されます。

9. 「初期化」ボタンを押すと、現在、表示されているスクラッチパッドが 0 で初期化されます。
このとき、スクラッチパッドを初期化してもよいか確認のダイアログが表示されますので、「はい」を選択します。



図 6-4 スクラッチパッド削除確認ダイアログ

10. 「再読込」ボタンを押すと、スクラッチパッドファイルの内容を全て再読込みし、スクラッチパッド管理ツールの表示時点もしくは「更新」を押した時点の状態に戻ります。
11. 「設定」ボタンを押すと、これまでの変更がスクラッチパッドファイルに書き出されます。
12. 「閉じる」ボタンを押すとスクラッチパッド管理ツールを終了します。

6.1.3. スクラッチパッドでJarファイルを扱う場合の注意事項

スクラッチパッドにJarを保存してアプリケーションで扱う場合、JarファイルのサイズをURLのlengthで指定するか、スクラッチパッドのサイズ（ADFのSPsize（「4.7. アプリケーションディスクリプタファイル（ADF）の編集」参照））とJarファイルのサイズが同じでないとフォーマットエラーになります。

6.2. スクラッチパッドを全て初期化

メニューから「プロジェクト」→「スクラッチパッドを全て初期化」を選択することで、現在読み込まれているプロジェクトのスクラッチパッドを全て0に初期化することができます。このとき、スクラッチパッドをすべて初期化してもよいか確認のダイアログが表示されますので、「はい」を選択します。

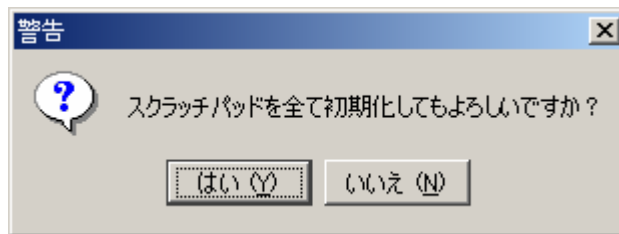


図 6-5 スクラッチパッド削除確認ダイアログ（全て）

スクラッチパッドが存在しない場合は、「スクラッチパッドがありません」というダイアログが表示されます。

6.3. 仮想 FeliCa データファイルビューワ

仮想 FeliCa データファイルビューワは、仮想 FeliCa データファイルを 16 進数でダンプします。仮想 FeliCa データファイルビューワを使用すれば、i アプリから仮想 FeliCa データファイルにデータが適切に書き込まれているかどうかを確認する事ができます。

6.3.1. 仮想 FeliCa データファイル

仮想 FeliCa データファイルは、以下のディレクトリに作成します：

C:\iDKDoJa4.1\apps\<プロジェクト名>\<プロジェクト名>.fc

6.3.2. 仮想 FeliCa データファイルビューワの起動

仮想 FeliCa データファイルビューワを表示するには、

1. メニューから「ツール」→「仮想FeliCaデータファイルビューワ...」を選択します。

「仮想FeliCaデータファイルビューワ」画面が表示されます（図 6-6）。

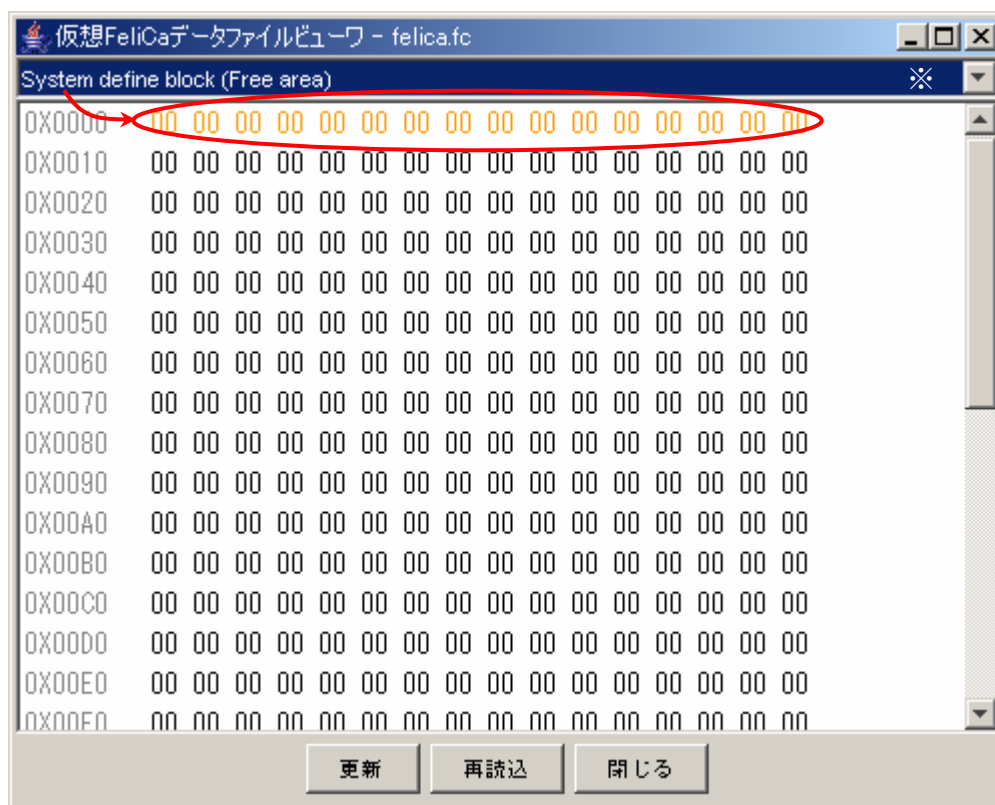


図 6-6 仮想 FeliCa データファイルビューワ画面

インストール直後のデータファイルは、データ領域を除き、適切な値が入っています。仮想FeliCaデータファイルの構造については、「9.14. 仮想FeliCaデータファイル」を参照してください。

2. 現在読み込まれているプロジェクトの仮想 FeliCa データファイルの内容が 16 進数で表示されます。
3. 上段のプルダウン（※）から仮想 FeliCa データファイルの記憶領域を選択します。該当する記憶領域のデータの色が変わります。これにより、実際に i アプリから書き込まれた領域を参照する事ができます。
4. 値を変更したい場合は、変更したい所の値を選択し直接変更します。仮想 FeliCa データファイルビューワでは、CPID Data Block, Data Block, PIN Data Block のみ値の変更が可能です。

00 00 00 → 00 1A 00
00 00 00 00 00 00

※ Area DefineBlock, Service Block 1, Service Block 2, PIN Service Block は変更しても無効です。

5. 「更新」ボタンを押すと、これまでの変更結果が仮想 FeliCa データファイルに書き出されます。
6. 「再読み込み」ボタンを押すと、仮想 FeliCa データファイルの内容を再度読み込み、仮想 FeliCa データファイルビューワの表示時点もしくは「更新」を押した時点の状態に戻ります。
7. 「閉じる」ボタンを押すと仮想 FeliCa データファイルビューワを終了します。

仮想 FeliCa データファイルビューワで表示されたフリー領域を扱う i アプリを利用するには、次の条件が満たされている必要があります：

- CPID Data Blockが、ADFのFeliCaCPIDと同じ値でなければなりません（「4.7. アプリケーションディスクリプタファイル（ADF）の編集」参照）。
- PIN Data Blockの先頭4バイトが、ADFのFeliCaPINの値を16進数表現した値でなければなりません（「4.7. アプリケーションディスクリプタファイル（ADF）の編集」参照）。

設定手順は：

1. 仮想 FeliCa データファイルビューワの上段にあるプルダウンから CPID Data Block を選択します。
2. CPID を格納する場所が桃色で表示されるので、この部分にカーソルを当て、ADF に設定した FeliCaCPID と同じ値に変更します。
3. 仮想 FeliCa データファイルビューワの上段にあるプルダウンから PIN Data Block を選択します。
4. PIN を格納する場所が紫色で表示されるので、この部分にカーソルを当て、先頭4バイトに ADF に設定した FeliCaPIN の値を16進数表現で設定します。例えば、ADF に設定した FeliCaPIN の値が 12345678 の場合は、PIN Data Block の先頭4バイトを 00BC614E に変更します。
5. 仮想 FeliCa データファイルビューワの「更新」ボタンを押します。
6. 「閉じる」ボタンを押します。

【注意事項】

- 仮想 FeliCa データファイルビューワを表示している状態で、i アプリから仮想 FeliCa データファイルヘデータを書き込んだ場合は、「再読込」ボタンを押してください。
- i アプリが起動している状態では、仮想 FeliCa データファイルビューワから値を変更しないでください。もし、変更した場合は i アプリが不正な動作をすることがあります。
- フリー領域を利用する i アプリで、ADF の FeliCaCPID、FeliCaPIN の値を設定して、仮想 FeliCa データファイルを作成しても、フリー領域の CPID Data Block、PIN Data Block には自動的に反映されません。

6.3.3. 仮想 FeliCa データファイルビューワの表示色

仮想 FeliCa データファイルビューワの表示で、ウィンドウ内の色は下記のようになっています。

橙 : SystemDefineBlock
 緑 : AreaDefineBlock
 赤 : Service Block
 水色 : PIN Service Block
 桃 : CPID Data Block および Data Block
 紫 : PIN Data Block

【F 1 2】を押すと全てを色分けして表示します。

6.3.4. ブロックの構成

仮想 FeliCa データファイルビューワでは各ブロックを下記のような構成になっています。

1 ブロックは 16 バイトで、以下のようなバイト列になっています。

D0	D1	D2	D3	・	・	・	・	・	D13	D14	D15
----	----	----	----	---	---	---	---	---	-----	-----	-----

D = バイト

- SystemDefineBlock
D0～D1 : System Code
- AreaDefineBlock
D0～D1 : Area Code
D2～D3 : End Service Code
- Service Block
D0～D1 : Service Code

仮想 FeliCa データファイルビューワで編集を行う場合のエンディアン表記については、全てビッグエンディアンになっています。

6.4. Javadoc 作成ツール

Javadoc 作成ツールは、i アプリのドキュメンテーションコメントから javadoc を作成します。
i アプリライブラリのjavadocを作成する場合は、スタブソースを外してから実行してください（i アプリライブラリについては、「6.7. i アプリライブラリ」参照）。

Javadoc 作成ツールで javadoc を作成するには：

1. メニューから「ツール」→「javadoc作成ツール...」を選択します（図 6-7）。

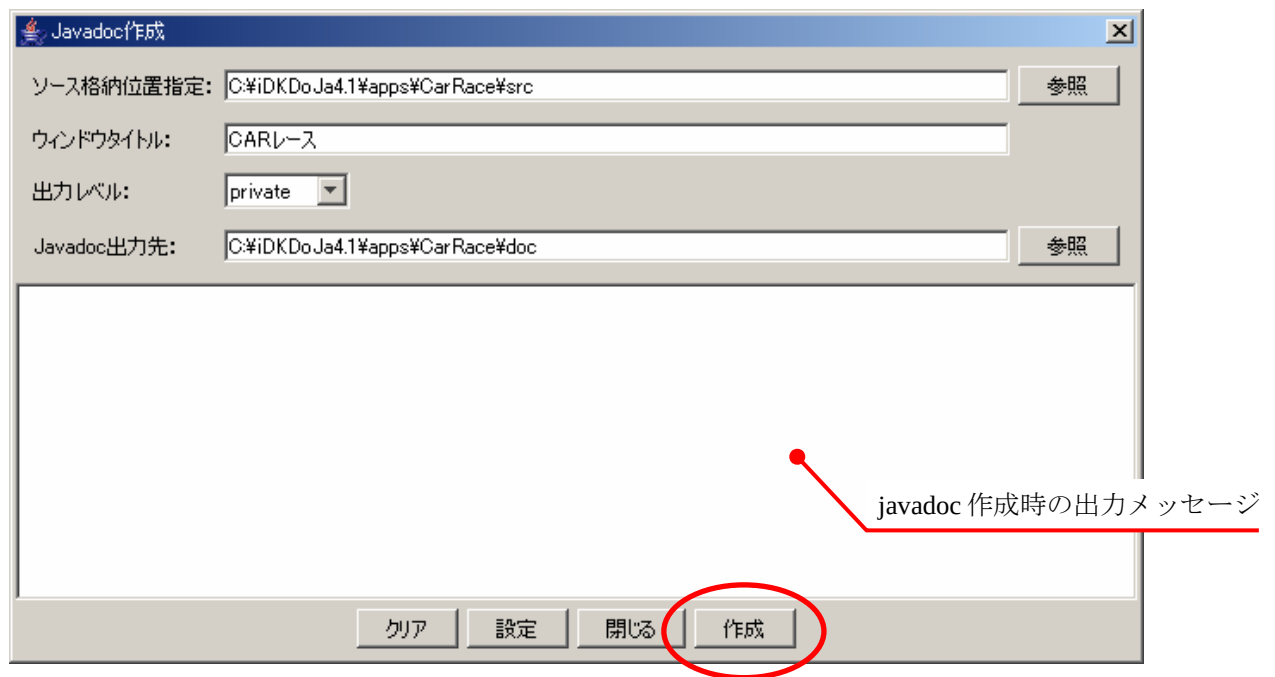


図 6-7 javadoc 作成ツール

2. 各設定項目を入力して「作成」ボタンを押すと javadoc が作成されます。
まずは、サンプルのCarRaceで試してみてください。
各設定項目は表 6-1 に示すとおりです。

表 6-1 ヘルプの設定項目

設定項目	説明	デフォルト
ソース格納位置指定	i アプリのソースが置かれたディレクトリ	読み込まれたプロジェクト
ウィンドウタイトル	javadoc のタイトル	読み込まれたプロジェクトの ADF の AppName
出力レベル	javadoc で出力するスコープ	protected
Javadoc 出力先	javadoc の出力先ディレクトリ	読み込まれたプロジェクト直下に doc で作成

3. 「設定」 ボタンを押すと設定を保存します。

設定された状態は、次のファイルに保存されています。

<プロジェクト名>\javadoc.properties

「javadoc.properties」 ファイルの記述方法は、以下のとおりです：

項目	内容
srcdir = C:\iDKDoJa4.1\apps\CarRace\src	ソース格納位置
windowtitle = CAR レース	ウィンドウタイトル
d = C:\iDKDoJa4.1\apps\CarRace\doc	javadoc 出力先
xxx = xxx	javadoc のパラメータで値指定のあるもの (※) <例> encoding = "Shift_JIS" footer = "フッター" ただし、以下のパラメータは扱えません。 • help • doclet <class> • docletpath <path> • breakiterator • source <release> • extdirs <dirlist> • verbose • J<flag> • docfilessubdirs • link • linkoffline • excludedocfilessubdir • quiet • serialwarn • taglet • tagletpath • stylesheetfile • bootclasspath <pathlist> • classpath <pathlist> bootclasspath、classpath は内部処理で作成する為、ユーザ指定は対応しません。
xxx	javadoc のパラメータで値指定のないもの (※) <例> private use

【注意】

本機能で設定可能なオプションは j2sdk1.4.2_04 に含まれる javadoc でサポートされている範囲のものであり（上記に記載されている対象外パラメータは除く）、それ以降に追加されたオプションについてはサポートしておりません。（無効なオプションとして処理されます。）

(※)

javadoc の各パラメータにつきましては、下記方針に従ってチェックを実施しております。
これにあわせて指定して下さい。

<形式チェック方針>

- 1) 値指定が必要なパラメータに対しては、値が設定されていること。
また、値指定が不要なパラメータに対しては、値が設定されていないこと。
- 2) ファイルを指定する場合は、指定ファイルが存在すること。
また、パスを指定する場合は、指定パスが存在すること。
- 3) javadoc に文字を埋め込むようなパラメータの値、及び encoding, docencoding, charset
の値は""で囲むこと。
上記以外のパラメータの値は""で囲まないこと。

6.5. ADFチェックツール

ADF チェックツールは、i アプリの ADF ファイル（.jam）が正しく作成されているかをチェックするためのツールです。

実際の i モード携帯電話機では、ADF が間違っているにもかかわらず単純なエラーメッセージしか表示されず、何が間違っているかが分からないケースがあります。

本ツールでは、こうした現象を解消するため、HTTP サーバ上に設置された ADF ファイルを読み取ってチェックする事ができます。

ADF チェックツールを起動するには：

1. メニューから「ツール」→「ADF チェック...」を選択します（図 6-8）。

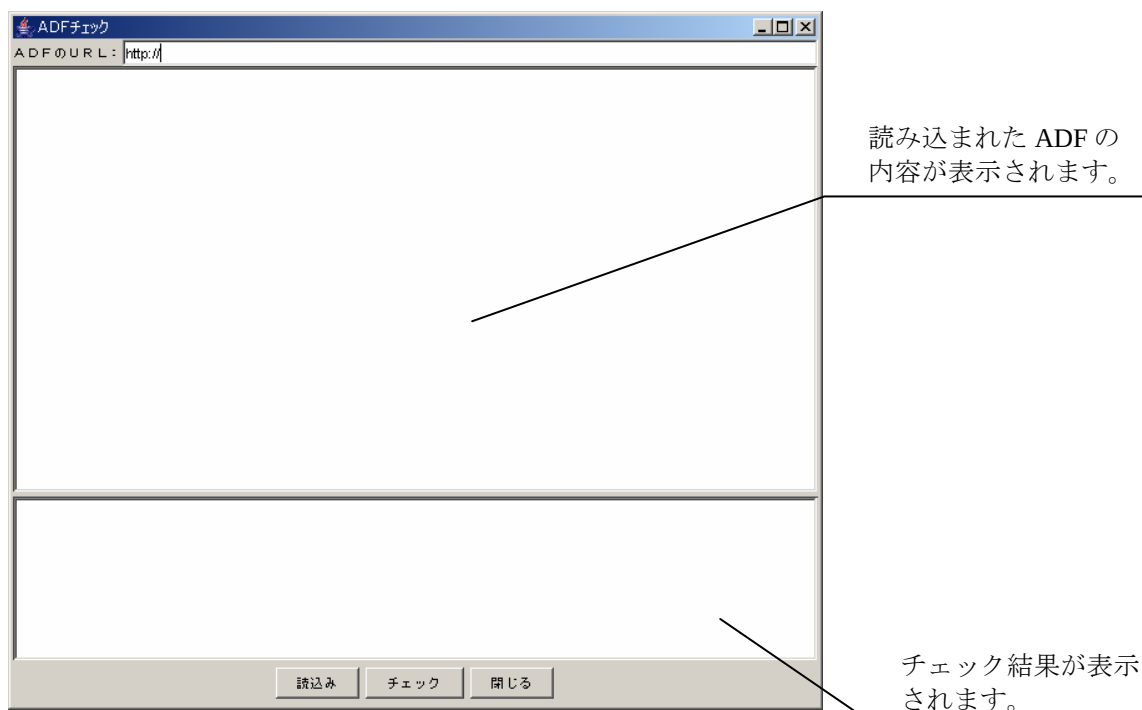


図 6-8 ADF チェックツール

2. 「ADFのURL」にチェックしたいADFのURLを指定します。
このとき、「file:ADFへのパス」を指定するとローカルにあるADFをチェックすることもできます。（例：file:c:/iDKDoJa4.1/apps/CarRace/bin/CarRace.jam）
3. 「読み込み」ボタンを押すとADFをHTTP経由で取得します。
4. 「チェック」ボタンを押すと読み込んだADFの構文が正しいかどうかをチェックします（図 6-9）。

「ADFのチェックは正常に終了しました」と表示されれば、ADFは正しいです。

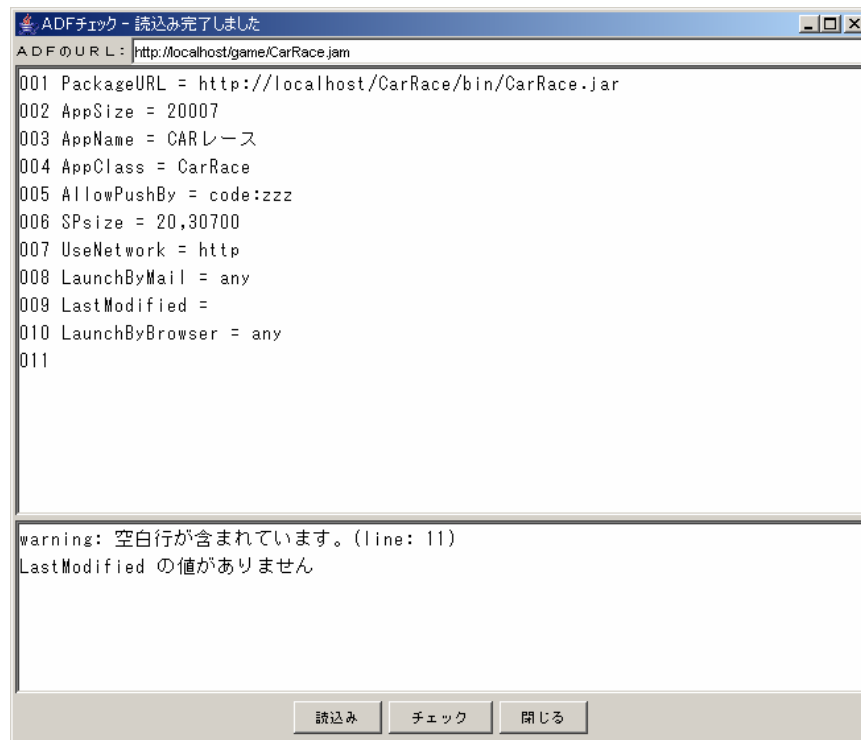


図 6-9 ADF のチェック実行

- ※ HTTP サーバで認証処理がある場合、ダウンロードする事ができない場合があります。
- ※ Jar との整合性はチェックしていませんので、もし、本ツールで正しくても i モード携帯電話上で実行できない場合は、PackageURL や AppClass が正しいかどうかを確認してみてください。

ワンポイント

ADFにはiアプリ(Jar)のサイズの指定(AppSize)がありますので、もし、iアプリをビルドしなおした場合はADFも更新されています。
そのため、Jarに変更があった場合は、必ずADFも置き換えるようにしてください。
(Jarのみの置き換えはNGと考えましょう。)

6.6. スクリーンショット

スクリーンショットは、端末ウィンドウの i アプリ実行画面をクリップボードにコピーする機能です。（クリップボードとは、Windows のカット＆ペーストをする作業領域です。）

スクリーンショットを実行するには：

1. i アプリを実行します。
 2. スクリーンショットを取りたい時に、メニューから「ツール」→「スクリーンショット」を選択します（図 6-10）。
- （あらかじめ、ショートカットキーを設定しておくとう便利です（5.15 参照）。）

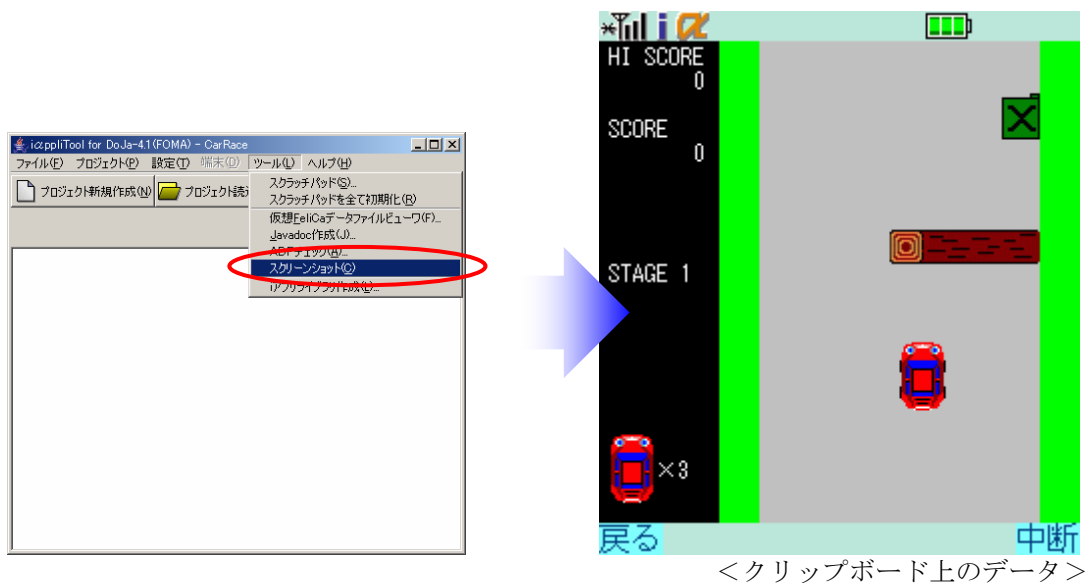


図 6-10 スクリーンショットの実行

3. クリップボードにコピーされているので、これを i アプリの説明などに利用します。

6.7. iアプリライブラリ

iアプリを作成すると同じような処理が多く存在してきます。
本機能では、これらの同じような処理をiアプリライブラリとしてまとめ、他のiアプリで利用できるようにします。

6.7.1. iアプリライブラリの作成

iアプリライブラリを作成するには：

1. まず、ライブラリ用のプロジェクトを作成します。
メニューから「プロジェクト」→「新規作成」を選択します。



図 6-1 1 iアプリライブラリ プロジェクト新規作成

2. プロジェクト名を入力して「作成」を押します。
3. 次にメニューから「プロジェクト」→「ソースファイルの新規作成」を選択して、ソースファイルを作成します。

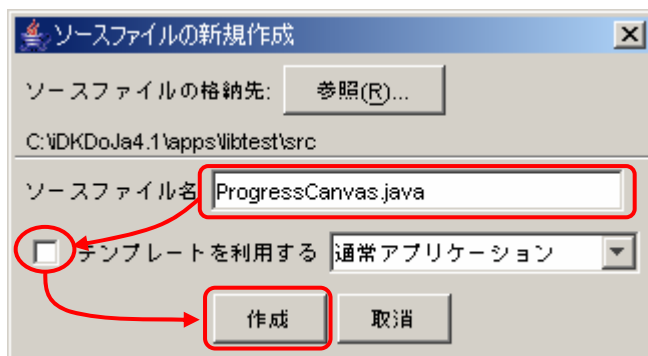


図 6-1 2 iアプリライブラリ ソースファイル新規作成

4. ソースファイル名を入力して、「作成」を押します。
(ここでは、「テンプレートを利用する」のチェックを外しておきます)

5. 開かれたソースファイルに共通処理のプログラムを書き込みます。



図 6-13 i アプリライブラリ プログラムの作成

(図の例は、サンプルアプリ CarRace の ProgressCanvas.java を書き換えたものです。)

6. 「ツール」 → 「i アプリライブラリ作成」を選択します。

7. 「i アプリライブラリを作成する」にチェックをして、このライブラリの属性「Jarファイル名、名前、バージョン、会社名、作者名」と「スタブソース名」(.java)を設定します(図 6-1 4)。

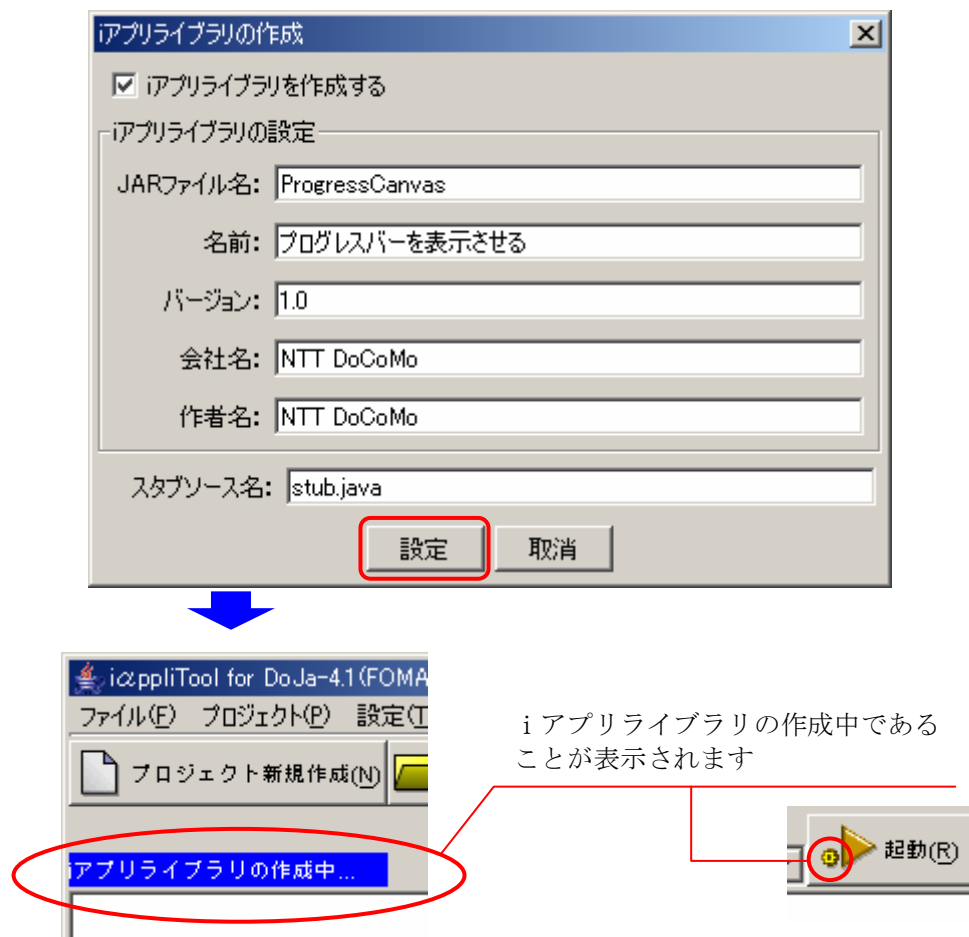


図 6-1 4 i アプリライブラリの作成

ここで設定した属性は、i アプリライブラリ中の JAR に META 情報で格納されます。
i アプリライブラリを使う i アプリや i アプリライブラリそのものには影響ありません。

各設定項目は表 6-2 に示すとおりです。

表 6-2 i アプリライブラリの設定項目

設定項目	説明	デフォルト
Jar ファイル名	i アプリライブラリの Jar ファイル名	なし
名前	i アプリライブラリの名前 (省略可)	なし
バージョン	ライブラリのバージョン番号 (省略可)	なし
会社名	作成者の会社名 (省略可)	なし
作者名	作者名 (省略可)	なし
スタブソース名	ライブラリをテストするためのプログラムです。あらかじめ作成しておかなくてもテンプレート ^[8] から自動的に作成されます (図 6-15)。 省略すると”プロジェクト名_stub.java”が作成されます。	xxx_stub.java

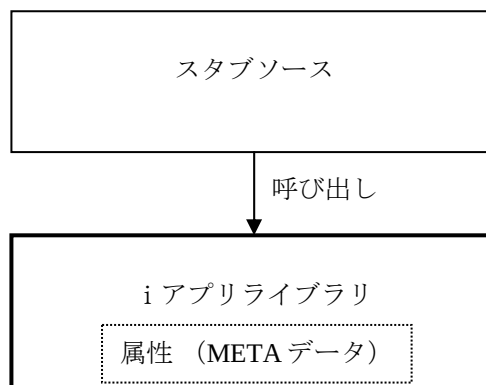


図 6-15 スタブソースと i アプリライブラリの関係

^[8] スタブソースのテンプレートは、以下のファイルに格納されています。
ファイル中の"{CLASSNAME}"がクラス名、"{AUTHOR}"が作者名、"{VERSION}"がバージョン、"{DATE}"が日時、"{COMP_NAME}"が名前に変更されます。

C:\iDKDoJa4.1\lib\ComponentStub.txt

8. 「プロジェクト」→「ソースファイルを開く」で、スタブソースを開き、ライブラリの呼び出しに合うようにプログラムを変更します。

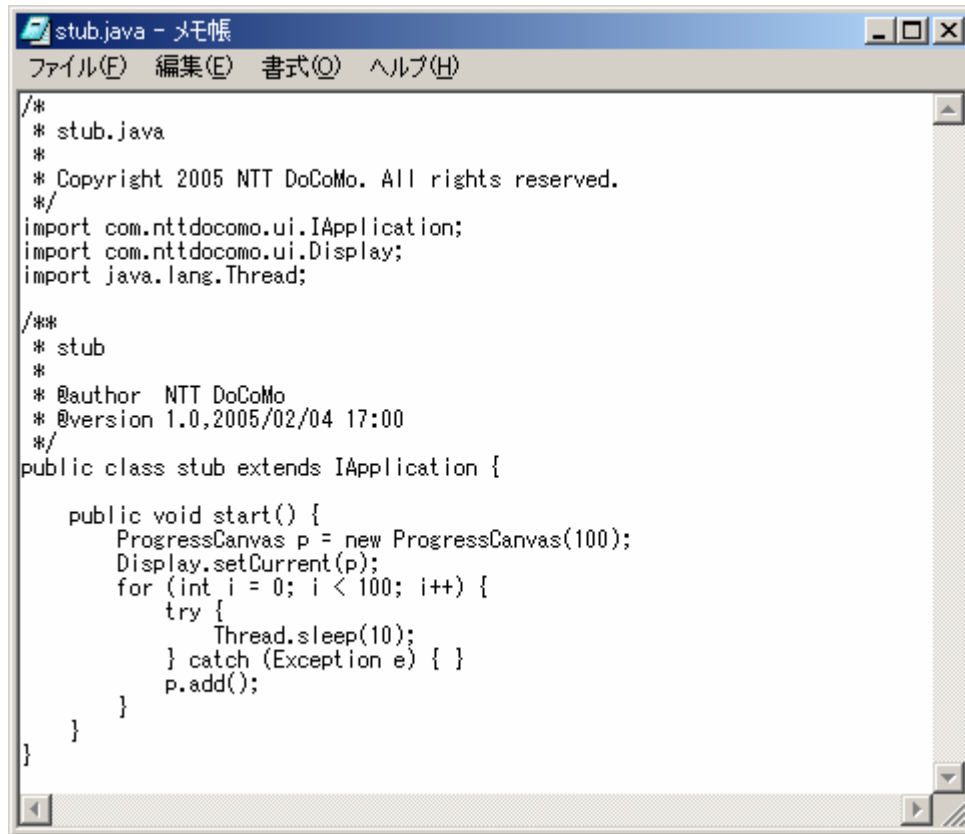


図 6-1 6 i アプリライブラリ スタブソースの書き換え

9. 「ビルド」を実行します。

10. 「起動」を押して i アプリライブラリを実行（テスト）します。

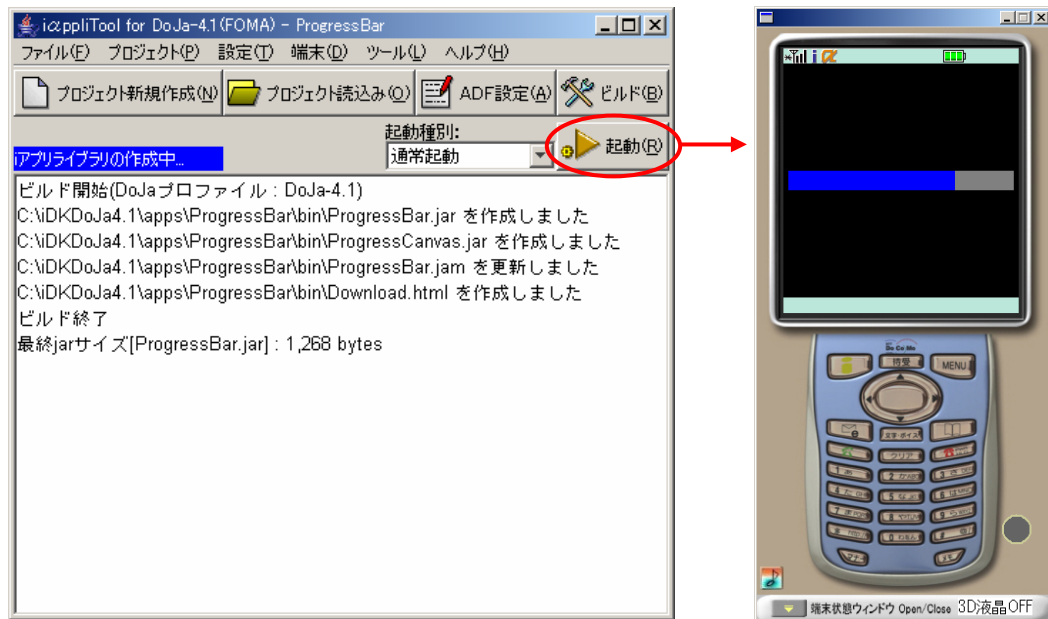


図 6-17 i アプリライブラリの実行

11. 次の場所に完成した i アプリライブラリができていますので配布して再利用します。

C:\iDKDoJa4.1\apps\プロジェクト名\bin\ライブラリ名.jar

注意：

スタブソースには、内部クラス（inner class）を作成しないでください。
もし内部クラスがある場合には、ライブラリ側に取り込まれてしまいます。
それでも実行上の問題はありませんが、不要なコードが含まれたライブラリになります。

旧エミュレータで作成された i アプリライブラリを本エミュレータで実行するには、必ず、ビルドしなおしてください。

6.7.2. iアプリライブラリの利用

i アプリライブラリを利用するには：

1. i アプリのプロジェクトを読み込みます。
2. メニューから「プロジェクト」→「i アプリライブラリ」を選択します。

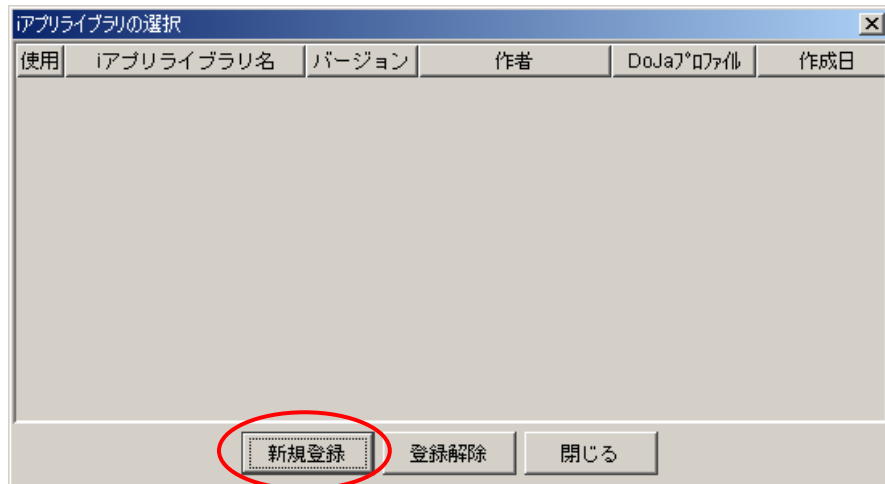


図 6-18 i アプリライブラリの選択画面

3. i アプリライブラリの選択画面が表示されるので「新規登録」を押します。
4. ファイル選択ダイアログで登録したいi アプリライブラリ(.jar)を選択するとi アプリライブラリの選択画面に追加されます。
ここで、登録されたライブラリは、他のプロジェクトでも利用できます。

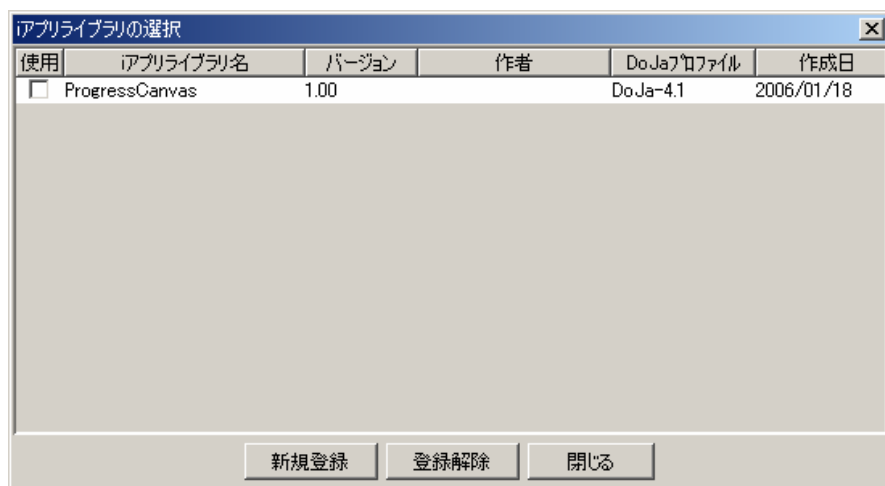


図 6-19 i アプリライブラリの選択画面 - ライブラリの追加

💡 ワンポイント

次の場所にi アプリライブラリを置いておくと自動的に登録できます。

C:\iDKDoJa4.1\lib\component

5. 不要になった i アプリライブラリは、一覧から選択して「登録解除」を押します。
(※ C:\iDKDoJa4.1\lib\component のライブラリは解除できません。)
6. 登録されたライブラリから使いたいライブラリだけにチェックをつけます。



図 6-20 ライブラリの選択

7. 「閉じる」を押します。
8. i アプリをビルドします。
これだけで、チェックのついた i アプリライブラリが自動的に組み込まれ、i アプリからライブラリ中のクラスやメソッドを呼び出すことができますようになります。

【ご注意】

i アプリライブラリの登録では、i アプリライブラリの格納先が記憶されているので、i アプリライブラリを別の場所に移動した場合は、登録しなおす必要があります。

i アプリライブラリの作成 (6.7.1) で別の i アプリライブラリを利用する事も可能です。ただし、別の i アプリライブラリは作成中の i アプリライブラリには組み込まれません。

6.8. CPBindingAuth 署名付与ツール

i アプリから SD メモリなどのストレージデバイスにアクセスするためには、ADF の CPBindingAuth と PackageHashCode の設定が必要になります。(ADF については、「4.7. アプリケーションディスクリプタファイル (ADF) の編集」参照)

本ツールは、証明書と i アプリの JAR ファイルから CPBindingAuth キーと PackageHashCode キーを生成して ADF ファイルに追加します。

本ツールは、ドコモが発行するユーザ証明書を使うもの (bindsigner) とユーザが独自に発行した証明書を使うもの (bindsigner2) の 2 通りがあります。

6.8.1. ドコモが発行したユーザ証明書を使って署名する方法

USBケーブルで繋いだFOMA端末からドコモが発行するユーザ証明書を使って、ADFのCPBindingAuthとPackageHashCodeを設定します。

ドコモが発行するユーザ証明書は、FOMA向けのクライアント認証サービス **FirstPass** で携帯端末上に取得することができます。

CPBindingAuth署名付与ツールの概要図を 図 6-2 1 に示します。

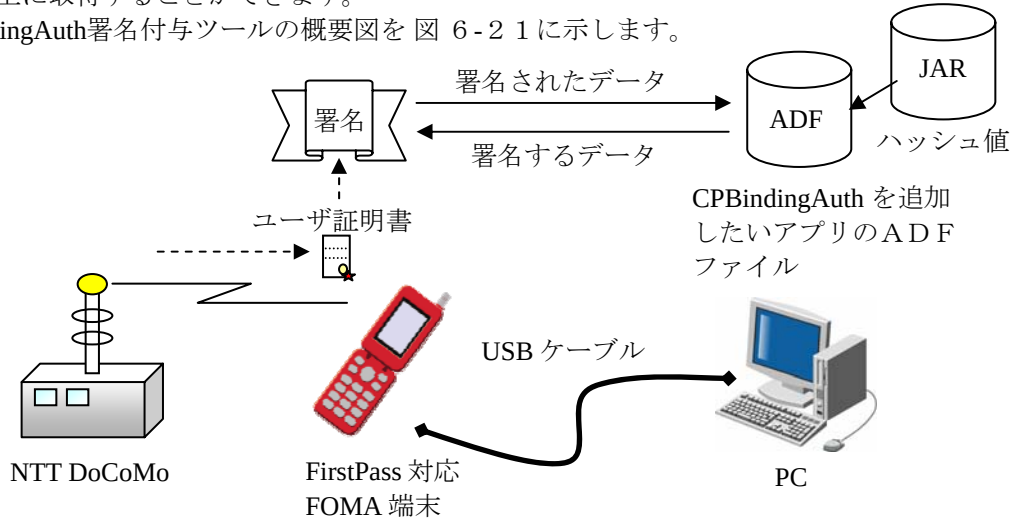


図 6-2 1 CPBindingAuth 署名付与ツール 概念図

CPBindingAuth を設定するには：

1. FOMA 端末で i モードメニューから FirstPass のユーザ証明書を取得しておきます。
2. FOMA 端末と PC を USB ケーブルで接続します。
USB ケーブルを接続するには、別途、機種毎にドライバが必要となります。
各ドライバは、<http://www.nttdocomo.co.jp/service/data/foma/> → 「通信設定ファイル」 から入手できます。
3. コマンドプロンプトを開き、C:\iDKDoJa4.1\bin に移動します。
4. CPBindingAuth 署名付与ツールを実行します。
ツールの実行形式は以下のとおりです。

```
bindsigner -adf ADF -jar JAR -pin2 PIN2_CODE -port COM_PORT | [-help]
```

ADF	: CPBindingAuth キーを追加したい ADF ファイル (UseStorage が設定されている必要があります。)
JAR	: i アプリの JAR ファイル
PIN2_CODE	: FOMA 端末の PIN2 コード
COM_PORT	: USB ケーブルを接続している COM ポート名 (Windows のデバイスマネージャで確認してください。)

<実行例>

```
bindsigner -adf CarRace.jam -jar CarRace.jar -pin2 0000 -port COM1
```

5. 「xxx.jam の CPBindingAuth に署名しました。」と表示されたら、ADF ファイルに CPBindingAuth と PackageHashCode が追加されています。
このとき既に ADF ファイルに CPBindingAuth や PackageHashCode が存在する場合は書き換えられます。

6.8.2. ユーザが発行した証明書を使って署名する方法

ユーザが発行した証明書を使って、ADF に CPBindingAuth の署名データを設定します。
この証明書は、**PKCS#12 形式**で秘密鍵が格納されていなければなりません。

CPBindingAuth を設定するには：

1. 認証用の PKCS#12 ファイルを用意します。
2. コマンドプロンプトを開き、C:\iDKDoJa4.1\bin に移動します。
3. CPBindingAuth 署名付与ツールを実行します。
ツールの実行形式は以下のとおりです。

```
bindsigner2 -adf ADF -jar JAR -p12 PKCS#12 | [-help]
```

ADF	: CPBindingAuth キーを追加したい ADF ファイル (UseStorage が設定されている必要があります。)
JAR	: i アプリの JAR ファイル
PKCS#12	: PKCS#12 ファイル

<実行例>

```
bindsigner2 -adf CarRace.jam -jar CarRace.jar -p12 usercert.p12
```

4. 「Enter Password:」で、PKCS#12 ファイルに設定されたパスワードの入力を聞いてくれるので、パスワードを入力します。
5. 「xxx.jam の CPBindingAuth に署名しました。」と表示されたら、ADF ファイルに CPBindingAuth と PackageHashCode が追加されています。
このとき既に ADF ファイルに CPBindingAuth や PackageHashCode が存在する場合は書き換えられます。

第7章. エミュレータの終了

エミュレータを終了するには、メニューから「ファイル」→「終了」を選択します。
エミュレータを終了してもエミュレータに設定した内容は消えません。また、一部の設定は、Windows ユーザ毎に保持されています。

第8章. iアプリ連携

「i アプリ連携」とは、i アプリから別の i アプリを起動することです。

i アプリ連携の起動方法には次の 2 通りがあります：

- 「ランチャモード」による起動
「ランチャモード」とは、起動元アプリケーションからあらかじめ端末内にダウンロードされた起動先アプリケーションのエントリ ID を指定して起動するモードです。
このモードは、起動先アプリケーションにパラメータを渡す事はできません。
このモードは、`!Application#launch(LAUNCH_AS_LAUNCHER)` で起動できます。
- 「連携モード」による起動
「連携モード」とは、起動元アプリケーションをダウンロードした同じホストからあらかじめ端末内にダウンロードされた起動先アプリケーションの ADF の URL を指定して起動するモードです。
このモードは、起動先のアプリケーションにパラメータを渡す事ができます。
このモードは、`!Application#launch(LAUNCH_IAPPLI)` で起動できます。

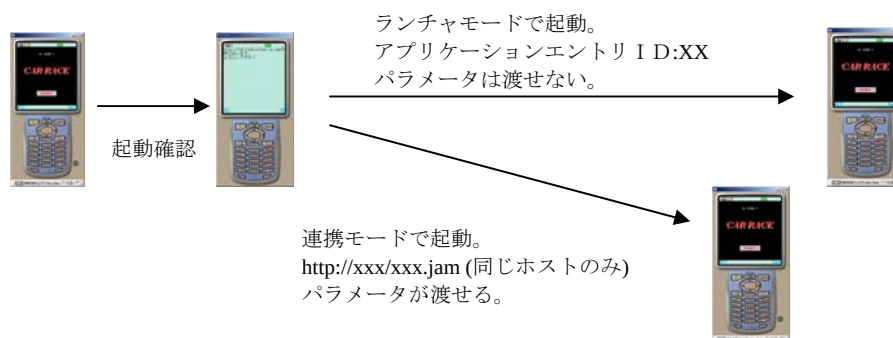


図 8-1 i アプリ連携の実行

i アプリ連携をするためには、以下の条件を満たしている必要があります:

- 起動元アプリケーションの ADF の項目で「LaunchApp」が「yes」に設定されていること。
- 起動元アプリケーションが即起動でないこと。
- 起動元アプリケーションが待ち受けアプリケーションの場合、非活性状態でないこと。
- ランチャモードの場合、起動先アプリケーションの ADF の項目で「LaunchByApp」が「deny」に設定されていないこと。
- 連携モードの場合、起動元アプリケーションの「トラステッド動作の設定」（「5.2 トラステッド動作の設定」を参照。）が 1 つでも"yes"に設定されている場合、起動先アプリケーションの ADF の項目で「TrustedAPID」が設定されていること（「4.7. アプリケーションディスクリプタファイル（ADF）の編集」を参照）。
- 起動先アプリケーションがデフォルトの格納先（「C:\iDKDoJa4.1\apps」）で作成されていること。

これは、エミュレータでの URL の参照先がデフォルトの格納先であるためです。

第9章. ファイルの記述方法

本章では、各種ファイルの記述方法を説明します。
各ファイルの場所は、以下のとおりです:

- ・ メール起動ファイル:
 C:\iDKDoJa4.1\lib\mail*.mail
- ・ バウンダリデータファイル:
 C:\iDKDoJa4.1\lib\mail\boundary.dat
- ・ メールアドレスファイル (※) :
 C:\iDKDoJa4.1\lib\mail\myAddress.dat
- ・ ブラウザ起動HTMLファイル:
 Web サーバの適切な場所
- ・ 着信履歴ファイル:
 C:\iDKDoJa4.1\lib\phonehistory\phonehistory.lst

 ここには、CallRecode#getLastRecode() で取得する
 最新の着信履歴を指定します。
- ・ 「アプリケーション名.properties」 ファイル (※) :
 C:\iDKDoJa4.1\apps\[アプリケーション名]\[アプリケーション名].properties
- ・ メールファイル (※) :
 C:\iDKDoJa4.1\lib\mail\ 配下の
 受信ボックス: received\[メッセージコード]\[受信日]_[通番].mail
 送信ボックス: sent\[メッセージコード]\[送信日]_[通番].mail
 未送信ボックス: unsent\[メッセージコード]\[送信日]_[通番].mail
- ・ 電話帳ファイル (※) :
 C:\iDKDoJa4.1\lib\phonebook\[エントリ ID]_[名前].tel
- ・ 電話帳グループファイル (※) :
 C:\iDKDoJa4.1\lib\broup\[グループ ID]_[グループ名].grp
- ・ マルチメディアファイル:
 C:\iDKDoJa4.1\lib\Multimedia\xxx.gif
 C:\iDKDoJa4.1\lib\Multimedia\xxx.jpg
- ・ ブックマーク登録ファイル (※) :
 C:\iDKDoJa4.1\lib\bookmark\[エントリ ID].bmk
- ・ スケジュール登録ファイル (※) :
 C:\iDKDoJa4.1\lib\schedule\[エントリ ID].sdl
- ・ コード認識結果ファイル:
 C:\iDKDoJa4.1\lib\Multimedia\xxx.dat
 (アプリ実行時にファイルダイアログで選択します。)

ここには、CodeReader#read() で取得するコード認識の結果を指定します。

- ・ 仮想 FeliCa データファイル (※) :
C:\iDKDoJa4.1\apps\[アプリケーション名]\[アプリケーション名].fc
- ・ スクラッチパッドデータファイル (※) :
C:\iDKDoJa4.1\sp\[アプリケーション名]XX.sp
- ・ LED表示点灯パターン登録ファイル:
C:\iDKDoJa4.1\lib\phonerresource\info_[登録 ID].txt
- ・ サウンド設定ファイル (※) :
C:\iDKDoJa4.1\lib\soundConf.properties
- ・ ストレージデバイスファイルパス:
C:\iDKDoJa4.1\lib\StorageDevice\xxx
- ・ トルカファイル:
C:\iDKDoJa4.1\lib\toruca\toruca[エントリ ID].dat

※ このファイルは、iα ppliTool から設定できます。

9.1. メール起動ファイルの記述方法

メール起動ファイル (*.mail) は、「4.1 6. メール起動」で使します。
 サンプルで、C:\iDKDoJa4.1\lib\mail\0.mail 1.mail が用意されています。
 メール起動ファイルの記述方法は、以下に示すとおりです：

1	From: xxx@docomo.ne.jp	送信者メールアドレス
2	To: zzz@docomo.ne.jp	受信者メールアドレス
3	Subject: i アプリ 起動 (IApplication)	Subject
4	Date: 2001/08/21 12:34	Date
5		
6	・・・メール本文・・・	本文
7		
8	--B:A	バウンダリ部
9	TEXT="iappli"	リンクを表示する際の文字列
10	ADF="http://*****/***.jam"	起動させる i アプリの ADF の URL
11	"A"="B"	i アプリに渡すパラメータ
12	"C"="D"	i アプリに渡すパラメータ

9.2. バウンダリデータファイル

バウンダリデータファイルは、メールに添付されたバウンダリ情報を格納するファイルです。
 このファイルの内容は、BASE64 でエンコードされた値でなければなりません。

9.3. メールアドレスファイル

メールアドレスファイルは、携帯電話に登録されたメールのアドレス情報を格納するファイルです。登録できるメールアドレスは1件のみです。

9.4. ブラウザ起動HTMLファイルの記述方法

ブラウザ起動HTMLファイルは、「4.17. ブラウザ起動」で使します。
 サンプルで、C:\iDKDoJa4.1\lib\sample.html が用意されています。
 ブラウザ起動HTMLファイルの記述方法は、以下に示すとおりです：

1	<HTML>	
2	<BODY>	
3	<OBJECT declare id ="application.declaration"	OBJECT タグ
4	data="http://*****/***.jam"	起動させる i アプリの ADF の URL ※
5	type="application/x-jam">	
6	<PARAM NAME="A" VALUE="B">	
7	<PARAM NAME="C" VALUE="D">	
8	</OBJECT>	
9	Download	アプリを連携するリンク ista="#[OBJECT タグ]" href="[OBJECT タグに対応したHTML のURL]" (変更不要)
10	</BODY>	
11	</HTML>	

※ dataに "****.jam" のみを指定した場合、プロジェクトの読み込みで指定されたプロジェクト格納先に存在するアプリが起動されます。

9.5. 着信履歴ファイル

i アプリから取得できる着信履歴が格納されたファイルです。
 着信履歴を変更したい場合は、このファイル内の履歴（4行で1履歴）を編集してください。
 ただし、i アプリから取得できるのは、新着履歴のみですので、最初の1件（最初の4行）
 だけになります。

着信履歴ファイルの記述方法は、以下のとおりです：

1	2002/08/01 07:12	着信日時
2	true	着信の成功可否 "true" または "false"
3	0	電話帳エントリの ID
4	09012345678	電話番号または、"非通知設定"、"公衆電話" ※

※ 4行目の電話番号に "非通知設定" または、"公衆電話" の文字列を指定すると、i アプリからこの文字列がそのまま取得できます。

9.6. 「アプリケーション名.properties」ファイル

端末固有の設定をこのファイルに記述します。

「アプリケーション名.properties」ファイルの記述方法は、以下のとおりです：

1	appMoveInfo = false	アプリケーションがメディア等により移動機外部から移されたものかどうかを指定 (true / false)
2	phoneBookEntry = 10	電話帳エントリの最大保存件数
3	xDcmMove = 0000	メディアリソースのHTTPヘッダ "X-Dcm-Move" で、0000 か 0001 を指定
4	moveFromOtherTerm = false	移動先と移動元の移動機が異なるかどうかを指定 (true / false)
5	adfUrl = http://*****	起動させる i アプリの ADF の URL
6	receivedMailMax = 200	受信メールの最大保存件数
7	sentMailMax = 50	送信/未送信メールの最大保存件数
8	bookmarkEntry = 10	ブックマークの最大保存件数
9	launchType = 0	i アプリ読み込み時の起動タイプ
10	groupEntry = 10	グループの最大保存件数
11	remoteParameter =	リモート制御の受信パラメータ
12	remotePushedTime =	リモート制御の受信時間
13	sound3dResource = 4	3D サウンドの制御を行うためのリソース数
14	cameraDevice = 1	カメラデバイス数
15	useChat = false	チャット機能の有効/無効 (true / false)
16	videoTrack = 1	再生可能な Video トラックの数
17	audioTrack = 1	再生可能な Audio トラックの数
18	textTrack = 1	再生可能な Text トラックの数
19	launchTime =	i アプリのスケジューラ起動時刻 n@YYYY/MM/DD/hh/mm/ss n:設定番号 1 ~ ; で区切って複数指定
20	fontBaselinePos =	フォントベースライン設定
21	stubSourceName = XXX	i アプリライブラリのスタブファイル名

9.7. メールファイル

メールファイルは、送受信メールを格納するファイルです。
メールファイルの記述方法は、以下に示すとおりです：

1	From: xxx@docomo.ne.jp	送信者メールアドレス
2	To: xxx@docomo.ne.jp	受信者メールアドレス
3	Subject: i アプリ 起動 (IApplication)	Subject
4	Date: 2002/12/01 12:34	Date
5		
6	・・・メール本文・・・	本文
7		
8	--B:P	バウンダリ部
9	MessageCode="*****"	ADF のメッセージコード
10	data="*****"	バイナリデータ ※
11	--B:P--	

※ バイナリデータは、アプリケーション固有のデータです（「5.1 3. 送受信メールの編集」を参照）。

9.8. 電話帳ファイル

電話帳ファイルは、電話帳情報を格納するファイルです。
「電話帳」ファイルの記述方法は、以下に示すとおりです：

1	1	エントリ I D
2	姓	姓
3	名	名
4	ふりがな姓	ふりがな姓
5	ふりがな名	ふりがな名
6	0001112222	電話番号 1
7	0001112222	電話番号 2
8	0001112222	電話番号 3
9	mail@xxx.xxx.xx	メール 1
10	mail@xxx.xxx.xx	メール 2
11	mail@xxx.xxx.xx	メール 3
12	1	グループ I D

9.9. 電話帳グループファイル

電話帳グループファイルは、電話帳グループ情報を格納するファイルです。
「電話帳グループ」ファイルの記述方法は、以下に示すとおりです：

1	1	エントリ I D
2	GROUP1	グループ名

9.10. マルチメディアファイル

マルチメディア領域から画像を取得する場合、ファイルダイアログが表示されるので、読み込みたい画像ファイルを指定します。

マルチメディア領域は、

C:\iDKDoJa4.1\lib\Multimedia\

になります。

ここに i アプリで利用するファイル（画像等）を置いてください。

（マルチメディア領域には、i アプリからは実際に保存されません。）

「i alpha Development Kit for DoJa-4.1(FOMA)」をインストールした時点では、あらかじめサンプルファイルが置いてあります。

9.11. ブックマーク登録ファイル

BookMark#addEntry() でブックマーク登録データが格納されたファイルです。
ブックマーク登録ファイルの内容は、以下のとおりです：

1	http://xxx.xxx.xxx/xxx	U R L
2	bookmark	ブックマーク名

9.12. スケジュール登録ファイル

Schedule#addEntry() でスケジュール登録データが格納されたファイルです。
スケジュール登録ファイルの内容は、以下のとおりです：

1	1	スケジュールサポートタイプ
2	schedule	内容
3	年/月/週/日/時/分	スケジュール日時
4	1	アラームの有無 0:FALSE 1:TRUE

9.13. コード認識結果ファイル

バーコード認識の i アプリオプシヨン A P I (CodeReader) を利用する事により、本ファイルに指定したデータで、J A N コードおよび、Q R コードを認識する事ができます。

「i α ppli Development Kit for DoJa-4.1(FOMA)」は、実際にコードを読み取れる機能を持ちませんので、バーコード認識 A P I で読み取るデータを次の様なファイルをマルチメディア領域に作成しておきます（「9.10. マルチメディアファイル」参照）。

sample1.dat

（拡張子が dat であればファイルの名前は何でも構いません。）

コード認識結果ファイル中のデータは、実際に読み込まれるデータを指定します。
読み込まれるデータは、それぞれ以下のとおりです。

1) J A N 8 コードの場合

XXXXXXXX	8 桁の数字（ショート J A N）
----------	--------------------

2) J A N 1 3 コードの場合

XXXXXXXXXXXX	1 3 桁の数字
--------------	----------

3) Q R コードの場合

[バイナリデータ]	本ファイルには、JPEG, GIF, MIDI(SMF), MFi のバイナリデータを直接指定します。 ▼ 実機で実際に認識する Q R コードは以下の通りです。 (QR コード基本仕様 JIS X 0510 参照。) ・バージョン： 1 ～ 1 0 ・誤り訂正レベル： L、M、Q、H ・モデル： モデル 2 ・対応モード： ① 数字：0～9(ASCII 値 0x30 ～ 0x39) ② 英数字：0～9(ASCII 値 0x30 ～ 0x39)、A～Z の 2 6 文字のアルファベット文字(ASCII 値 0x41～0x5A)、スペース、\$、%、*、+、-、.、/、:、の 9 個の記号(ASCII 値 0x20, 0x24, 0x25, 0x2A, 0x2B, 0x2D～0x2F, 0x3A) ③ 8-ビットバイト：JIS X 0201(ASCII 値 0x00～0xFF)に基づく文字 ④ 漢字：JIS X 0208 に基づく Shift-JIS 漢字
-----------	--

以下のようなQRコードの解析結果によって、電話帳登録、Mail To、ブックマーク登録、iアプリ連携の4つの連携データを取得することができます。
エミュレータでは連携されませんが、実機では連携して動作します。

1) 電話帳登録の場合 (※)

MECARD:	電話帳登録を表す識別子
N: [姓], [名];	名前
SOUND: [フリガナ姓], [フリガナ名];	振り仮名
TEL: [12345678901234567890123456];	電話番号 (複数指定可)
TEL- <u>AV</u> : [12345678901234567890123456];	T V 発信用電話番号 (複数指定可) 《T V 電話機能付き移動機のみ》
EMAIL: [xxxxx@docomo.ne.jp];	メールアドレス (複数指定可)
NOTE: [メモ];	メモ 《対応機種のみ》
;	終端

2) Mail To (※)

MATMSG:	メール連携を表す識別子
TO: [xxxxx@docomo.ne.jp];	宛先
SUB: [題名];	表題
BODY: [あいうえお];	本文
;	終端

3) ブックマーク登録 (※)

MEBKM:	ブックマーク登録を表す識別子
TITLE: [NTTドコモ];	サイト名
URL: [http://www.nttdocomo.co.jp/];	URL
;	終端

4) iアプリ連携 (※)

LAPL:	iアプリ起動を表す識別子
ADFURL: [http\://xxx.xx.xx/xxx.jam];	起動するiアプリのADFの取得先URL 最低でも255Byteまで指定可。 (注) コードリーダー起動を実行する場合は、 httpの後に\を忘れず付与してください。
CMD: [起動コマンド];	起動コマンドへセットする文字列 最大250Byteまで指定可。
PARAM: [NAME, VALUE];	iアプリに渡すパラメータで名前と値をカンマで区切って指定する (複数指定可) 最大255Byte/件、16件まで指定可。
;	終端

「iアプリ Development Kit for DoJa-4.1(FOMA)」は、KToolBar から「コードリーダー」を選択する事により、バーコードデータのパラメータを渡してiアプリを起動することができます (「4.18 コードリーダー起動」参照)。

※ 表記上改行していますが、実際には改行を入れないでください。

【記述例】

```
MECARD:N: ドコモ, 太郎;SOUND:ドコモ, タウ;TEL:090XXXXXXXX;TEL:03XXXXXXXX;EMAIL:XXX@docomo.ne.jp;;
```

9.14. 仮想 FeliCa データファイル

仮想 FeliCa データファイルは、FeliCa チップを仮想的に実現したもので、このチップデータをファイルで管理しています。仮想 FeliCa データファイルは、FeliCa#open()を実行すると自動的に作成されます。

本エミュレータで扱える仮想 FeliCa データファイルは、SystemDefineBlock と AreaDefineBlock を先頭に 3 つのフリーエリアに分かれています。

仮想 FeliCa データファイルのフォーマットは以下のとおりです：

領域	属性	ブロック数	備考
フリー領域	SystemDefineBlock	1	先頭 2 バイトは、SystemCode
	AreaDefineBlock	1	先頭 2 バイトは、Area0000
	AreaDefineBlock	1	Data Block を除き、先頭 2 バイトは、AreaCode/ServiceCode
	Service Block 1	1	
	CPID Data Block	1	
	Service Block 2	1	
	Data Block 0	1	
	Data Block 1	1	
	Data Block 2	1	
	Data Block 3	1	
	PIN Service Block	1	
	PIN Data Block	1	
	AreaDefineBlock	1	Data Block を除き、先頭 2 バイトは、AreaCode/ServiceCode
	Service Block 1	1	
	CPID Data Block	1	
	Service Block 2	1	
	Data Block 0	1	
	Data Block 1	1	
	Data Block 2	1	
	Data Block 3	1	
	PIN Service Block	1	
	PIN Data Block	1	
	AreaDefineBlock	1	Data Block を除き、先頭 2 バイトは、AreaCode/ServiceCode
	Service Block 1	1	
	CPID Data Block	1	
	Service Block 2	1	
	Data Block 0	1	
	Data Block 1	1	
	Data Block 2	1	
	Data Block 3	1	
	PIN Service Block	1	
	PIN Data Block	1	

1 ブロック = 1 6 バイト

SystemDefineBlock	= システム定義ブロック
AreaDefineBlock	= エリア定義ブロック
Service Block 1	= CPID が格納されているサービスのサービス定義ブロック
CPID Data Block	= CPID が格納されているユーザブロック
Service Block 2	= CP がデータを格納するサービスのサービス定義ブロック
Data Block 0~3	= CP がデータを格納するサービスのユーザブロック
PIN Service Block	= PIN サービスのサービス定義ブロック
PIN Data Block	= PIN が格納されているユーザブロック

仮想FeliCaデータファイルの内容を編集するには、仮想FeliCaデータファイルビューを使用します（「6.3. 仮想FeliCaデータファイルビュー」を参照）。

9.15. スクラッチパッドデータファイル

スクラッチパッドのデータを格納しておくファイルです。
このファイルは、i アプリからスクラッチパッドに書き込む事で作成されます。

スクラッチパッドの編集方法については、「6.1. スクラッチパッド管理ツール」を参照してください。

9.16. LED表示点灯パターン登録ファイル

i アプリ オプション A P I com.nttdocomo.opt.ui.PhoneResource#setInfomationDisplay() を実行して登録されたデータが格納されるファイルです。

「LED表示点灯パターン登録ファイル」に格納される内容は、以下に示すとおりです：

[illegible]

9.17. サウンド設定ファイル

サウンド設定で設定されたデータが格納されるファイルです。
サウンド設定ファイルのフォーマットは以下のとおりです：

1	CH_1=SM_0	MF i のチャンネル 1 の代替音
:	:	:
16	CH_16=-1	MF i のチャンネル 1 6 の代替音
17	DEVICE=null	デバイス名
18	MODE=0	再生モード 0:通常再生 1:ビジュアル

9.18. ストレージデバイス ファイルパス

ストレージデバイスでアクセスするファイルパスです。
SDBindingFile は、このフォルダ上のファイルにアクセスします。

ファイルは、ext0/SD_BIND/SVCXXXXYYYYZZZZ 配下に作成されます。

XXXX:共有識別とアクセス識別の値

YYYY:アプリケーション共有の場合：ADF のファイル名

CP 共有の場合：トラステッドAPI D (指定が無ければ 1 1 桁の 0)

ZZZZ:アクセス識別

UIM : レジストリの StorageAccessUIM の値

機器 : レジストリの StorageAccessModel の値

シリーズ : レジストリの StorageAccessSeries の値

シリーズと機器が指定された場合は機器の値になります。

9.19. トルカファイル

トルカのデータが格納されたファイルです。
トルカファイルは、次の様なファイルで作成します。

```
torucaxxxxxxxxx.dat
(xxxxxxxx: エントリ I D)
```

トルカファイルのフォーマットは以下のとおりです：

1	[ToruCa]	固定
2	Version=257	バージョン (2 バイト表記 上位バイト:メジャー 下位バイト:マイナー)
3	Type=0	トルカ種別 (0:SNIP / 1:CARD)
4	URL=http://xxx/xxx/xxx.rc	取得先URL
5	Data1=xxxxx	データ 1 (Base64 形式)
6	Data2=xxxxx	データ 2 (Base64 形式)
7	Data3=xxxxx	データ 3 (Base64 形式)
8	Cat=005D	カテゴリアイコン 2 バイト 16 進数表記
9	Body=[MIME-Version: 1.0 xxxxxxxxxxxxxxxxxxxx]	[]内に記述します

あらかじめサンプルとして、SampleToruca.dat ファイルが置いてあります。

このフォーマットは、実際のトルカファイルとは異なります。

実際のトルカ仕様のバイナリデータを読み込ませたい場合は、HTTP 通信等でバイト配列として取得し、Toruca クラスのコンストラクタ Toruca(byte[] data) を利用して Toruca クラスを生成してください。

9.20. ファイルの記述具体例

「http://xxx.yyy.zzz」の「CarRace.jam」をメール、ブラウザ、コードリーダーから起動する場合:

- メール起動
 - 起動したい i アプリ (CarRace) の ADF の「LaunchByMail」に「any」を設定します。
 - メール起動ファイルに「C:\iDKDoJa4.1\lib\mail\0.mail」を用います。
 - 「0.mail」ファイルの ADF には、
「ADF="http://xxx.yyy.zzz/CarRace.jam"」と記述します。
 - メール起動で、[Sample1](#) を起動します。
- ブラウザ起動
 - 起動したい i アプリ (CarRace) の ADF の「LaunchByBrowser」に「any」を設定します。
 - ブラウザ起動 HTML ファイルに「C:\iDKDoJa4.1\lib\sample.html」を用います。
 - 「sample.html」の OBJECT タグの data には、
「data="http://xxx.yyy.zzz/CarRace.jam"」と記述して、
<OBJECT>タグにダウンロードして起動する i アプリの ADF を記述します。
(プロジェクト読み先に CarRace が存在する場合は、
「data="CarRace.jam"」でも起動可能です。)
 - ブラウザ起動で、URL に「file:c:/iDKDoJa4.1/lib/sample.html」と指定して起動します。
- コードリーダー起動
 - 起動したい i アプリ (CarRace) の ADF の「AllowPushBy」に
「Code:command」(command は 9.13. コード認識結果ファイルの CMD: の値)を設定します。
 - コード認識結果ファイルに
「C:\iDKDoJa4.1\lib\multimedia\sample1.dat」を用います。
 - 「sample1.dat」ファイルの起動する i アプリ (ADFURL:) には、
「ADFURL:http://xxx.yyy.zzz\CarRace.jam」 と記述します。
 - コードリーダー起動で、sample1.dat を起動します。

第10章. 証明書管理

「iα ppliTool」上で HTTPS 通信を行う際に使用する証明書は、OS 組み込みのものを使用します。よって、「iα ppliTool」では管理しません。

そこで、証明書を管理するためには OS にあらかじめインストールされている証明書管理アプリケーションを使用します。

1. 「コントロールパネル」→「インターネットオプション」をダブルクリックします（図 10-1）。

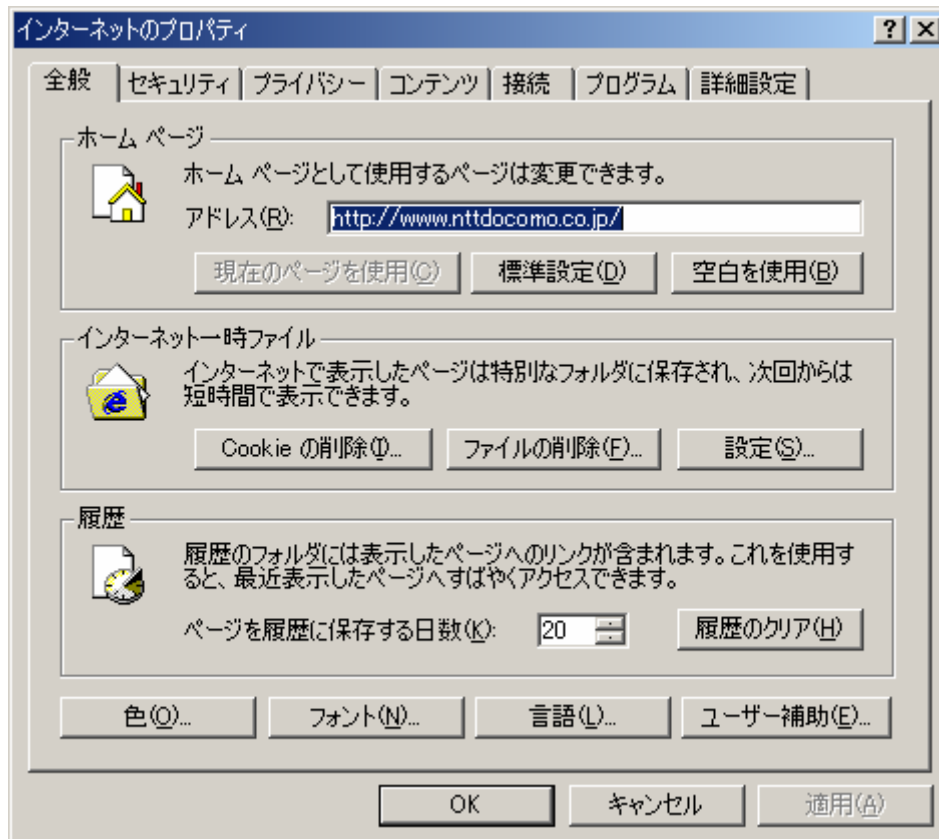


図 10-1 インターネットのプロパティ

2. 図 1 0-1 の「コンテンツ」タブを選択します（図 1 0-2）。

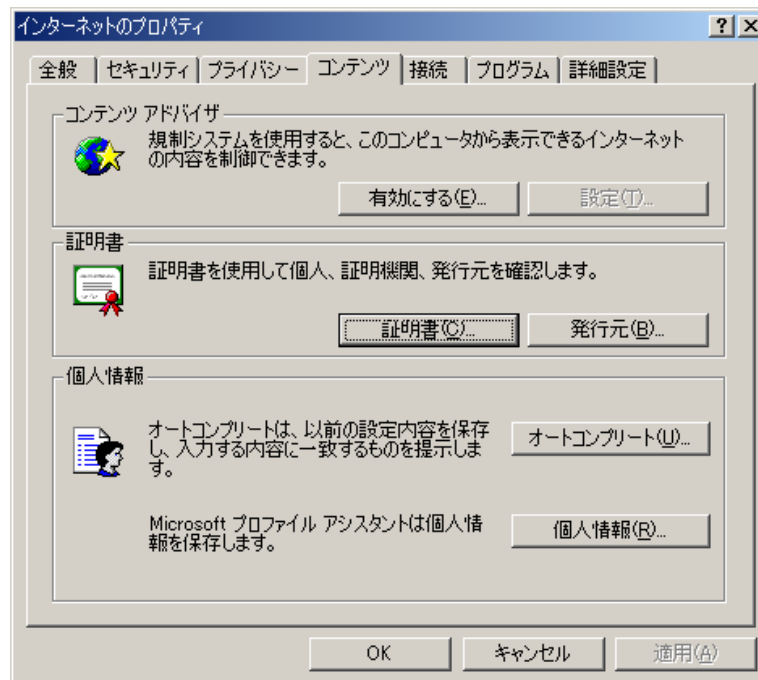


図 1 0-2 インターネットのプロパティ（コンテンツ）

3. 図 1 0-2 の「証明書 (C) ...」 ボタンを押します（図 1 0-3）。

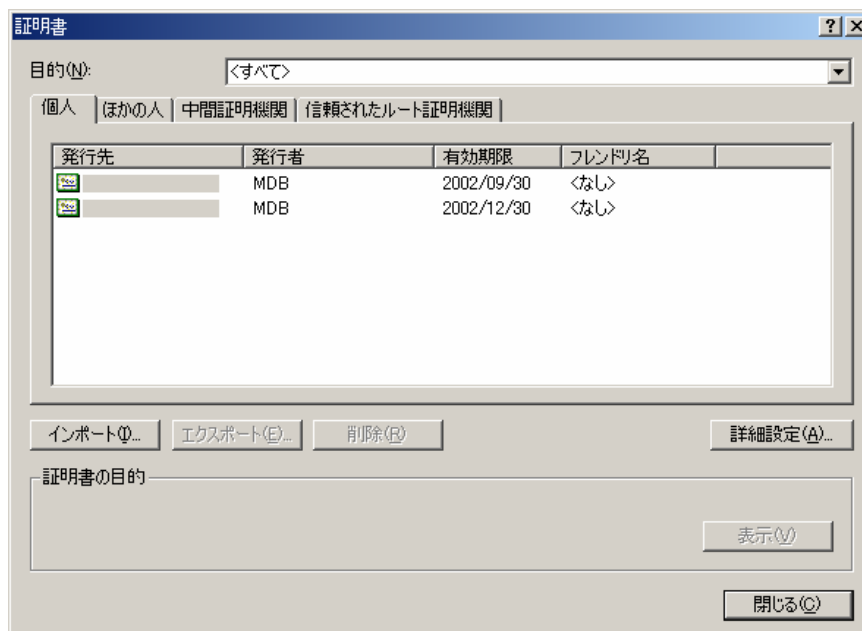


図 1 0-3 証明書画面

4. 図 1 0-3 の証明書画面で証明書の管理を行います。使用方法については、OSのドキュメントを参照してください。

第11章. OBEX の利用

「iα ppli Development Kit for DoJa-4.1(FOMA)」では、TCP/IP もしくは赤外線を用いて OBEX 通信することができます。

TCP/IP を用いて OBEX 通信を行う場合、「iα ppliTool」の実行ファイルが置かれているディレクトリ（「C:\iDKDoJa4.1\bin」に置かれる）にある libobex_tcp.dll を libobex.dll に名前を変更してください。通信先のホストの設定は、「iα ppliTool」のメニューから「設定」→「ネットワークの設定...」の選択で表示される画面の「OBEX 設定」で行います。何も設定しない場合には、127.0.0.1（ローカルホスト）に通信します。使用される TCP のポート番号は 650 番です。ポート番号を変更することはできません。

NetBeans IDE 4.1 での設定は「1 3.4.1. デバイスの設定」を参照してください。

赤外線を用いて OBEX 通信を行う場合、「iα ppliTool」の実行ファイルが置かれているディレクトリにある libobex_irda.dll を libobex.dll に名前を変更してください。赤外線を利用する場合は、ホストの設定の必要はありません。

なお、インストール時の libobex.dll は libobex_tcp.dll であり、TCP/IP での通信となります。

第12章. 再配布不可識別子

i アプリのダウンロードやメールの添付などで取得したメディアデータ（画像、動画や音楽）には、i モード携帯電話より外に出させないようにするため、再配布不可識別子を付与することができます。

エミュレータでは、GIF(89a、87a)、JPEG (EXIF、JFIF)、MFi、SMF、MP4 のメディアデータに対して再配布不可識別子のチェックをしています。

読み込んだメディアデータが再配布不可であるかどうかは、MediaResource#isRedistributable() を実行することで確認することができます。

selectEntry()、getEntry()を実行する際は再配布不可識別子のチェックは行いません。

エミュレータでは、マルチメディア領域へのデータの保存はできませんので、メディアデータに再配布不可識別子を設定することはできません。

第13章. NetBeans IDE の利用

13.1. はじめに

この章では、NetBeans IDE 4.1 と「NetBeans IDE 4.1 モジュール for iα ppli」の拡張モジュールについて説明します。

13.1.1. NetBeans IDE について

NetBeans IDE 4.1 に「NetBeans IDE 4.1 モジュール for iα ppli」をインストールする事で、NetBeans IDE 4.1 からエミュレータを直接起動することができます。「NetBeans IDE 4.1 モジュール for iα ppli」は、NetBeans IDE 4.1 で i アプリを作成するための NetBeans IDE 4.1 の拡張モジュールです。

13.2. 「NetBeans IDE 4.1 モジュール for iα ppli」のインストール

この節では、「NetBeans IDE 4.1 モジュール for iα ppli」のインストール方法について説明します。

13.2.1. インストール

「NetBeans IDE 4.1 モジュール for iα ppli」をインストールするには、まず、「2.2. インストール手順」のセットアップタイプの選択画面（図 2-6）で、「カスタム」を選択し、機能の選択画面で、「NetBeans IDE 4.1 モジュール」をチェックします（図 1 3-1）。

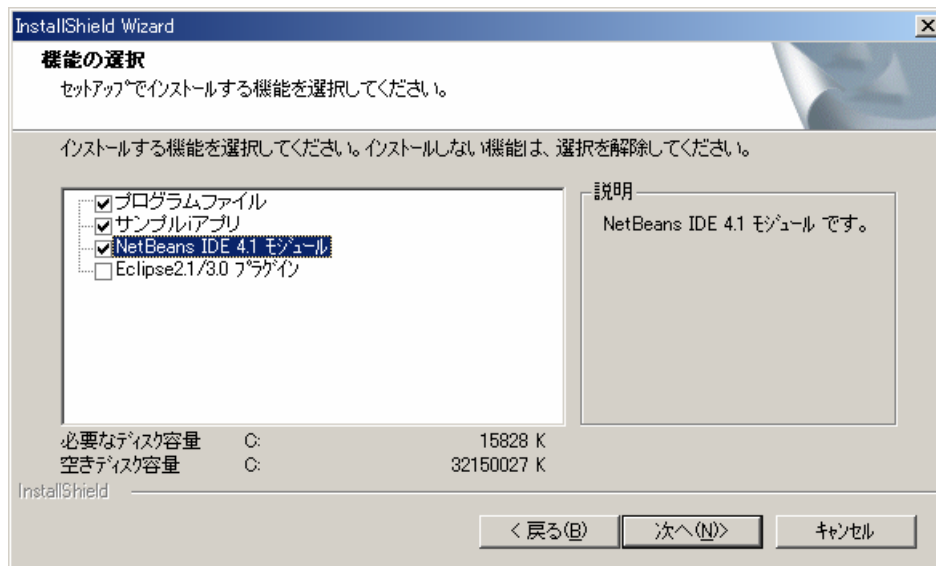


図 1 3-1 NetBeans IDE 4.1 モジュール のインストール

注意：本モジュールはデバッグ非対応です。

そのため J A R にはデバッガ情報は含まれませんので、i モード携帯電話にダウンロードする J A R を iα ppliTool でビルドしなおす必要はありません。

13.2.2. NetBeans IDE にモジュールを組み込む

「NetBeans IDE 4.1 モジュール for iα ppli」は、次の場所に置かれています。

➤ C:\jdkDoJa4.1\plugin\org.netbeans.modules.iapli.nbm

このモジュールは、NetBeans IDE の機能「アップデートセンター」で組み込む事によって使えるようになります。そのため、以下の手順を踏まないと NetBeans IDE で i アプリを作成できませんので、必ず実施してください。

NetBeans IDE に「NetBeans IDE 4.1 モジュール for iα ppli」を組み込むには、次の手順に従います。

1. NetBeans IDE を起動します。
2. 「ツール」→「アップデートセンター」を選択します（図 1 3-2）。

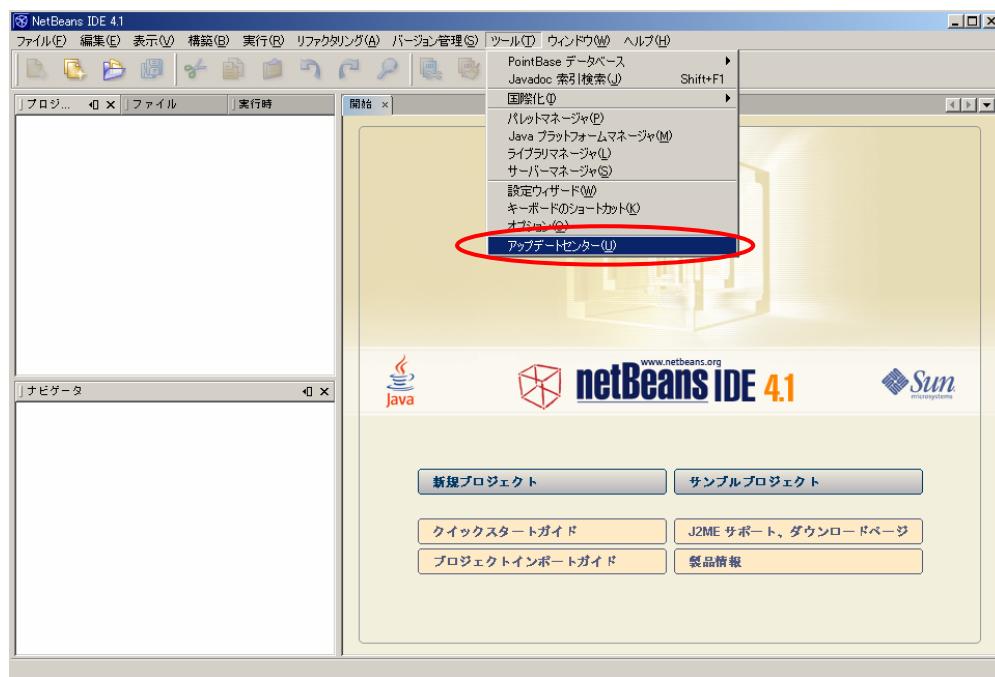


図 1 3-2 アップデートセンターの実行

3. 「モジュールの場所を選択」で、「手動でダウンロードしたモジュールをインストール」にチェックを付けて、「次へ」を押します（図 1 3-3）。

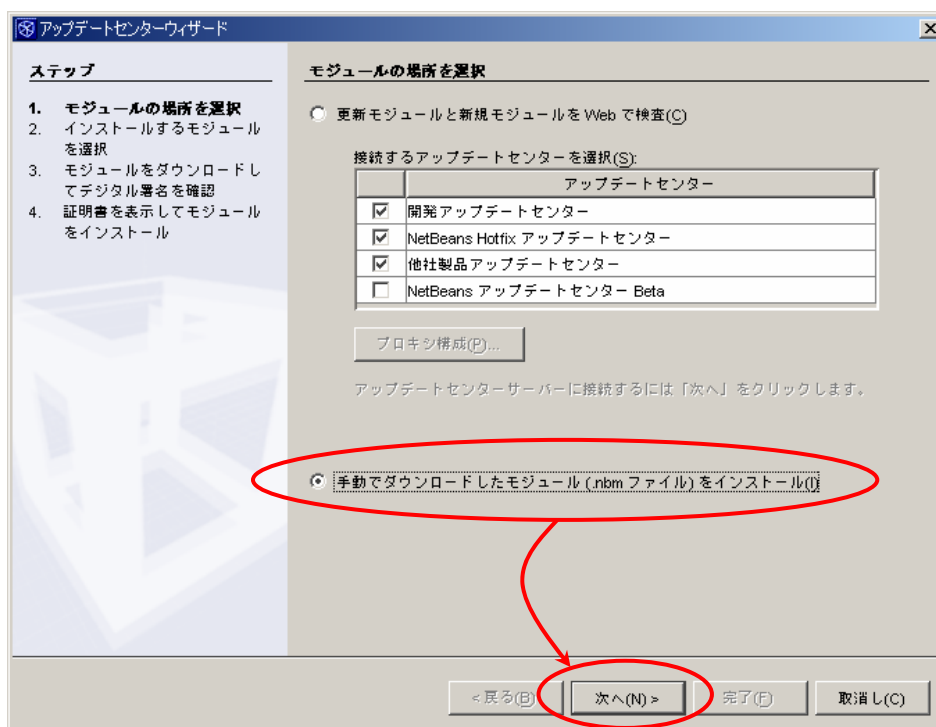


図 1 3-3 モジュールの場所を選択

4. 「インストールするモジュールを選択」で、「追加」ボタンを押して、「C:\jdkDoJa4.1\plugin\org.netbeans.modules.iappli.nbm」を追加します（図 1 3-4）。

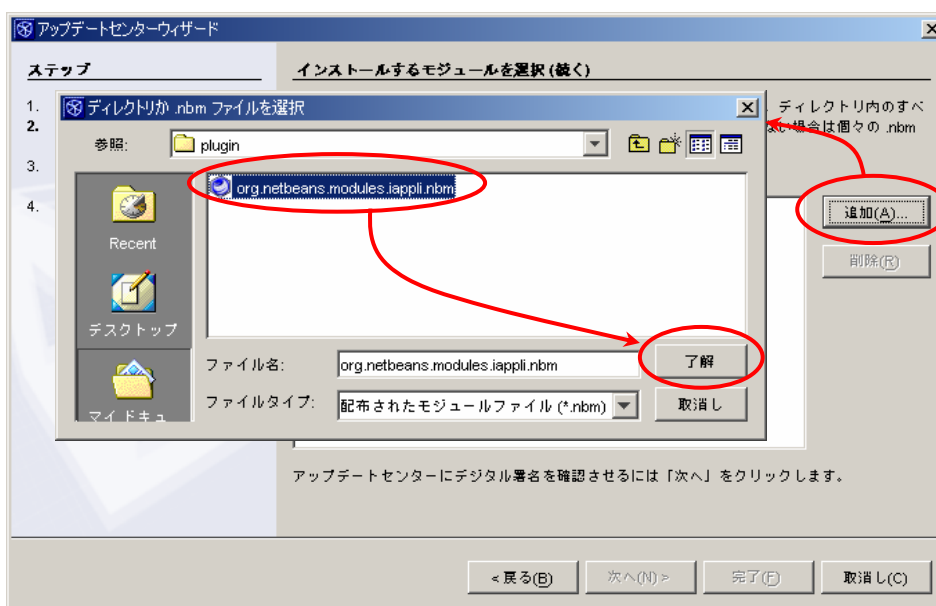


図 1 3-4 モジュールの選択

5. 「次へ」を押すと、モジュールの確認画面が表示されるので、「インストールを含む」の一覧に **DoJa-4.1** が含まれていることを確認して、「次へ」を押します（図 1 3-5）。

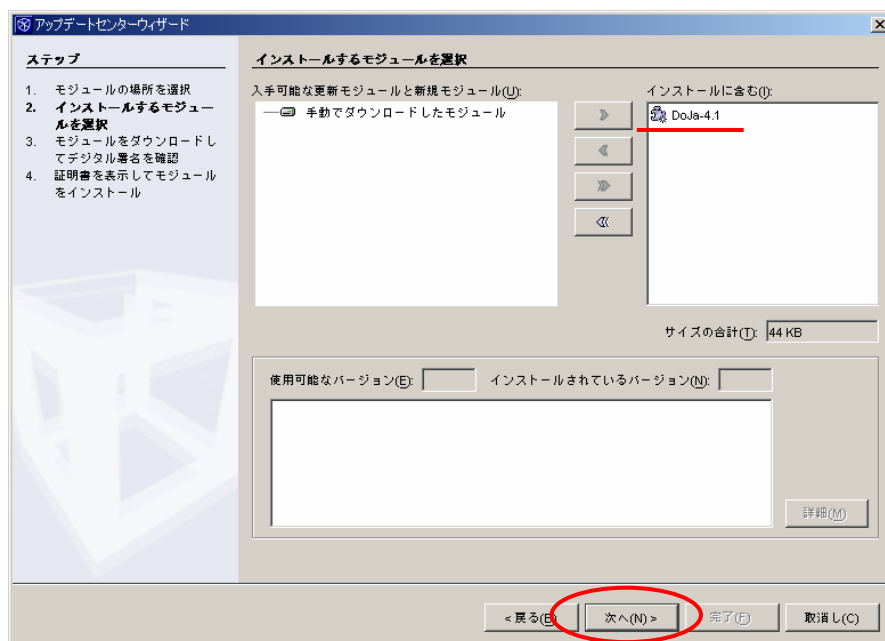


図 1 3-5 モジュールの確認

6. デジタル署名の確認が実行されるので、完了したら「次へ」を押します（図 1 3-6）。

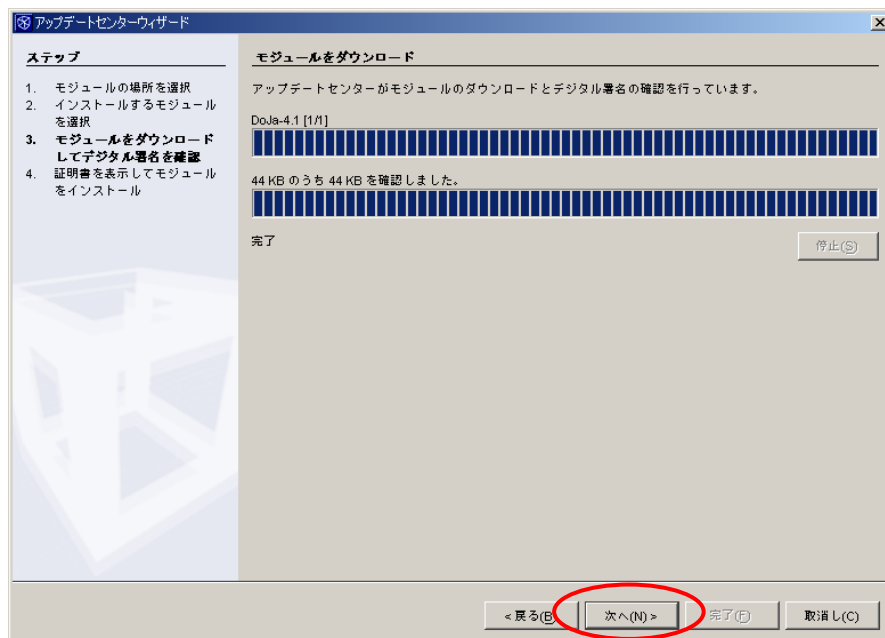


図 1 3-6 モジュールのデジタル署名

7. インストールするモジュールの一覧から「DoJa-4.1」のモジュールにチェックを付けて、「完了」を押します。
この時、署名がない旨の確認が表示されますので「はい」を押してください（図 1 3-7）。

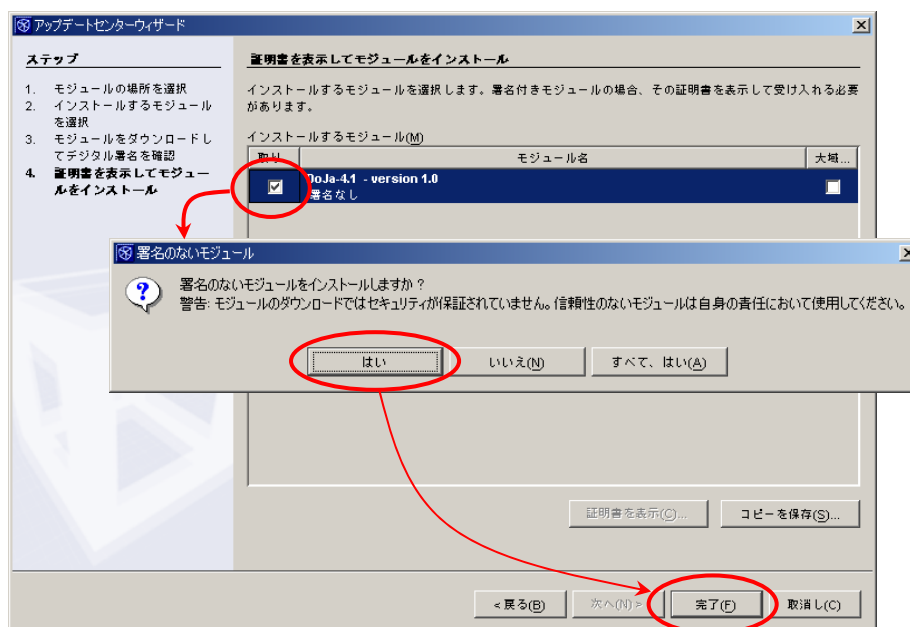
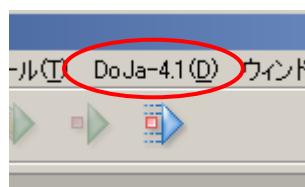


図 1 3-7 モジュールのインストール

8. NetBeans IDE に DoJa-4.1 メニューが表示されます。



9. エミュレータを「C:\iDKDoJa4.1」以外にインストールした場合は、「DoJa-4.1」→「環境設定」を選択して、「DoJa-4.1 のインストール先」を正しいパスに変更してください（図 1 3-8）。

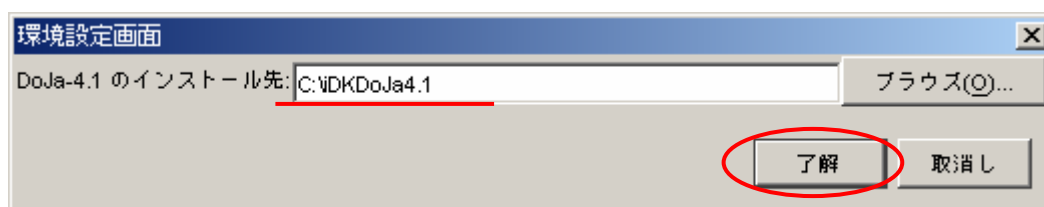


図 1 3-8 エミュレータのインストール先 設定

以上で、「NetBeans IDE 4.1 モジュール for iα ppli」の組み込みは完了です。

13.3. 新規アプリケーションの作成

この節では、新しいアプリケーションの作成方法について説明します。
主なステップは次のとおりです。

- プロジェクトの作成
- ソースファイルの作成
- アプリケーションディスクリプタファイル（ADF）の設定
- トラストド動作の設定
- アプリケーションの構築と実行

13.3.1. プロジェクトの作成

アプリケーションのソースファイルと属性ファイルを収容するパッケージを作成します。
パッケージを作成するには、次の手順に従います。

1. 「ファイル」→「新規プロジェクト」を選択します（図 1 3-9）。

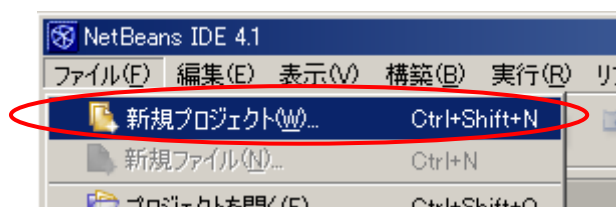


図 1 3-9 新規プロジェクトメニュー

2. 「DoJa」 - 「DoJa-4.1 プロジェクト」を選択して、「次へ」を押します（図 1 3-10）。

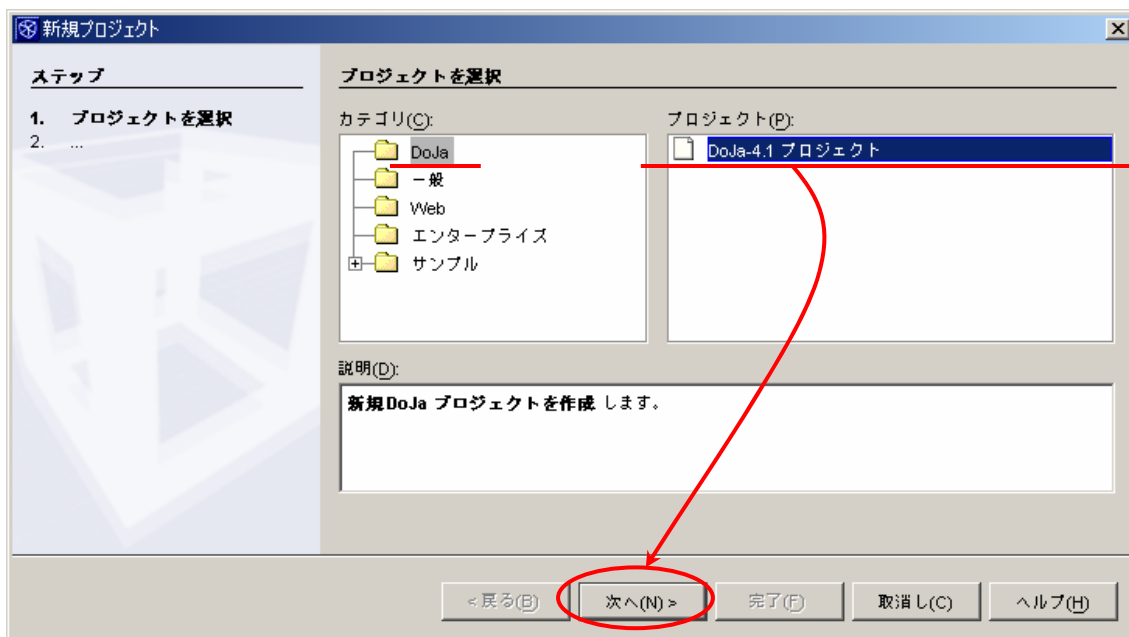


図 1 3-10 プロジェクトの種類

3. 「プロジェクト名」を入力して、「完了」を押します（図 1 3-1 1）。
例えば、プロジェクト名に「hello」と入力します。

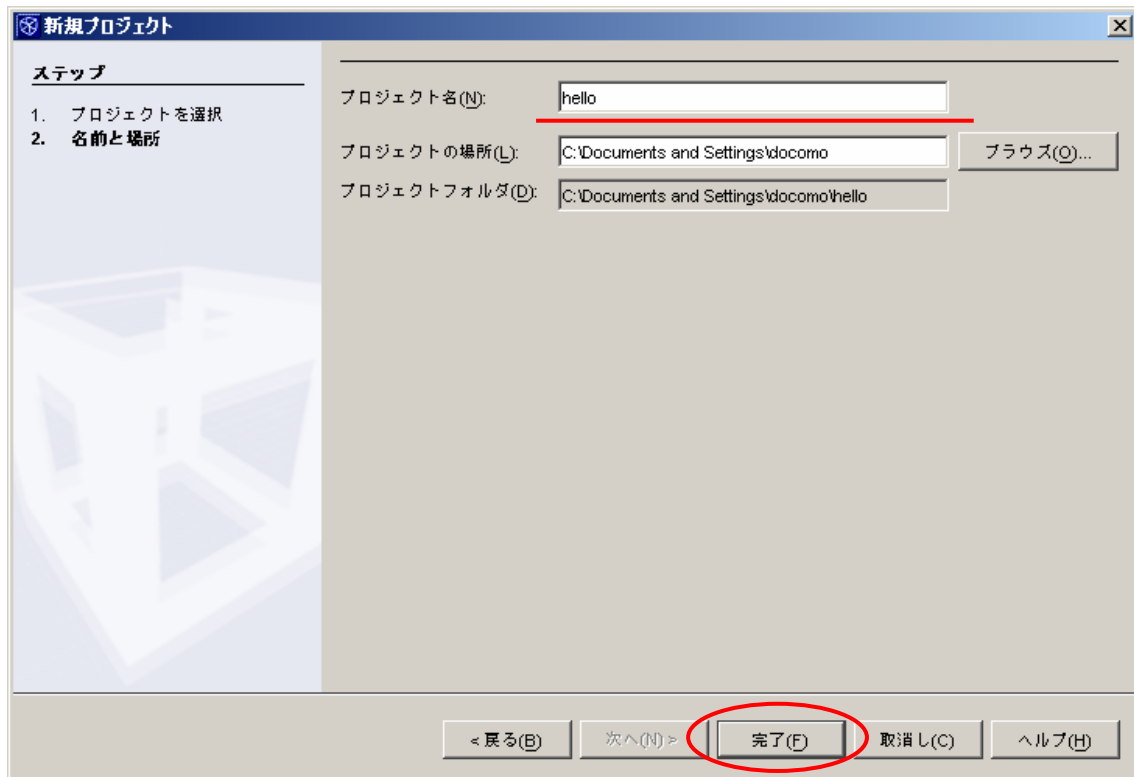


図 1 3-1 1 プロジェクト名とプロジェクトの場所

4. 作成したプロジェクトが表示されます（図 1 3-1 2）。

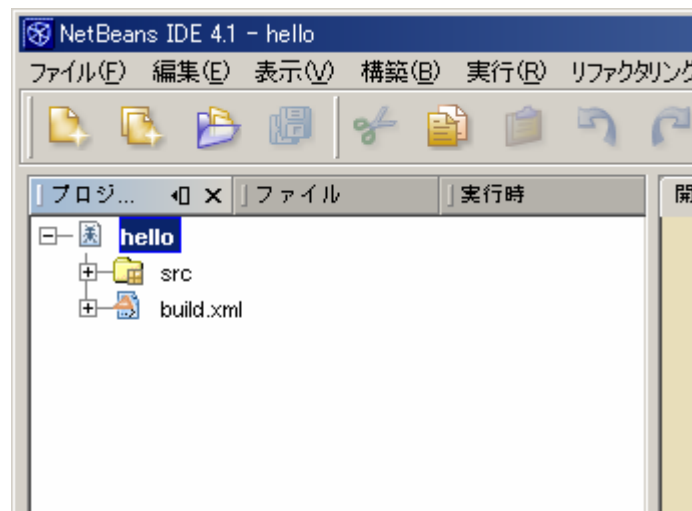


図 1 3-1 2 プロジェクトウィンドウ

13.3.2. ソースファイルの作成

アプリケーションのソースファイルを作成するには、次の手順に従います。

1. プロジェクトウィンドウの「src」を右クリックして、「新規」→「Java クラス」を選択します。

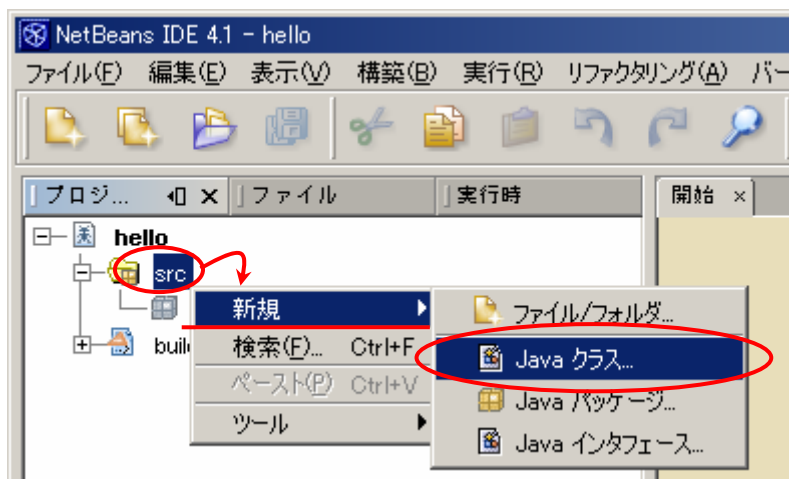


図 1 3-1 3 Java ソースファイルの作成

2. クラス名とパッケージ名を入力して、「完了」を押します。
例えば、クラス名に「HelloWorld」、パッケージ名に「hello」と入力します。

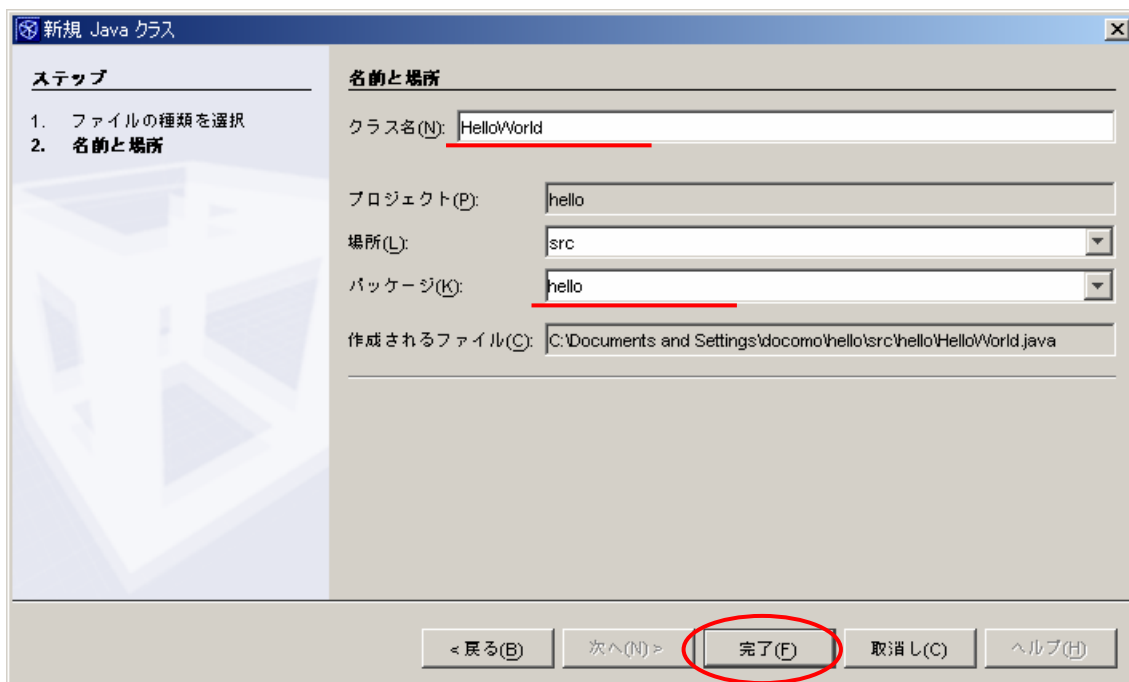


図 1 3-1 4 クラス名の設定

3. プロジェクトに空のクラスのソースファイルが追加されます。
 たとえば、ソースファイル中で、
 package hello; の後に
import com.nttdocomo.ui.IApplication;
 public class HelloWorld に続けて、
extends IApplication を追加し、
 public class HelloWorld extends IApplication { の後に
public void start() {
 System.out.println("Hello World");
}
 を追加します。
 また、コンストラクタ **public HelloWorld() { }** は不要なので削除 します。

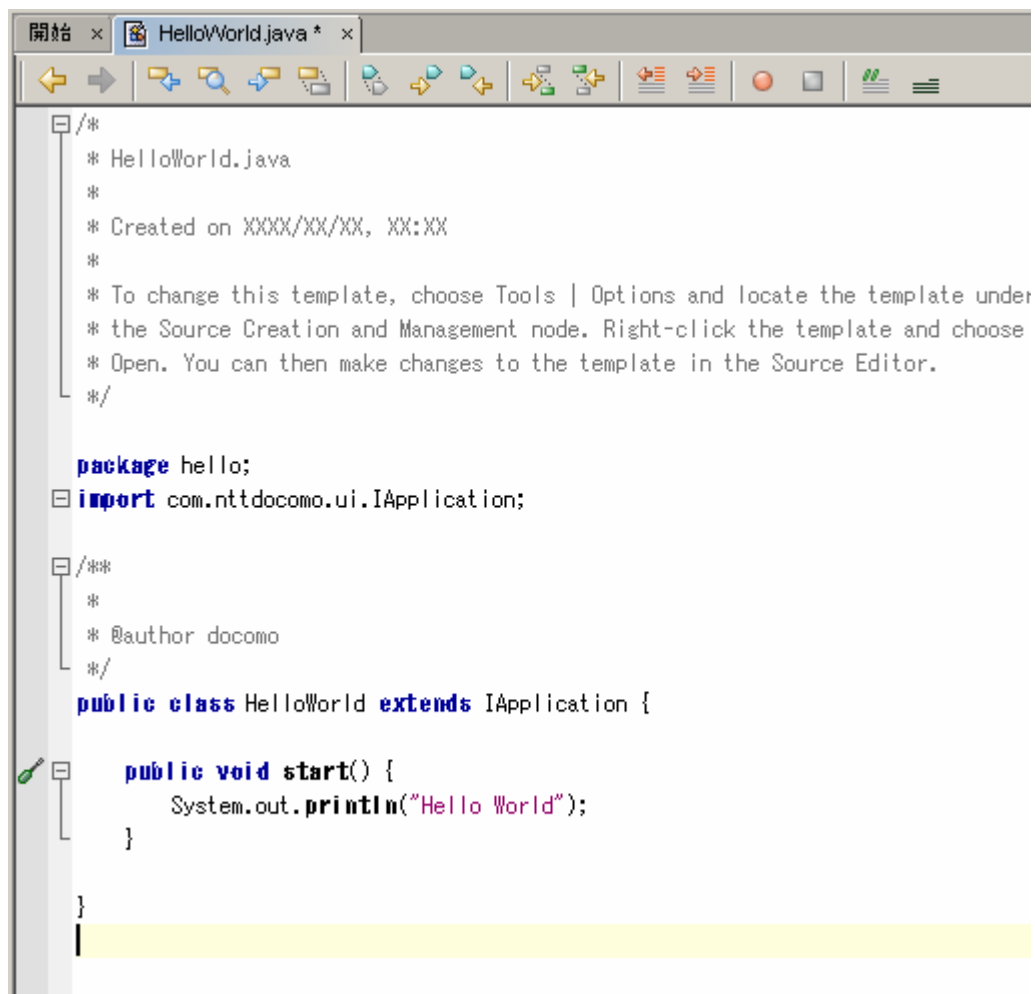


図 13-15 ソースファイルの編集

13.3.3. アプリケーションディスクリプタファイル(ADF)の設定

1. 「DoJa-4.1」 → 「ADF/トラステッド動作設定」を選択します。
2. ADF/トラステッド動作設定画面が表示されるので、「ADF設定」タブを選択して、**AppName と AppClass を設定します**。(AppSize と LastModified は自動的に設定されますが、AppName と AppClass は手動で設定する必要があります。)
3. 必要に応じて各キーの属性値を変更してください(図 13-16)。
4. 「了解」を押すと保存されます。

ADF/トラステッド動作設定画面

ADF設定 | トラステッド動作設定

ADFキー	値
AppName	HelloWorld
AppVer	
PackageURL	hello.jar
AppSize	
ConfigurationVer	
ProfileVer	
Sysize	
AppClass	hello.HelloWorld
AppParam	
LastModified	
UseNetwork	
TargerDevice	
LaunchAt	
MyConcierge	
UseTelephon	
UseBrowser	
LaunchByMail	
LaunchByBrowser	
AllowPushBy	
AppTrace	
DrawArea	
GetSysInfo	
TrustedAPID	
TrustedLMD	
PackageHashCode	

了解 取消し

AppName と AppClass は、必ず手動で設定してください。

図 13-16 ADF設定画面

各ADFの設定項目については、「4.7. アプリケーションディスクリプタファイル(ADF)の編集 表 4-3 アプリケーションディスクリプタファイル(ADF)の設定項目」を参照してください。

13.3.4. トラステッド動作の設定

トラステッド動作の設定をするには、次の手順に従います。

1. 「DoJa-4.1」 → 「ADF/トラステッド動作設定」を選択します。
2. ADF/トラステッド動作設定画面が表示されるので、「トラステッド動作設定」タブを選択して、必要に応じて各キーの属性値を変更してください（図 1 3-1 7）。
3. 「了解」を押すと保存されます。

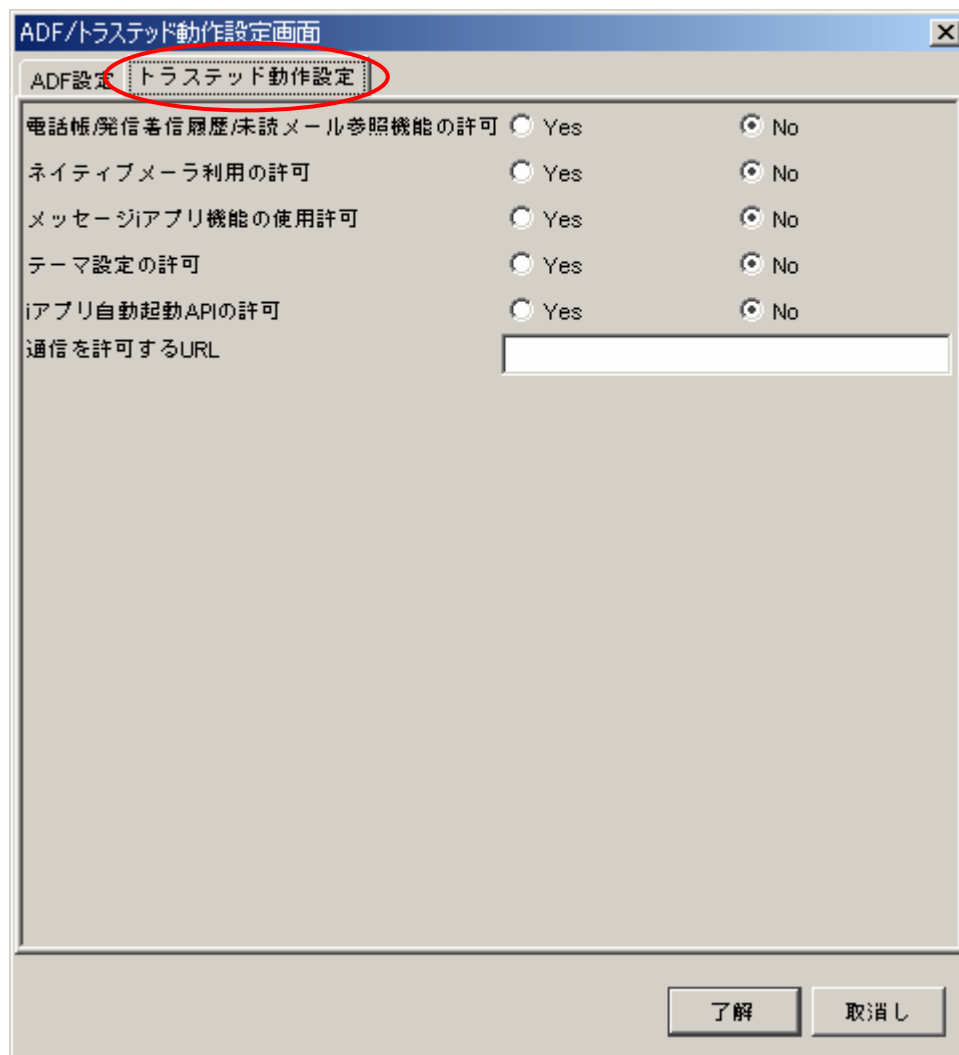


図 1 3-1 7 トラステッド動作設定画面

各トラステッド動作設定の設定項目については、「5.2 トラステッド動作の設定」を参照してください。

13.3.5. アプリケーションの構築と実行

アプリケーションを構築および実行するには、次の手順に従います。

1. プロジェクトウィンドウで、プロジェクトを選択します。

右クリックメニューから「プロジェクトを構築」を選択します（

2. 図 13-18）。
3. プロジェクトがビルドされ、ビルドの経過が出力ウィンドウに表示されます。

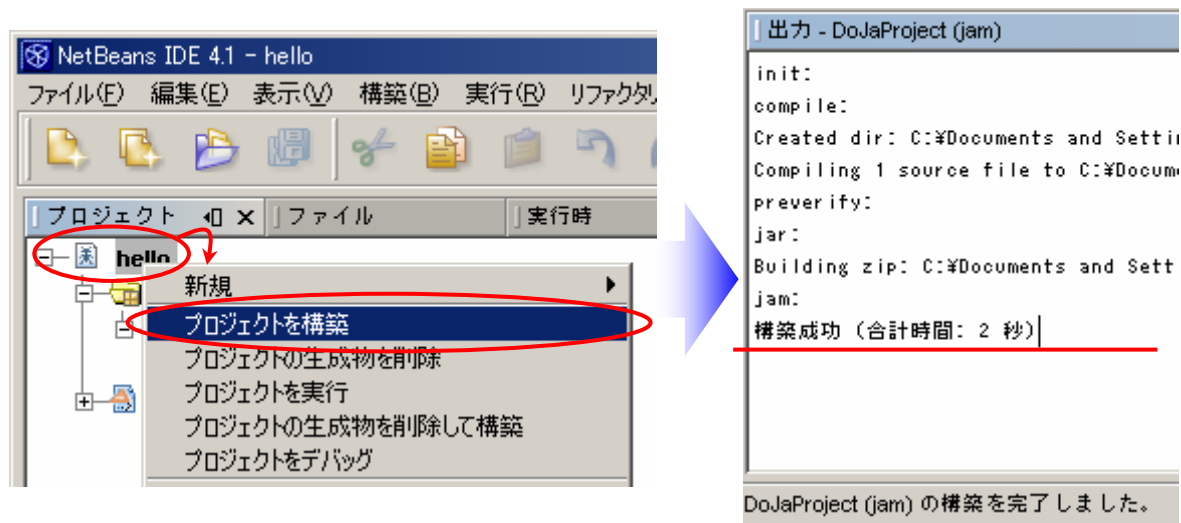


図 13-18 プロジェクト構築

4. プロジェクトのビルドに成功したら、右クリックメニューから「プロジェクトを実行」を選択します（図 13-19）。

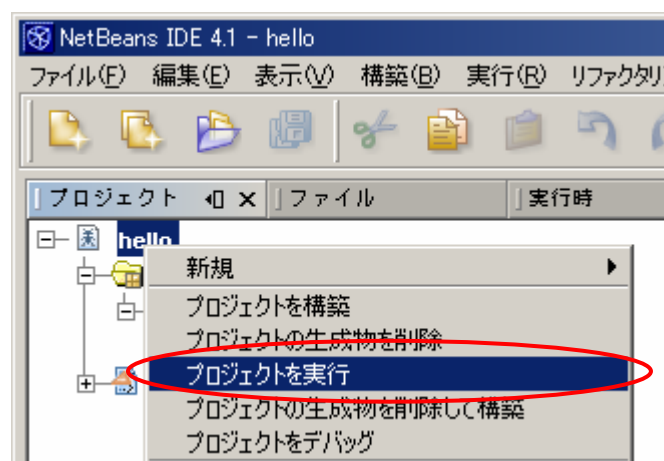


図 13-19 プロジェクト実行

5. 端末ウィンドウが表示されて、i アプリが実行されます（図 1 3-2 0）。



図 1 3-2 0 i アプリの実行

13.4. エミュレータの設定

13.4.1. デバイスの設定

エミュレータのデバイスを変更するには、次の手順に従ってください。

1. 「DoJa-4.1」 → 「起動設定」を選択すると、起動設定画面が表示されます（図 1 3-2 1）。
2. 「Device type」でエミュレーション端末の種類を選択します。
3. 「Launch type」で起動方法を選択します。
4. 「了解」を押します。



図 1 3-2 1 起動設定画面

13.4.2. ネットワークの設定

ネットワークを設定するには、次の手順に従ってください。

1. 「DoJa-4.1」 → 「ネットワーク設定」を選択すると、ネットワーク設定画面が表示されます（図 1 3-2 2）。
2. ネットワークの設定を変更して「了解」を押します。

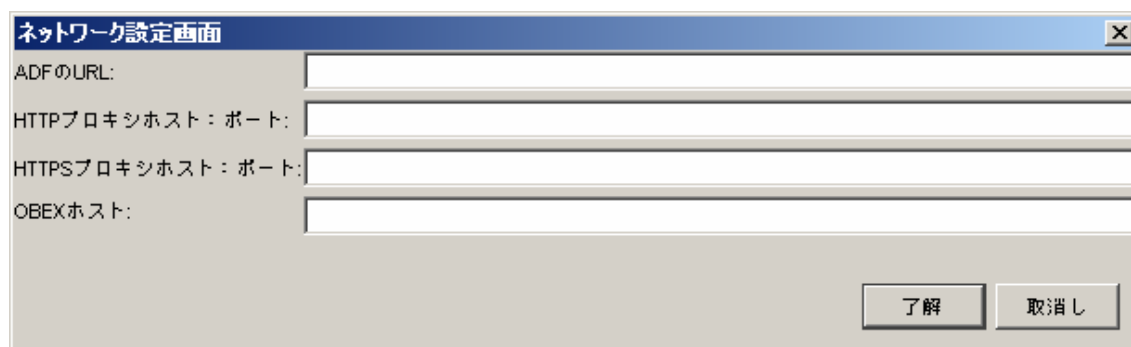


図 1 3-2 2 ネットワーク設定

ネットワークの各設定項目の詳細については、「5.1. ネットワークの設定」を参照してください。

13.4.3. NetBeans IDE からモジュールを削除する

NetBeans IDE から「NetBeans IDE 4.1 モジュール for iα ppli」を削除するには、次の手順に従います。

1. NetBeans IDE を起動します。
2. 「ツール」→「オプション」を選択します（図 1 3-2 3）。
3. 「IDE 構成」→「システム」→「モジュール」→「DoJa-4.1」を選択します。
4. 右クリックメニューから「削除」を選択します。

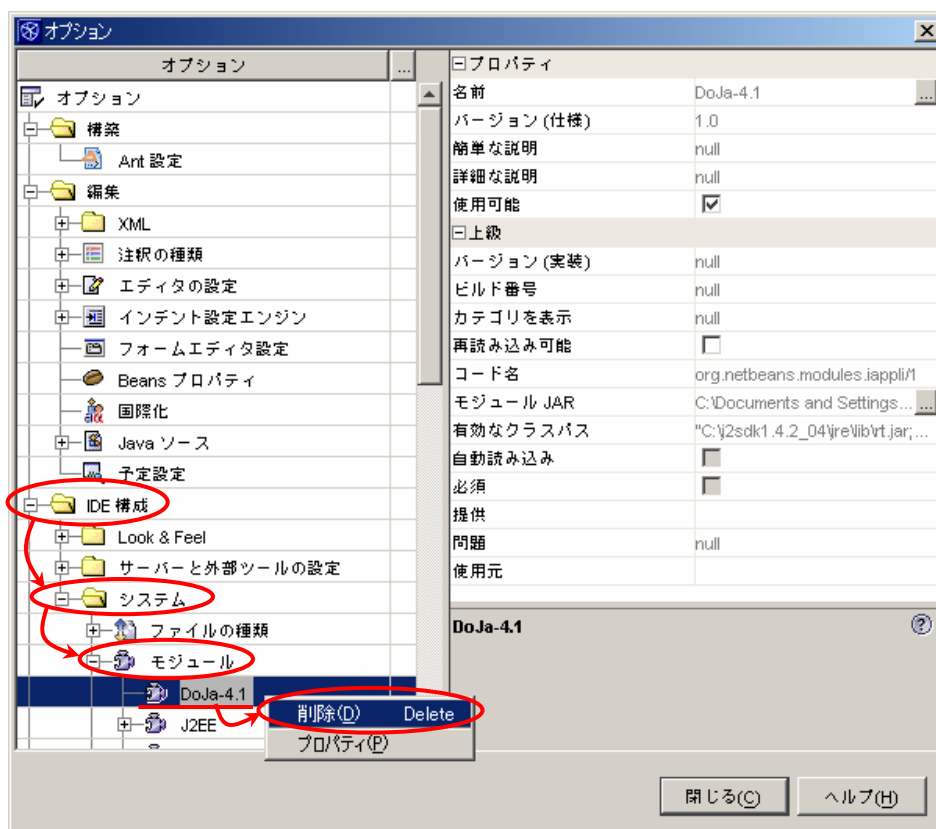


図 1 3-2 3 モジュールの削除

5. 削除の確認ダイアログが表示されるので、「はい」を押します（図 1 3-2 4）。
6. 「NetBeans IDE 4.1 モジュール for iα ppli」が削除され、「DoJa-4.1」メニューが消えます。

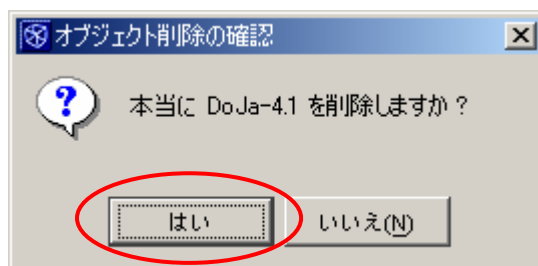


図 1 3-2 4 モジュールの削除確認

第14章. Eclipse の利用

14.1. はじめに

この章では、Eclipse と「Eclipse プラグイン for iα ppli」の拡張モジュールについて説明します。

14.1.1. Eclipse について

Eclipse に「Eclipse プラグイン for iα ppli」をインストールする事で、Eclipse からエミュレータを直接起動することができます。「Eclipse プラグイン for iα ppli」は、Eclipse で i アプリを作成するための Eclipse のプラグインです。

「Eclipse プラグイン for iα ppli」は、Eclipse2.1 および Eclipse3.0 で開発しています。そのため「Eclipse プラグイン for iα ppli」は、Eclipse2.1 か Eclipse3.0 で使われる事を推奨します。**(※ 推奨範囲ではありませんが、Eclipse3.1 でも動作可能です。)**

Eclipse は「Language Pack」をダウンロードして、Eclipse のインストール（解凍）先に上書きする事で日本語に対応する事ができます。

Eclipse のダウンロードや、Eclipse の使用とインストールに関する詳細なドキュメントは、Eclipse の Web サイト <http://www.eclipse.org/> を参照してください。

注意：Eclipse のインストール先には、「スペース（空白）」及び「2 バイト文字（全角文字）」は含めないようにしてください。

また、古いプラグインや他のプロファイルのプラグインが入っている場合は上書きされます。もし、DoJa-3.5 や DoJa-4.0 と共存させたい場合は、別の Eclipse 環境にインストールしてください。

14.2. 「Eclipse プラグイン for iα ppli」のインストール

本節では、Eclipse 用の 拡張モジュールをインストールする手順を説明します。

14.2.1. 始める前に

Eclipse で「Eclipse プラグイン for iα ppli」を使用するには、Java 2 Platform, Standard Edition SDK (J2SE SDK) と Eclipse をインストールしてから、「Eclipse プラグイン for iα ppli」をインストールする必要があります。

- J2SE SDK は、次の Web サイトからダウンロードできます。
<http://java.sun.com/j2se/>
 - J2SE SDK のインストール後は、**環境変数の PATH に J2SE SDK の bin を追加してください** (コントロールパネル → システム → 詳細 → 「環境変数」)。
これは、iα ppliTool では自動的に検出しています。
- Eclipse と、その Language Pack は、次の Web サイトからダウンロードできます。
<http://www.eclipse.org/>

14.2.2. インストール

「Eclipseプラグイン for iα ppli」をインストールするには、2.2. インストール手順のセットアップタイプの選択画面（図 2-6）で、「カスタム」を選択し、機能の選択画面で、「Eclipse2.1/3.0 プラグイン」をチェックします（図 1 4-1）。

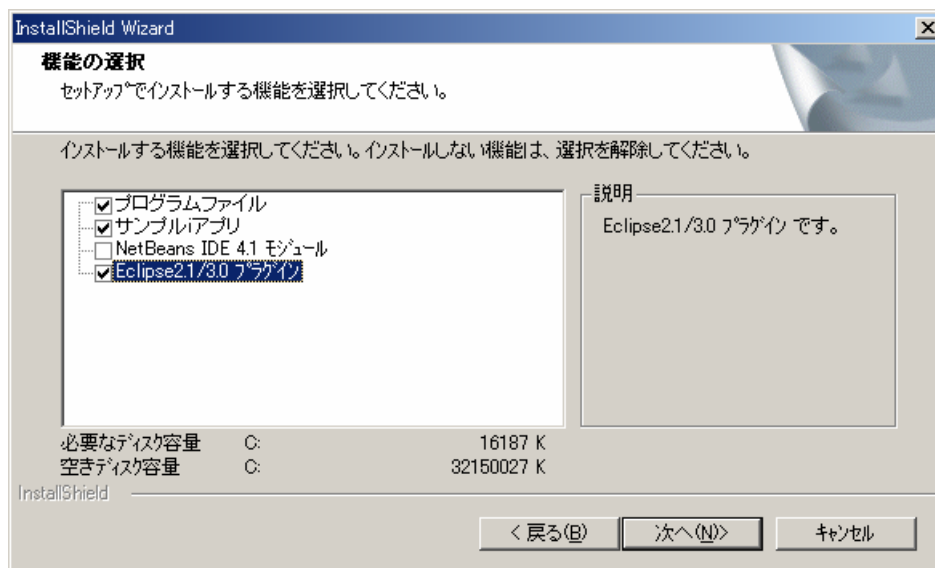


図 1 4-1 Eclipse プラグインのインストール

Eclipse は、インストーラが無いため、どこにあるかを指定する必要があります。

「Eclipse2.1/3.0 プラグイン」をチェックして、「次へ」を押すと、Eclipse のインストール先ディレクトリを指定する画面が表示されます。

注意：Eclipseで作成されたJARにはデバッガ情報が含まれています。そのため、iモード携帯電話にダウンロードするJARは、必ずエミュレータの「iα ppliTool」を使ってビルドしてください。（「iα ppliTool」については、1.1.1. iα ppliTool を参照してください。）

14.3. 新規アプリケーションの作成

この節では、新しいアプリケーションの作成方法について説明します。
主なステップは次のとおりです。

- DoJaを利用する前に
- プロジェクトメニューのカスタマイズ
- プロジェクトの作成
- ソースファイルの作成
- アプリケーションディスクリプタファイル（ADF）設定
- トラストッド動作の設定
- アプリケーションの構築と実行
- プロジェクトの設定
- リモート制御の設定
- ファイル配置カスタマイズ
- i アプリライブラリの作成
- i アプリライブラリの利用
- デバッグの実行

14.3.1. DoJa を利用する前に

Eclipse で DoJa を利用できるようにするためには、Eclipse に「iα ppli Development Kit for DoJa-4.1」のインストール先を設定しておく必要があります。

「Eclipse プラグイン for iα ppli」を組み込んだ状態で、最初に設定されている「iα ppli Development Kit for DoJa-4.1」のインストール先は、「C:\iDKDoJa4.1」になっています。Eclipse に「iα ppli Development Kit for DoJa-4.1」のインストール先を設定するには、次の手順に従います。

1. 「ウィンドウ」→「設定」を選択します。
2. 設定ウィンドウが開くので、設定ウィンドウから「ワークベンチ」→「DoJa-4.1 Environment」を選択すると、「iα ppli Development Kit for DoJa-4.1」のインストール先を設定する画面になります。

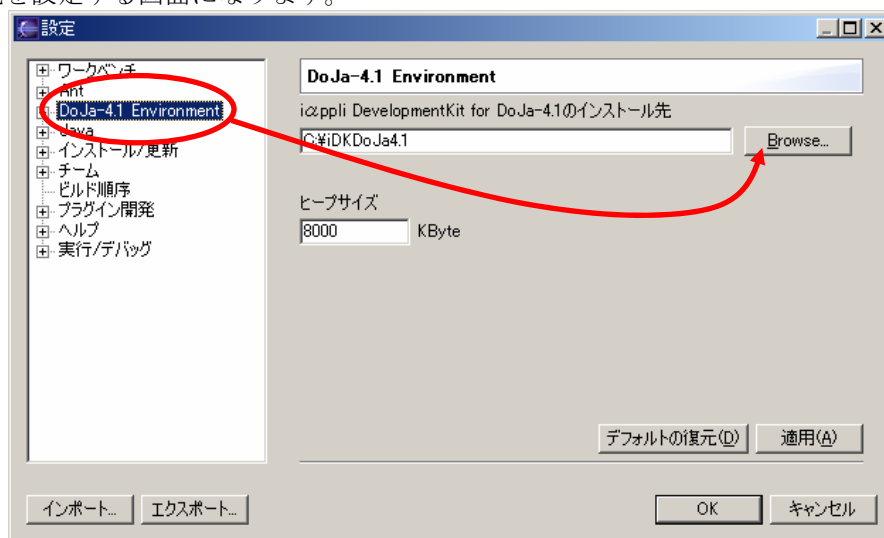


図 1 4-2 Eclipse 設定ウィンドウ

3. 「Browse(B)...」を押して「iα ppli Development Kit for DoJa-4.1」のインストール先を設定し、「OK」を押します。

14.3.2. プロジェクトメニューのカスタマイズ

Eclipse で DoJa の設定を行えるようにするには、次の手順が必要になります。

新たにプラグインをインストールしなおした場合にも必要です。

1. 「ウィンドウ」→「パースペクティブのカスタマイズ」を選択します。
2. Eclipse2.1 系の場合：
「その他」→「DoJa-4.1 設定」にチェックを付けます。
Eclipse3.0 系の場合：
「コマンド」タブ→「DoJa-4.1 設定」にチェックを付けます。
一度チェックをすると、次回の Eclipse 起動時にはチェックされた状態になります。

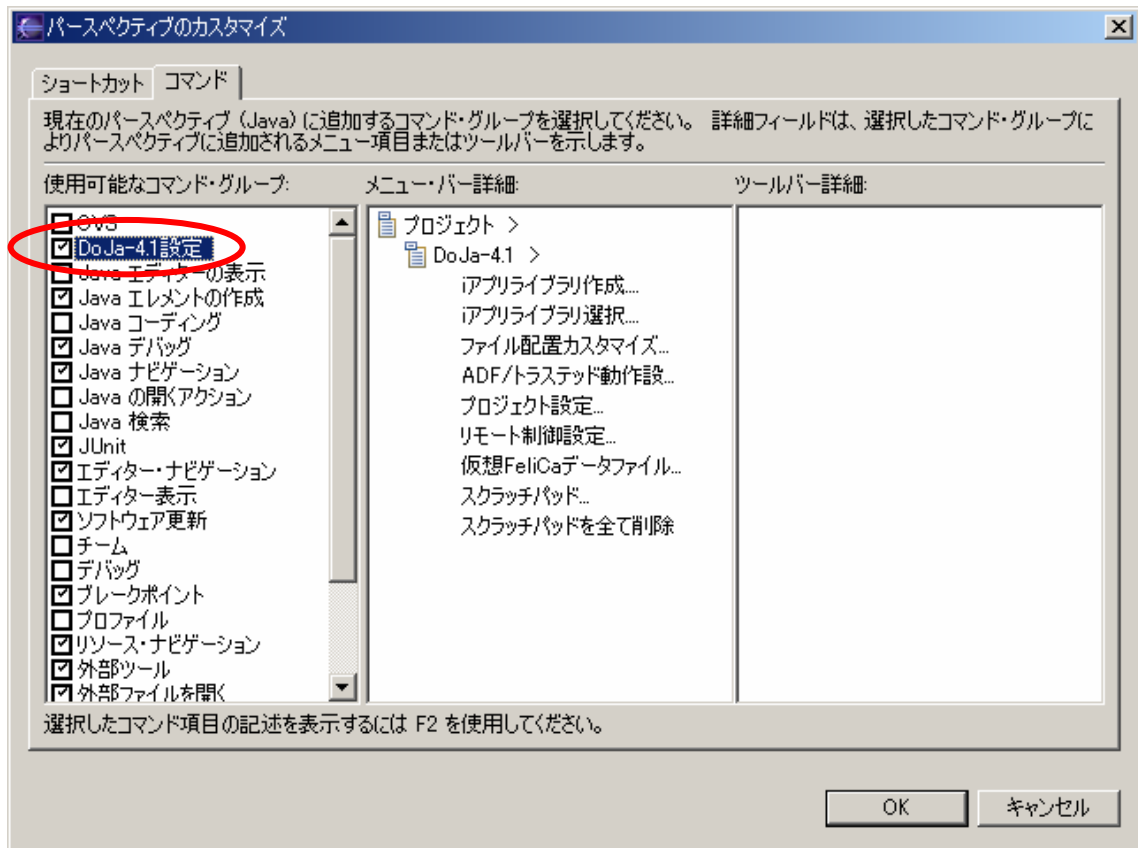


図 14-3 プロジェクトメニューのカスタマイズ

14.3.3. プロジェクトの作成

アプリケーションのソースファイルと属性ファイルを収容するプロジェクトを作成します。プロジェクトを作成するには、次の手順に従います。

1. 「ファイル」→「新規」→「プロジェクト」を選択します。
2. 新規プロジェクトウィンドウが表示されるので、「Java」の「DoJa-4.1 プロジェクト」を選択し、「次へ」を押します。

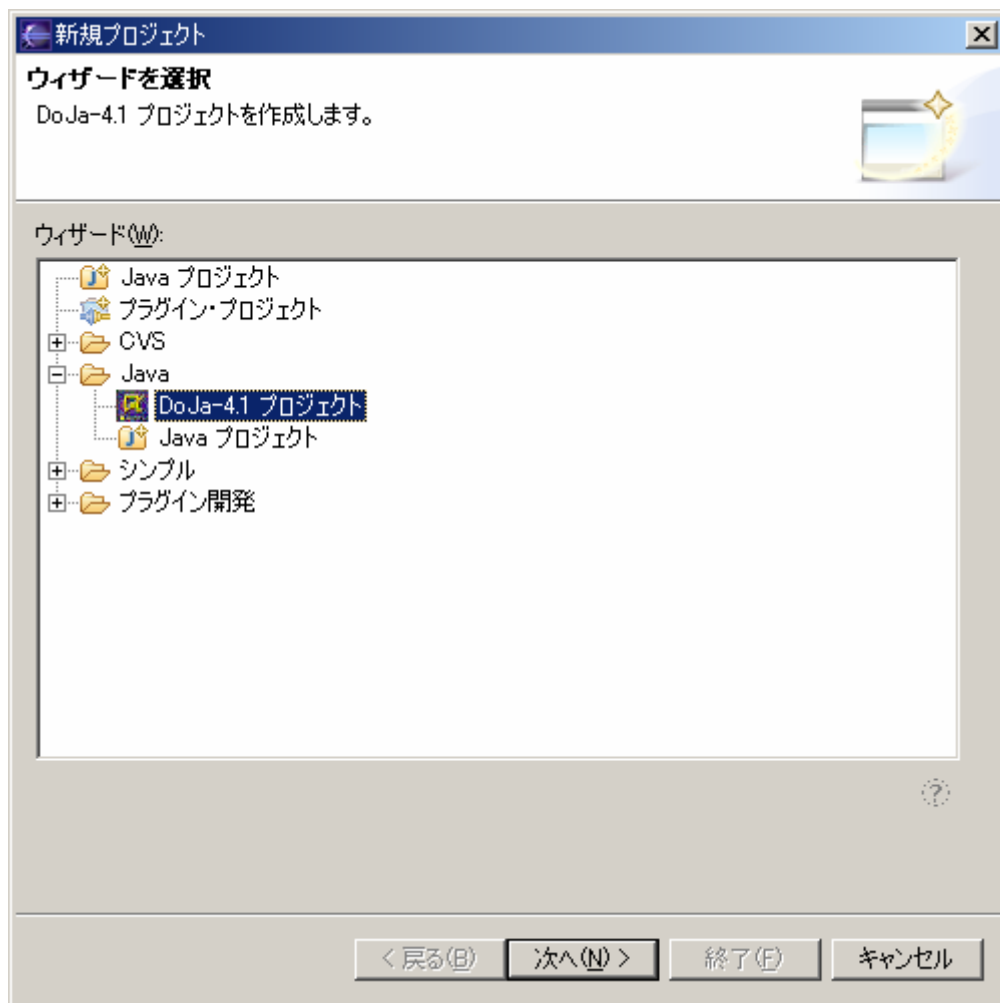


図 1 4-4 新規プロジェクトウィンドウ

3. プロジェクト名設定ウィンドウが表示されるので、プロジェクト名を入力して「次へ」を押します。
たとえば、hello と入力します。

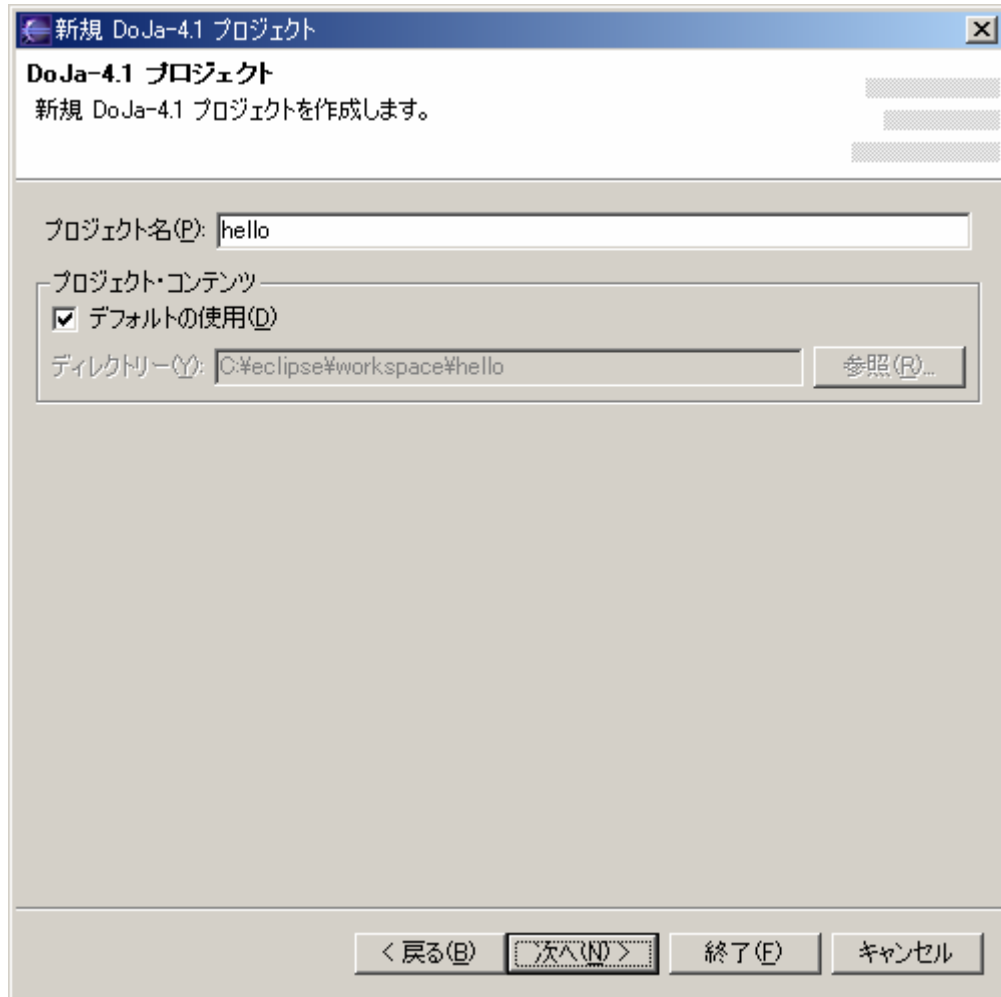


図 1 4-5 プロジェクト名設定ウィンドウ

- 「プロジェクトの内容」の「デフォルトの使用」のチェックを外す事で、既に「iα ppli Development Kit for DoJa-4.1」で作成されたプロジェクトを「ディレクトリ」で指定する事もできます。

4. Java 設定ウィンドウが表示されるので、もしビルド環境を変更したい場合は、設定内容を変更して「終了」を押します。
(ここでは、何も変更せずに「終了」を押します。)

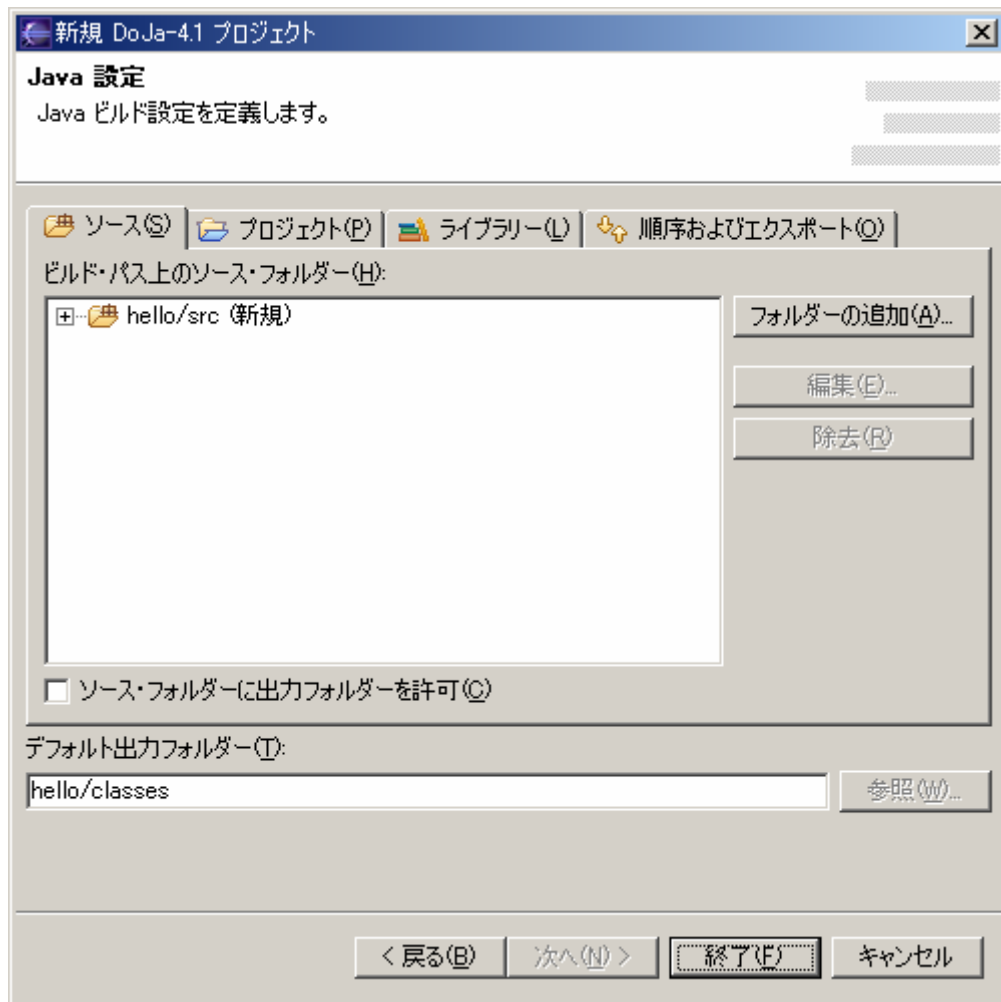


図 1 4-6 Java 設定ウィンドウ

5. ナビゲーターウィンドウにプロジェクト名 (hello) が追加されます。

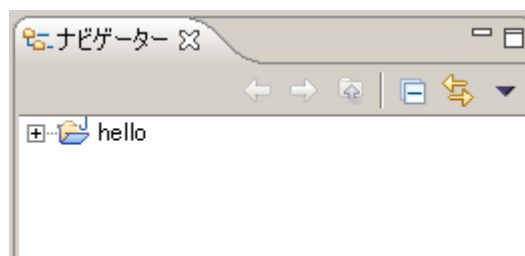


図 1 4-7 ナビゲーターウィンドウ

14.3.4. ソースファイルの作成

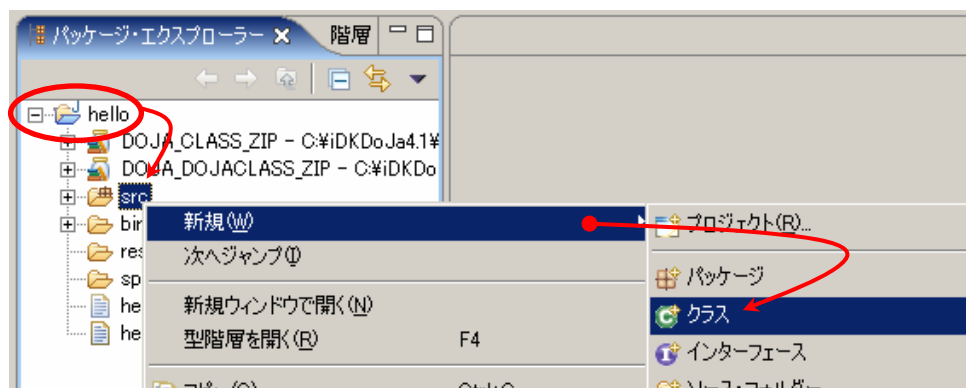
アプリケーションのソースファイルを作成するには、次の手順に従います。

1. パースペクティブの選択から「Java」パースペクティブを選択します。



図 1 4 - 8 Java パースペクティブの選択

2. パッケージ・エクスプローラーウィンドウに切り替わるので、プロジェクト(hello)→「src」をクリックし、マウスの右ボタンから「新規」→「クラス」を選択します(図 1 4 - 9)。



3. 新規クラス設定ウィンドウが表示されるので、パッケージ名とクラス名を入力し、「終了」を押します。
たとえば パッケージに `hello`、名前に `HelloWorld` と入力します。

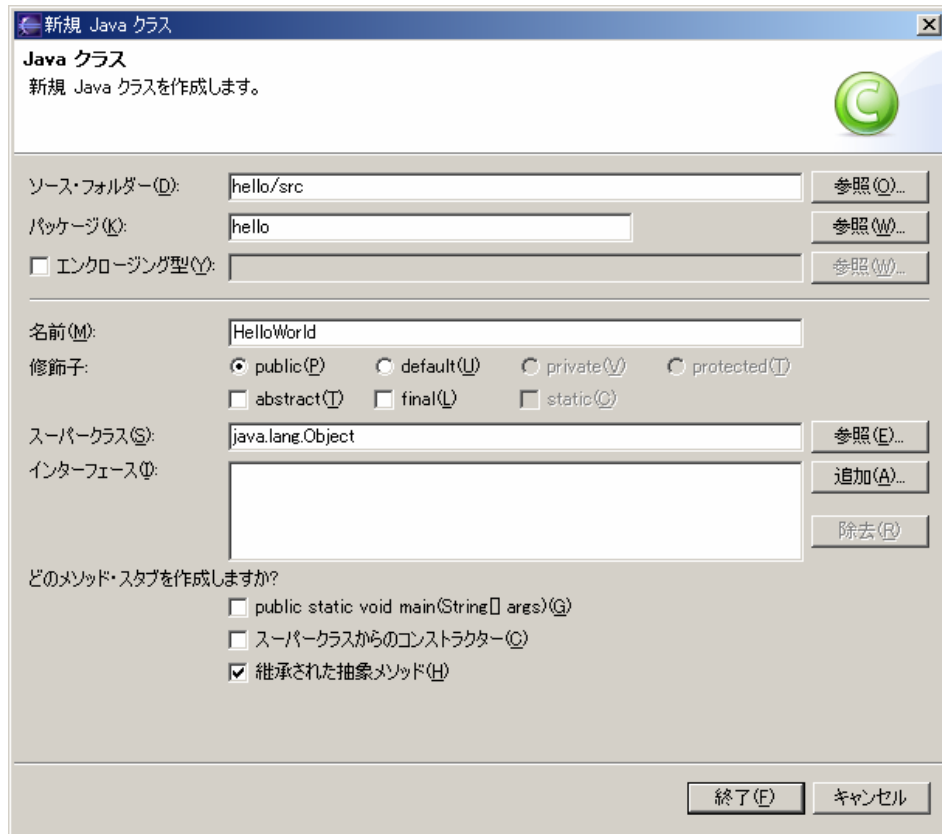


図 1 4 - 1 0 新規クラス設定ウィンドウ

4. ソースファイルを編集します。
 ソースファイルは、1～3の手順以外に、マウスのドラッグ&ドロップで既存のソースを挿入する事もできます。
 たとえば、ソースファイル中で
 package hello; の後に
import com.nttdocomo.ui.IApplication;
 public class HelloWorld に続けて、
extends IApplication を追加し、
 public class HelloWorld extends IApplication { の後に
public void start() {
System.out.println("Hello World");
}
 を追加します（図 1 4-1 1）。

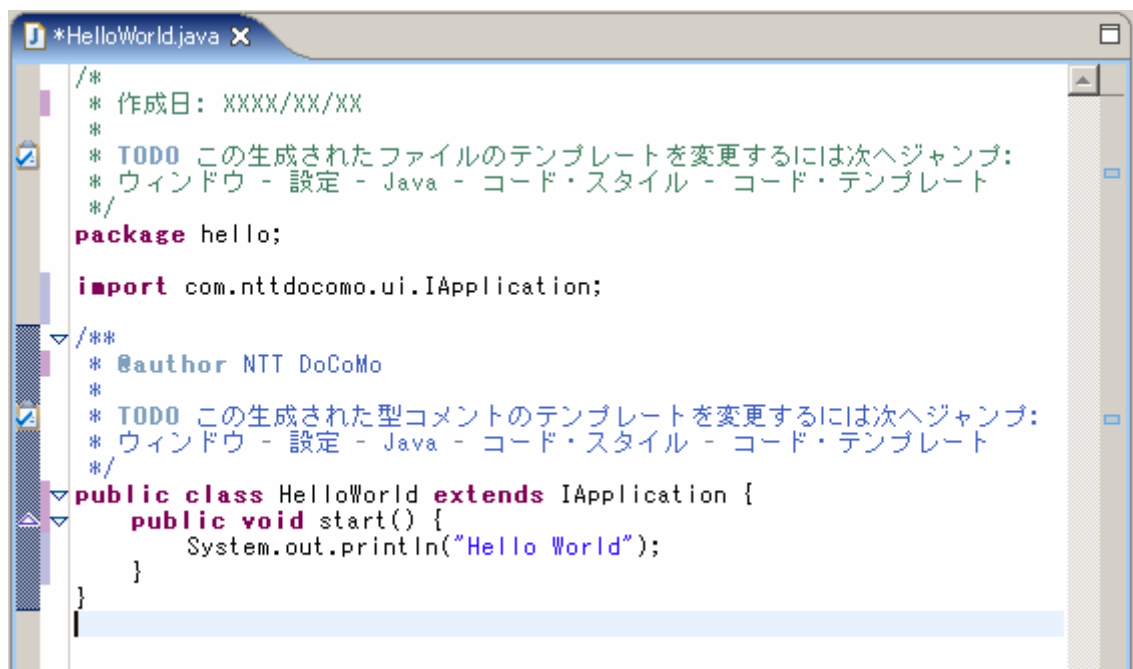


図 1 4-1 1 ソースファイルの編集

5. ソースファイルを保存します。
 「ファイル」→「すべて保管」の順でメニュー項目を選択するか、**Ctrl-S** を押して、ソースファイルを保存します。
 この時点で、ソースファイルは自動的にコンパイルされます。
 もし、ソースファイルの横に  や  が表示されたら、その行かその行に関連する処理に誤りがあります。マークの上にカーソルを置くとその理由が表示されますのでソースを修正してください。

14.3.5. プロジェクト設定からのソースファイルの作成

アプリケーションのソースファイルを作成するには、次の手順から行えます。

1. パッケージ・エクスプローラーウィンドウから**プロジェクト (hello)** をクリックして、「プロジェクト」→「DoJa-4.1」→「ソースファイルの新規作成」をクリックします。
(※ **プロジェクトを選んでいないと選択できません。**)
2. プロジェクト設定ウィンドウが表示されるので、「ソースファイルの新規作成」タブから、必要に応じて各キーの属性値を変更してください。

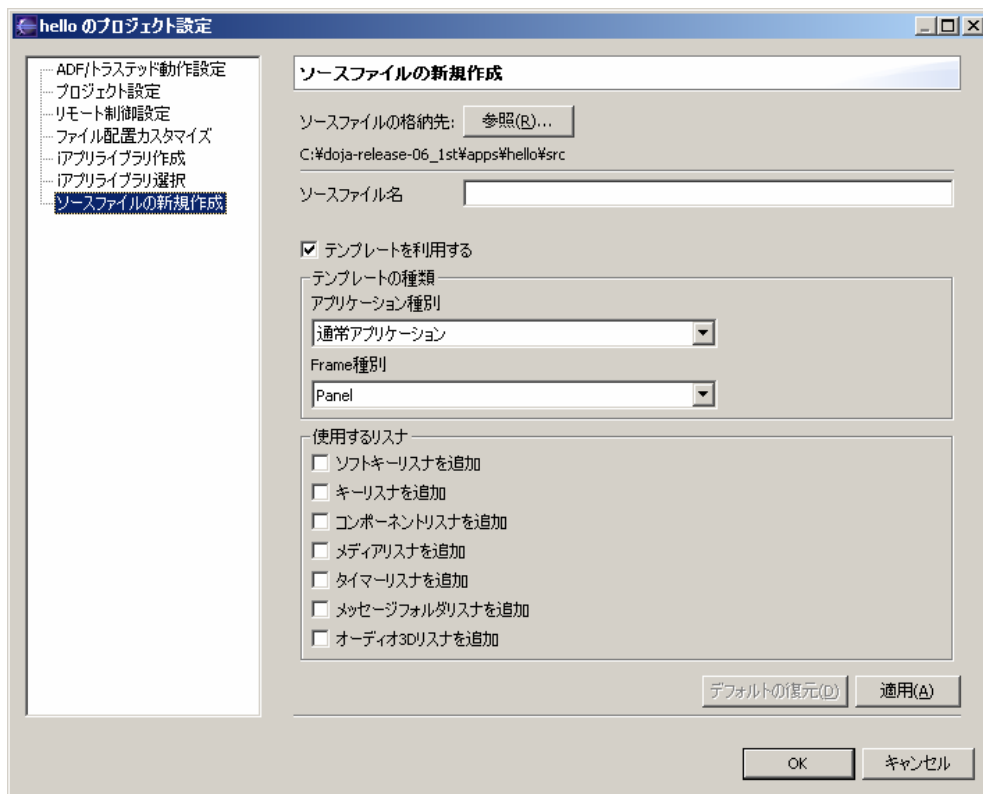


図 1 4 - 1 2 ソースファイルの新規作成画面

各ソースファイルの新規作成の設定項目については、「4.2.1 ソースファイルの作成」を参照してください。

14.3.6. アプリケーションディスクリプタファイル(ADF)設定

ADFの設定をするには、次の手順に従います。

1. パッケージ・エクスプローラーウィンドウから**プロジェクト (hello)** をクリックして、「プロジェクト」→「DoJa-4.1」→「ADF/トラステッド動作設定」をクリックします。
(※ **プロジェクトを選んでいないと選択できません。**)
2. プロジェクト設定ウィンドウが表示されるので、「ADF 設定」タブから、必要に応じて各キーの属性値を変更してください。

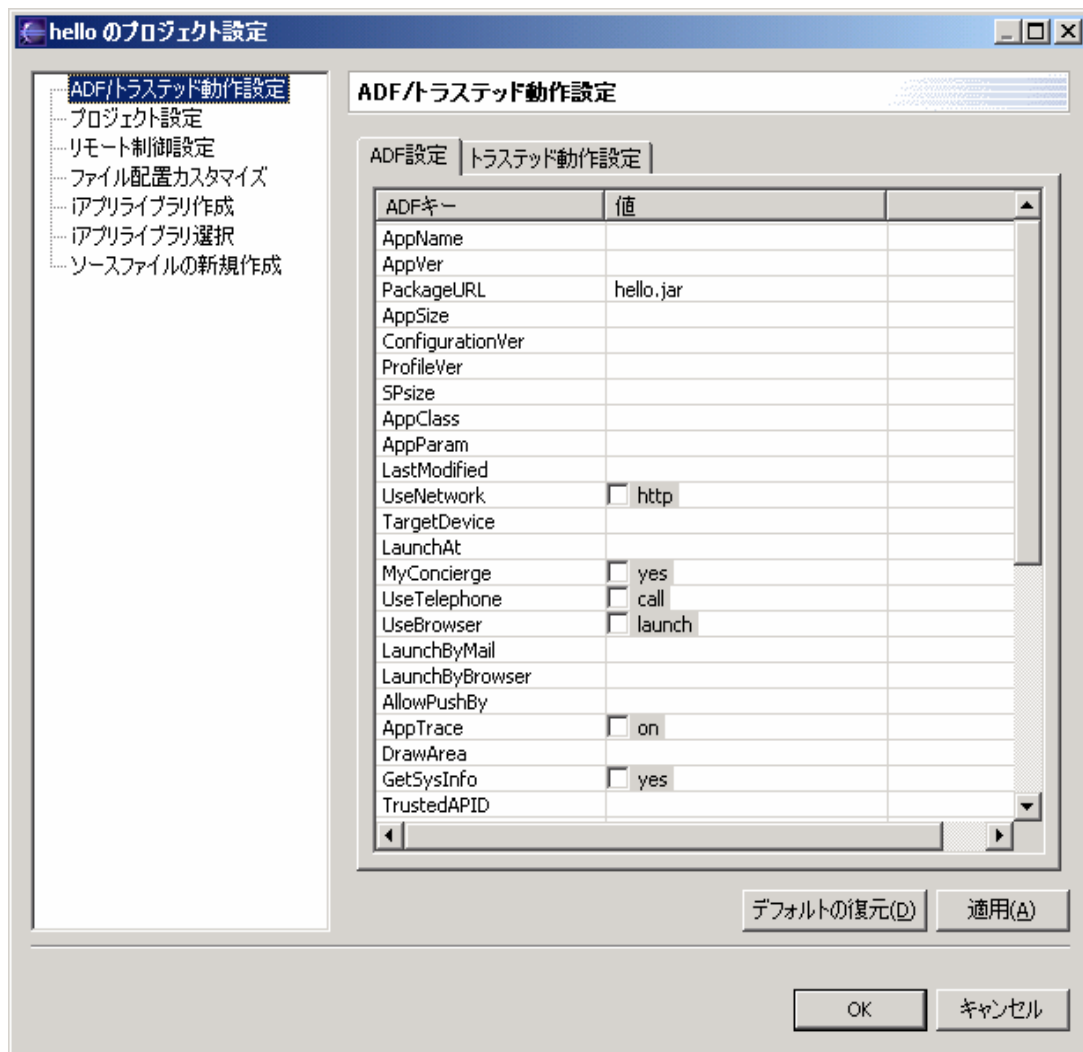


図 1 4 - 1 3 ADF 設定画面

各ADFの設定項目については、「4.7. アプリケーションディスクリプタファイル (ADF) の編集 表 4-3 アプリケーションディスクリプタファイル(ADF)の設定項目」を参照してください。

14.3.7. トラストッド動作の設定

トラステッド動作の設定をするには、次の手順に従います。

1. パッケージ・エクスプローラーウィンドウからプロジェクト (hello) をクリックして、「プロジェクト」→「DoJa-4.1」→「ADF/トラステッド動作設定」をクリックします。
(※プロジェクトを選んでいないと選択できません。)
2. プロジェクト設定ウィンドウが表示されるので、「トラステッド動作設定」タブをクリックし、必要に応じて各キーの属性値を変更してください。

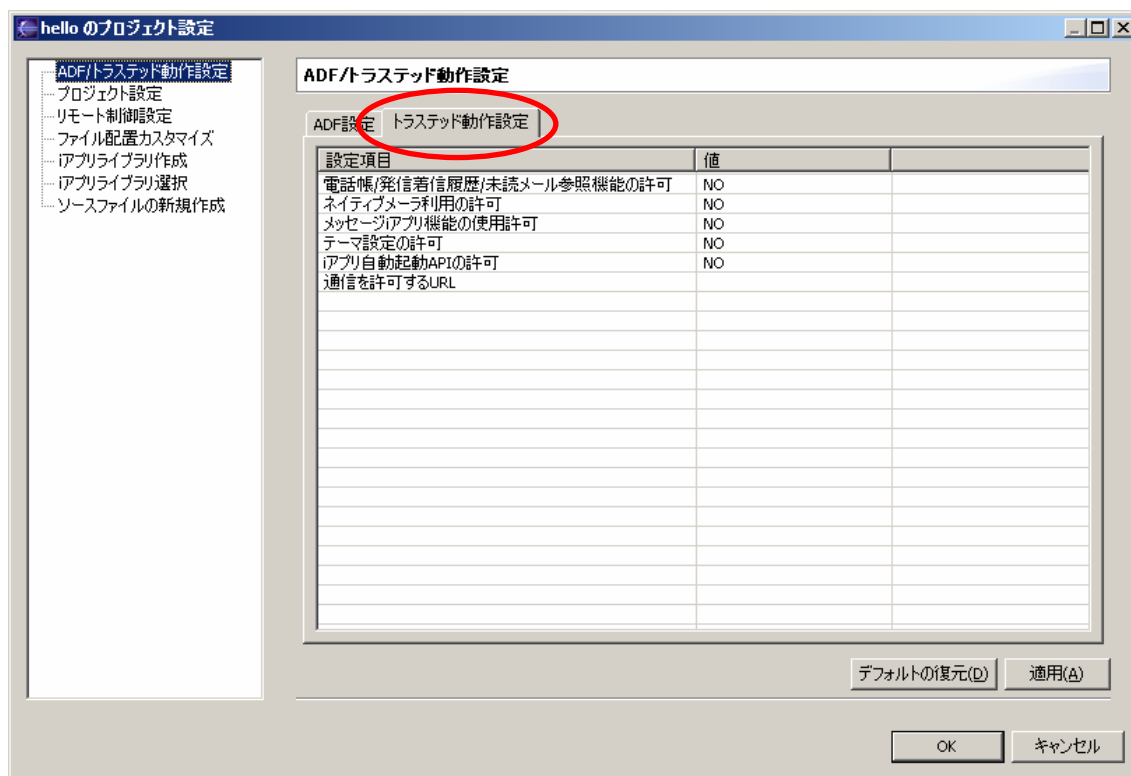


図 14-14 トラストッド動作設定画面

各トラステッド動作設定の設定項目については、「5.2.トラステッド動作の設定」を参照してください。

14.3.8. アプリケーションの構築と実行

アプリケーションを構築および実行するには、次の手順に従います。

1. 「実行」→「実行」を押します。
2. 起動構成ウィンドウが表示されるので、起動構成の「DoJa-4.1 アプリケーション」をマウスでダブルクリック、もしくは、「新規」ボタンを押します。
3. 「新規構成」が追加されるので、これを選択します。
「新規構成」の名前は、必要に応じてアプリケーション名などに変更してください。
4. 「Project」で実行するアプリケーションのプロジェクトを選択します。
5. 「Device type」でエミュレーション端末の種類を選択します。
6. 「Launch type」で起動方法を選択します。
7. 「実行」を押します。

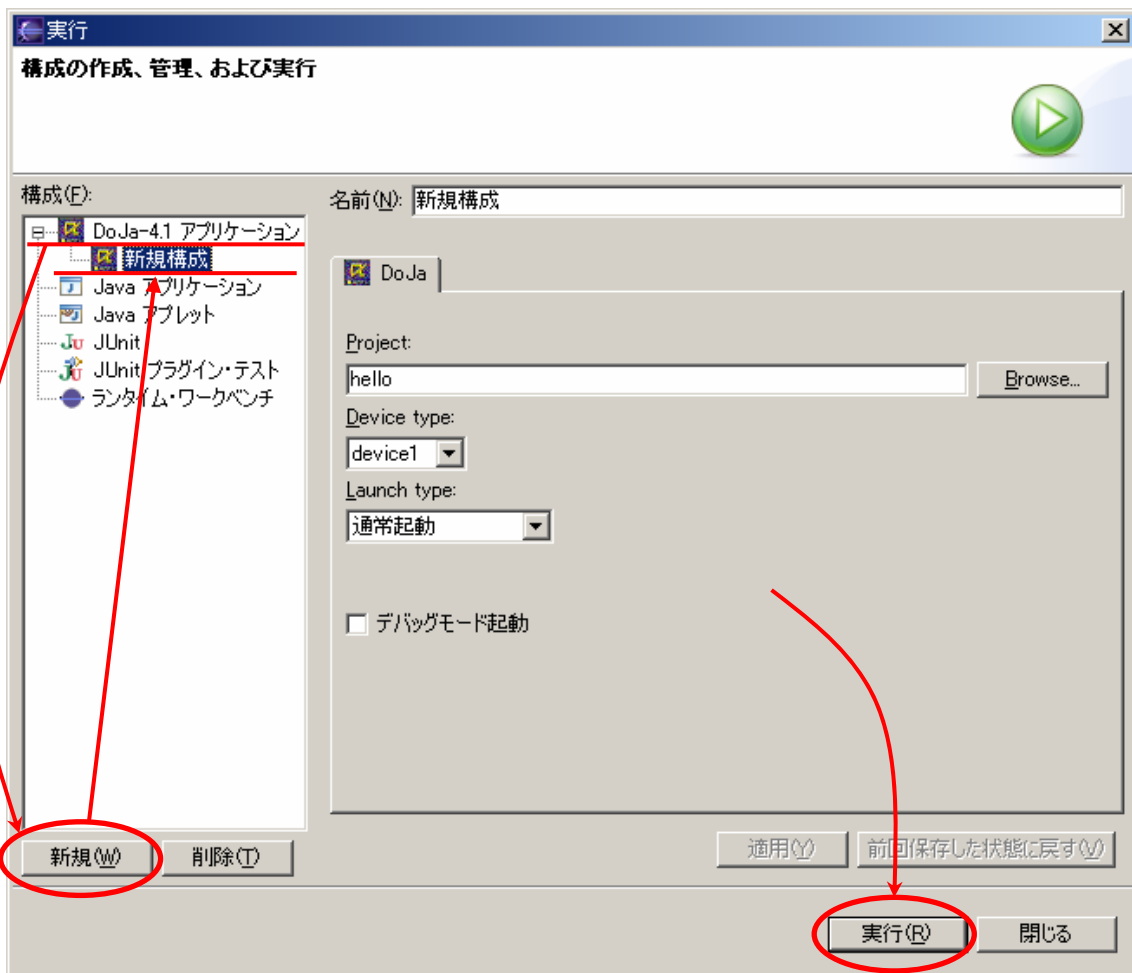
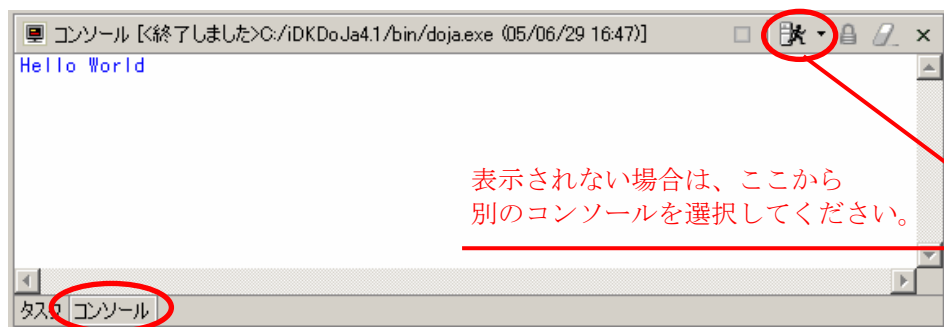
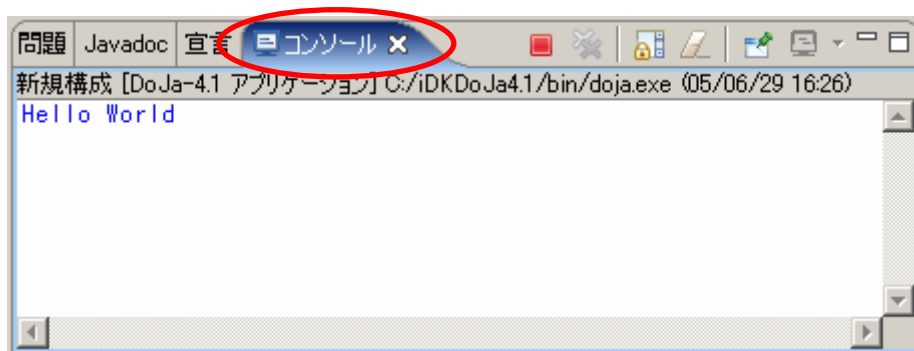


図 1 4 - 1 5 起動構成ウィンドウ

8. preverify（事前検証）、jar ファイルの作成後に端末ウィンドウが起動し、アプリケーションが実行されます。
9. コンソールに出力された内容は、Eclipseのコンソールに出力されます（図 1 4-1 6）。



Eclipse2.1 系



Eclipse3.0 系

図 1 4-1 6 コンソールに出力された内容の表示

14.3.9. プロジェクトの設定

プロジェクトの設定をするには、次の手順に従います。

1. 「プロジェクト」→「DoJa-4.1」→「プロジェクト設定」を選択します。
2. プロジェクト設定ウィンドウが開きます（図 14-17）。

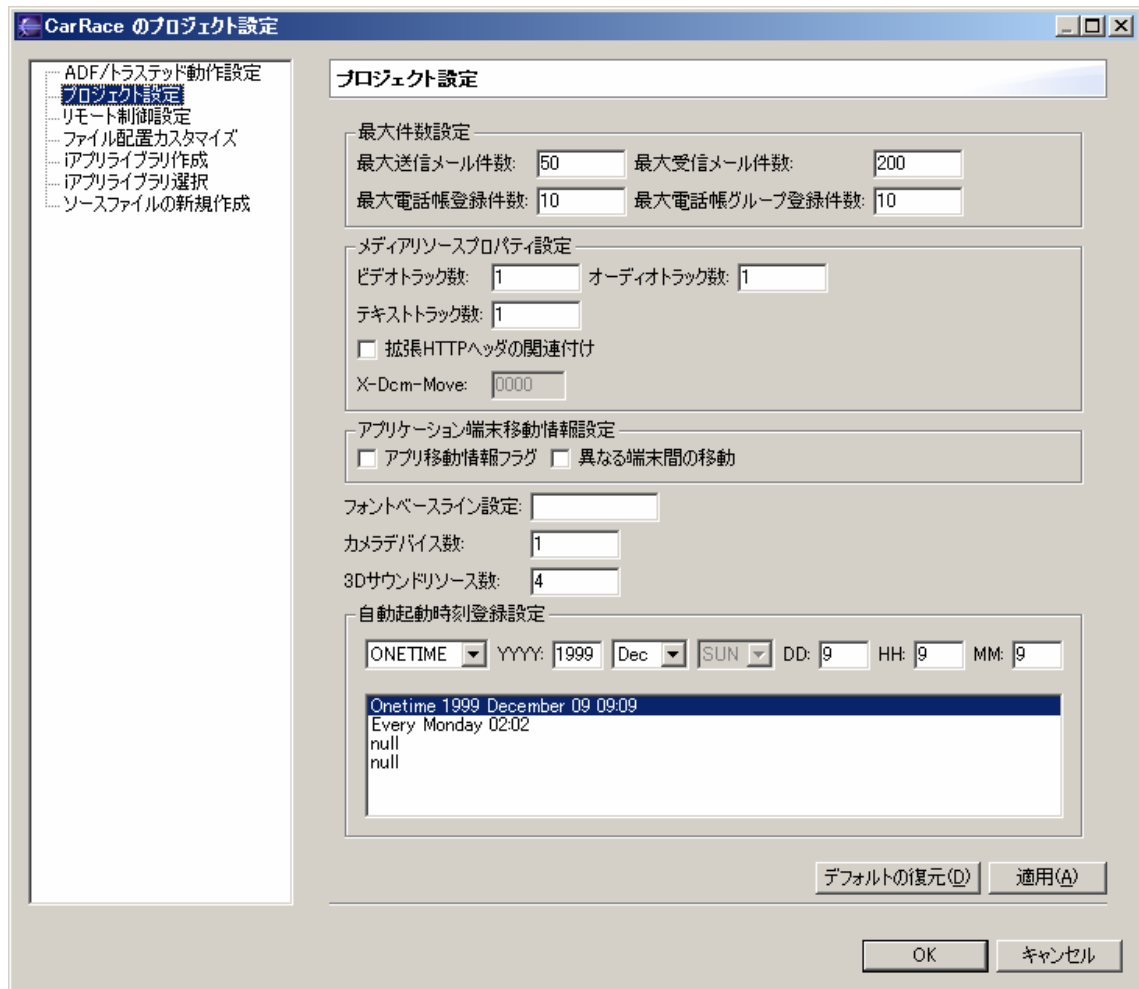


図 14-17 プロジェクト設定画面

各プロジェクト設定の設定項目については、「4.5. プロジェクトの設定」を参照してください。

14.3.10. リモート制御の設定

リモート制御の設定をするには、次の手順に従います。

1. 「プロジェクト」→「DoJa-4.1」→「リモート制御設定」を選択します。
2. プロジェクト設定ウィンドウが開きます。（図 14-18）。

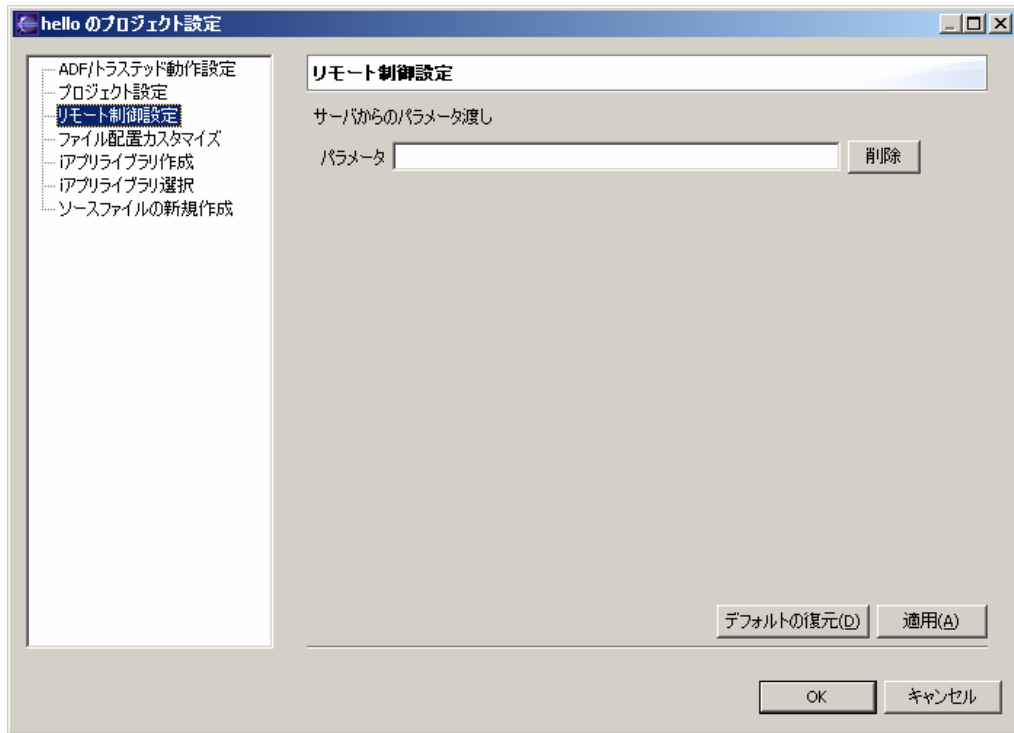


図 14-18 リモート制御設定画面

3. テキストフィールドにパラメータを入力し、「適用」、または「OK」を押します。
「アプリケーション名.properties」ファイルにパラメータが追加されます。
4. 「削除」ボタンを押した直後に「適用」ボタン、または「OK」ボタンを押すと、「アプリケーション名.properties」ファイルからパラメータが削除されます。

設定したパラメータをアプリケーションから取得する方法については、「5.4 リモート制御の設定」を参照してください。

14.3.11. ファイル配置カスタマイズ

ファイル配置カスタマイズの設定をするには、次の手順に従います。

1. 「プロジェクト」→「DoJa-4.1」→「ファイル配置カスタマイズ」を選択します。
2. プロジェクト設定ウィンドウが開きます（図 1 4-1 9）。

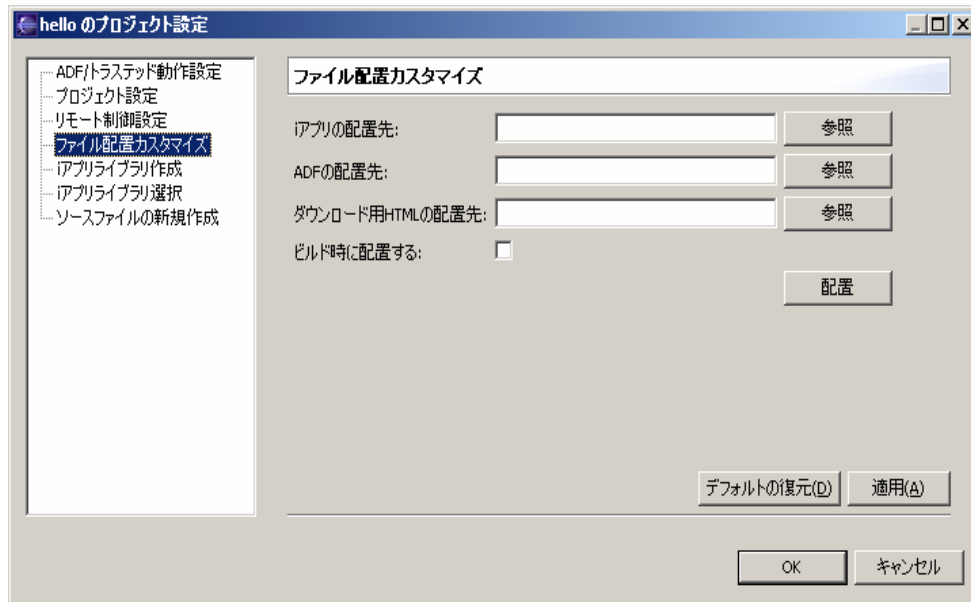


図 1 4-1 9 ファイル配置カスタマイズ

3. それぞれの配置先を入力し、「適用」または「OK」を押します。
このとき、ファイル名まで指定するとそのファイル名に変更されます。
4. 「ビルド時に配置する」にチェックをつけた場合、i アプリのビルド時に自動的に配置処理を実行します。

詳細については、「4.6. ファイル配置カスタマイズ」を参照してください。

14.3.12. iアプリライブラリの作成

i アプリライブラリの作成を設定するには、次の手順に従います。

1. 「プロジェクト」→「DoJa-4.1」→「i アプリライブラリ作成」を選択します。
2. プロジェクト設定ウィンドウが開きます（図 1 4-2 0）。

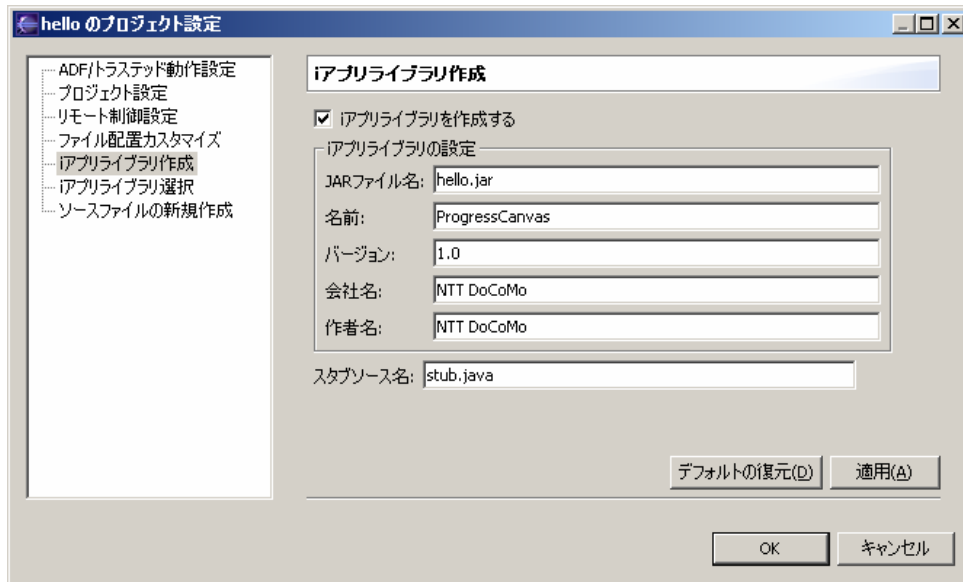


図 1 4-2 0 i アプリライブラリ作成

3. 「i アプリライブラリを作成する」にチェックをします。
4. このライブラリの属性（「JAR ファイル名」、「名前」、「バージョン」、「会社名」、「作者名」）と「スタブソース名」（.java）を設定します。
スタブソース名は、ライブラリをテストするためのプログラムで、あらかじめ作成しておかなくてもテンプレートから自動的に作成されます。
5. 「OK」を押します。
これで i アプリライブラリが作成できるようになります。

詳細については、「6.7.1. i アプリライブラリの作成」を参照してください。

14.3.13. iアプリライブラリの利用

i アプリライブラリを利用するには、次の手順に従います。

1. 「プロジェクト」→「DoJa-4.1」→「i アプリライブラリ選択」を選択します。
2. プロジェクト設定ウィンドウが開きます(図 14-21)。

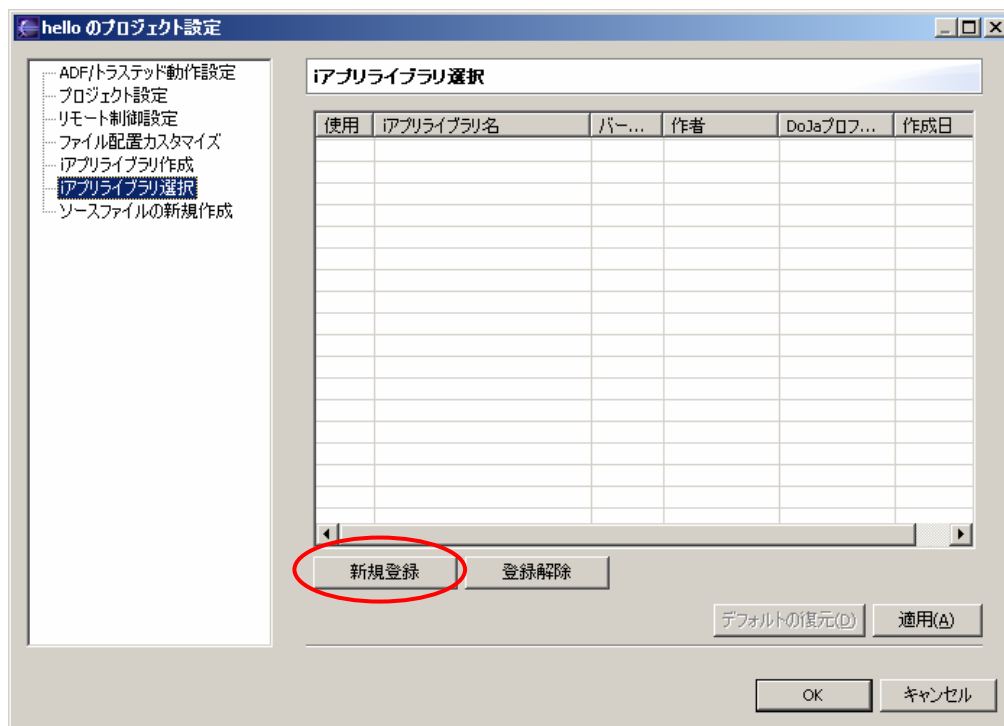


図 14-21 i アプリライブラリ選択

3. i アプリライブラリ選択画面が表示されるので「新規登録」を押します。
4. ファイル選択ダイアログで登録したい i アプリライブラリ(.jar)を選択すると i アプリライブラリの選択画面に追加されます(図 14-23)。
ここで、登録されたライブラリは、他のプロジェクトでも利用できます。
追加した i アプリライブラリは、右下の「キャンセル」を押しても消えませんが、削除したい場合は、「登録解除」で消してください。

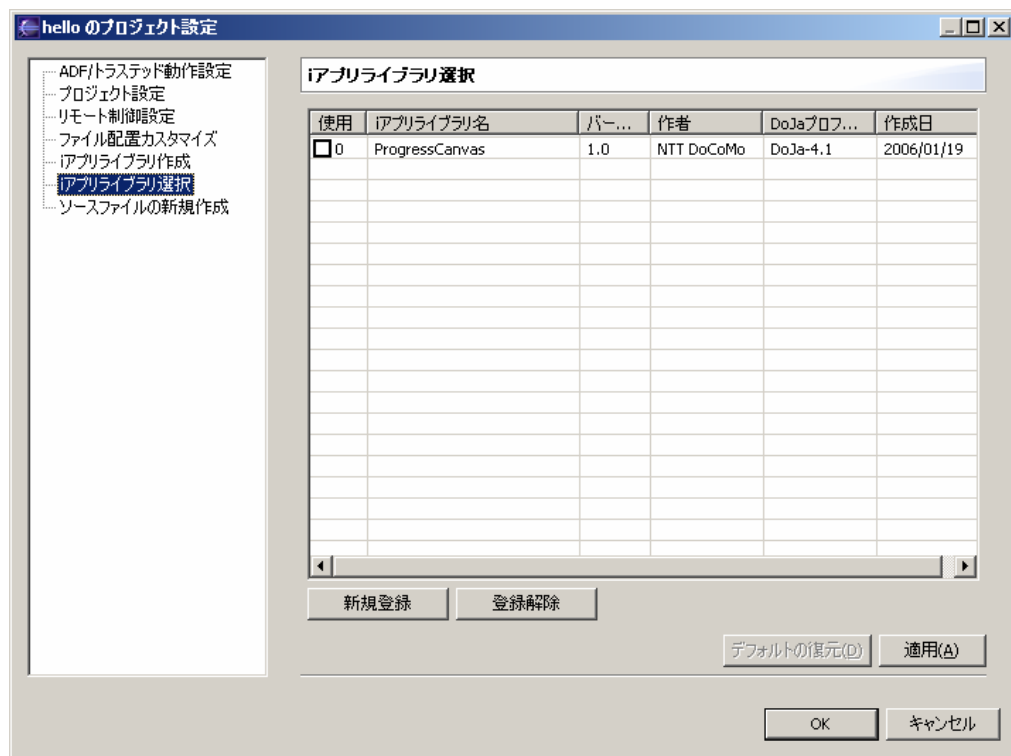


図 14-22 i アプリライブラリの選択画面 - ライブラリの追加



 ワンポイント

次の場所に i アプリライブラリを置いておくと自動的に登録できます。

C:\iDKDoJa4.1\lib\component

5. 不要になった i アプリライブラリは、一覧から選択して「登録解除」を押します。
(※ C:\iDKDoJa4.1\lib\component のライブラリは解除できません。)
6. 登録されたライブラリから使いたいライブラリだけにチェックをつけます。

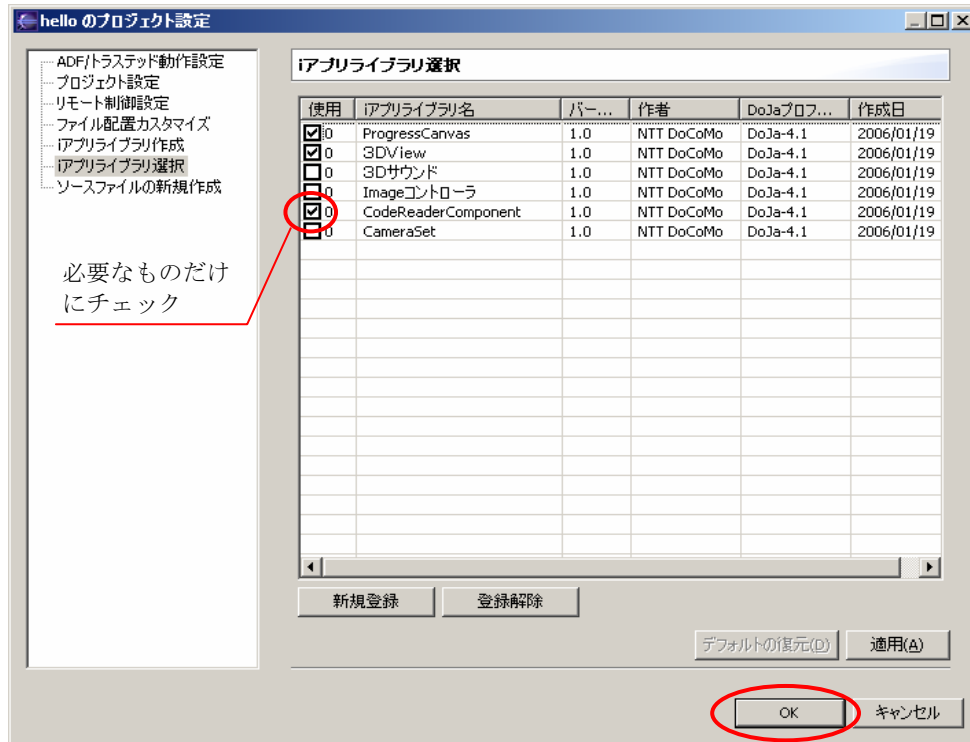


図 1 4-2 3 ライブラリの選択

7. 「OK」を押します。
チェックのついた i アプリライブラリが自動的に組み込まれ、i アプリからライブラリ中のクラスやメソッドを呼び出すことができますようになります。
8. i アプリを実行します。

詳細については、「6.7.2. i アプリライブラリの利用」を参照してください。

14.3.14. デバッグの実行

アプリケーションをデバッグするには、次の手順に従います。^[9]

1. パッケージ・エクスプローラーからソース（HelloWorld.java）を開きます。
ソースに変数を定義して、変数の内容を表示させるとします。
たとえば、
 int a = 1;
と
 System.out.println("a = " + a);
を追加します。

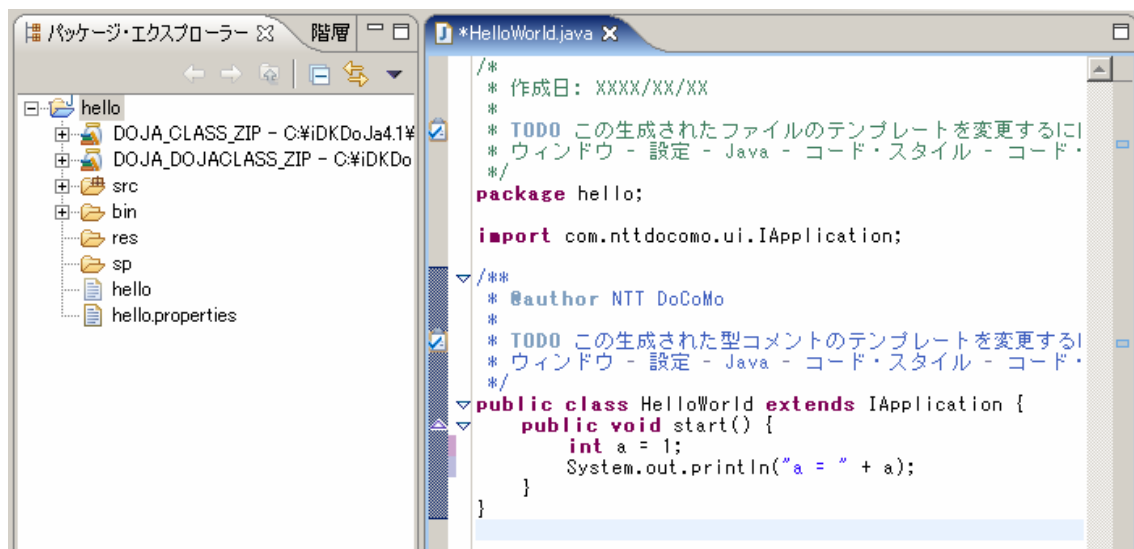


図 1 4-2 4 変数の表示

^[9] エミュレータには、米国 Sun Microsystems 社が提供する CLDC の KDWP DEBUG PROXY (DEBUG AGENT) TOOL が組み込まれています。

2. デバッグで止めたい行（ブレークポイント）の左枠をクリックしてチェックを付けます（図 1 4-2 5）。

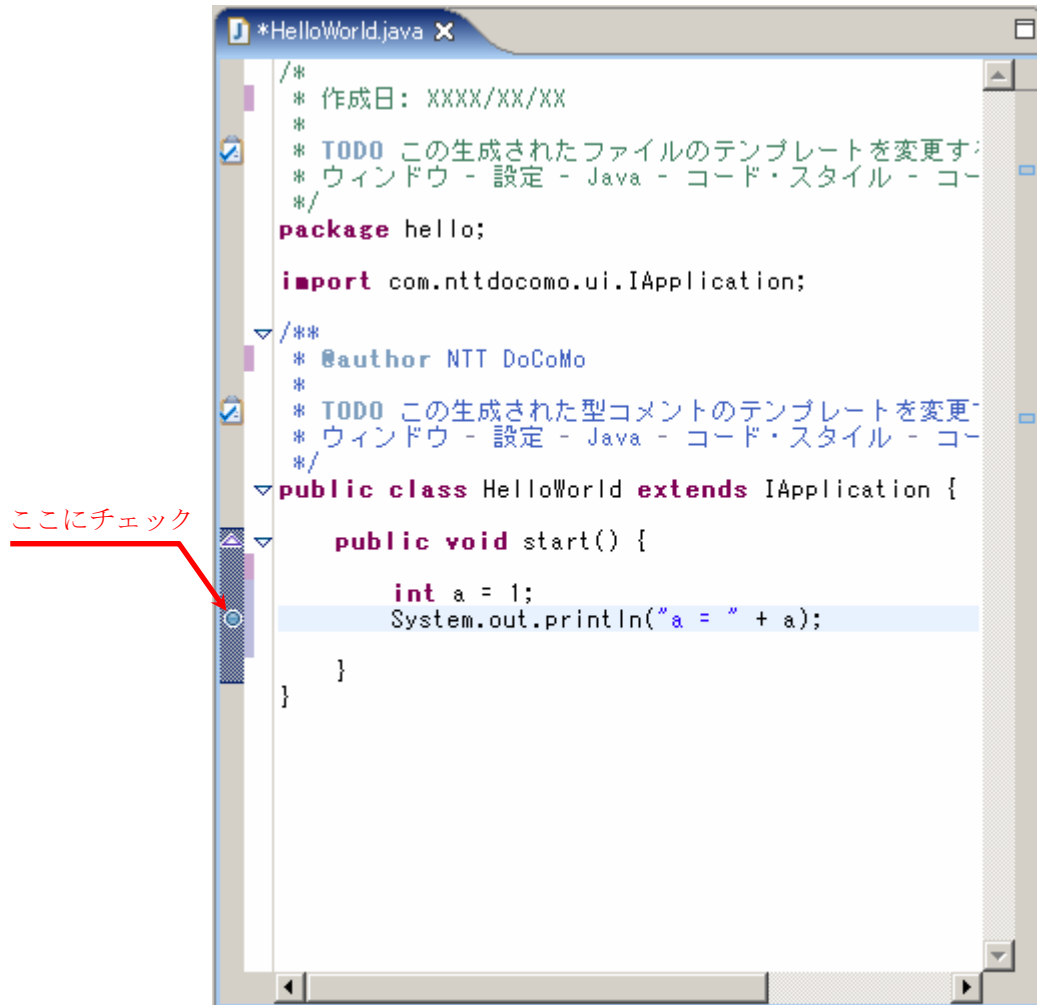


図 1 4-2 5 ブレークポイントの設定

3. 「実行」→「**実行**」を押します。
4. 起動構成ウィンドウが表示されるので、起動構成の「デバッグモード起動」にチェックマークを入れて、「実行」を押します。

もし、デバッグ操作を無効にする場合は、実行画面の「デバッグモード起動」のチェックを外して下さい。

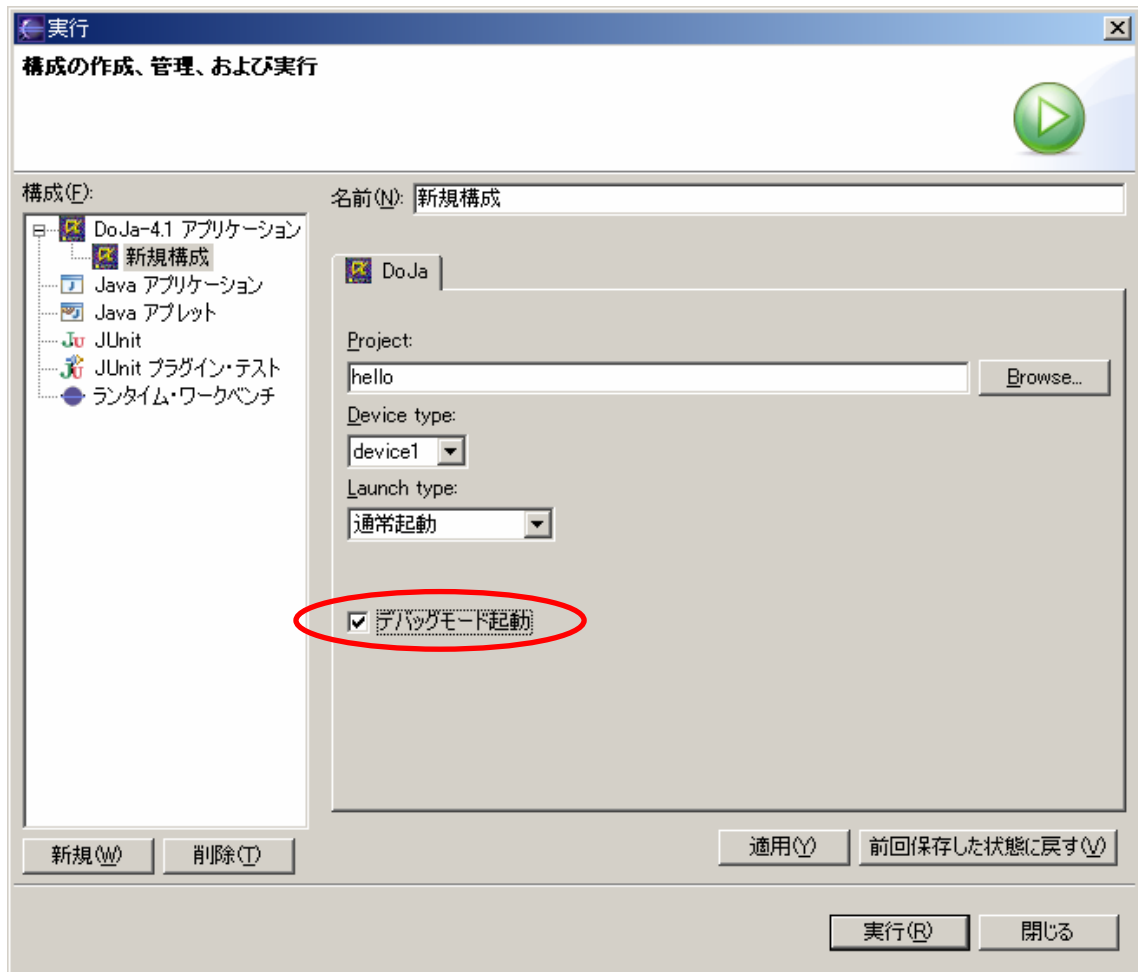
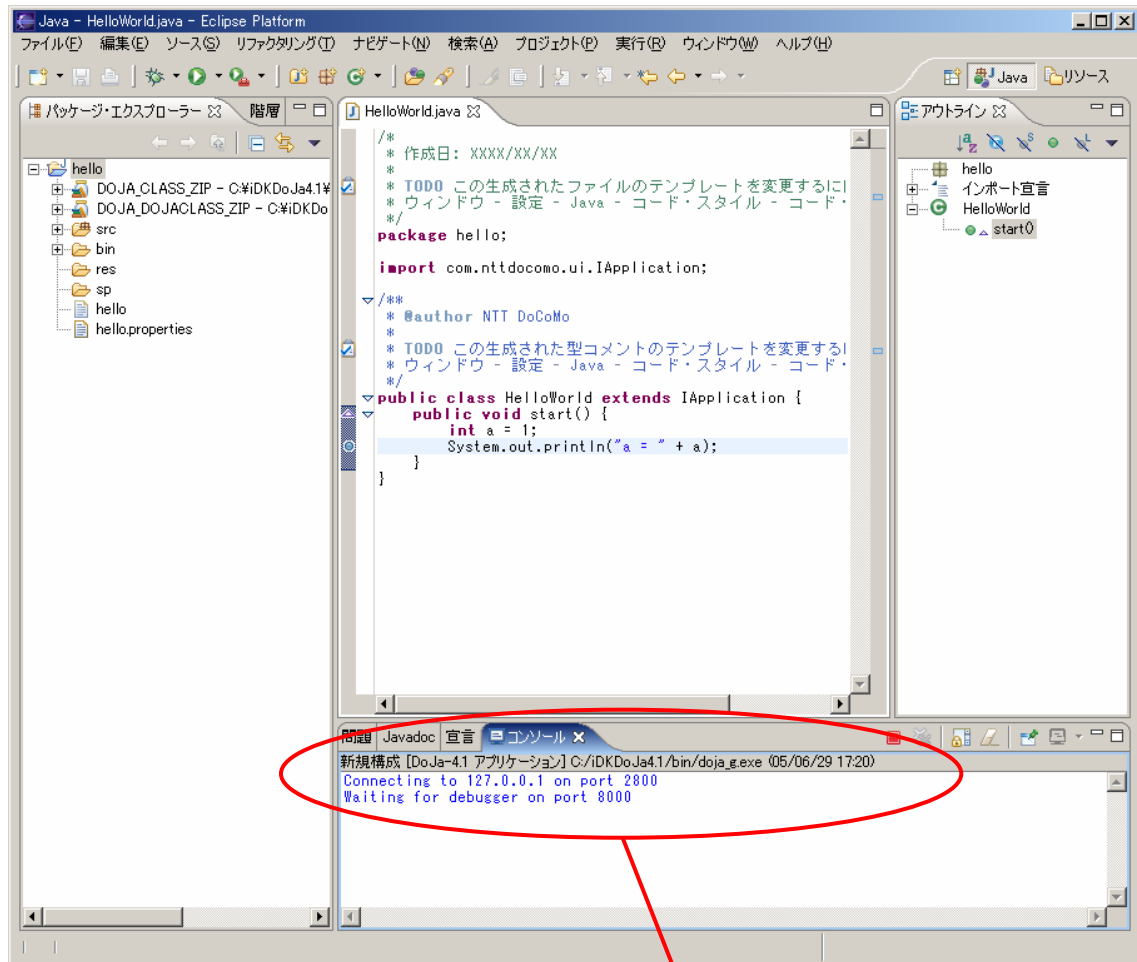


図 1 4-2 6 デバッグモードでの実行

5. 端末ウィンドウが表示されて、実行が止まったままになります（図 1 4-2 7）。この状態でエミュレータはデバッガ命令を待ち受けています。



待ち受けている状態

図 1 4-2 7 デバッグ実行

6. 次に「実行」 → 「**デバッグ**」を選択します。
7. デバッグ起動構成ウィンドウが表示されるので、構成の「**リモート Java アプリケーション**」をマウスでダブルクリック、もしくは、「新規」ボタンを押します。
8. ここでは「HelloWorld」が追加されるので、これを選択します。
「HelloWorld」の名前は、必要に応じて変更することができます。
9. 「接続プロパティ」に、
ホストを 「localhost」
ポートを 「8000」
に設定し「デバッグ」を押します（図 1 4 - 2 8）。
（この値は、一度設定するとそのまま保持されます。）

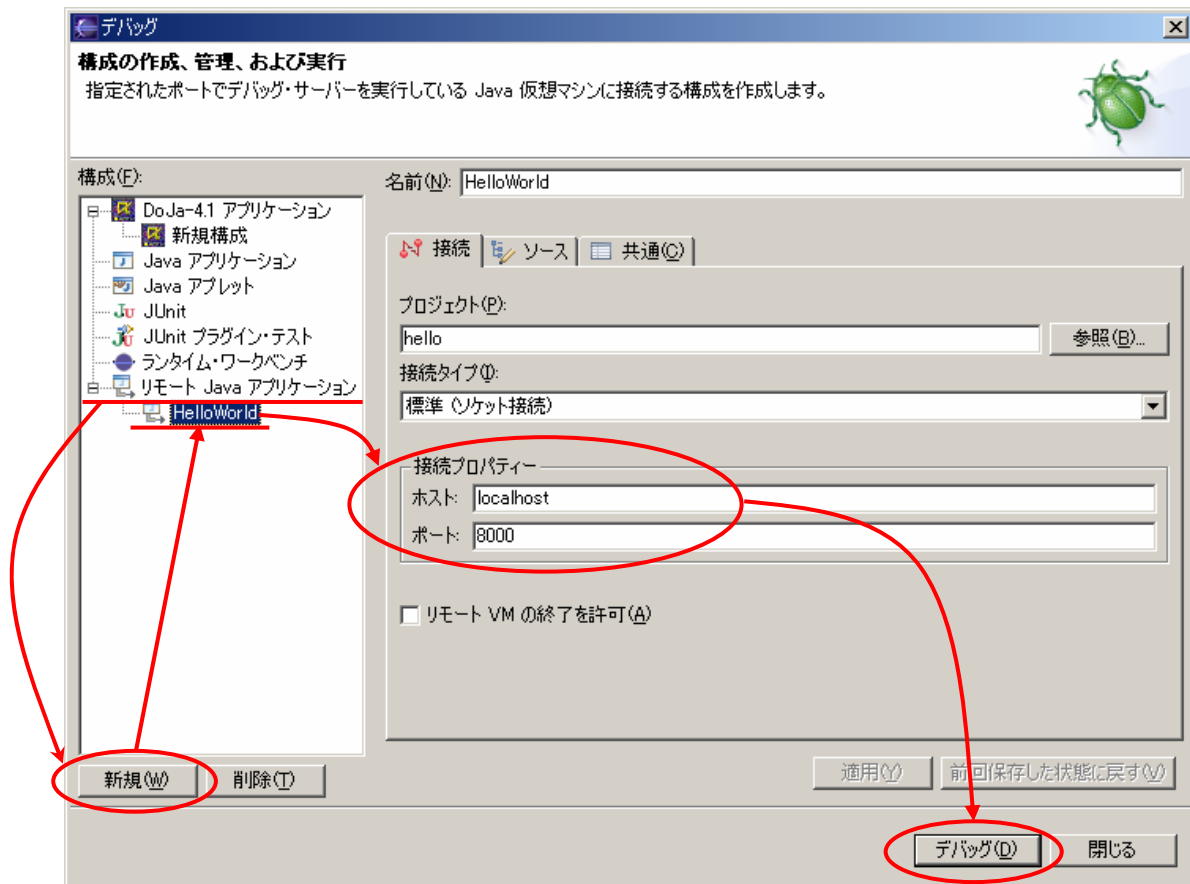


図 1 4 - 2 8 デバッグ接続画面

10. デバッグパースペクティブと端末ウィンドウが表示され、ブレークポイントで処理が中断します（図 1 4-2 9）。

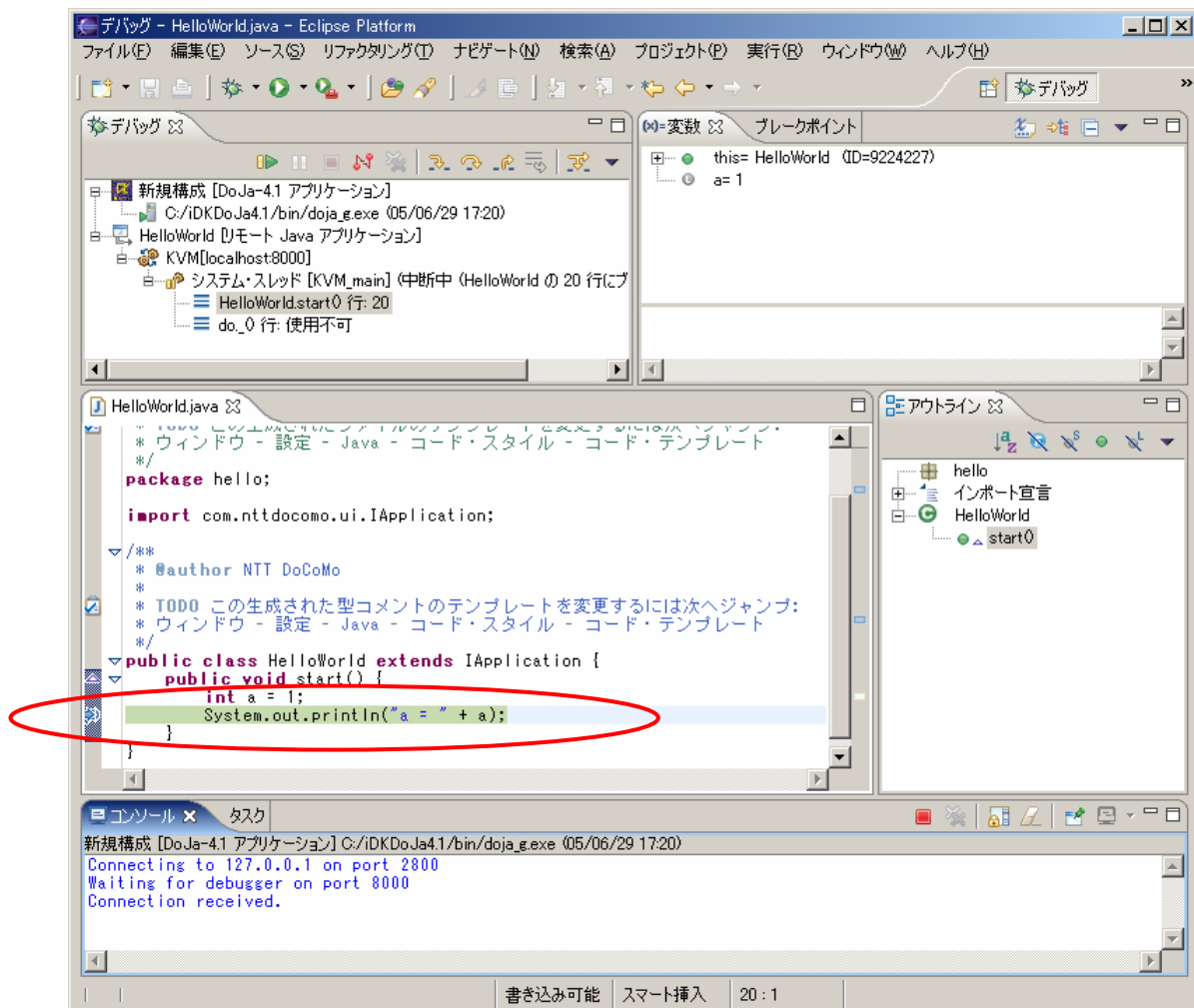


図 1 4-2 9 ブレークポイントでの処理の中断

11. 変数ウィンドウのウォッチポイントで、「a=1」を右クリックし、「値の変更」(もしくは「変数の変更」)を選択します(図 1 4-3 0)。

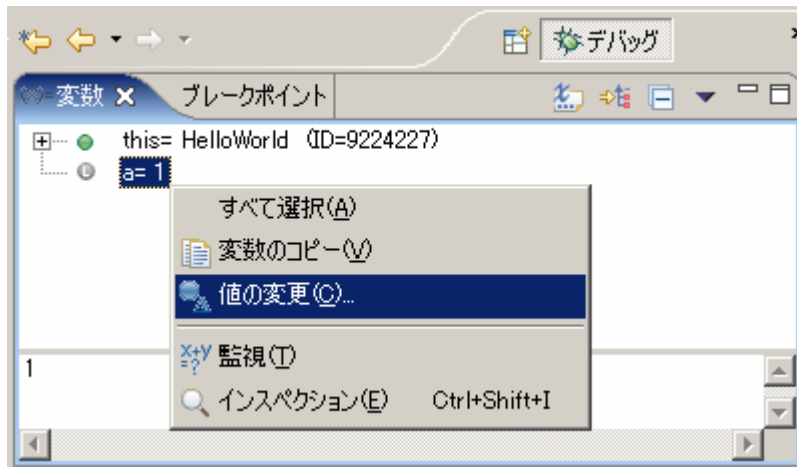


図 1 4-3 0 変数の変更

12. 「変数値を設定」ダイアログが表示されるので、変数の値を変更します。たとえば、aの値を 10 に変更します(図 1 4-3 1)。

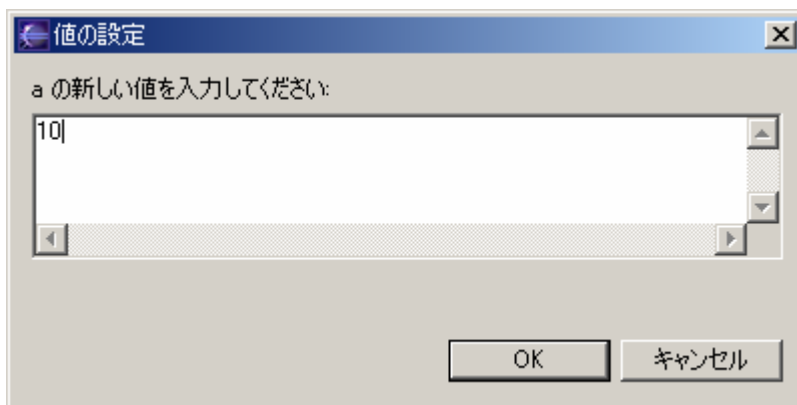


図 1 4-3 1 変数値の設定

13. 「デバッグビュー」 → 「再開」で中断した位置から処理を再開させます。
変数に変更された値で処理が再開されます。
「実行」メニューからも操作する事が可能です。
デバッグの実行は、デバッグビューでも様々なパターンをテストできます。

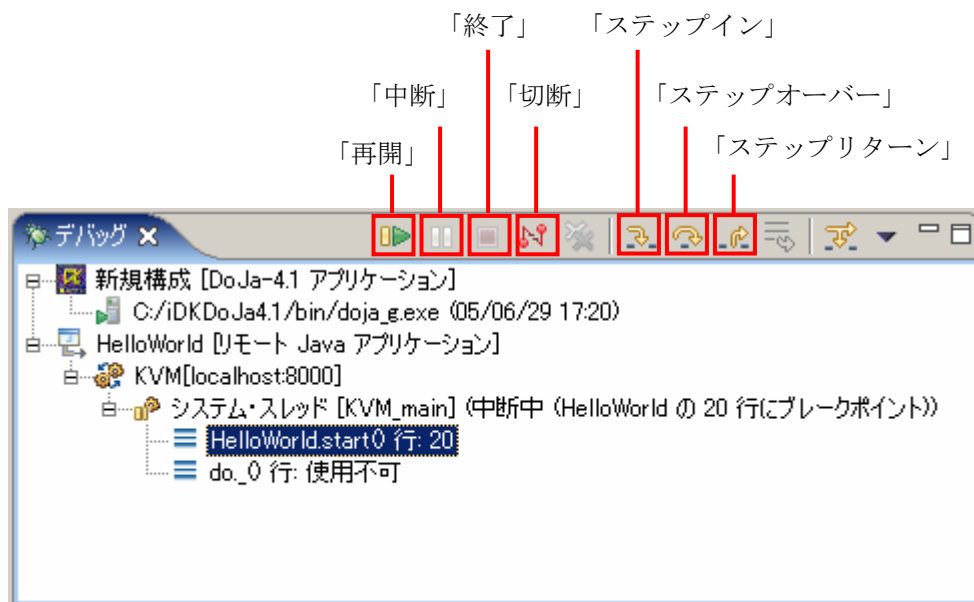


図 1 4-3 2 デバッグビューのボタン

14. 実行結果がコンソールウィンドウに `a = 10` と表示されます。
もし、表示されない場合は、別のコンソールウィンドウに切り替えてください。



図 1 4-3 3 デバッグ実行結果

15. 端末ウィンドウの終話ボタンを押すか、「デバッグビュー」 → 「切断」でデバッグを終了します。

14.4. エミュレータの設定

14.4.1. インストール先の設定

エミュレータのインストール先を設定することができます。設定をするには、次の手順に従ってください。

1. 「ウィンドウ」→「設定」を選択します。
2. Eclipse設定ウィンドウが表示されるので、「DoJa-4.1 Environment」を選択します（図 1 4-3 4）。

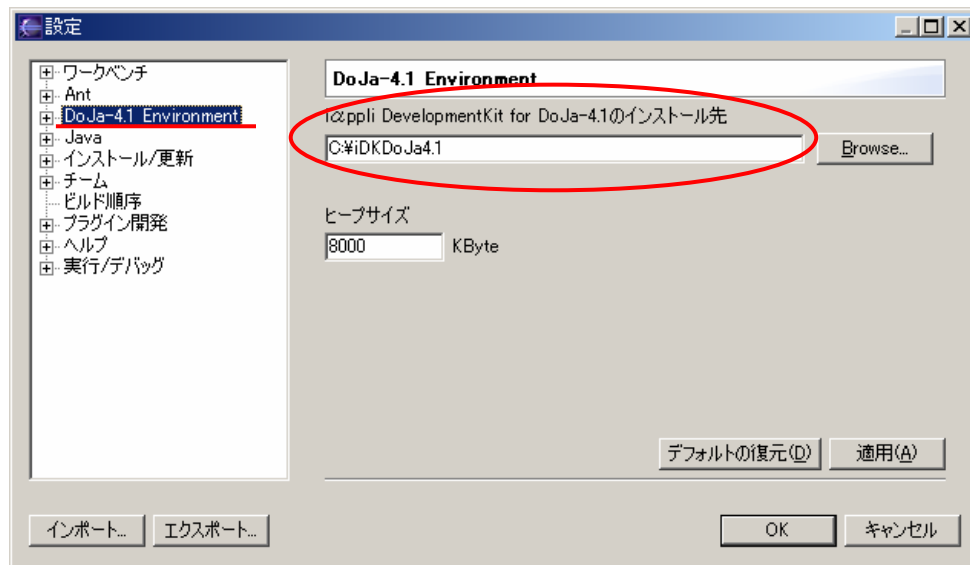


図 1 4-3 4 Eclipse 設定ウィンドウ

3. インストール先を設定して「OK」を押します。

14.4.2. ヒープサイズの設定

エミュレータのヒープサイズを設定することができます。設定をするには、次の手順に従ってください。

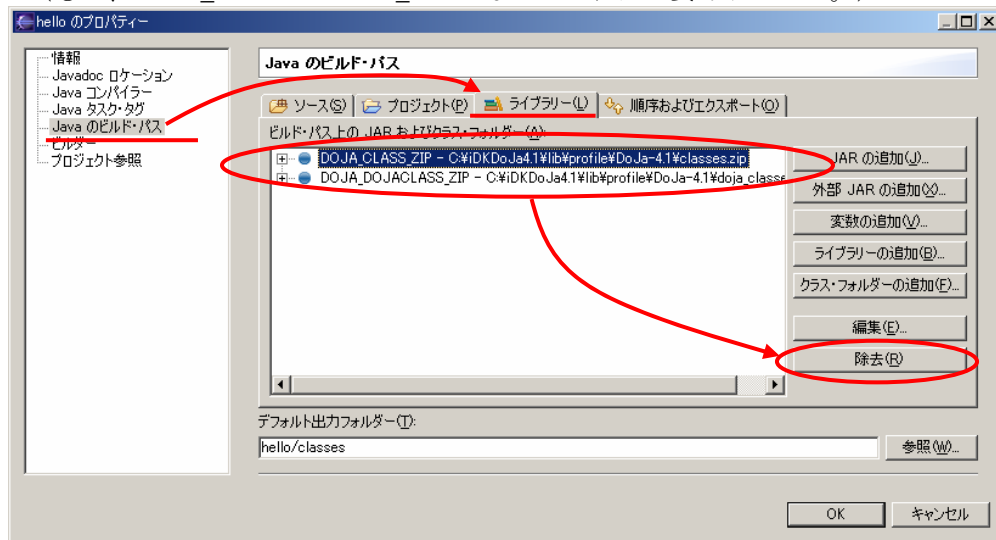
1. 「ウィンドウ」→「設定」を選択します。
2. Eclipse設定ウィンドウが表示されるので、「DoJa-4.1 Environment」を選択します（図 1 4-3 4）。
3. ヒープサイズを設定して「OK」を押します。

14.4.3. ターゲットプロファイルの変更

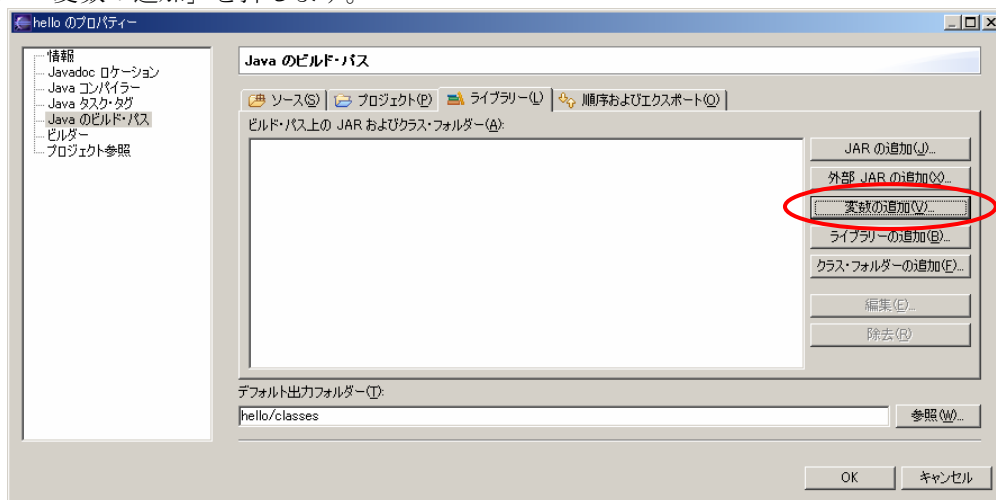
i アプリをビルドするターゲットプロファイルを変更します。
ターゲットプロファイルを変更するには、次の手順に従ってください。

1. 旧バージョンのプラグインで作成されたプロジェクトの場合は、以下の手順を実施してください。（プロジェクト毎に初回のみです。）
新規プロジェクトの場合は、手順2から実施してください。

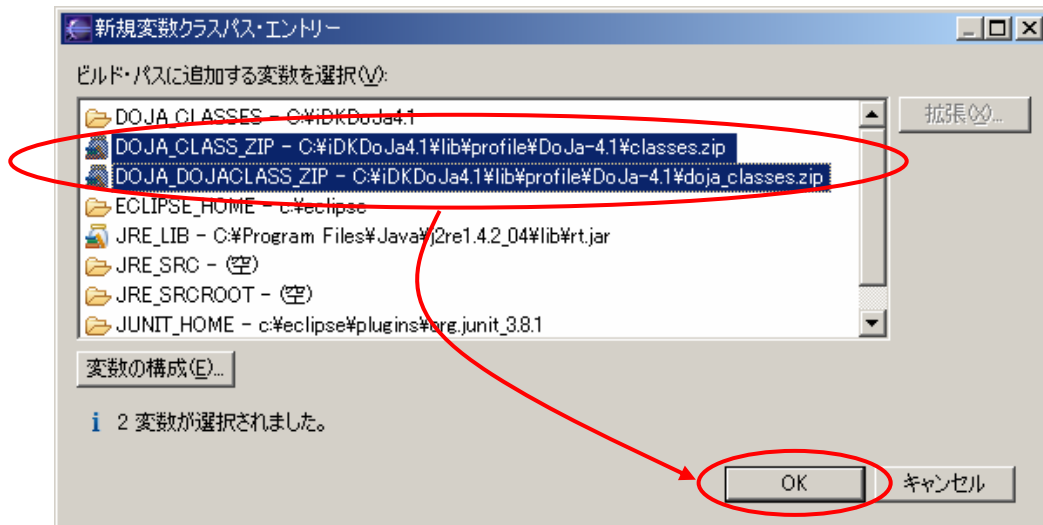
- ① 「プロジェクト」→「プロパティ」→「Java のビルド・パス」→「ライブラリ」を選択します。
- ② 「DOJA_CLASSES/lib/classes.zip」ライブラリを選択して「除去」を押します。
（もし、DOJA_CLASS_ZIP になっていれば必要ありません。）
- ③ 同様に「DOJA_CLASSES/lib/doja_classes.zip」ライブラリを選択して「除去」を押します。
（もし、DOJA_DOJACLASSES_ZIP になっていれば必要ありません。）



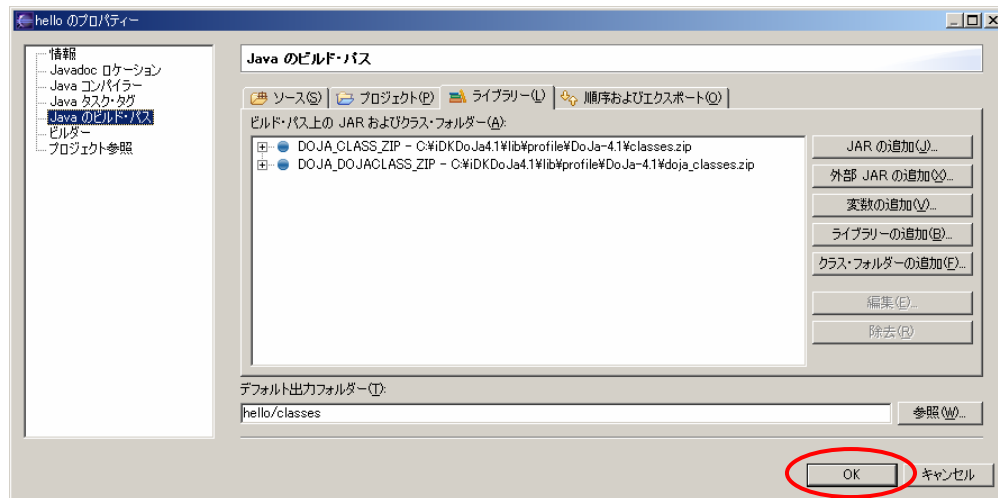
- ④ 「変数の追加」を押します。



- ⑤ 変数一覧から「DOJA_CLASS_ZIP」と「DOJA_DOJACLASS_ZIP」を選択して「OK」を押します。



- ⑥ 一覧に選択した変数（「DOJA_CLASS_ZIP」と「DOJA_DOJACLASS_ZIP」）が追加されていることを確認して「OK」を押します。



2. 「ウィンドウ」→「設定」を選択します。
3. Eclipse設定ウィンドウが表示されるので、「DoJa-4.1 Environment」→「ビルド設定」を選択します（図 1 4 - 3 5）。

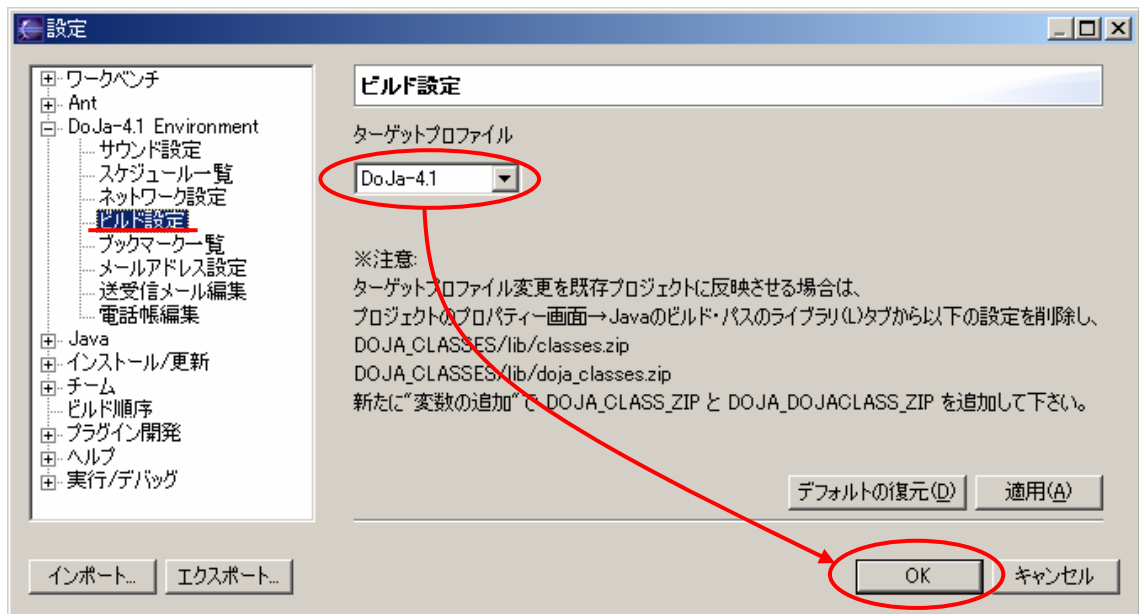


図 1 4 - 3 5 ビルド設定

4. ビルド対象にするターゲットプロファイルを選択して「OK」を押します。

※ ここで設定したプロファイルは、i アプリのビルド時に使われており、
i アプリの動作には影響しません。

詳細については、「5.1 4.2. ターゲットプロファイルの設定」を参照してください。

14.4.4. ネットワークの設定

エミュレータのネットワークの設定を行うことができます。設定を行うには、次の手順に従ってください。

1. 「ウィンドウ」→「設定」選択します。
2. Eclipse設定ウィンドウが表示されるので、「DoJa-4.1 Environment」→「ネットワーク設定」を選択します（図 14-36）。

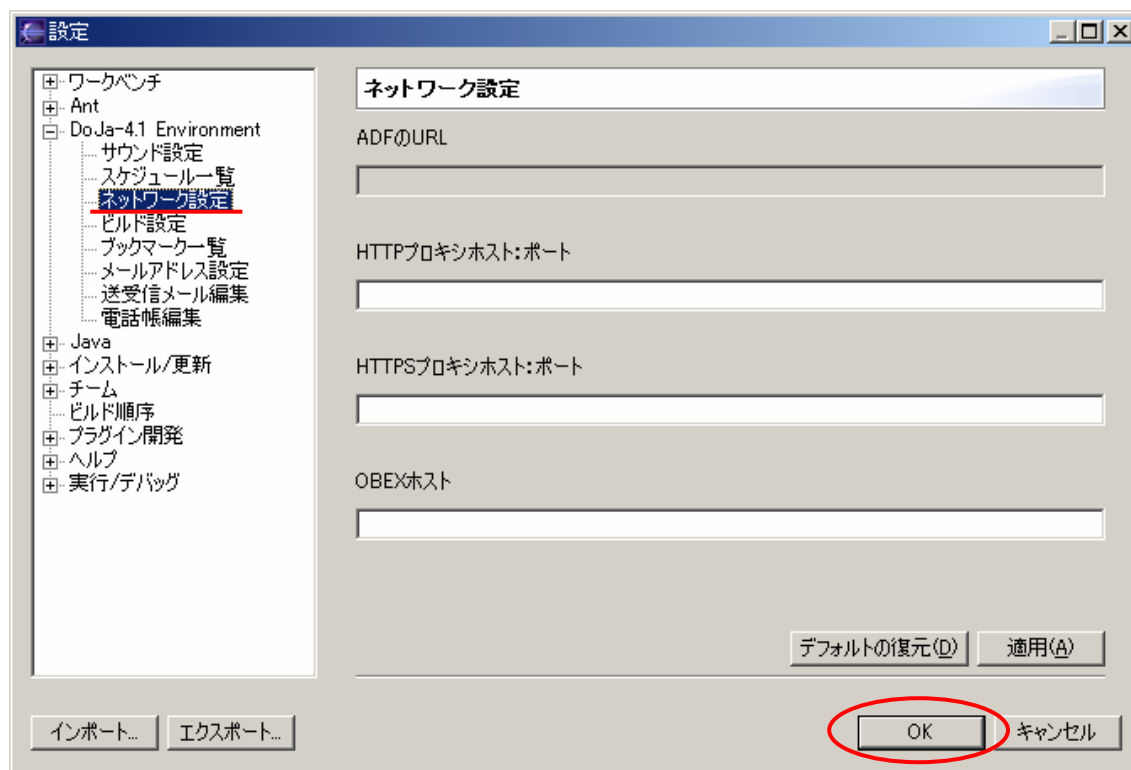


図 14-36 ネットワーク設定ウィンドウ

3. ネットワークの設定を変更して「OK」を押します。

ネットワークの各設定項目の詳細については、「5.1. ネットワークの設定」を参照してください。

14.4.5. メールアドレスの設定

エミュレータのメールアドレスの設定をすることができます。設定をするには、次の手順に従ってください。

1. 「ウィンドウ」→「設定」を選択します。
2. Eclipse設定ウィンドウが表示されるので、「DoJa-4.1 Environment」→「メールアドレス設定」を選択します（図 1 4-3 7）。

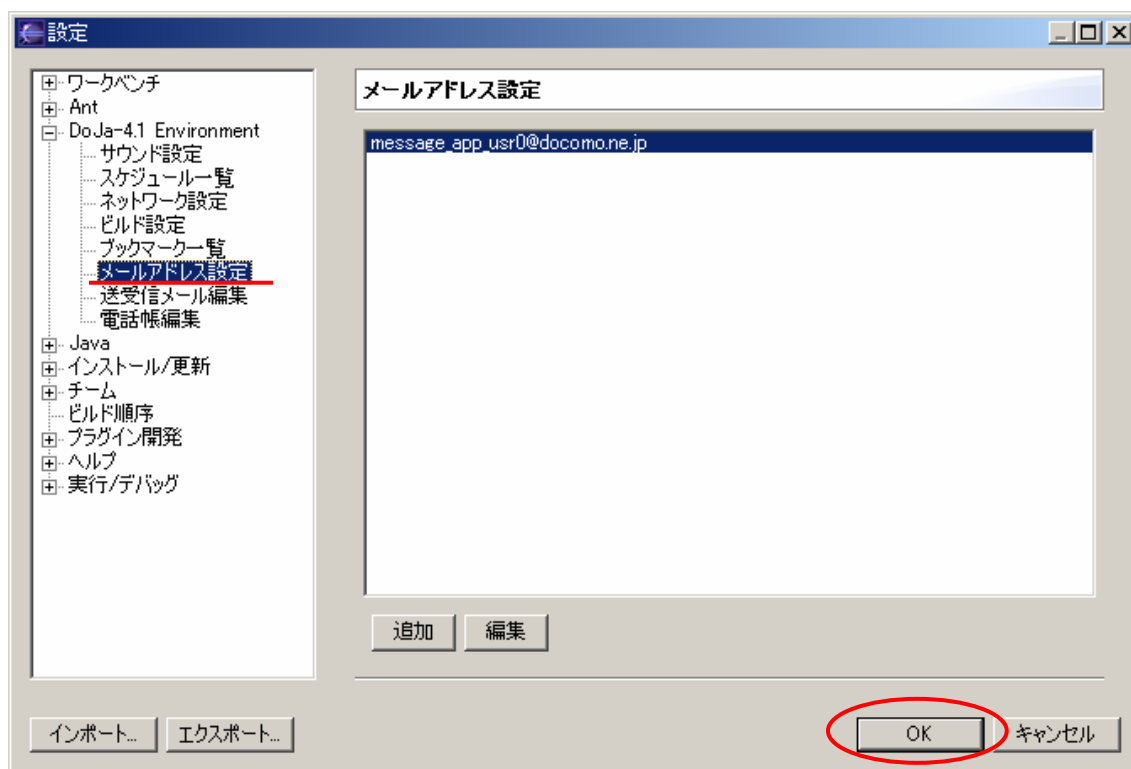


図 1 4-3 7 メールアドレス設定画面

3. メールアドレスを追加するには「追加」、変更するには変更したいアドレスを選択し、「編集」を押します。
メールアドレス変更画面が表示されます（図 1 4-3 8）。



図 1 4-3 8 メールアドレス変更画面

4. アドレスを変更後、「OK」を押します。
メールアドレス設定画面の一覧にアドレスの変更が反映されます。

14.4.6. 電話帳の編集

エミュレータの電話帳の編集をすることができます。設定をするには、次の手順に従ってください。

1. 「ウィンドウ」→「設定」を選択します。
2. Eclipse設定ウィンドウが表示されるので、「DoJa-4.1 Environment」→「電話帳編集」を選択します（図 14-39）。

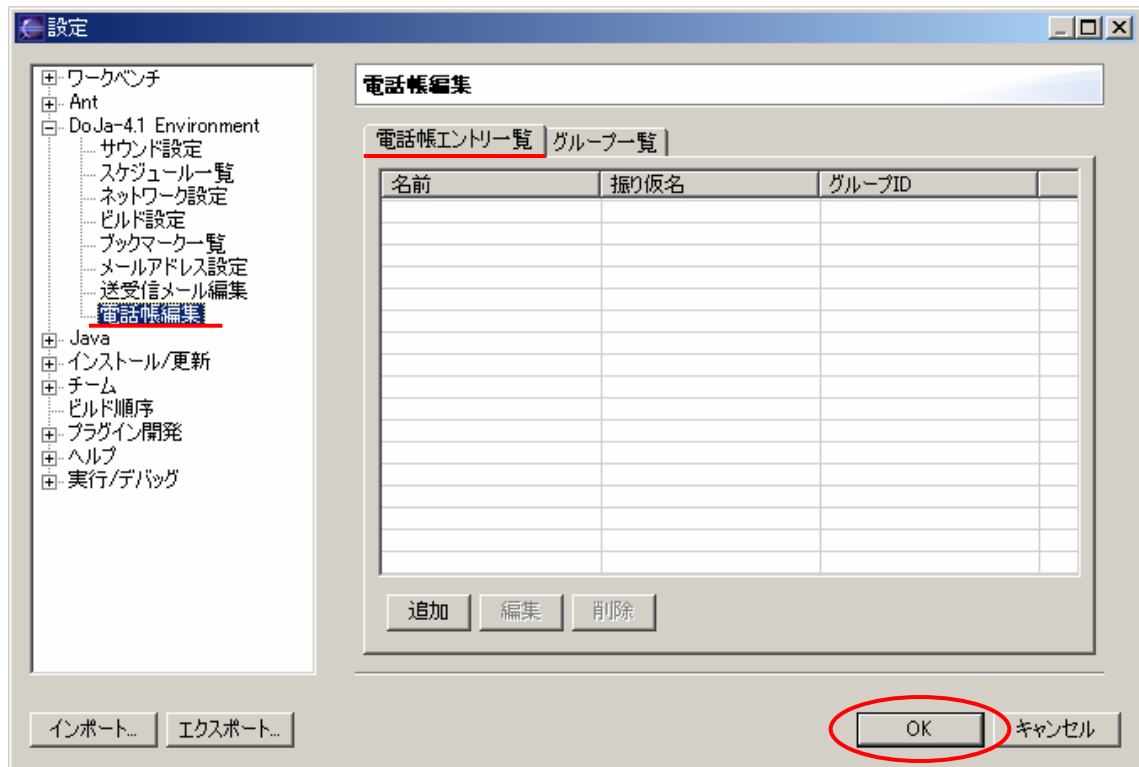


図 14-39 電話帳編集画面

3. 電話帳エントリの追加をするには「追加」、編集をするには「編集」を押します（図 1 4 - 4 0）。



電話帳エントリ登録

名前 姓 DoCoMo

名 太郎

振り仮名 姓 ドコモ

名 タロウ

電話番号 1 090XXXXXXXX

2 03XXXXXXXX

3

メールアドレス 1 message_app_usr1@

2

3

グループID 1

設定 取消

図 1 4 - 4 0 電話帳エントリ画面

4. 各設定項目を設定します。
5. 「設定」を押すと、電話帳一覧に追加されます。

各設定項目の詳細については、「表 5 - 4 . 電話帳エントリの設定項目」を参照してください。

6. 電話帳エントリを削除するには、一覧から電話帳エントリを選択して「削除」を押します。
電話帳エントリファイルが削除され、電話帳一覧からエントリが削除されます。

14.4.7. グループの編集

エミュレータのグループの編集をすることができます。設定をするには、次の手順に従ってください。

1. 「ウィンドウ」→「設定」を選択します。
2. Eclipse設定ウィンドウが表示されるので、「DoJa-4.1 Environment」→「電話帳編集」を選択し、グループ一覧タブをクリックします（図 1 4-4 1）。

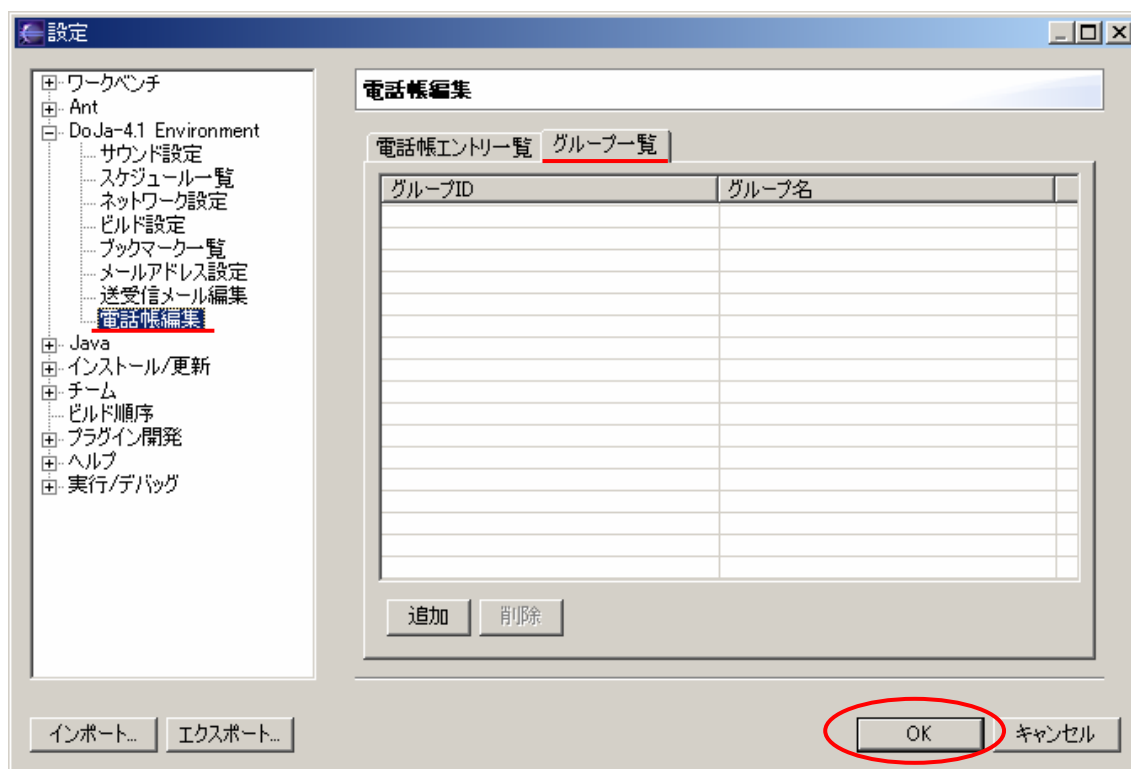


図 1 4-4 1 Eclipse 設定ウィンドウのグループ編集画面

3. グループを追加したい場合は「追加」を押します（図 1 4-4 2）。

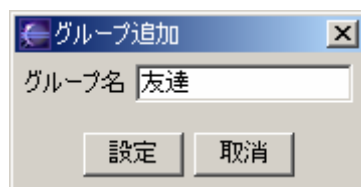


図 1 4-4 2 グループ追加画面

4. 「設定」を押します。
グループ一覧にグループ名が追加されます。
5. グループを削除したい場合は、一覧からグループを選択し、「削除」を押します。
グループファイルが削除され、一覧からグループ名が削除されます。

14.4.8. 送受信メールの編集

エミュレータの送受信メールの編集をすることができます。設定をするには、次の手順に従ってください。

1. 「ウィンドウ」→「設定」を選択します。
2. Eclipse設定ウィンドウが表示されるので、「DoJa-4.1 Environment」→「送受信メール編集」を選択します（図 1 4-4 3）。

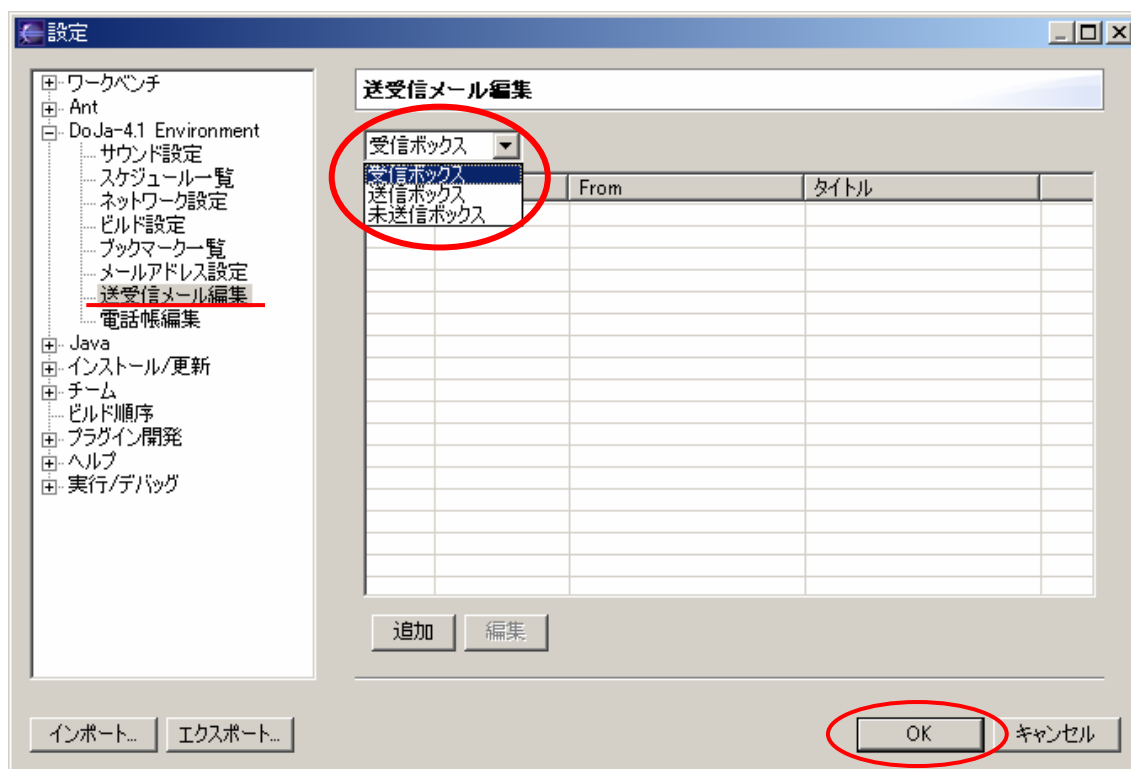


図 1 4-4 3 Eclipse 設定ウィンドウの送受信メール編集画面

3. 編集したいメールボックスを選択すると、一覧が切り替わります。
メールボックスの種類は、「受信ボックス」「送信ボックス」「未送信ボックス」の3種類があります。
4. メールを追加したい場合は「追加」、メールを編集したい場合は一覧から編集したいメールを選択して、「編集」ボタンを押します(図 1 4-4 4、図 1 4-4 5)。

メール編集(受信ボックス)

To: message_app_usr0@docomo.ne.jp

From: message_app_usr0@docomo.ne.jp

タイトル: おめでとう

受信日: 2002 年 12 月 01 日 10 時 11 分

MessageCode: 0000000000

ネイティブメール: ☐

状態: ☒ 既読 ☐ 未読

本文

お誕生日おめでとう。

設定 取消

図 1 4-4 4 受信メール編集画面

メール編集(送信ボックス)

To: message_app_usr0@docomo.ne.jp

From: message_app_usr0@docomo.ne.jp

タイトル: Re:おめでとう

送信日: 2002 年 12 月 01 日 10 時 22 分

MessageCode: 0000000000

ネイティブメール: ☐

宛先状態: ☒ XString ☐ 通常文字列

本文

ありがとう。

設定 取消

図 1 4-4 5 送信/未送信メール編集画面

5. 各設定項目を設定します。
6. 「OK」を押すと、送受信メール編集一覧に追加されます。

各設定項目の詳細については、「5.1 3. 送受信メールの編集」を参照してください。

14.4.9. スケジュール一覧

エミュレータのスケジュール一覧を編集することができます。編集するには、次の手順に従ってください。

1. 「ウィンドウ」→「設定」を選択します。
2. Eclipse設定ウィンドウが表示されるので、「DoJa-4.1 Environment」→「スケジュール一覧」を選択します（図 1 4-4 6）。

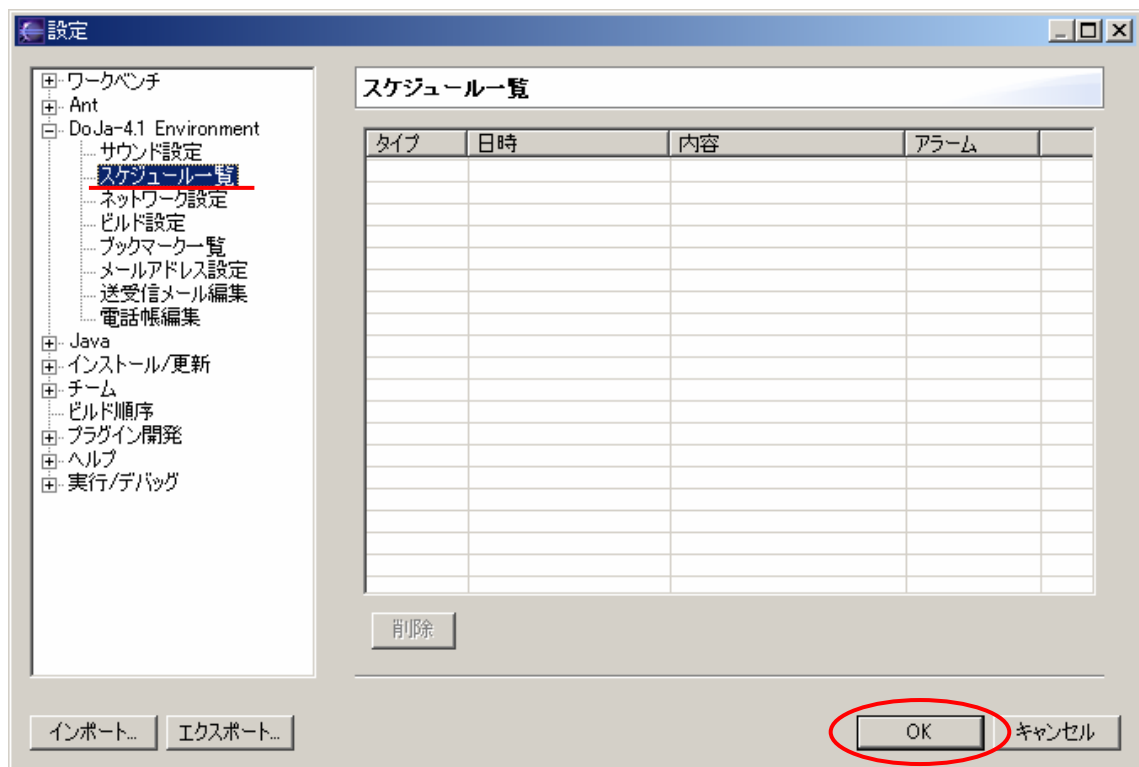


図 1 4-4 6 スケジュール一覧画面

3. スケジュールを削除するには、一覧からスケジュールを選択して「削除」を押します。スケジュールファイルが削除され、一覧からスケジュールが削除されます。

14.4.11. サウンド設定

サウンドの再生モードと代替音を設定することができます。設定するには、次の手順に従ってください。

1. 「ウィンドウ」→「設定」を選択します。
2. Eclipse設定ウィンドウが表示されるので、「DoJa-4.1 Environment」→「サウンド設定」を選択します（図 1 4-1 7）。



図 1 4-4 8. サウンド設定画面

3. 各設定項目を設定します。
デバイス設定 と 代替音設定 は、再生モードが「通常再生モード」の場合に有効です。
「通常再生モード」でのサウンド再生には、別途、**DirectX** が必要です。
4. 「OK」を押すと、設定されます。

各設定項目の詳細については、「5.6. サウンド設定」を参照してください。

14.5. 仮想 FeliCa データファイルビューワ

仮想 FeliCa データファイルビューワは、仮想 FeliCa データファイルを 16 進数でダンプします。

仮想 FeliCa データファイルビューワを使用すれば、アプリケーションから仮想 FeliCa データファイルにデータが適切に書き込まれているかどうかを確認する事ができます。

14.5.1. 仮想 FeliCa データファイルビューワの起動

1. メニューから「プロジェクト」→「DoJa-4.1」→「仮想 FeliCa データファイルビューワ…」を選択します。

「仮想FeliCaデータファイルビューワ」画面が表示されます（図 1 4-4 9）。

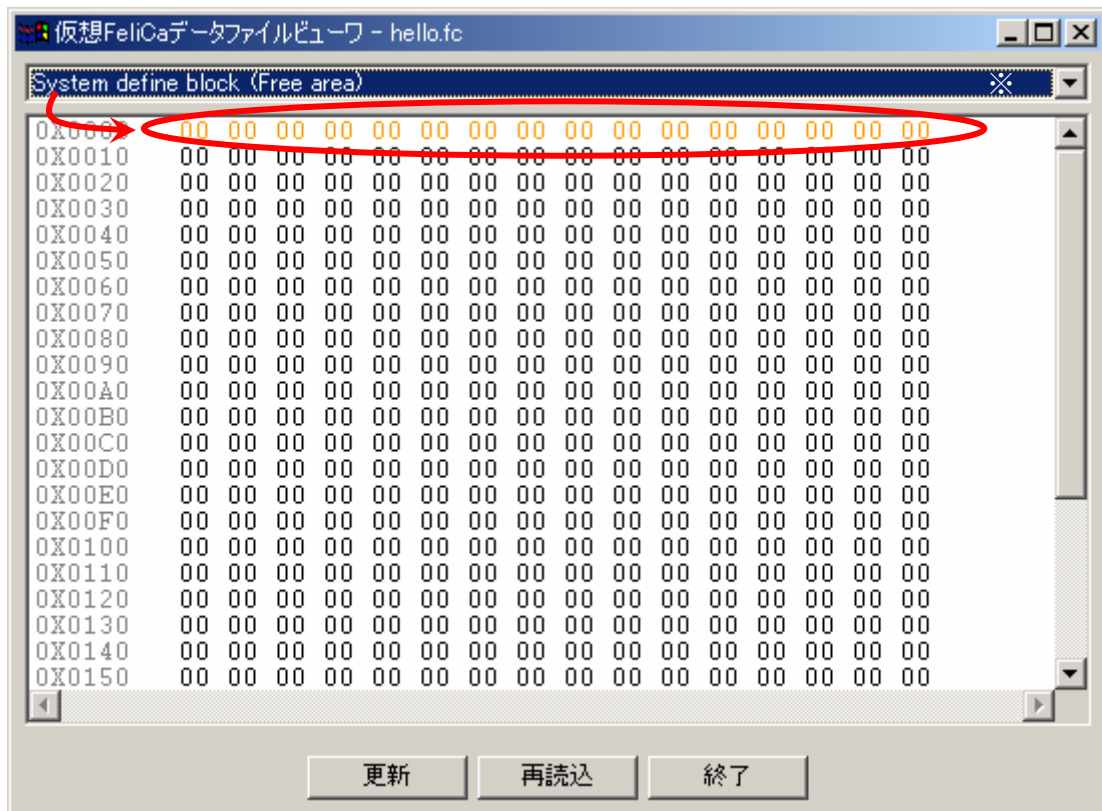
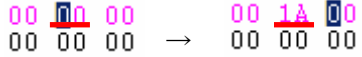


図 1 4-4 9 仮想 FeliCa データファイルビューワ画面

インストール直後のデータファイルは、データ領域を除き、適切な値が入っています。仮想FeliCaデータファイルの構造については、「9. 1 4. 仮想FeliCaデータファイル」を参照してください。

4. 現在読み込まれているプロジェクトの仮想 FeliCa データファイルの内容が 16 進数で表示されます。

5. 上段のプルダウン（※）から仮想 FeliCa データファイルの記憶領域を選択します。
該当する記憶領域のデータの色が変わります。
これにより、実際にアプリケーションから書き込まれた領域を参照する事ができます。
6. 値を変更したい場合は、変更したい所の値を選択し直接変更します。
仮想 FeliCa データファイルビューワでは、CPID Data Block, Data Block, PIN Data Block のみ値の変更が可能です。

- ※Area DefineBlock, Service Block 1, Service Block 2, PIN Service Block は変更しても無効です。
7. 「更新」を押すと、これまでの変更結果が仮想 FeliCa データファイルに書き出されます。
8. 「再読込」を押すと、仮想 FeliCa データファイルの内容を再読込みし、仮想 FeliCa データファイルビューワの表示時点もしくは「更新」を押した時点の状態に戻ります。
9. 「閉じる」を押すと仮想 FeliCa データファイルビューワを終了します。

仮想 FeliCa データファイルビューワで表示されたフリー領域を扱うアプリケーションを利用するには、次の条件が満たされている必要があります：

- CPID Data Blockが、ADFのFeliCaCPIDと同じ値でなければなりません（「4.7.アプリケーションディスクリプタファイル（ADF）の編集」参照）。
- PIN Data Blockの先頭4バイトが、ADFのFeliCaPINの値を16進数表現した値でなければなりません（「4.7.アプリケーションディスクリプタファイル（ADF）の編集」参照）。

設定手順は：

1. 仮想 FeliCa データファイルビューワの上段にあるプルダウンから CPID Data Block を選択します。
2. CPID を格納する場所が桃色で表示されるので、この部分にカーソルを当て、ADF に設定した FeliCaCPID と同じ値に変更します。
3. 仮想 FeliCa データファイルビューワの上段にあるプルダウンから PIN Data Block を選択します。
4. PIN を格納する場所が紫色で表示されるので、この部分にカーソルを当て、先頭4バイトに ADF に設定した FeliCaPIN の値を16進数表現で設定します。例えば、ADF に設定した FeliCaPIN の値が 12345678 の場合は、PIN Data Block の先頭4バイトを 00BC614E に変更します。
5. 仮想 FeliCa データファイルビューワの「更新」ボタンを押します。
6. 「閉じる」ボタンを押します。

【注意事項】

- 仮想 FeliCa データファイルビューワを表示している状態で、アプリケーションから仮想 FeliCa データファイルヘデータを書き込んだ場合は、「再読込」ボタンを押してください。
- アプリケーションが仮想 FeliCa データファイルにアクセスしている状態で、仮想 FeliCa データファイルを編集した場合、アプリケーションで例外が発生する可能性があります。
- フリー領域を利用するアプリケーションで、ADF の FeliCaCPID、FeliCaPIN の値を設定して、仮想 FeliCa データファイルを作成しても、フリー領域の CPID Data Block、PIN Data Block には自動的に反映されません。

14.5.2. 仮想 FeliCa データファイルビューワの表示色

仮想 FeliCa データファイルビューワの表示で、ウィンドウ内の色は下記のようになっています。

橙 : SystemDefineBlock
 緑 : AreaDefineBlock
 赤 : Service Block
 水色 : PIN Service Block
 桃 : CPID Data Block および Data Block
 紫 : PIN Data Block

【F 1 2】を押すと全てを色分けして表示します。

14.5.3. ブロックの構成

仮想 FeliCa データファイルビューワでは各ブロックを下記のような構成になっています。

1 ブロックは 16 バイトで、以下のようなバイト列になっています。

D0	D1	D2	D3	・	・	・	・	・	・	D13	D14	D15
----	----	----	----	---	---	---	---	---	---	-----	-----	-----

D = バイト

- SystemDefineBlock
D0～D1 : System Code
- AreaDefineBlock
D0～D1 : Area Code
D2～D3 : End Service Code
- Service Block
D0～D1 : Service Code

仮想 FeliCa データファイルビューワで編集を行う場合のエンディアン表記については、全てビッグエンディアンになっています。

14.6. スクラッチパッド

14.6.1. スクラッチパッド管理ツール

スクラッチパッド管理ツールは、アプリケーションのスクラッチパッドファイルを 16 進数でダンプして編集するためのツールです。

スクラッチパッド管理ツールを使用すれば、スクラッチパッドのデータを作成したり、アプリケーションからスクラッチパッドにデータが適切に書込まれているかどうかを確認したりすることができます。

Eclipse のスクラッチパッド管理ツールは、複数のスクラッチパッドを 1 つの領域で表示します。各スクラッチパッドで編集したい場合は、*ia ppliTool* をご利用ください。

14.6.2. スクラッチパッド管理ツールの起動

1. メニューから「プロジェクト」→「DoJa-4.1」→「スクラッチパッド...」を選択します。
2. 「スクラッチパッド管理ツール」画面が表示されます（図 1 4-5 0）。

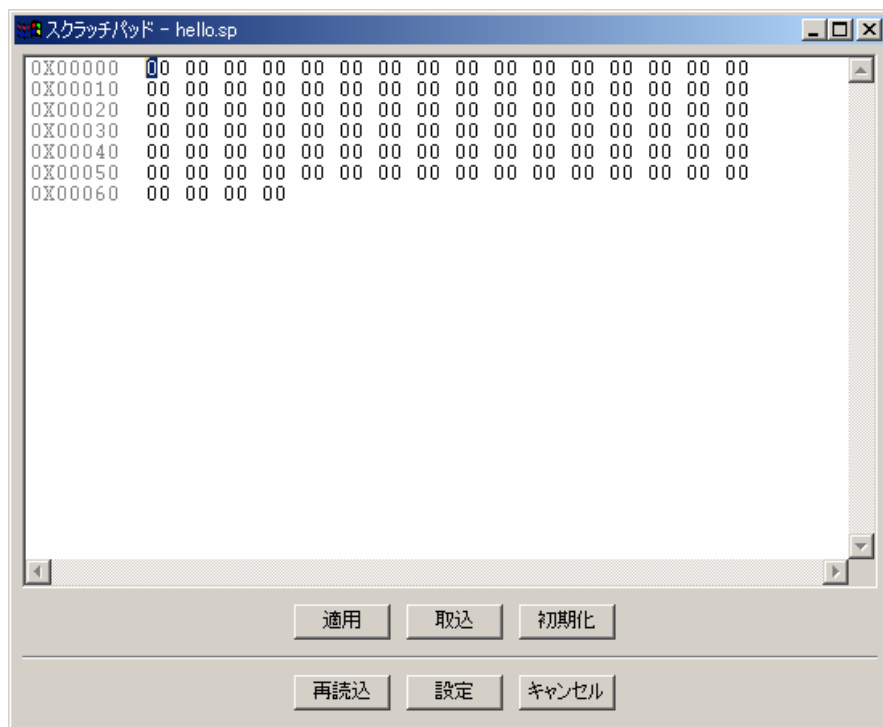


図 1 4-5 0 スクラッチパッド管理ツール画面

3. 現在、読み込まれているプロジェクトのスクラッチパッドの内容が、16 進数で表示されます。

4. 「適用」を押すと、これまでの変更結果をスクラッチパッドファイルに書き込みます。
5. 「取込」は、カーソルの位置から外部のファイルを取り込むことができます。
まず、取り込みたい位置を選択します（図 1 4-5 1）。

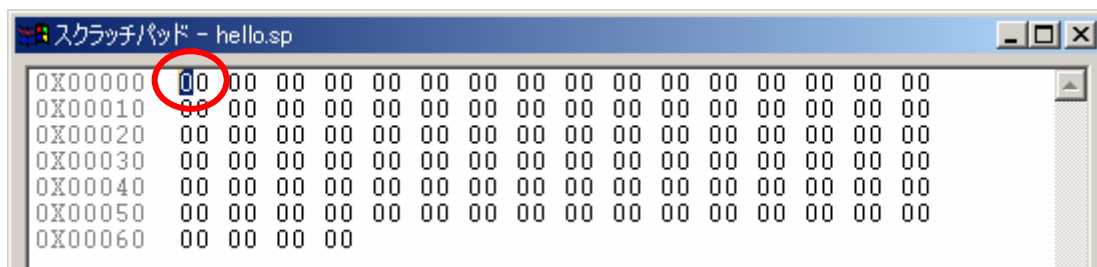


図 1 4-5 1 スクラッチパッド 置換開始位置選択

「取込」を押すと、「スクラッチパッドファイル選択」ダイアログが表示されるので、取り込みたいファイルを選択し、「開く」を押します（図 1 4-5 2）。

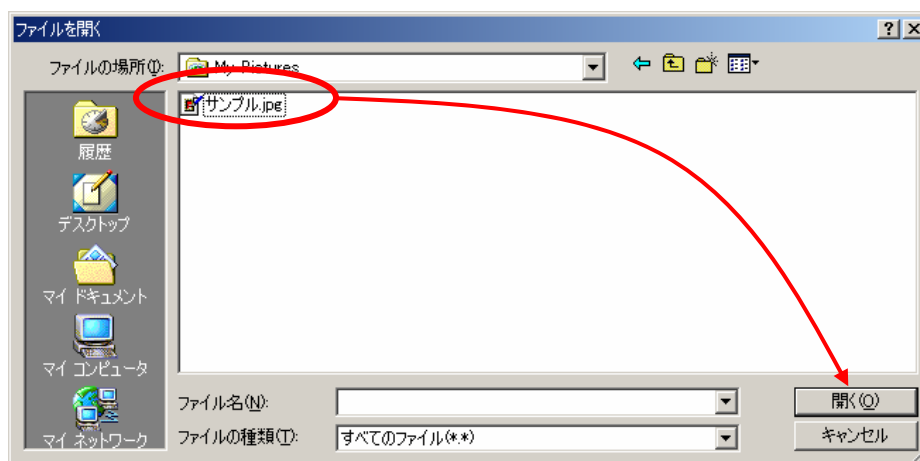


図 1 4-5 2 置換ファイル選択ダイアログ

カーソルの位置から選択したファイルの内容が取り込まれます（図 1 4-5 3）。
ファイルのサイズがスクラッチパッドのサイズを超えた場合は途中で切り捨てられ警告されます。

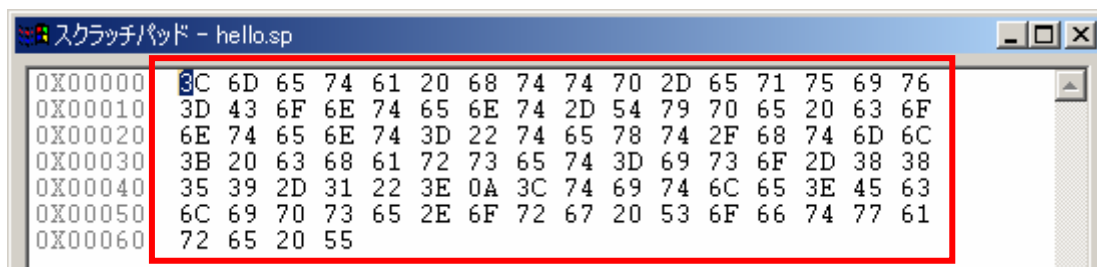


図 1 4-5 3 置換後のスクラッチパッド

6. 「初期化」を押すと、スクラッチパッドの表示内容を0で初期化します。
7. 「再読込」を押すと、スクラッチパッドファイルの内容を再読込みし、スクラッチパッド管理ツールの表示時点もしくは、「適用」を押した時点の状態に戻ります。
8. 「設定」を押すと、これまでの変更結果をスクラッチパッドファイルに書き込んで終了します。
9. 「キャンセル」を押すと、スクラッチパッドファイルは変更せずに終了します。

14.6.3. スクラッチパッドを全て初期化する

メニューから「プロジェクト」→「DoJa-4.1」→「スクラッチパッドを全て初期化」を選択することで、現在読込まれているプロジェクトのスクラッチパッドを全て削除することができます。スクラッチパッドが存在しない場合、「スクラッチパッドがありません」というダイアログが表示されます。

14.6.4. スクラッチパッドでJarファイルを扱う場合の注意事項

スクラッチパッドにJarを保存してアプリケーションで扱う場合、JarファイルのサイズをURLのlengthで指定するか、スクラッチパッドのサイズ（ADFのSPsize（「4.7. アプリケーションディスクリプタファイル（ADF）の編集」参照。））とJarファイルのサイズが同じでないとフォーマットエラーになります。

付録.1 エミュレーション端末の設定

エミュレーション端末設定ファイルを用意することで、任意の端末画像を使ってエミュレータを実行することが可能になります。エミュレーション端末設定ファイルは、"`lib\skin\端末名\skin.conf`"です。

1. コメント行

先頭に `#` がある行はコメント行として無視されます。

2. エミュレータウィンドウの設定

```
screen_width=[エミュレータウィンドウの幅]
screen_height=[エミュレータウィンドウの高さ]
default_image=[スキン画像ファイル名]
pressed_image=[ボタンが押された状態のスキン画像ファイル名]
```

通常エミュレータウィンドウの幅と高さはスキン画像の幅と高さに合わせます。

3. LCD 領域の設定

```
lcd_position=[LCD 位置の X 座標],[LCD 位置の Y 座標]
LCD の位置はエミュレータウィンドウの左上を原点とする。
lcd_width=[LCDの幅][10]
lcd_height=[LCDの高さ]
lcd_system_height=[LCD のシステムアイコン表示領域の高さ]
lcd_system_width=[アンテナアイコンの幅],[バックライトマークの幅],
                  [バイブレータマークの幅],[スクロールマークの幅],
                  [i アプリマークの幅]
lcd_color=[RED],[GREEN],[BLUE]
system_image=[システムアイコン画像ファイル名]
num_cameras=[カメラデバイスの数]
max_image_length=[撮影できる動画の最大サイズ]
```

^[10] `Display.getWidth()`は、`lcd_width` の値と同じになります。

`Display.getHeight()`は、`lcd_height` の値から `lcd_system_height` の値とソフトウェア領域の高さ（デフォルトフォントの高さ）を引いた値になります。

4. フォントの設定

`font_size`=[タイニーフォントサイズ],[スモールフォントサイズ],
[ミディアムフォントサイズ],[ラージフォントサイズ]

`font_size`を指定しない場合のフォントサイズは、12, 16, 24, 28 になります。

注意：実際にエミュレータで利用されるフォントサイズはエミュレータの実行環境に依存します。

5. ボタン設定

- システムボタンの設定

`system_button`=[ボタン番号],[ボタン位置の X 座標],[ボタン位置の Y 座標],
[ボタンの幅],[ボタンの高さ]

- 通常ボタンの設定

`button`=[ボタン番号],[ボタン位置の X 座標],[ボタン位置の Y 座標],
[ボタンの幅],[ボタンの高さ]

ボタンの位置はエミュレータウィンドウの左上を原点とします。

システムボタンのボタン番号は

- 1 --- 通話開始ボタン
- 2 --- クリアボタン
- 3 --- 通話終了ボタン

通常ボタンのボタン番号は DoJa のキー番号に対応しています。

6. ピクチャーライトの設定

SH用 i アプリオプション API `PictureLight` の表示領域を指定できます。

`picture_light_position`=[ライト位置の X 座標],[ライト位置の Y 座標]
`picture_light_width`=[ライトの幅]
`picture_light_height`=[ライトの高さ]
`picture_light_color`=[RED],[GREEN],[BLUE]

7. サウンドライブラリの設定

サウンドライブラリのプロパティを指定できます。

`sound_param_audio3d_resources` = [3D仮想音源数]

指定値：0 ～ 8

未指定：4

`sound_param_adpcm_resources` = [ADPCM同時発音数]

指定値：0 ～ 4

未指定：4

`sound_param_block_num` = [ブロック数(ストリーミングバッファサイズ)]

指定値：制限なし

未指定：2

`sound_param_wait_time` = [DryTrackの3DTrackに対する遅延調整時間]

指定値：0 ～ 500

未指定：0

※ サウンドライブラリの設定値を指定する場合は、新たにファイルに追加してください。

8. 例(デフォルト)

```
# skin.conf
# Unknown key KEY_F00KUP
# Unknown key KEY_F00KDOWN
screen_width = 300
screen_height = 610
default_image = device1-off.jpg
pressed_image = device1-on.jpg
system_image = system.gif
warning_image = warn.gif
error_image = error.gif
info_image = info.gif
lcd_color = 186, 231, 217
lcd_position = 27, 22
lcd_width = 240
lcd_height = 273
lcd_system_height = 15
lcd_system_width = 36, 12, 13, 11, 20, 25, 25, 17, 36, 15, 15, 16, 18
max_image_length = 102400
system_button = 1, 71, 425, 47, 20
button = 32, 123, 431, 48, 17
system_button = 3, 176, 425, 48, 20
button = 9, 176, 488, 48, 20
button = 21, 71, 334, 47, 27
button = 19, 124, 395, 46, 12
button = 39, 125, 410, 44, 15
button = 22, 176, 334, 47, 27
button = 16, 106, 367, 16, 25
button = 0, 123, 515, 48, 17
button = 10, 71, 509, 47, 20
button = 1, 71, 446, 47, 20
button = 40, 82, 534, 32, 19
button = 24, 124, 335, 46, 15
button = 2, 123, 452, 48, 17
button = 33, 71, 402, 47, 22
button = 11, 176, 509, 48, 20
button = 3, 176, 446, 48, 20
button = 34, 181, 534, 32, 19
button = 18, 172, 367, 16, 25
button = 4, 71, 467, 47, 20
button = 37, 176, 402, 48, 22
button = 5, 123, 473, 48, 17
button = 6, 176, 467, 48, 20
button = 17, 124, 352, 46, 12
button = 7, 71, 488, 47, 20
button = 20, 126, 369, 43, 22
button = 8, 123, 494, 48, 17
picture_light_position = 250, 500
picture_light_width = 30
picture_light_height = 30
picture_light_color = 100, 100, 100
sub_button = 0, 15, 586, 36, 20
sub_button = 1, 3, 553, 24, 24
```

付録.2 端末子ウィンドウの設定

端末子ウィンドウ設定ファイルを用意することで、任意の端末子ウィンドウ画像を使ってエミュレータを実行することが可能になります。端末子ウィンドウ設定ファイルは、`"lib\skin\端末名\subwindow.conf"` です。

1. コメント行

先頭に # がある行はコメント行として無視されます。

2. 端末子ウィンドウの設定

```
default_image = [スキン画像ファイル名]
pressed_image = [ボタンが押された状態のスキン画像ファイル名]
subwindow_position = [子ウィンドウの X 座標], [子ウィンドウの Y 座標]
radiobutton_position = [ 2 択状態設定の X 座標], [ 2 択状態設定の Y 座標],
                        [ 2 択状態設定の幅], [ 2 択状態設定の高さ]
combobox_position = [状態設定の X 座標], [状態設定の Y 座標],
                    [状態設定の幅], [状態設定の高さ]
skin_position = [スキン画像の X 座標], [スキン画像の Y 座標],
                [スキン画像の幅], [スキン画像の高さ]
```

通常エミュレータウィンドウの幅と高さはスキン画像の幅と高さに合わせます。

3. ボタン設定

```
button = [ボタン番号], [ボタン位置の X 座標], [ボタン位置の Y 座標],
         [ボタンの幅], [ボタンの高さ]
```

ボタン番号は、エミュレータウィンドウと共通です。

4. 例(デフォルト)

```
# subwindow.conf
# Unknown key KEY_F00KUP
# Unknown key KEY_F00KDOWN
default_image = subwindow-off.jpg
pressed_image = subwindow-on.jpg
subwindow_position = 15, 349
radiobutton_position = 4, 0, 296, 120
combobox_position = 4, 120, 296, 140
skin_position = 300, 0, 170, 260
button = 48, 33, 20, 38, 18
button = 49, 97, 20, 38, 18
button = 50, 19, 69, 38, 18
button = 51, 65, 69, 38, 18
button = 52, 111, 69, 38, 18
button = 38, 19, 117, 38, 18
button = 41, 65, 117, 38, 18
button = 42, 111, 117, 38, 18
button = 36, 19, 145, 38, 18
button = 53, 65, 145, 38, 18
button = 35, 111, 145, 38, 18
button = 57, 19, 193, 38, 18
button = 56, 65, 193, 38, 18
button = 58, 111, 193, 38, 18
button = 60, 19, 220, 38, 18
button = 61, 65, 220, 38, 18
button = 59, 111, 220, 38, 18
```