



DoJa-3.0 API i α ppli Development Kit ユーザーズガイド

平成 16 年 9 月 2 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	-	2003年4月 制定 <505i 対応 完全実装版 - DoJa-3.0>
1.10	4.16	追加	改版内容	画面状態「可視／不可視」を追加しました。
			改版理由	505i S 向け
1.10	4.16	追加	改版内容	端末ウィンドウに F 用 i アプリオプション API 「指紋センサ・ナビゲーション」を追加しました。
			改版理由	505i S 向け
1.10	5.1 5.2	変更	改版内容	プロジェクト新規作成、プロジェクト読み込み時に、今まで apps 固定だったプロジェクトの格納先を選択できるように しました。 ただし、メーラ起動、ブラウザ起動、コードリーダー 起動、i アプリ連携は、 apps 以外のプロジェクトから 起動させる事はできません。
			改版理由	エミュレーションの機能性向上に伴う追加
1.10	5.3	変更	改版内容	ADFの説明文を変更しました。
			改版理由	505i S 向け
1.10	5.19	追加	改版内容	コードリーダー起動を追加しました。
			改版理由	505i S 向け
1.10	5.20	追加	改版内容	状態遷移にコードリーダー起動を追加しました。
			改版理由	505i S 向け
1.10	8.8	追加	改版内容	コード認識起動ファイルの設定内容を追加しました。
			改版理由	505i S 向け
1.10	付録.17	追加	改版内容	指紋センサ・ナビゲーションの設定を追加しました。
			改版理由	505i S 向け
1.20	4.16.1	追加	改版内容	端末子ウィンドウを追加しました。
			改版理由	エミュレーションの機能性向上に伴う追加
1.20	4.10 4.12	追加	改版内容	位置情報編集機能、測位データ編集機能を追加しまし た。
			改版理由	GPS搭載移動機端末 向け
1.20	4.16.1	追加	改版内容	端末子ウィンドウの操作を追加しました。
			改版理由	エミュレーションの機能性向上に伴う追加
1.20	4.3	追加	改版内容	トラステッド動作設定に「位置情報取得APIの許可」を追 加しました。
			改版理由	GPS搭載移動機端末 向け
1.20	5.5.1 5.5.2 5.5.3	追加	改版内容	ソースコードの作成機能、ソースコードの修正機能、 ソースエディタの設定機能を追加しました。
			改版理由	エミュレーションの機能性向上に伴う追加
1.20	5.11 5.12	追加	改版内容	位置情報起動、位置情報画像起動 を追加しました。
			改版理由	GPS搭載移動機端末 向け
1.20	5.15 5.17 5.18	追加	改版内容	外部機器起動、メーラ起動、ブラウザ起動に位置情報 利用確認画面 を追加しました。
			改版理由	GPS搭載移動機端末 向け
1.20	8.9 8.11	追加	改版内容	位置情報設定ファイル、測位データ設定ファイル を追 加しました。
			改版理由	GPS搭載移動機端末 向け
1.20	付録.2	追加	改版内容	端末子ウィンドウの設定を追加しました。
			改版理由	エミュレーションの機能性向上に伴う追加

1.30	5.3 5.6	追加	改版内容	ビルド時に AppName, AppClass が未設定であれば、自動的に設定する機能を追加しました。
			改版理由	エミュレーションの機能性向上に伴う追加
1.30	4.2 4.9.2	追加	改版内容	メールアドレスの設定画面を複数設定できるように変更し、メールの送受信をエミュレートする機能を追加しました。
			改版理由	エミュレーションの機能性向上に伴う追加
1.30	8.5	追加	改版内容	新たに useChat を追加して、i アプリオプシオンAPI のチャット機能の有効／無効を切り替えられるようにしました。 チャット機能が有効の場合、IApplication#LAUNCH_CHAT が使え、LAUNCH_SCHEDULER が使えなくなります。
			改版理由	5 0 5 i S 向け (追加)
1.40	-	-	改版内容	API リファレンスの変更に基づく修正版
			改版理由	5 0 5 i S 向け
1.40	8	変更	改版内容	MyPicture を multimedia に変更しました。
			改版理由	5 0 5 i S 向け
1.50		変更	改版内容	GPS 機能に測地系計算ライブラリを搭載しました。
			改版理由	GPS 搭載移動機端末 向け
1.50	4.11 8.10	追加	改版内容	位置情報画像編集、位置情報画像設定ファイルを追加しました。
			改版理由	GPS 搭載移動機端末 向け
1.50	5.12	変更	改版内容	位置情報画像起動の確認画面で表示する項目を変更しました。
			改版理由	GPS 搭載移動機端末 向け
1.50	-	変更	改版内容	HTTP ブラウザの不具合を修正しました。
			改版理由	5 0 5 i S 向け
2.00	2 14	追加	改版内容	Eclipse2.1.X プラグイン - デバッガ対応 搭載。
			改版理由	アプリの開発効率向上のため < 5 0 5 i S 対応 完全実装版 >
2.01	-	変更	改版内容	JBuilder 向けの実装を施しました。
			改版理由	ポーランド社 製品 JBuilder10 対応
3.00	全体	変更	改版内容	iαppliTool 画面の張り替え。
			改版理由	5 0 6 i C 向け対応、エミュレータの機能向上により画面デザインを変更したため
3.00	4.2	追加	改版内容	ダウンロードアプリケーション設定機能を追加しました。
			改版理由	エミュレーションの機能向上に伴う追加
3.00	4.5 4.6	追加	改版内容	リモート制御設定機能を追加しました。
			改版理由	エミュレーションの機能向上に伴う追加
3.00	4.13	追加	改版内容	ヘルプの設定機能を追加しました。
			改版理由	エミュレータ機能向上に伴う追加
3.00	5.4	追加	改版内容	プロジェクト設定機能を追加しました。
			改版理由	エミュレーションの機能向上に伴う追加
3.00	4.16.1①	追加	改版内容	端末初期状態設定機能を追加しました。
			改版理由	エミュレーションの機能向上に伴う追加
3.00	5.3	追加	改版内容	ADF にリモート制御と FeliCa の項目を追加しました。
			改版理由	5 0 6 i、5 0 6 i C 向け。

3.00	1.1.1	変更	改版内容	起動種別の選択をラジオボタンからプルダウンに変更しました。
			改版理由	エミュレータ機能向上に伴う変更
3.11	-	変更	改版内容	Graphics#fillpolygon を paint() 内で実行した場合にメモリ開放漏れが発生する不具合を修正しました。
			改版理由	不具合対応
3.20	9	追加	改版内容	仮想 FeliCa データファイルビューワ機能を追加しました。
			改版理由	5 0 6 i C 向け
3.21	-	追加	改版内容	仮想 FeliCa 機能強化、アプリ連携不具合修正
			改版理由	5 0 6 i C 向け
3.22	-	変更	改版内容	KToolBar メニューの表示フォント修正
			改版理由	不具合修正

目 次

1. はじめに	7
1.1. 「iappli Development Kit for DoJa-3.0」の開発環境	7
1.1.1. 「iappliTool」開発環境	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	9
1.3. 「iappli Development Kit for DoJa-3.0」によるアプリケーションの開発	12
2. 「iappli Development Kit for DoJa-3.0」のインストール	14
2.1. インストールする前に	14
2.2. インストールの手順	14
2.3. アンインストール手順	21
3. 「iappliTool」の起動	22
4. エミュレータの設定	24
4.1. ネットワークの設定	24
4.2. ダウンロードアプリケーションの設定	27
4.3. トラステッド動作の設定	28
4.4. メールアドレスの設定	30
4.5. リモート制御の設定	31
4.6. リモート制御のパラメータ領域の削除	32
4.7. 電話帳の編集	33
4.8. グループの編集	35
4.9. 送受信メールの編集	36
4.9.1. ショートメールの扱いについて	39
4.9.2. メールを送受信エミュレート機能	41
4.10. 位置情報編集	42
4.11. 位置情報画像編集	43
4.12. 測位データ編集	44
4.13. ヘルプの設定	45
4.14. プロキシサーバの設定	46
4.15. エミュレーション端末の選択	46
4.16. エミュレーション端末の操作	48
4.16.1. 端末子ウィンドウの操作	51

4.17. コンパイラ (javac) の選択	53
5. プロジェクト	54
5.1. プロジェクトの作成から起動まで	54
5.2. 既存プロジェクトの読み込み	55
5.3. アプリケーションディスクリプタファイル (ADF) の編集	57
5.3.1. LastModified 項目の表記方法	67
5.4. プロジェクトの設定	68
5.5. プロジェクトの新規作成	70
5.5.1. ソースコードの作成	72
5.5.2. ソースコードの修正	73
5.5.3. ソースエディタの設定	75
5.6. プロジェクトのビルドと通常起動	76
5.7. 待ち受け起動	79
5.7.1. 待ち受けアプリケーション	79
5.7.2. 待ち受け起動手順	81
5.8. 受信フォルダ起動	84
5.9. 送信フォルダ起動	85
5.10. 未送信フォルダ起動	86
5.11. 位置情報起動	87
5.12. 位置情報画像起動	89
5.13. 初回起動	91
5.14. タイマ起動	92
5.15. 外部機器起動	93
5.16. 即起動	95
5.16.1. 即起動手順	95
5.17. メーラ起動	96
5.17.1. メーラ起動手順	96
5.18. ブラウザ起動	101
5.18.1. ブラウザ起動手順	101
5.19. コードリーダー起動	107
5.19.1. コードリーダー起動手順	107
5.19.2. CodeReader の実行手順	111
5.20. 各種起動方法による画面遷移	112
6. コンソールと「iappliTool」の終了	114
6.1. デバッグトレースの利用	114
7. i アプリ連携	116

8. ファイルの記述方法	118
8.1. 「*.mail」 ファイルの記述方法	119
8.2. 「HTML」 ファイルの記述方法	119
8.3. 「JAM.properties」 ファイルの記述方法	120
8.4. 着信履歴ファイル	120
8.5. 「アプリケーション名.properties」 ファイル	121
8.6. メールファイル	121
8.7. マルチメディアファイル	122
8.8. コード認識結果ファイル	122
8.9. 位置情報設定ファイル	125
8.10. 位置情報画像設定ファイル	125
8.11. 測位データ設定ファイル	126
8.12. 仮想 FeliCa データファイル	127
8.13. ファイルの記述具体例	129
9. 仮想 FeliCa データファイルビューワ	130
9.1. 仮想 FeliCa データファイルビューワの表示色	132
9.2. ブロックの構成について	132
10. スクラッチパッド管理	133
10.1. スクラッチパッド管理ツール	133
10.2. スクラッチパッドファイル	133
10.3. スクラッチパッド管理ツールの起動	133
10.4. スクラッチパッドの内容表示	134
10.5. スクラッチパッドの削除	135
10.6. スクラッチパッド管理ツールの終了	135
10.7. 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	135
11. 証明書の管理	136
12. OBEX の利用	138
13. Sun ONE Studio 4, Community Edition の利用	139
13.1. はじめに	139
13.1.1. Sun ONE Studio 4, Community Edition について	139
13.2. 「Sun ONE Studio Module for iappli」 のインストール	139
13.2.1. 始める前に	139
13.2.2. 「Sun ONE Studio Module for iappli」 のインストール	140

13.3. 新規アプリケーションの作成	142
13.3.1. パッケージの作成	142
13.3.2. アプリケーション・ディスクリプタ・ファイル (ADF) の作成	143
13.3.3. ソースファイルの作成	146
13.3.4. アプリケーション・ディスクリプタ・ファイル (ADF) の設定	149
13.3.5. トラステッド動作の設定	151
13.3.6. jar ファイルの内容の設定	152
13.3.7. アプリケーションの構築と実行	154
13.3.8. デバッグの実行	155
13.4. エミュレータの設定	162
13.4.1. デバイスの設定	162
13.4.2. プロキシサーバの設定	166
13.4.3. ログメッセージのトレース	166
14. Eclipse の利用	167
14.1. はじめに	167
14.1.1. Eclipse について	167
14.2. 「Eclipse Module for iappli」 のインストール	167
14.2.1. 始める前に	167
14.2.2. 「Eclipse Module for iappli」 のインストール	168
14.3. 新規アプリケーションの作成	169
14.3.1. DoJa を利用する前に	169
14.3.2. プロジェクトの作成	170
14.3.3. ソースファイルの作成	173
14.3.4. プロジェクトメニューのカスタマイズ	175
14.3.5. アプリケーション・ディスクリプタ・ファイル (ADF) 設定	176
14.3.6. トラステッド動作の設定	177
14.3.7. アプリケーションの構築と実行	178
14.3.8. デバッグの実行	180
14.4. エミュレータの設定	188
14.4.1. ネットワークの設定	188
付録.1 エミュレーション端末の設定	189
1. コメント行	189
2. エミュレータウィンドウの設定	189
3. LCD 領域の設定	189
4. フォントの設定	190
5. ボタン設定	190
6. ピクチャーライトの設定	190
7. 指紋センサ・ナビゲーションの設定	190
8. 例 (デフォルト)	191
付録.2 端末子ウィンドウの設定	192

1. コメント行.....	192
2. 端末子ウィンドウの設定.....	192
3. ボタン設定.....	192
4. 例（デフォルト）	193

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

Sun、Sun Microsystems、Java、J2SE SDK、Sun ONE Studio、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 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。

「i モード」、「i アプリ／アイアプリ」、「i-α ppli」ロゴ、「DoJa」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 2000-2004 NTT DoCoMo, Inc. 2-11-1 Nagata-cho, Chiyoda-ku, Tokyo, 100-6150, Japan.

1. はじめに

このマニュアルでは、「iα ppli Development Kit for DoJa-3.0」のインストール、設定及び使用方法について説明します。

このマニュアルで述べる「iα ppli Development Kit for DoJa-3.0」は、Java 対応 i モード携帯電話で動作する Java アプリケーションの開発をサポートします。

このマニュアルを理解するには、Java プログラミング、CLDC、および NTT ドコモが提供する DoJa-3.0 API の知識が必要です。

本章では、「iα ppli Development Kit for DoJa-3.0」についての概要を説明します。
本章の内容は、次の通りです:

- ① 「iα ppli Development Kit for DoJa-3.0」の開発環境
- ② エミュレータの概要
- ③ 「iα ppli Development Kit for DoJa-3.0」によるアプリケーションの開発

1.1. 「iα ppli Development Kit for DoJa-3.0」の開発環境

「iα ppli Development Kit for DoJa-3.0」は、以下の環境での動作を保証します。

- ・ Windows 2000 (SP4) / Windows XP (SP1)
- ・ Java 2 SDK, Standard Edition v 1.3.1 (J2SE SDK)

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. 「iαppliTool」開発環境

「iαppliTool」開発環境の画面は、図 1-1 のように表示されます。

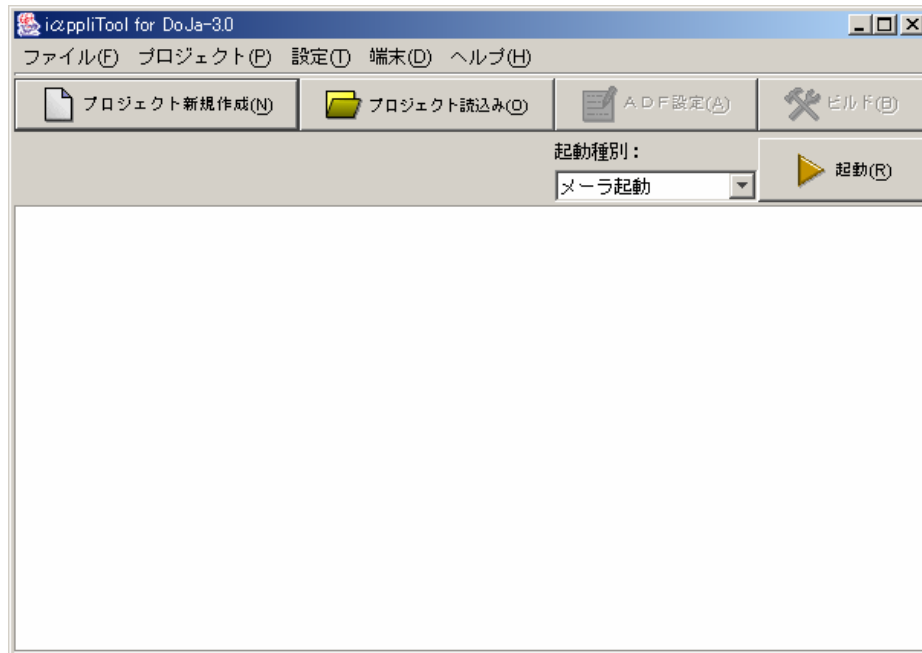


図 1-1 「iαppliTool」画面

「iαppliTool」は、次の機能を備えた最小限の開発環境です。

- ビルド
 - コンパイル
 - バイトコードの検証
 - パッケージング
- アプリケーションディスクリプタファイルの作成
- テスト（エミュレーション）

アプリケーションファイルは、エミュレータでのテストが完了したら、Web サイトにアップロードして、実際のデバイスでテストします。

開発サイクルについては、「1.3. 「iαppli Development Kit for DoJa-3.0」によるアプリケーションの開発」を参照してください。

1.2. エミュレータの概要

エミュレータは、ターゲットデバイスの多様なエミュレーションを PC 上で行うことによってアプリケーション開発をサポートします。エミュレータを使用すれば、ターゲットデバイスで行われるアプリケーションの動作を PC 上で検証できます。

1.2.1. エミュレータの利点

エミュレータの利点には、次のものがあります：

- 開発に使用したのと同じ PC プラットフォームでアプリケーションをテストできます。
- 実際のターゲットデバイスは、最終的なテストのときにしか必要ありません。
- 多様なターゲットデバイスでのアプリケーションの実行を同じ PC 上からエミュレートできます。

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、実行の機能が統合されており、ユーザの生産性を向上させることができます。

「iα ppli Development Kit」は、Sun ONE Studio 4, Community Edition および Eclipse2.1.2 の統合開発環境でアプリケーション開発サイクルを行う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は、アプリケーションの編集、コンパイル、ビルド、及びエミュレータによる実行とデバッグを行うことができます。

詳細については「13. Sun ONE Studio 4, Community Edition の利用」もしくは、「14 Eclipse の利用」を参照してください。

- Sun ONE Studio 4 : <http://jp.sun.com/software/sundev/jde41/>
- Eclipse2.1.2 : <http://www.eclipse.org/>

1.2.3. エミュレーションの範囲

エミュレータをテストの目的で使用するには、エミュレータがエミュレーションできる範囲を理解しておくことが大切です。

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、実際の電話機の上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。例えば、アプリケーションの更新は、エミュレータが実行しない処理の 1 つです。

アプリケーションマネージャに関する動作のうち実際の電話機と異なる動作として以下があります。

- 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。
- アプリケーションからマルチメディアに保存する事はできません。
- エミュレータでは、Win32API を用いているため、実際の電話機とは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は行っています。従って、アプリケーションとして特に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「5.3. 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、携帯電話固有の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、携帯電話固有の多くの機能（待ち受け設定やダイヤルロックなど）はエミュレートされません。

携帯電話固有の機能に関して、実際の電話機と異なる動作として以下があります。

- 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。
- 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。

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、電話機に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ません。

待ち受けアプリケーションに関して、実際の電話機と異なる動作として以下があります。

- 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電話機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としての起動はエミュレータの起動オプションにより指定します。
- 待ち受けアプリケーションが休眠状態の際の省電力機能はエミュレートしません。

5. ネイティブアプリケーション連携

エミュレータでは、ネイティブアプリケーション連携のうちブラウザ、メーラ、音声通話の各機能との連携をエミュレートする環境を提供しますが、外部機器連携をエミュレートする環境は提供しません。

ネイティブアプリケーション連携に関して、実際の電話機と異なる動作として以下があります。

- 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。
- アプリケーションからブラウザを起動する API (IApplication クラスの launch()) を呼び出した場合、確認ダイアログの表示まで行われますが、ブラウザの起動は行いません。
- 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。
- カメラ撮影、動画撮影をする機能はありません。

6. アプリケーション連携

エミュレータではアプリケーションの連携ができる機能を提供します。

7. 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。

8. 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. 「iα ppli Development Kit for DoJa-3.0」によるアプリケーションの開発

本節では、上述の「iα ppliTool」と Sun ONE Studio 4 開発環境でサポートされる開発サイクルについて説明します。

「iα ppli Development Kit for DoJa-3.0」でサポートする開発サイクルには、次のフェーズがあります：

1. テキストエディタを使用して.java ファイルを編集する（「iα ppli Development Kit for DoJa-3.0」の範囲には含まれません）。
2. ビルド
 - 2.1. 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。
 - 2.2. 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行うと共に、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検査します。
 - 2.3. 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アプリケーションリソースファイルを JAR ファイルとしてパッケージングします。
 - 2.4. アプリケーションディスクリプタファイル (ADF)
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、アプリケーション情報を記述するテキストファイルです。
3. テスト（エミュレーション）
異なるターゲットデバイスに対してエミュレータを使用してアプリケーションを実行します。

上述の開発サイクルを図示すると、図 1-2 のようになります。

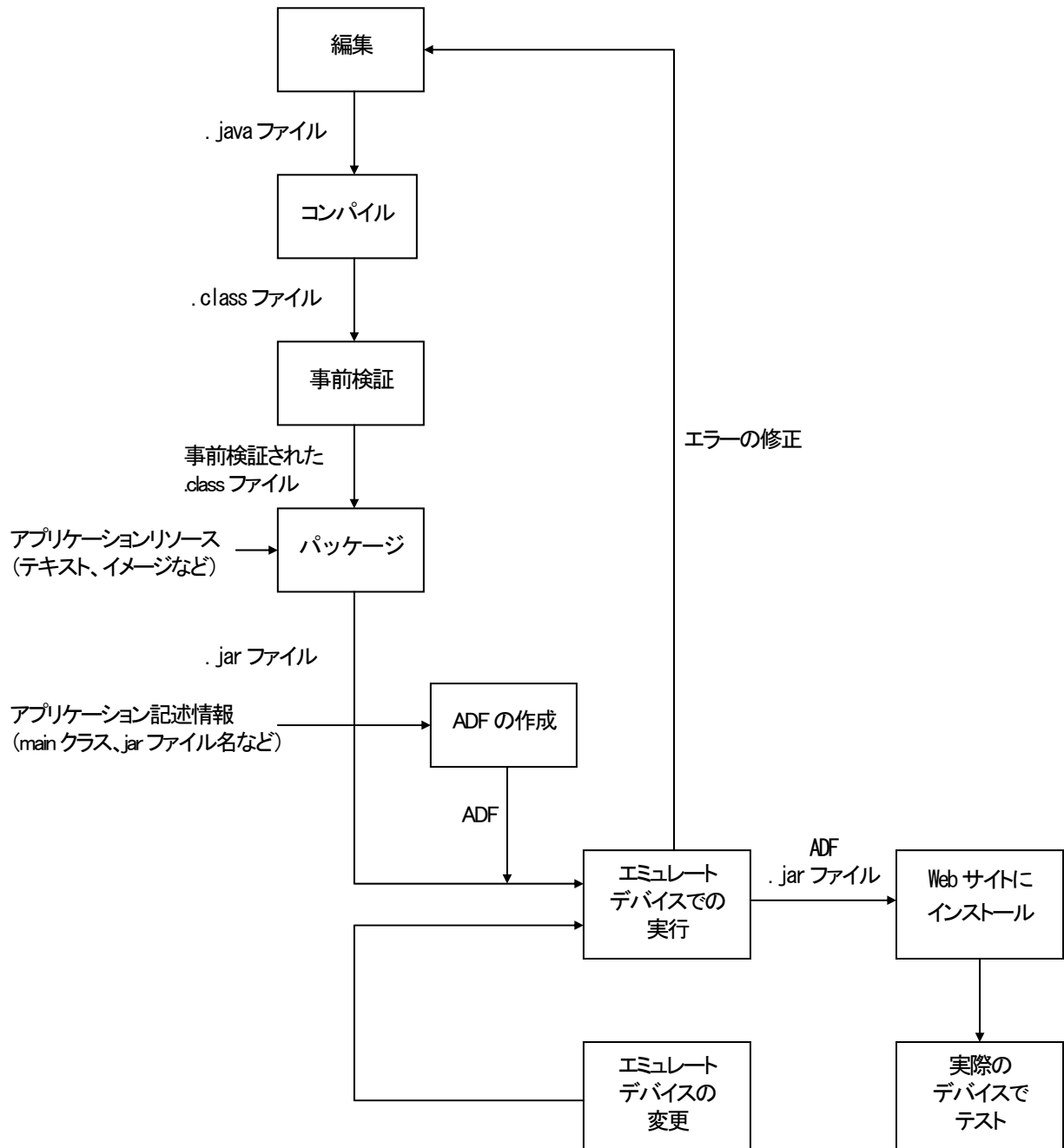


図 1-2 アプリケーション開発サイクル

2. 「iα ppli Development Kit for DoJa-3.0」のインストール

2.1. インストールする前に

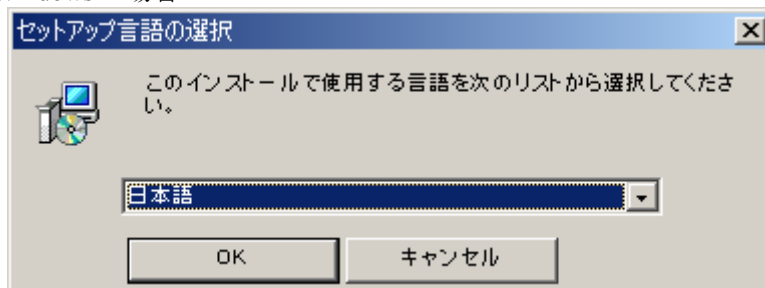
- 「iα ppli Development Kit for DoJa-3.0」は、J2SE SDK がインストールされている必要があります。
もし、J2SE SDK がインストールされていない場合には、J2SE SDK を入手してインストールしてください。
- インストール（アンインストール）する際には、Administrator 権限のあるユーザでログインしてください。
- 「iα ppli Development Kit for DoJa-3.0」をインストールする前に、「iα ppli Development Kit for DoJa-3.0」の以前のバージョンが既にインストールされている場合には、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。
- Eclipse を DoJa-3.5 と共存させたい場合は、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_DoJa3.0、C:\Eclipse_DoJa3.5）

2.2. インストールの手順

「iα ppli Development Kit for DoJa-3.0」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：

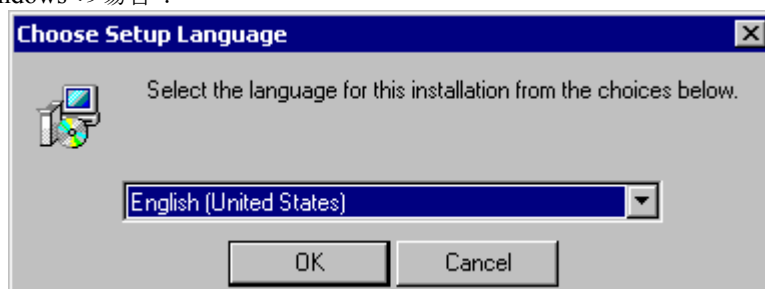


図 2-1 セットアップ言語の選択画面

3. ウィザードに従い「次へ」を押します。

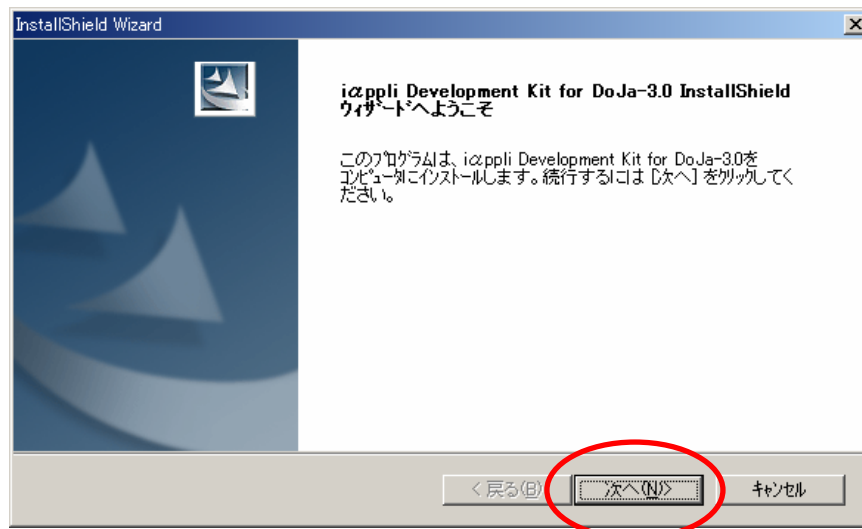


図 2-2 ようこそ画面

4. 「使用許諾契約」と「Binary Code License Agreement」が表示されるので、よく読んでから「はい」を押します。

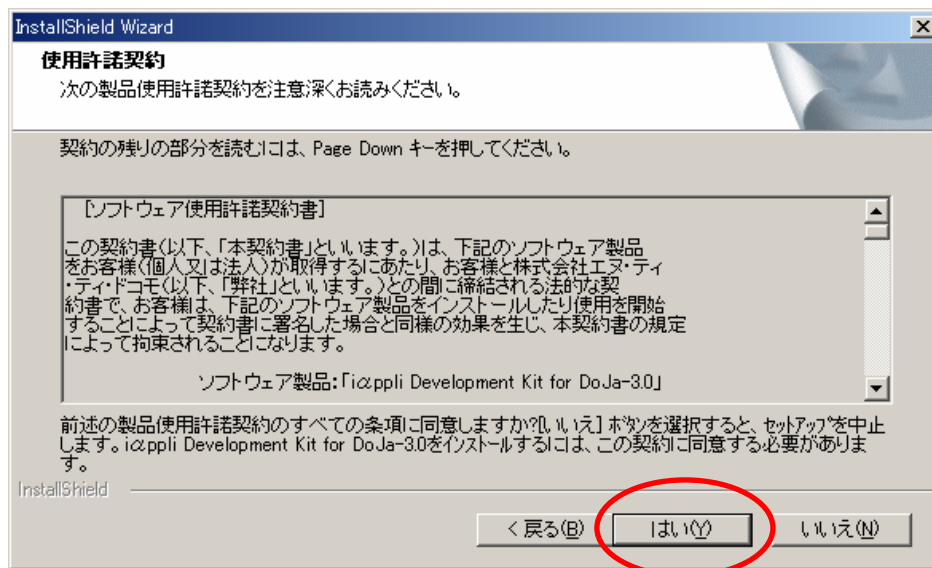


図 2-3 使用許諾契約画面

5. 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。

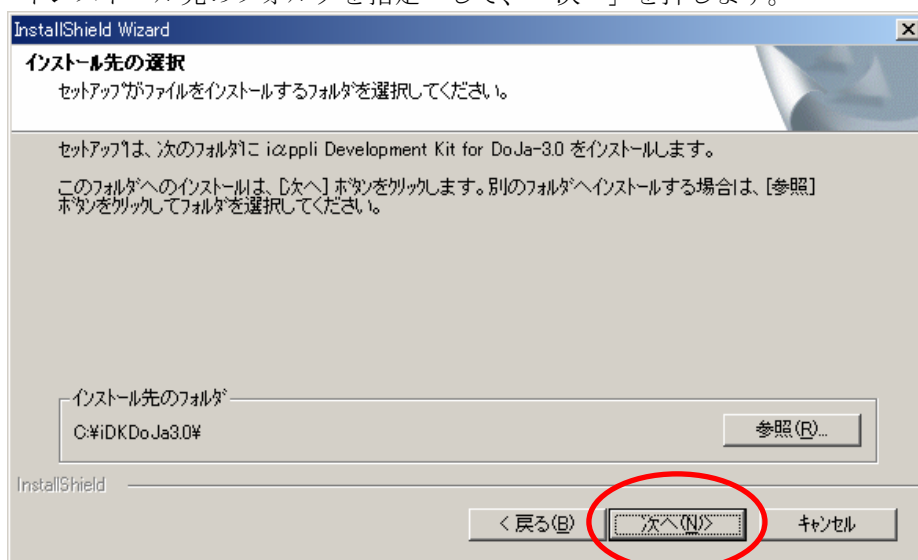


図 2-4 インストール先の選択画面

6. セットアップの種類を選択して、「次へ」を押します。

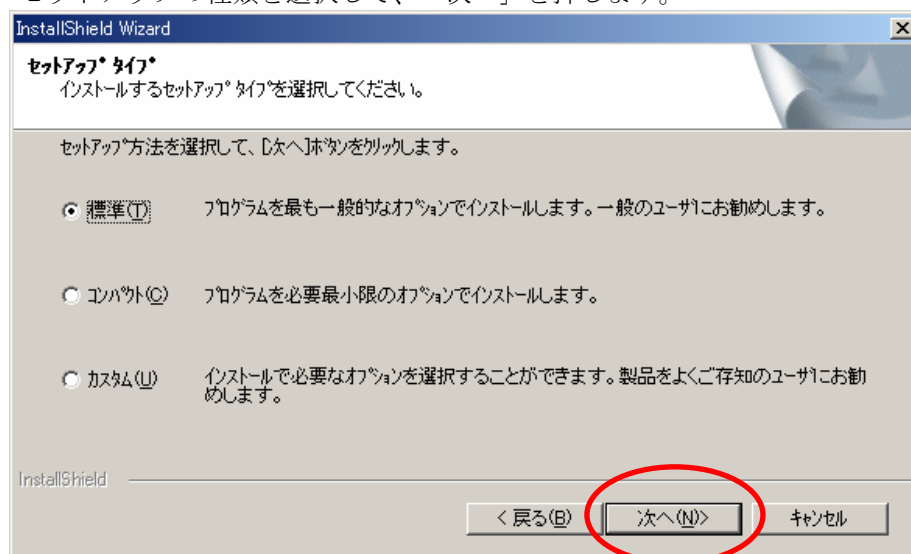


図 2-5 セットアップタイプの選択画面

- ・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できます。
もし、Sun ONE Studio や Eclipse2.1.X 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。

7. 手順6で、「カスタム」以外を選択した場合は、手順8へ進んでください。
インストールしたい機能を選択して、「次へ」を押します。

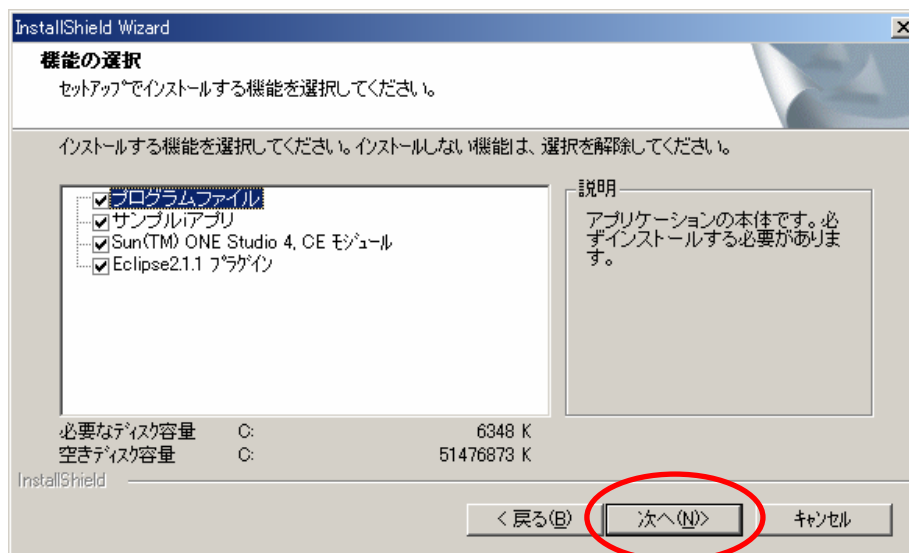


図 2-6 機能の選択画面

- 「Sun(TM) ONE Studio 4, CE モジュール」を選択すると、Sun ONE Studio 4, Community Edition を使って i アプリを開発することができます。
ただし、あらかじめ Sun ONE Studio 4, Community Edition がインストールされている必要があります。
注意：DoJa-3.5 のモジュールがインストールされている場合は、上書きされます。
 - 「Eclipse2.1.X プラグイン」を選択すると、Eclipse2.1.X を使って i アプリを開発することができます。
ただし、あらかじめ Eclipse2.1.X がインストールされている必要があります。
注意：DoJa-3.5 のプラグインがインストールされている場合は、上書きされます。
これは、別の Eclipse 環境を用意することで共存できるようになります。
8. 前項で「Eclipse2.1.X 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。

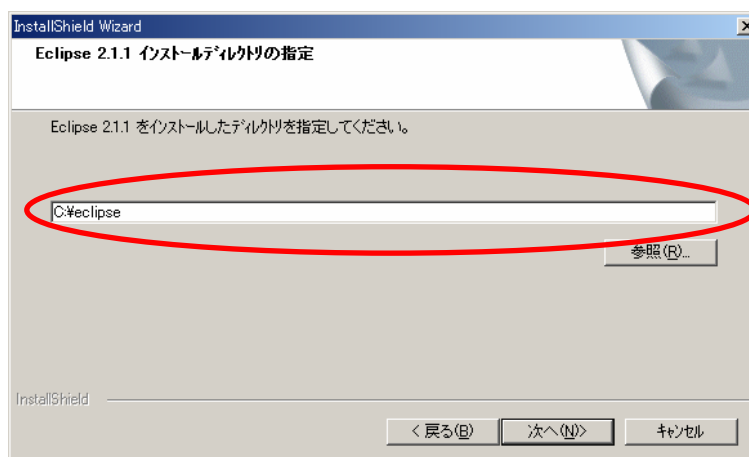


図 2-7 Eclipse プラグインインストール先設定画面

9. 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。

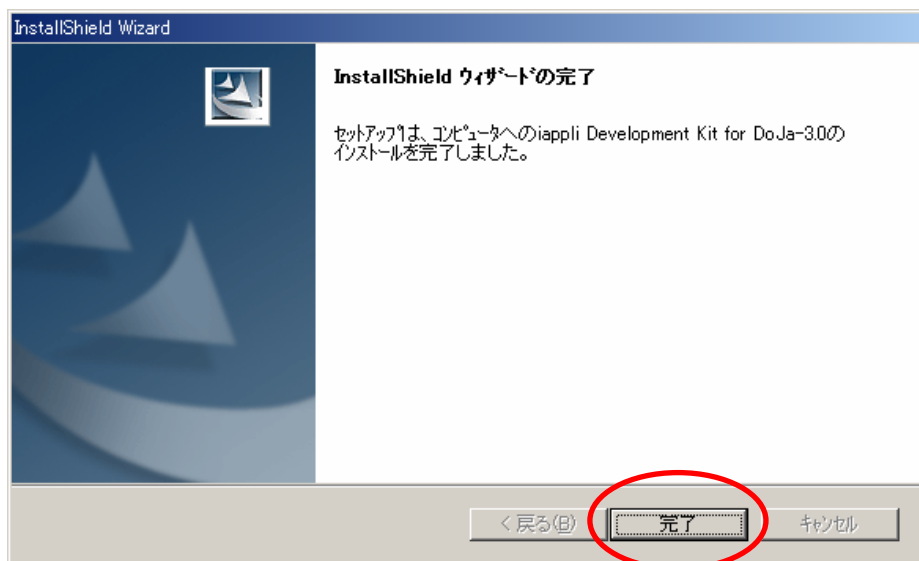


図 2-8 インストール完了画面

10. 日本語で「iα ppli Development Kit for DoJa-3.0」をインストールした場合、「絵文字（外字）」を設定するための外字設定画面が表示されますので、外字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 2-9）。

インストール後に実行したい場合は、「iα ppli Development Kit for DoJa-3.0」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」を実行します。

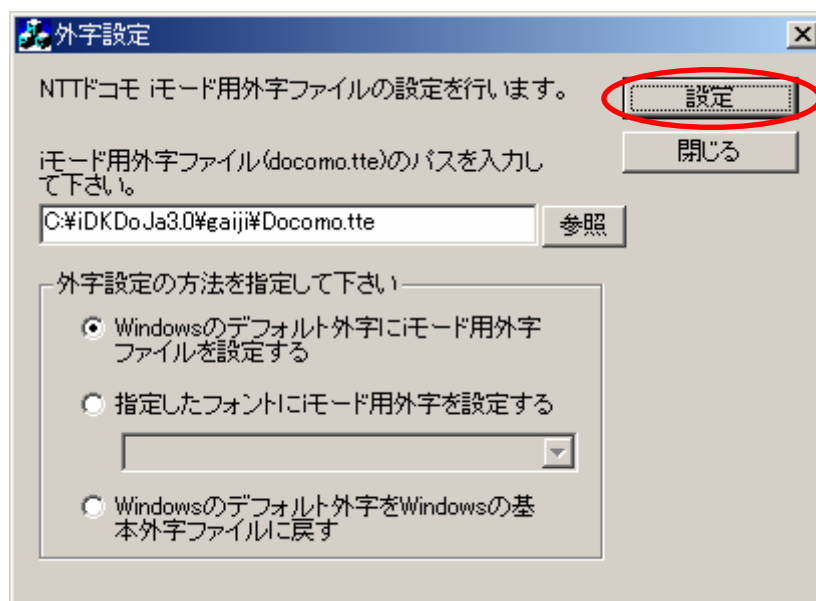


図 2-9 「ChangeEUDC.exe」の実行（外字設定画面）

11. 設定終了の確認画面が表示されたら、「OK」を押します。（図 2-10）
12. 外字設定画面の「閉じる」を押します（図 2-9）。

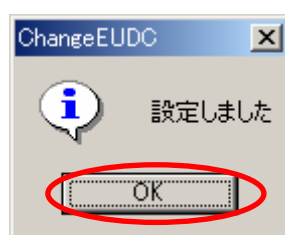


図 2-10 絵文字（外字）設定終了の確認

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 2-9 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。

13. 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:\iDKDoJa3.0」にインストールした場合、「C:\iDKDoJa3.0\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事ができます。

14. 必要に応じて、APIリファレンスを http://www.nttdocomo.co.jp/p_s/imode/make/ から取得し、「C:\iDKDoJa3.0」にインストールした場合、「C:\iDKDoJa3.0\doc」の配下に置いておきます。

インストール後、「i appli Development Kit for DoJa-3.0」のヘルプメニューからAPIリファレンス、操作ガイド（本書：ユーザーズガイド）および、リリースノート参照できるようになります（「4.13. 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）を置いてみてください。

2.3. アンインストール手順

「iα ppli Development Kit for DoJa-3.0」をアンインストールするには、

1. 「スタート」→「プログラム」→「iα ppli Development Kit for DoJa-3.0」→
「Remove iα ppli Development Kit for DoJa-3.0」を実行します（図 2-11）。
（「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。）

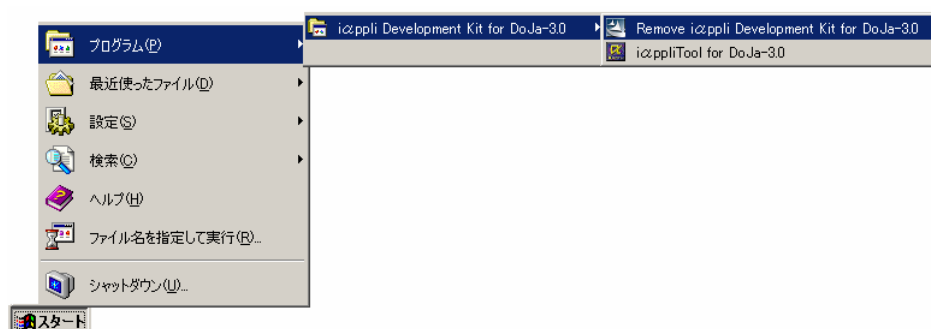


図 2-11 アンインストールの実行

2. ファイル削除の確認画面が表示されるので、「OK」を押します。（図 2-12）

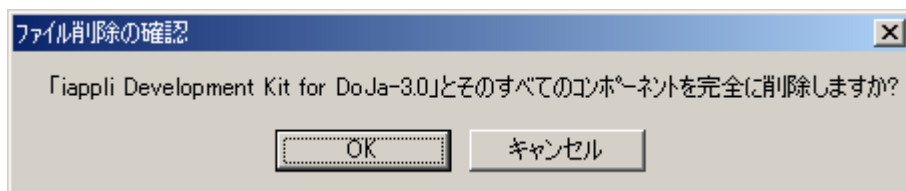


図 2-12 ファイル削除の確認画面

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、そのまま残りますので、不要の場合は、インストール先のディレクトリ
「C:\iDKDoJa3.0」を直接削除してください。

3. 「iαppliTool」の起動

「iαppliTool」を起動する前に、HTTP で通信を行うアプリケーションを使用する場合、以下の設定を行ってください。

- Windows のコントロールパネル→インターネット オプションの「詳細設定」タブ内「HTTP1.1 設定」の「HTTP1.1 を使用する」項目のチェックボックスのチェックをはずしてください（図 3-1）。

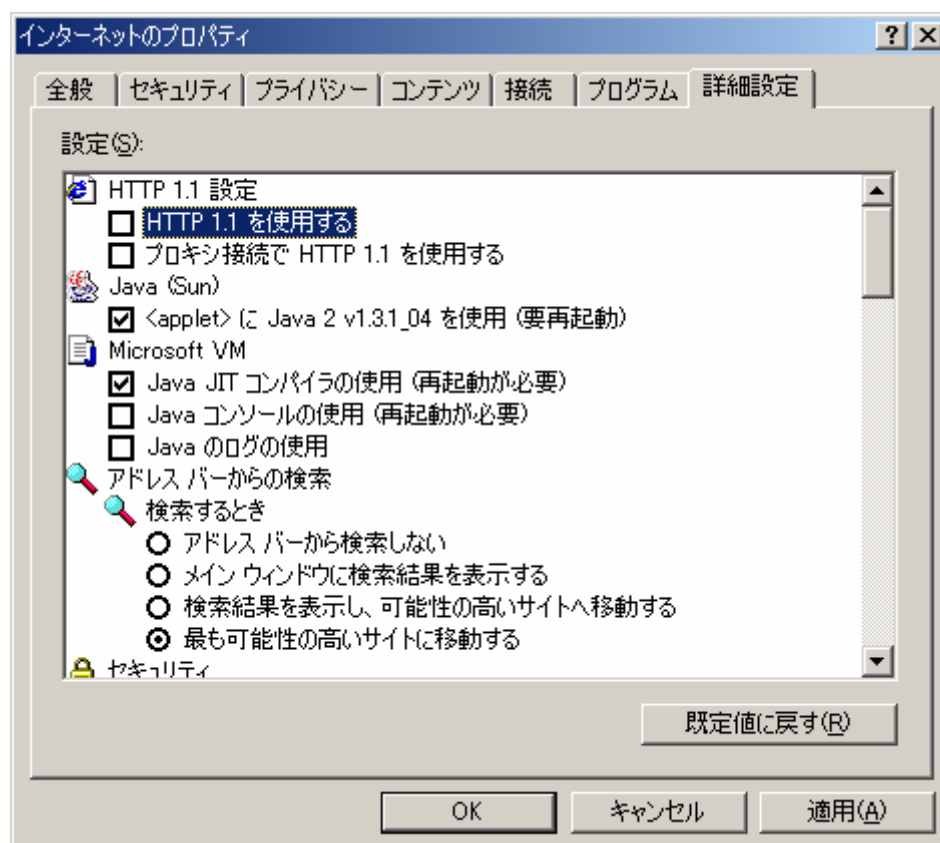


図 3-1 HTTP アプリケーションを使用する際の設定

「iα ppliTool」を起動するには、「スタート」→「プログラム」→「iα ppli Development Kit for DoJa-3.0」→「iα ppliTool for DoJa-3.0」メニューを選択します^[2]。

「iα ppliTool」を起動すると、図 3-2、図 3-3 に示す最初の画面が表示されます。図 3-2 は、「iα ppliTool」のメイン画面であり、ソースコードのコンパイル、ADF の編集、ネットワークの設定、アプリケーションの実行などを行います。

図 3-3 は、「エミュレータ端末」画面であり、実際のアプリケーションを表示する画面です。この画面がエミュレータの待ち受け画面となります。

この時点で図 3-2 では、次の操作が可能となります：

- ドキュメントの参照（「4.13. ヘルプの設定」を参照）
- エミュレーション端末の選択（「4.15. エミュレーション端末の選択」を参照）
- コンパイラ（javac）の選択（「4.17. コンパイラ（javac）の選択」を参照）
- 既存プロジェクトを読み込む（「5.2. 既存プロジェクトの読み込み」を参照）
- 新規プロジェクトを作成する（「5.5. プロジェクトの新規作成」を参照）
- メーラ起動（「5.17. メーラ起動」を参照）
- ブラウザ起動（「5.18. ブラウザ起動」を参照）
- コードリーダー起動（「5.19. コードリーダー起動」を参照）
- その他、ネットワーク、電話帳、送受信メール等の初期設定

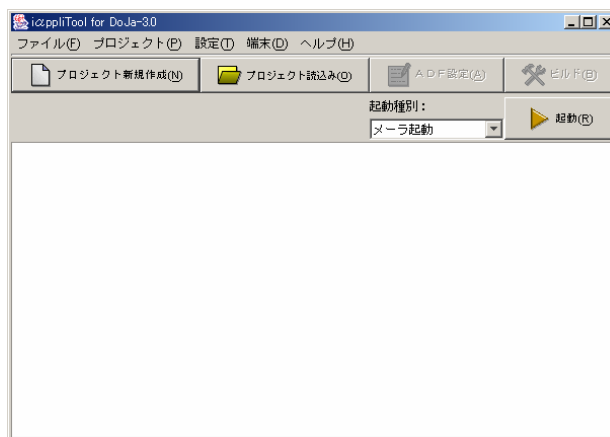


図 3-2 「iα ppliTool」のメイン画面



図 3-3 エミュレータ端末画面

^[2] 「iα ppliTool」を起動後、プロジェクトのビルド（「5.6. プロジェクトのビルドと通常起動」参照）を行う際、「iα ppliTool」内で J2SE SDK がインストールされているディレクトリを正しく取得できない場合があります。この場合は、「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ますので、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、あらためて「iα ppliTool」を起動してください。

4. エミュレータの設定

4.1. 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」→「ネットワーク設定...」を選択します（図 4-1）。

ネットワーク設定

ADFのURL

UserAgent

HTTP設定

プロキシHOST

プロキシポート

HTTPS設定

プロキシHOST

プロキシポート

OBEX設定

ホスト

設定 取消

図 4-1 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 4-1 に示すとおりです。

表 4-1 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		アプリケーションの 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。アプリケーションはこの 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/1.0/D OJA_INET_CLI ENT
HTTP 設定	プロキシ ホスト	アプリケーション及び「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。「4.14. 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	アプリケーション及び「iα ppliTool」のブラウザが Web サーバとの通信に使用する HTTP プロキシサーバのポート番号。 「4.14. 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	アプリケーションが 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。「4.14. 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	アプリケーションが 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。「4.14. 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
OBEX 設定	ホスト	アプリケーションが 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。「12. 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「iα ppliTool」起動時に自動的に設定されます。	なし

4.2. ダウンロードアプリケーションの設定

ダウンロードアプリケーションの設定を行います。

ダウンロードアプリケーションの設定ファイルについては、「8.3. 「JAM.properties」ファイルの記述方法」を参照のこと。

ダウンロードアプリケーションの設定をするには：

1. メニューから「設定」→「ダウンロードアプリケーション設定...」を選択します（図 4-2）。

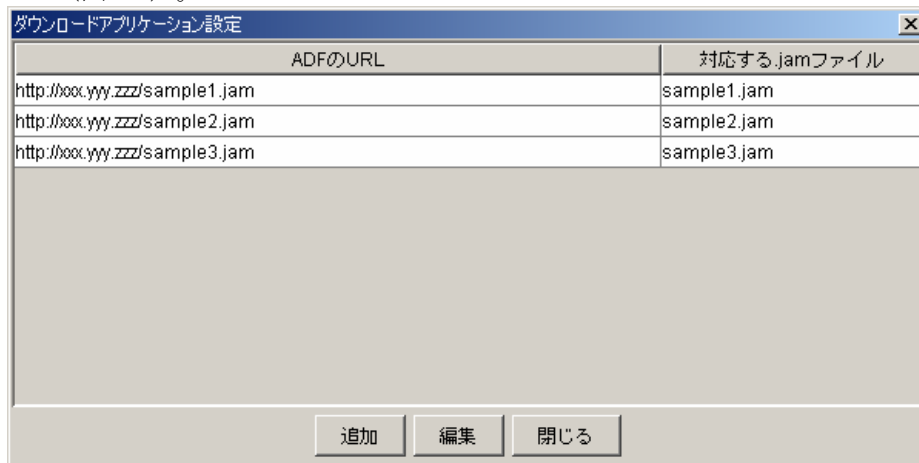


図 4-2 ダウンロードアプリケーションの設定画面

2. URL を新たに追加する場合は「追加」、変更する場合は一覧を選択して「編集」を押します（図 4-3）。



図 4-3 ダウンロードアプリケーション情報の編集

3. 「ADF の URL」にダウンロード元のURL、
「対応する .jam ファイル」に起動したいアプリケーションの ADF
を入力して「設定」を押します。
設定した内容がダウンロードアプリケーションの設定画面 と JAM.properties フ
ァイルに反映されます。

各設定項目は 表 4-2 に示すとおりです。

表 4-2 ダウンロードアプリケーションの設定項目

設定項目	説明	デフォルト
ADF の URL	ダウンロード元の URL。	http://xxx.yyy.zzz/samp le1.jam
対応する jam ファイル	起動したいアプリケーションの ADF の名称。	sample1.jam

4.3. トラストッド動作の設定

アプリケーションがトラストッド動作機能を利用する**トラストッドiアプリ（iアプリDX）**である場合に設定します。

トラストッド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
- 端末位置情報の取得

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、トラストッド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。

トラストッド動作を設定するには、

1. メニューから「設定」→「トラストッド動作設定...」を選択します。
「トラストッド動作設定」画面が表示されます（図 4-4）。

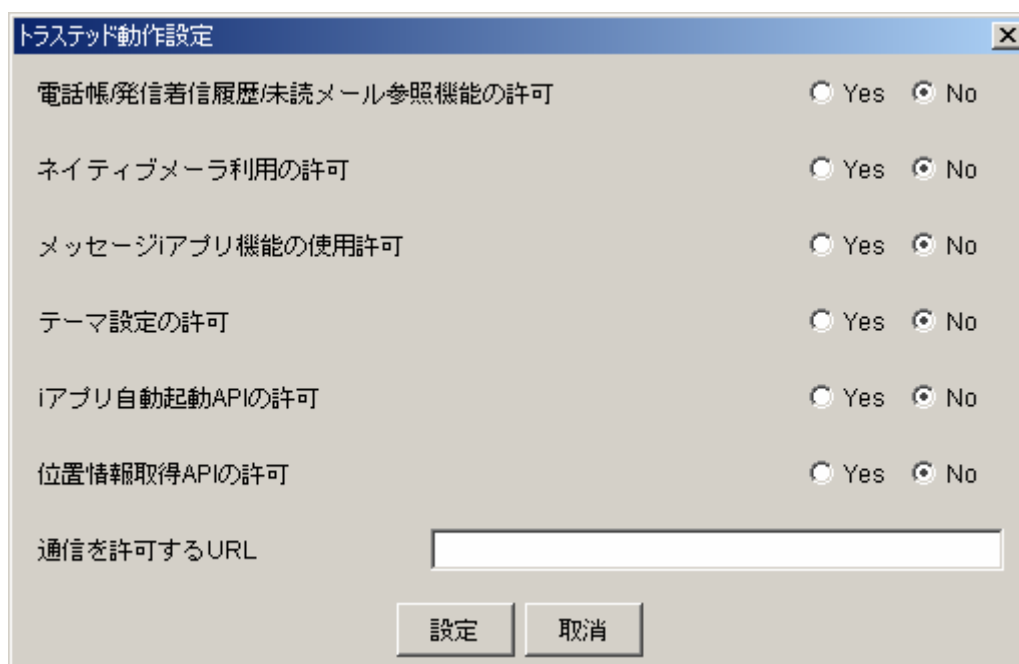


図 4-4 トラストッド動作設定画面

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 4-3 に示すとおりです。

表 4-3 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)
位置情報取得 A P I の許可	位置情報を取得する A P I を利用する場合に yes に変更します。 (関連: Location, LocationProvider)
通信を許可する U R L	ダウンロード元以外にこのアプリケーションが通信を許可する U R L 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、アプリケーション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。)

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 O O O 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 U R L から入手することができます:

http://www.nttdocomo.co.jp/p_s/imode/make/

4.4. 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」→「メールアドレス設定...」を選択します（図 4-5）。



図 4-5 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 4-6）。



図 4-6 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」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。)

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アプリケーションから送信すると、そのメールアドレスを選択しているアプリケーションでメールを受信する事ができます。（受信ボックスに格納されています。）
これにより、エミュレータでもアプリケーション同士で相互にメールをやり取りする事ができます（「4.9.2. メールの送受信エミュレート機能」を参照）。

4.5. リモート制御の設定

プッシュ機能により、サーバからアプリケーションに渡すパラメータを設定します。

(関連：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 4-7)。

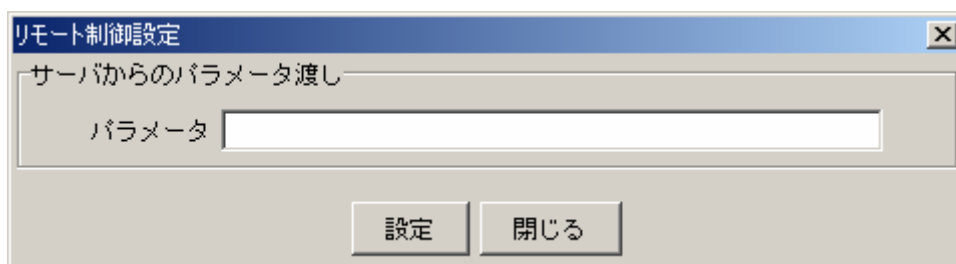


図 4-7 リモート制御パラメータの設定画面

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アプリケーションから取得するには、

1. アプリケーション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、アプリケーションにてパラメータを取得できます(図 4-8)。

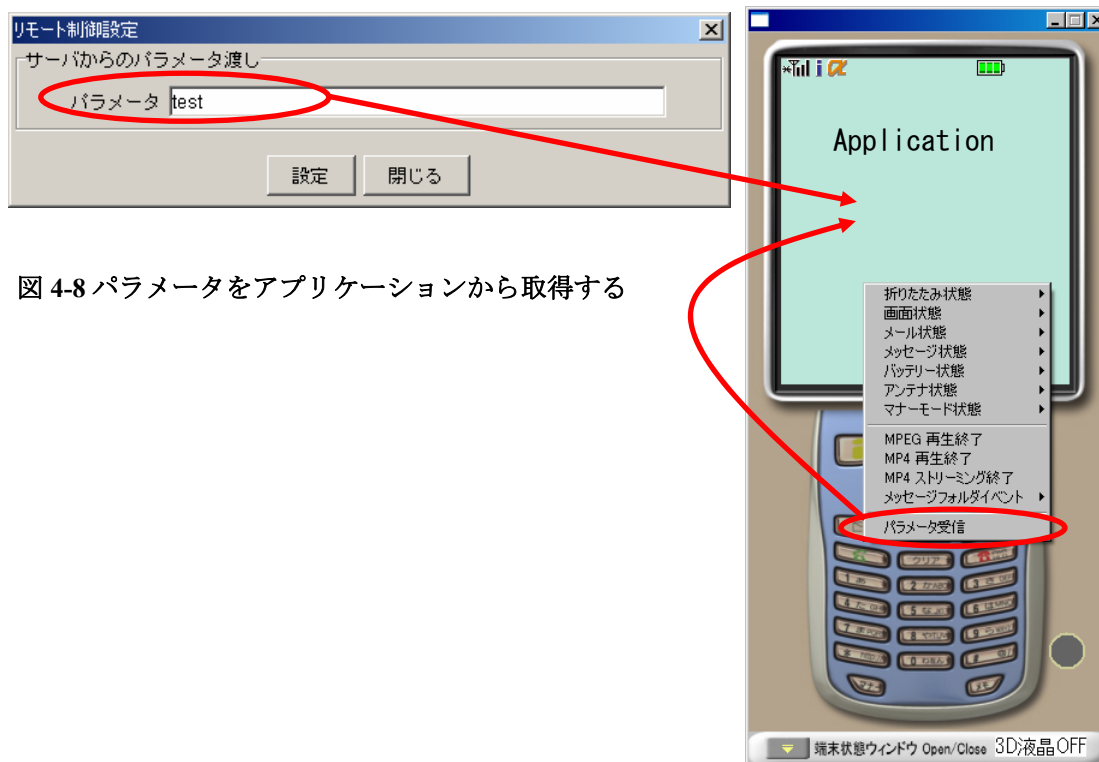


図 4-8 パラメータをアプリケーションから取得する

4.6. リモート制御のパラメータ領域の削除

「4.5. 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4-9）。
これにより、“一度もパラメータを受信していない場合”の端末を試験する事ができます。

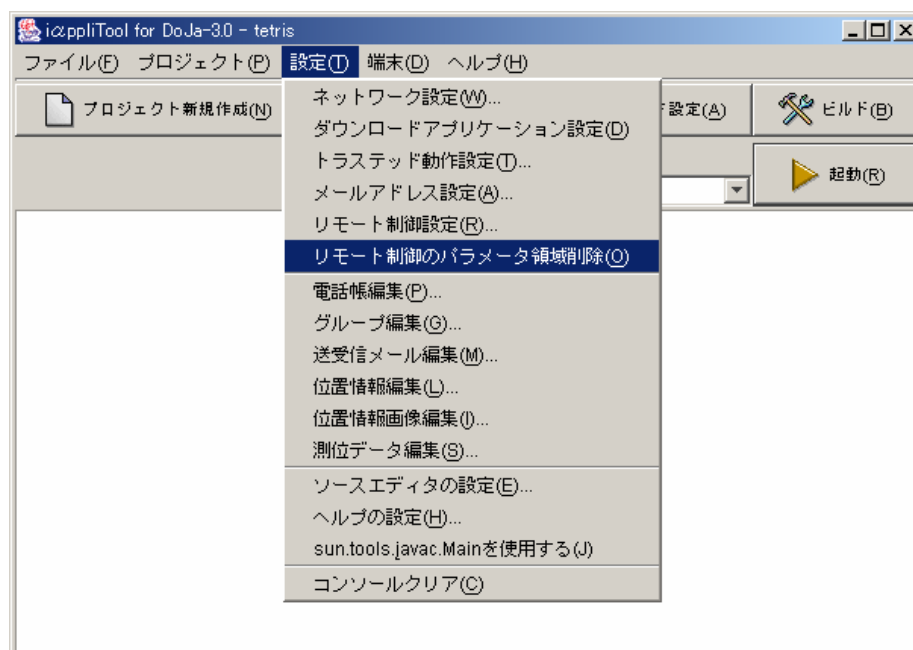


図 4-9 リモート制御パラメータ領域削除

4.7. 電話帳の編集

アプリケーション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」→「電話帳の編集...」を選択します（図 4-10）。

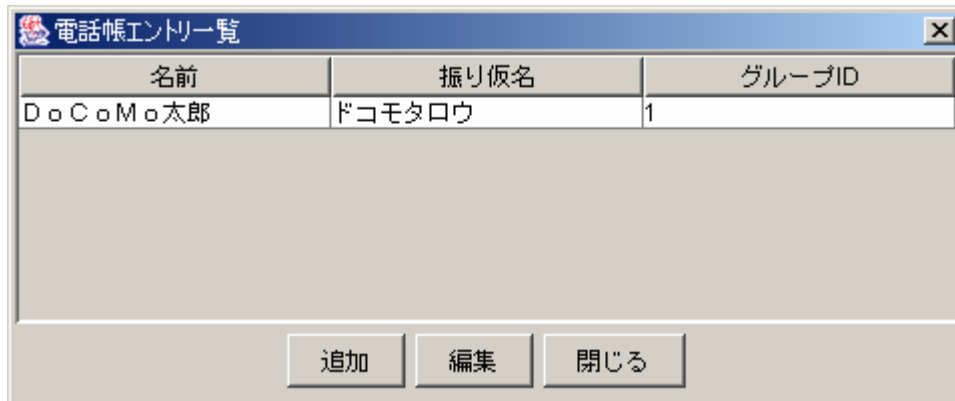


図 4-10 電話帳エントリー一覧画面

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 4-11）。

図 4-11 電話帳エントリ登録画面

3. 各設定項目を設定して「設定」を押します。

各設定項目は表 4-4 に示すとおりです。

表 4-4 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 4-13）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 4-12）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

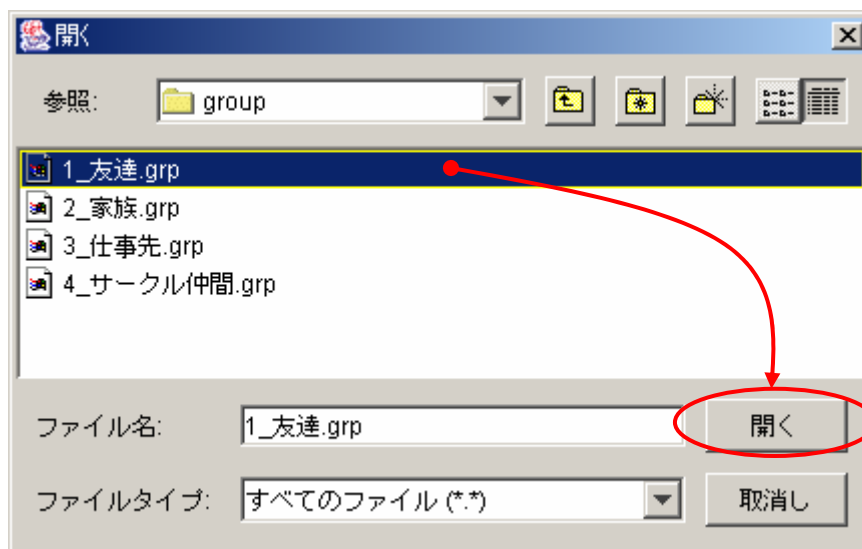


図 4-12 グループの選択ダイアログ

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。

もし、電話帳エントリ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。

4.8. グループの編集

アプリケーション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」→「グループの編集...」を選択します（図 4-13）。

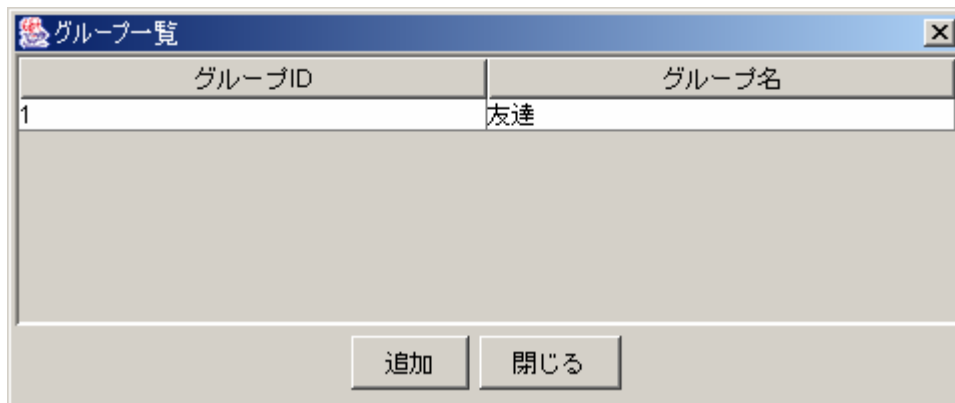


図 4-13 グループ一覧画面

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 4-14）。



図 4-14 グループ追加画面

3. 各設定項目を設定して「設定」を押します。

各設定項目は表 4-5 に示すとおりです。

表 4-5 グループの設定項目

設定項目	説明	デフォルト
グループ名	グループの名前。	なし

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。
もし、グループ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。

4.9. 送受信メールの編集

アプリケーション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」→「送受信メール編集...」を選択します（図 4-15）。

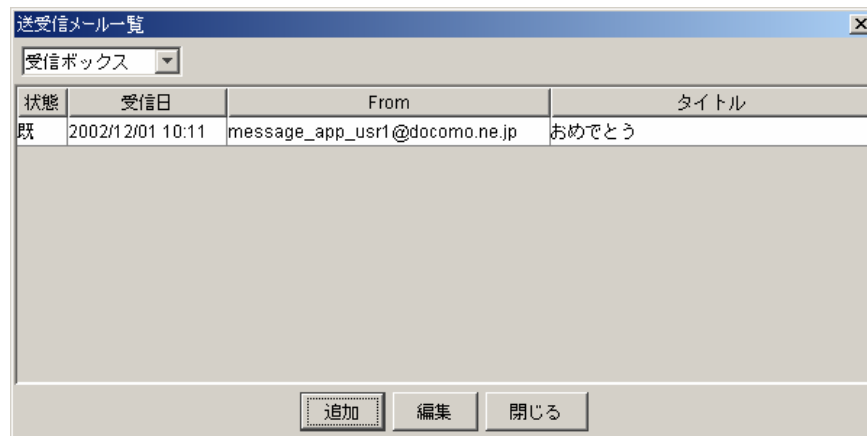


図 4-15 送受信メール一覧画面

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。
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 4-16、図 4-17）。

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: 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 4-16 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: 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 4-17 メール編集（送信ボックス）画面

4. 各設定項目を設定して「設定」を押します。

各設定項目は表 4-6 に示すとおりです。

表 4-6 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\メールアドレス\received\メッセージコード\受信日_通番.mail
 送信ボックス： lib\mail\メールアドレス\sent\メッセージコード\送信日_通番.mail
 未送信ボックス： lib\mail\メールアドレス\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、「8.6. 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（「5.3. 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、アプリケーション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、BASE64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\メールアドレス\receive\メッセージコード\mail.state
宛先状態 : lib\mail\メールアドレス\sent\メッセージコード\xstring.state
lib\mail\メールアドレス\unsent\メッセージコード\xstring.state
移動機のメールアドレス : 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。

4.9.1. 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 4-18、図 4-19）

- 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: 2002年12月01日10時0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 4-18 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: 2002年12月01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 4-19 送信ボックス／未送信ボックスのショートメール作成

4.9.2. メールの送受信エミュレート機能

メールを扱うアプリケーションは、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
(「4.4 メールアドレスの設定」を参照)。
2. エミュレートするアプリケーションの MessageCode を同一の値にします
(「5.3. 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 (アプリケーション) でのメールはやり取りできません。
3. エミュレータからアプリケーションで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移動機に設定したアプリケーションが存在しない、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 4-20)。

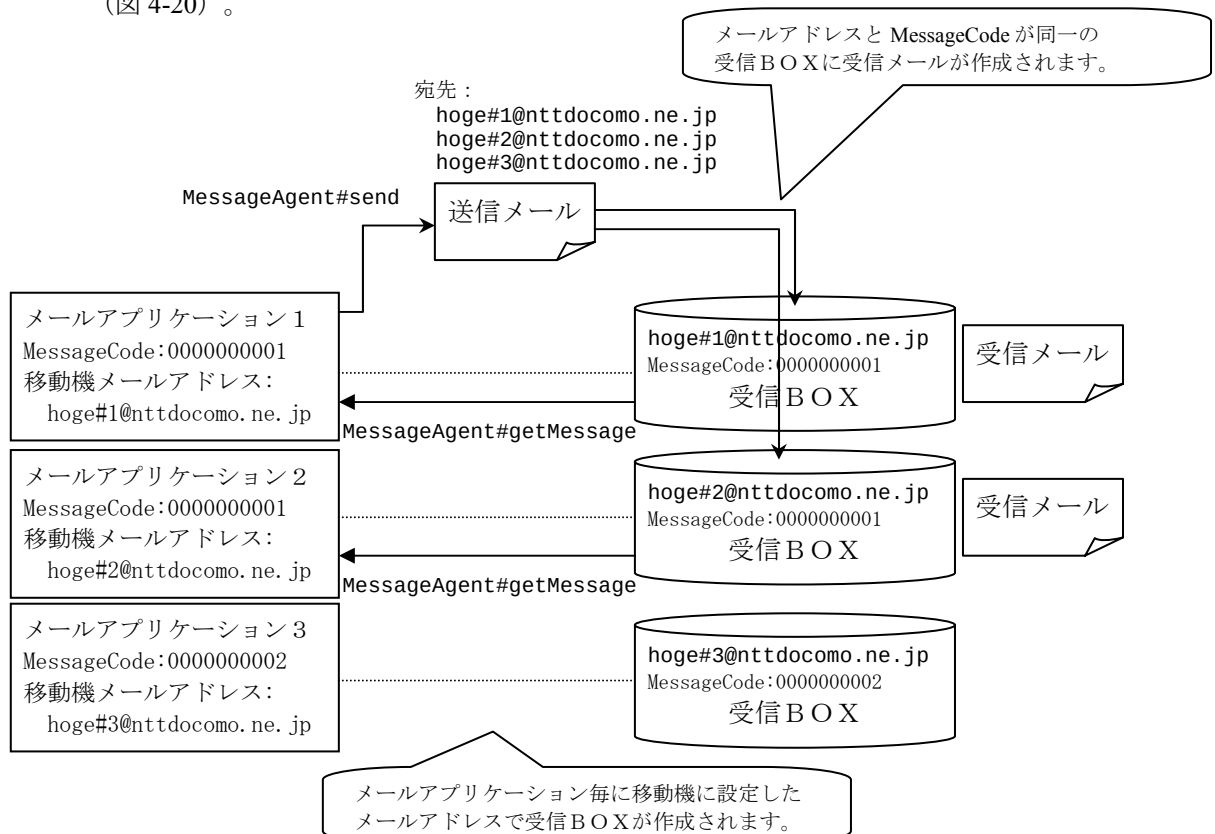


図 4-20 アプリケーションによるメールの送受信

4.10. 位置情報編集

アプリケーションから利用する位置情報を編集します。

位置情報の編集をするには、

1. メニューから「設定」→「位置情報編集...」を選択します（図 4-21）。

図 4-21 位置情報編集画面

2. 各設定項目を設定して「設定」を押します。

各設定項目は表 4-7 に示すとおりです。

表 4-7 位置情報の設定項目

設定項目	説明	デフォルト
名称	位置情報の名前を指定します。	位置メモ 1
位置測位日時	測位を実行した日時を指定します。	2003/09/01 00:00:00.000
測位レベル	測位レベルを 1 ～ 3 で設定します。 レベル 1 の最悪値：2147483647 [m] レベル 2 の最悪値：299 [m] レベル 3 の最悪値：49 [m]	1
緯度	DMS 単位で、N:北緯 S:南緯 と 度、分、秒、ミリ秒を指定します。	N60.00.00.00
経度	DMS 単位で、E:東経 W:西経 と 度、分、秒、ミリ秒を指定します。	E60.00.00.00

4.11. 位置情報画像編集

アプリケーションから利用する位置情報画像に埋め込まれるデータを編集します。

位置情報画像に埋め込まれるデータの編集をするには、

1. メニューから「設定」→「位置情報画像編集...」を選択します（図 4-22）。

図 4-22 位置情報画像編集画面

2. 各設定項目を設定して「設定」を押します。

各設定項目は表 4-8 に示すとおりです。

表 4-8 位置情報画像の設定項目

設定項目	説明	デフォルト
エントリ ID	位置情報画像のエントリ ID を指定します。	0
位置測位日時	測位を実行した日時を指定します。	2003/09/01 00:00:00.000
測位レベル	測位レベルを 1 ～ 3 で設定します。 レベル 1 の最悪値：2147483647 [m] レベル 2 の最悪値：299 [m] レベル 3 の最悪値：49 [m]	1
緯度	DMS 単位で、N:北緯 S:南緯 と 度、分、秒、ミリ秒を指定します。	N60.00.00.00
経度	DMS 単位で、E:東経 W:西経 と 度、分、秒、ミリ秒を指定します。	E60.00.00.00

4.12. 測位データ編集

アプリケーションから利用する測位データを編集します。

測位データの編集をするには、

1. メニューから「設定」→「測位データ編集...」を選択します（図 4-23）。

図 4-23 測位データ編集画面

2. 各設定項目を設定して「設定」を押します。

各設定項目は表 4-9 に示すとおりです。

表 4-9 測位データの設定項目

設定項目	説明	デフォルト
測位レベル	測位レベルを 1 ～ 3 で設定します。 レベル 1 の最悪値：2147483647 [m] レベル 2 の最悪値：299 [m] レベル 3 の最悪値：49 [m]	1
緯度	DEGREE 単位で、N:北緯 S:南緯 と 角度を小数で指定します。	N60.000000
経度	DEGREE 単位で、E:東経 W:西経 と 角度を小数で指定します。	E60.000000
測位時間	測位が完了するまでの時間を秒単位で指定します。 ここで設定した測位時間が、 LocationPrvider#getLocation で指定したタイムアウト値よりも大きい場合、測位に失敗します。	30

4.13. 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ための設定を行います。

ヘルプの設定をするには、

1. メニューから「設定」→「ヘルプの設定...」を選択します（図 4-24）。

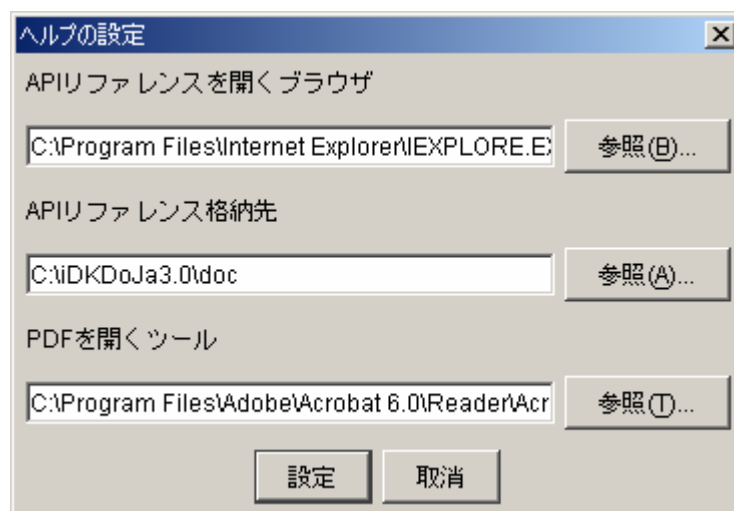


図 4-24 ヘルプの設定画面

2. 各設定項目を設定します。
各設定項目は表 4-10 に示すとおりです。

表 4-10 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アプリケーションを設定します。 ■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。

4.14. プロキシサーバの設定

エミュレータのもとで動作するアプリケーション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。アプリケーションは、HTTP か HTTPS を使用して Web サーバに接続しますが、同じアプリケーション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 4-1 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「11. 証明書の管理」を参照してください。

4.15. エミュレーション端末の選択

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 4-25）。

このメニューには、ビルドしたアプリケーション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「iαppli Tool」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 4-26 に示します。

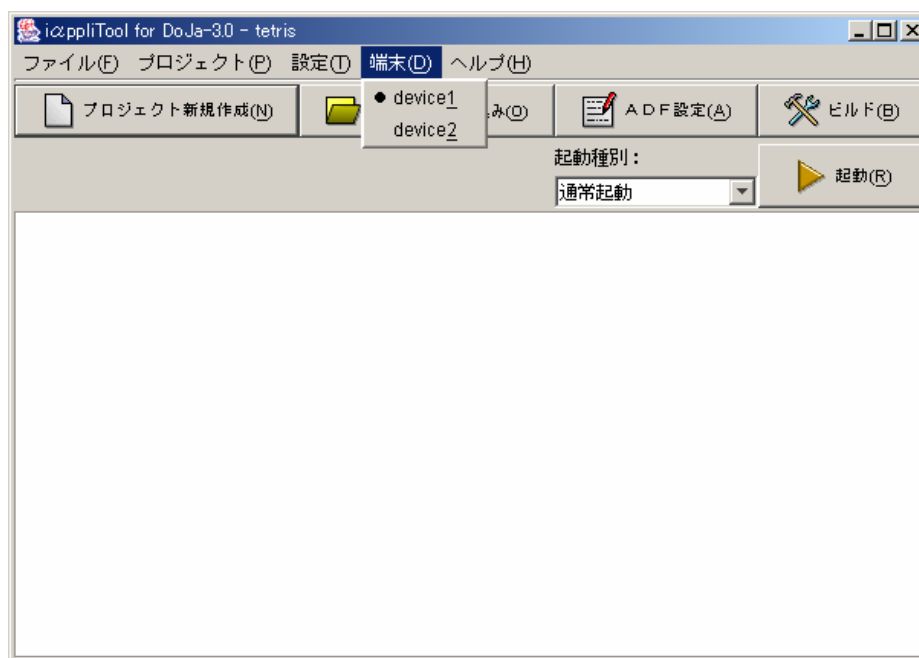


図 4-25 端末の選択

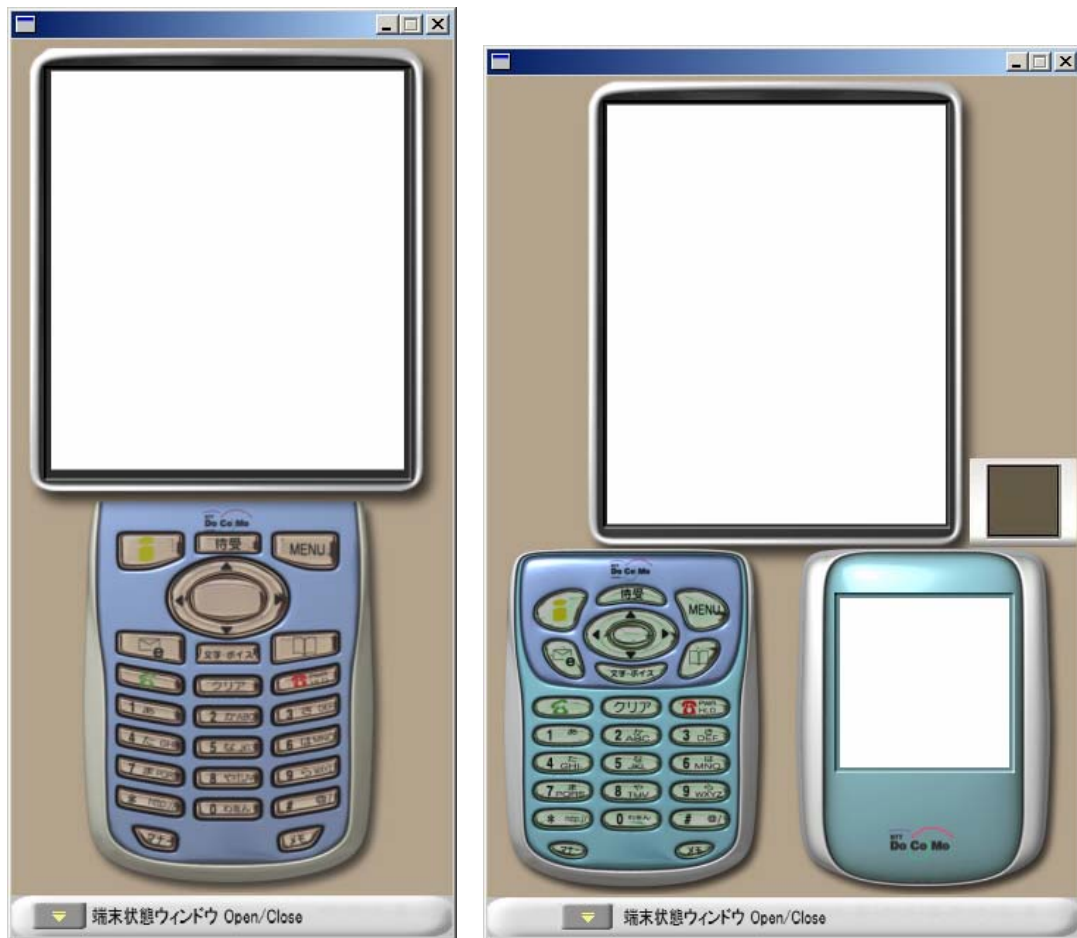


図 4-26 エミュレーション端末（左から「device1」、「device2」）

4.16. エミュレーション端末の操作

図 4-27 を用いてエミュレーション端末の基本操作を説明します。
なお、基本操作はどのエミュレータ端末でも同様です。

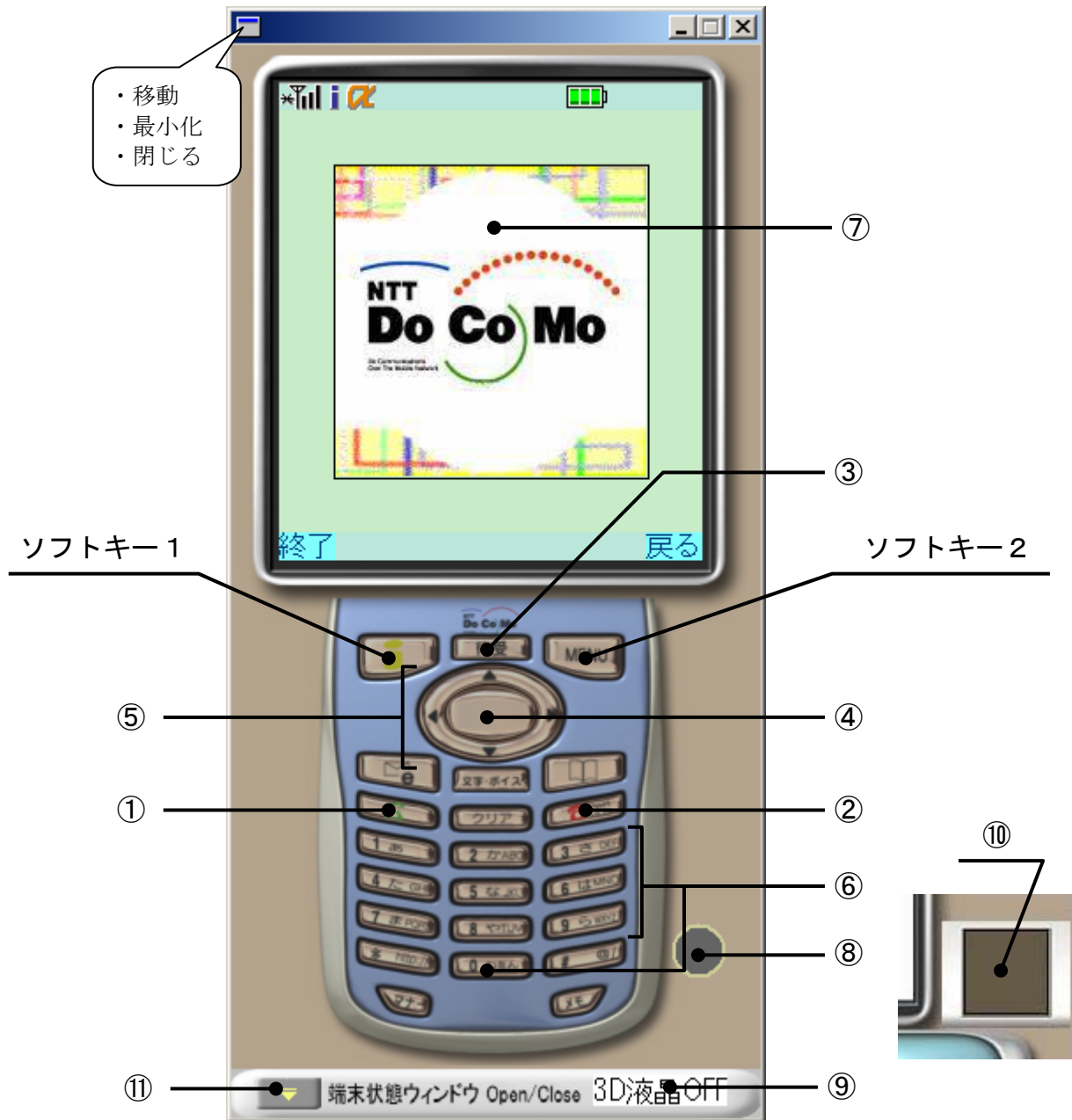


図 4-27 エミュレーション端末

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、アプリケーションはサスペンドします。

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アプリケーション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（「5.7.1. 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

- アプリケーション起動中に 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。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。
状態には、それぞれ以下に示す属性があります：

- 「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
 - 「GPS 測位開始(native)」

このメニューを選択すると、「GPS 測位中」の画面に切り替わり、実行中のアプリケーションはサスペンドされます。

これにより、ネイティブで実行されるGPSの定期測位との競合を試験することができます。

測位状態を終了するには、終話ボタンを押下します。
 - 「パラメータ受信」
- 上記の状態をアプリケーションから取得するには、`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（「5.3. アプリケーションディストリブタファイル（ADF）の編集」を参照）。

⑧. ピクチャーライト

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

⑨. 3D液晶ON/OFF

- i アプリオプションAPI `StereoScreen` を使った場合に切り替えます。

⑩. 指紋センサ・ナビゲーション

- i アプリオプションAPI `FingerSensor` を使った場合にイベントを通知します。

⑪. 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、「4.16.1. 端末子ウィンドウの操作」で説明します。

上記以外に、「ソフトキー1（左側）」「ソフトキー2（右側）」は、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、テキスト編集では無効です）。

4.16.1. 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（4.16. エミュレーション端末の操作 参照）。

図 4-28 を用いて端末子ウィンドウの基本操作を説明します。

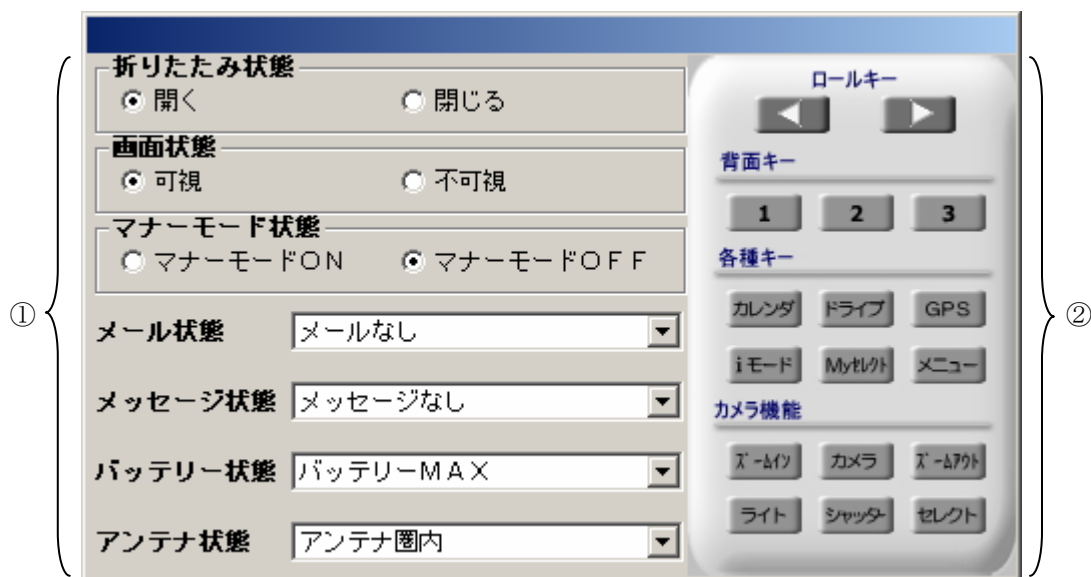


図 4-28 子ウィンドウ

- 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、アプリケーション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

※ アプリケーション実行中、GPSキーでの測位はおこなわれません。

4.17. 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 4-29)。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 (「5.6. 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「iα ppliTool」起動時に自動的に設定されます。

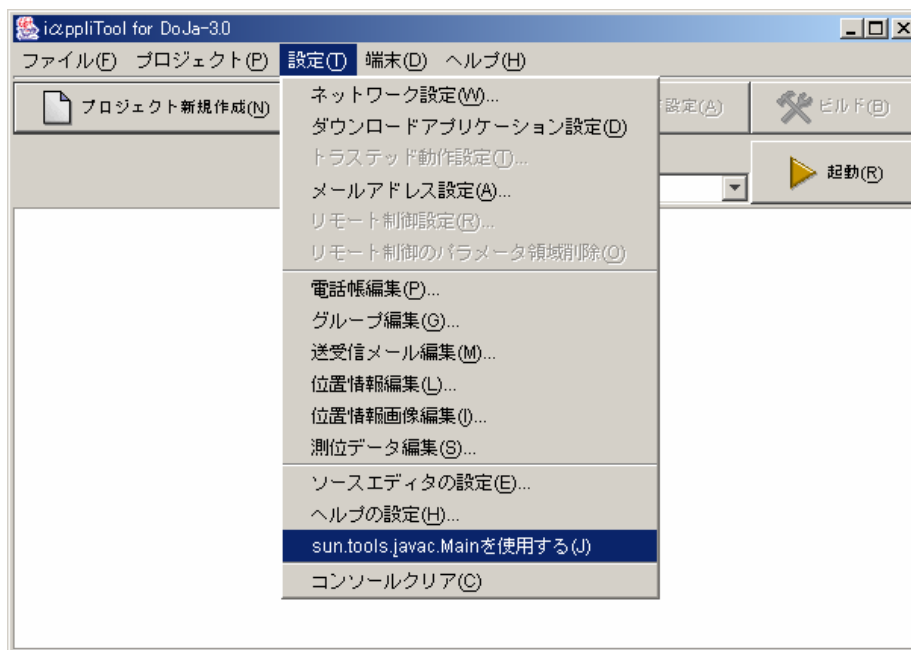


図 4-29 コンパイラ (javac) の選択

5. プロジェクト

「プロジェクト」とは、1つのアプリケーションに対するファイルの集合です。これには、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) に至る全てのファイルが含まれます。プロジェクトの目的は、開発の様々な段階でプロジェクト名を指定するだけでアプリケーションを呼び出せるようにすることです。

プロジェクトにおける開発作業項目には、以下のものがあります:

- プロジェクトのビルド

プロジェクトのコンパイル、事前検証、パッケージ化を1つの操作で連続的に行います。

- プロジェクトの起動

選択するエミュレーション端末上でプロジェクトを起動し、アプリケーションの動作をエミュレートします。

5.1. プロジェクトの作成から起動まで

「iαppliTool」では、プロジェクトの作成から起動まで、以下に示すことができます:

- 既存プロジェクトの読み込み (「5.2. 既存プロジェクトの読み込み」を参照)
- アプリケーションディスクリプタファイル (ADF) の変更 (「5.3. 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)
- 新規プロジェクトの作成 (「5.5. プロジェクトの新規作成」を参照)
- プロジェクトのビルドと通常起動 (「5.6. プロジェクトのビルドと通常起動」を参照)

5.2. 既存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 5-1）。



図 5-1 プロジェクト読み込み画面

2. プロジェクトの格納先（C:\iDKDoJa3.0\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。

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 5-2）。

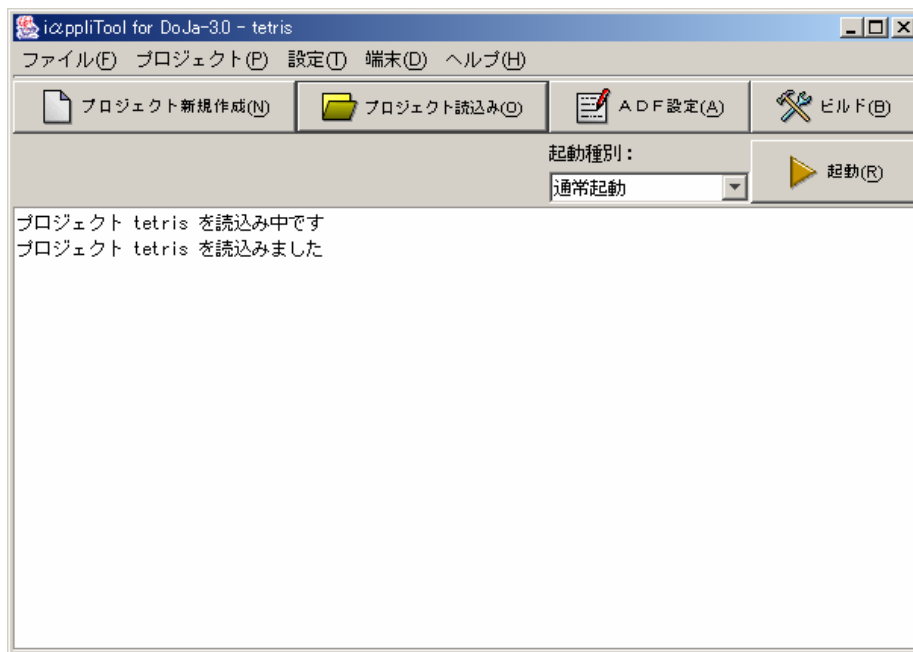


図 5-2 既存プロジェクト読み込み後のメイン画面

5.3. アプリケーションディスクリプタファイル(ADF)の編集

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 5-3）。

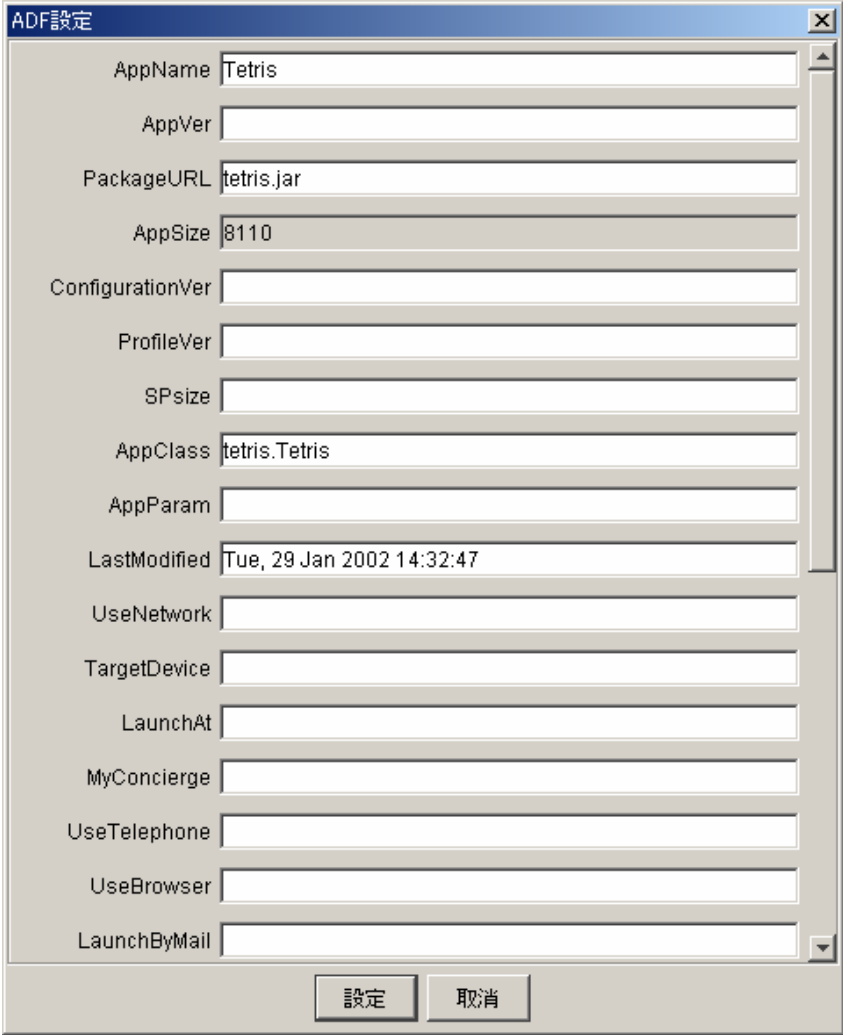


図 5-3 ADF 設定画面

2. 必要な設定項目を入力します。

「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 5-1 に示すとおりです。

表 5-1 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

設定項目	説明	種別
AppName	アプリケーション名（最大 16 バイト）。 設定する文字列は、SJIS を使用します。 <u>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。</u> <u>（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。）</u>	必須
AppVer	アプリケーションのバージョン（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	アプリケーションの場所を表す 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、「4.1. 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。 アプリケーションはこの PackageURL のホストとポートのみ通信が可能です。 <u>新規プロジェクトを作成したときには、アプリケーションの JAR ファイルの名前になっているので、適切な URL（アプリケーションのダウンロード先）に変更してください。</u>	必須
AppSize	アプリケーションサイズ（バイト単位）。 0 または負数が指定された場合は、アプリケーションの起動を禁止します。 <u>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。</u>	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” とし、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。	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" (503シリーズ対応) ・"DoJa-2.0" (504シリーズ対応) ・"DoJa-2.1" (FOMA02仕様モデル対応) ・"DoJa-2.2" (FOMA03前期仕様モデル対応) ・"DoJa-3.0" (505/506シリーズ対応) ・"DoJa-3.5" (FOMA03後期/04前期仕様モデル対応)	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。	オプション
AppClass	アプリケーションのメインクラス名（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	メインクラスのパラメータ（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	アプリケーションの最終更新時刻。 設定する文字列の形式については「5.3.1. LastModified 項目の表記方法」を参照。 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	アプリケーションで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、ネットワーク接続不可なアプリケーション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が発生します。	オプション
TargetDevice	アプリケーションが動作する機種名（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アプリケーション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は、"X506i,Y506i,Z506i"（FOMA の場合、"X2004i,Y2004i,Z2004i"）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、"all506" 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。アプリケーション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: l h（"h" は起動間隔（時間 単位）。l と h の間には 1 つのスペースを入れる。例: "l 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（「5.7. 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アプリケーションであるとみなし、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	アプリケーション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	アプリケーション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	アプリケーションの起動を許可する 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	アプリケーション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アプリケーションの起動を許可します（<起動コマンド>は、アプリケーション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	アプリケーション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、アプリケーション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	アプリケーション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 アプリケーション起動時の画面を、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、アプリケーション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、アプリケーション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、アプリケーションはトラステッド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
LaunchByApp	ランチャモードによる他のアプリケーション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アプリケーション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 "00000000000" 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	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、"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 5-4 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（9. 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
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（9. 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

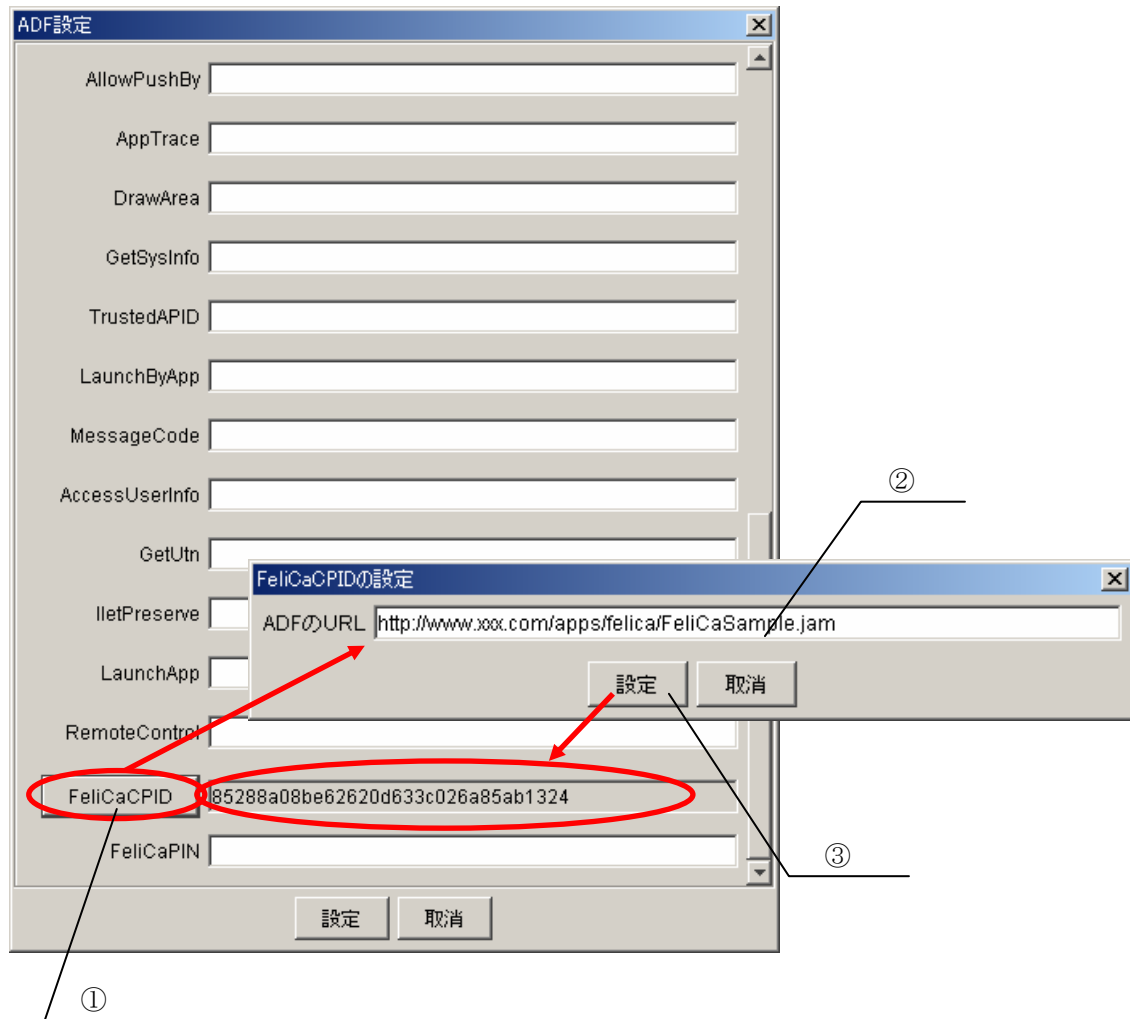


図 5-4 FeliCaCPID 設定画面

- ① DF 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 5-4)。
- ② 「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 「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。

5.3.1. 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”) の可能とします。

5.4. プロジェクトの設定

プロジェクトで使用するプロパティファイルの設定を行います。

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 5-5）。

プロパティ

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

フォントベースライン設定:

自動起動時刻登録可能件数: 4 登録可能件数設定

ONETIME YYYY: 2003 Dec SUN DD: 6 HH: 10 MM: 0 } 自動起動時刻設定

Onetime 2003 December 06 10:00
Every January 01 00:00
null
null } 自動起動時刻一覧

チャット機能: ☐ ON ☒ OFF

チャット機能をONにした場合、スケジューラ起動を使う事はできません。

設定 取消

図 5-5 プロジェクトの設定

2. 各設定項目を設定します。

各設定項目は表 5-2 に示すとおりです。

表 5-2 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
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
自動起動時刻登録可能件数	自動起動時刻の登録可能件数を入力し「登録可能件数設定」ボタンを押します。 自動起動時刻一覧の設定項目が増減します。	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
チャット機能	チャット機能を使用する場合「ON」にします。	OFF

詳細については、8.5. 「アプリケーション名.properties」ファイル 参照

5.5. 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 5-6）。

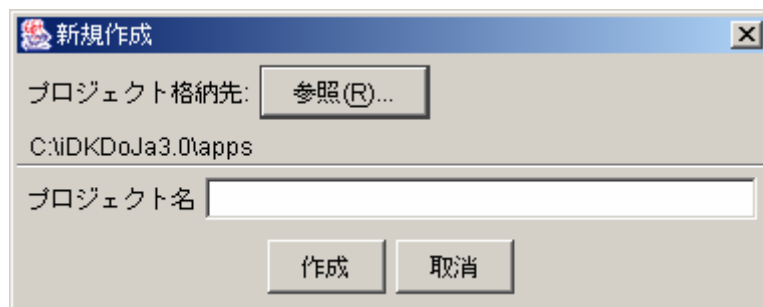


図 5-6 プロジェクト新規作成画面

2. プロジェクトの格納先（C:\iDKDoJa3.0\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:\iDKDoJa3.0」にインストールした場合、「C:\iDKDoJa3.0\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 5-7）。

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：

^[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：

「i aplli Development Kit for DoJa3.0」を C:\iDKDoJa3.0 にインストールした場合

ソースコードの格納場所: C:\iDKDoJa3.0\apps\<プロジェクト名>\src

リソースファイルの格納場所: C:\iDKDoJa3.0\apps\<プロジェクト名>\res

スクラッチパッドの格納場所: C:\iDKDoJa3.0\apps\<プロジェクト名>\sp

JAR、JAM ファイルの格納場所: C:\iDKDoJa3.0\apps\<プロジェクト名>\bin

JAM ファイル: C:\iDKDoJa3.0\apps\<プロジェクト名>\bin\<プロジェクト名>.jam

この時点での JAM ファイルの内容は「PackageURL =<プロジェクト名>.jar」のみです。また、ビルドした際に設定された ADF の項目や「5.3. アプリケーションディスクリプタファイル (ADF) の編集」で編集された ADF はこの JAM ファイルに保存されます。

「iα ppli Development Kit for DoJa3.0」を C:\iDKDoJa3.0 にインストールした場合

ソースコードの場所: “C:\iDKDoJa3.0\apps\<プロジェクト名>\src”

リソースファイルの場所: “C:\iDKDoJa3.0\apps\<プロジェクト名>\res”

スクラッチパッドの場所: “C:\iDKDoJa3.0\apps\<プロジェクト名>\sp”

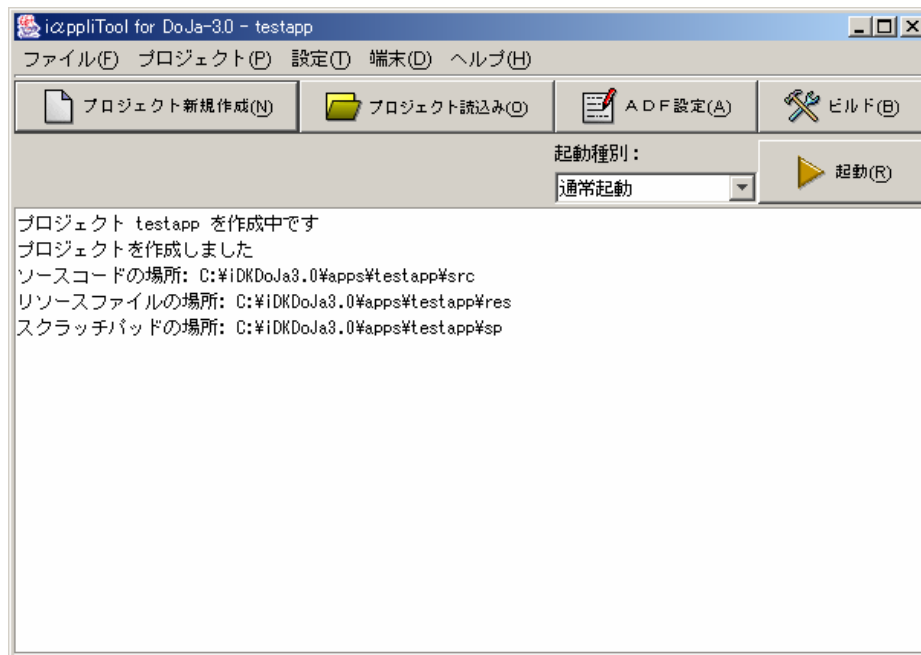


図 5-7 新規プロジェクト作成後のメイン画面

5.5.1. ソースコードの作成

ソースコードを記述したソース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。
既にソースコードがある場合は、直接ディレクトリに置いてください。

「ソースファイルの新規作成」が表示されます（図 5-8）。



図 5-8 ソースファイルの新規作成画面

ソースファイル名を入力して、「作成」を押します。
実行時に参照するリソースファイルがあれば直接ディレクトリに置いてください。

「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。

テンプレートの種類には、

- 「通常アプリケーション（IApplication）」
- 「待ち受けアプリケーション（MApplication）」

を選択する事ができます。^[4]

待ち受けアプリケーションに関しては、「5.7. 待ち受け起動」で説明します。

^[4] 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、"{DATE}"が作成日時に変更されます。

「iα ppli Development Kit for DoJa3.0」を C:\iDKDoJa3.0 にインストールした場合

- 通常アプリケーション：
C:\iDKDoJa3.0\lib\IApplication.txt
- 待ち受けアプリケーション：
C:\iDKDoJa3.0\lib\MApplication.txt

2. 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、5.5.3 ソースエディタの設定 参照) ※1

(例)

"HelloiAppli" と入力し、通常アプリケーションのテンプレートで作成した場合、図 5-9 のようにファイルが開かれます。

ソースコードは、

C:\iDKDoJa3.0\apps\<プロジェクト名>\src\HelloiAppli.java
で作成されます。

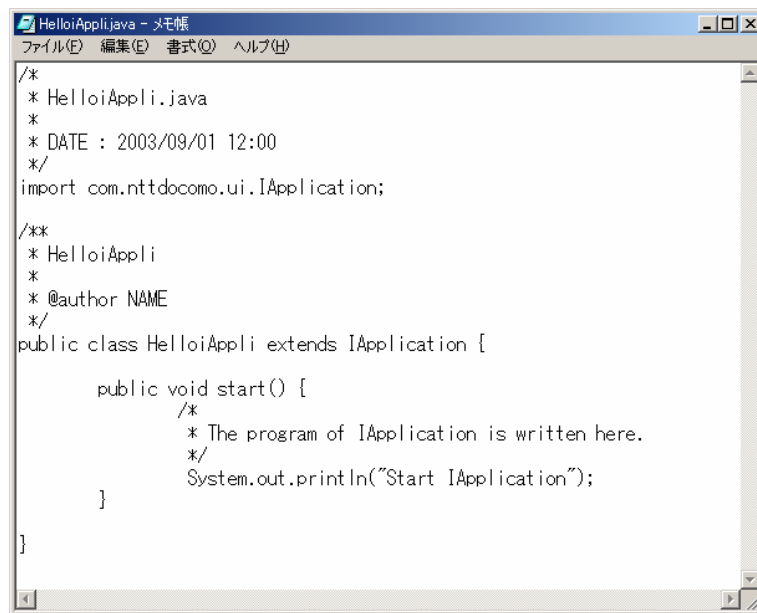


図 5-9 (例) Hello iAppli ソースコード

※1

ソースコードを作成するには、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/

5.5.2. 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 (5.6 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 5-10）。

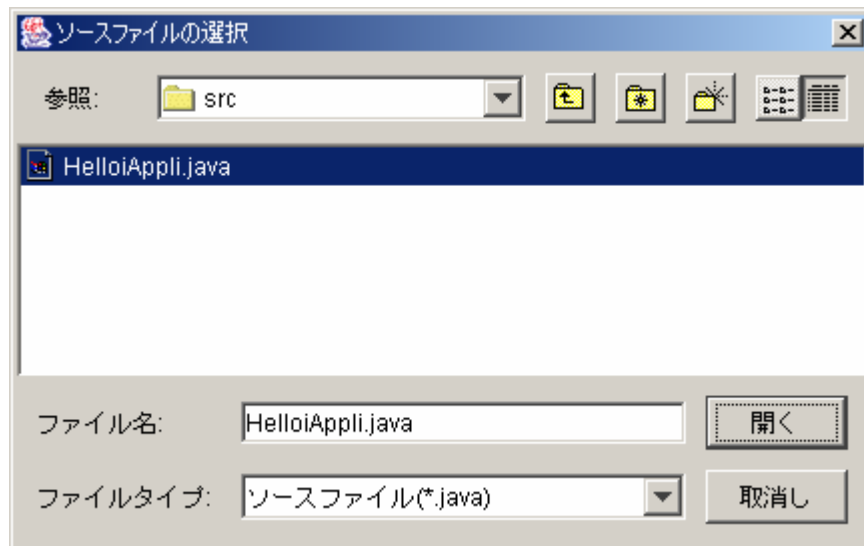


図 5-10 ソースファイルの選択画面

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、5.5.3 ソースエディタの設定 参照)

5.5.1 ソースコードの作成の図 5-9（例）Hello iAppli ソースコードで作成したソースコードの場合は、図 5-11 のように開かれます。

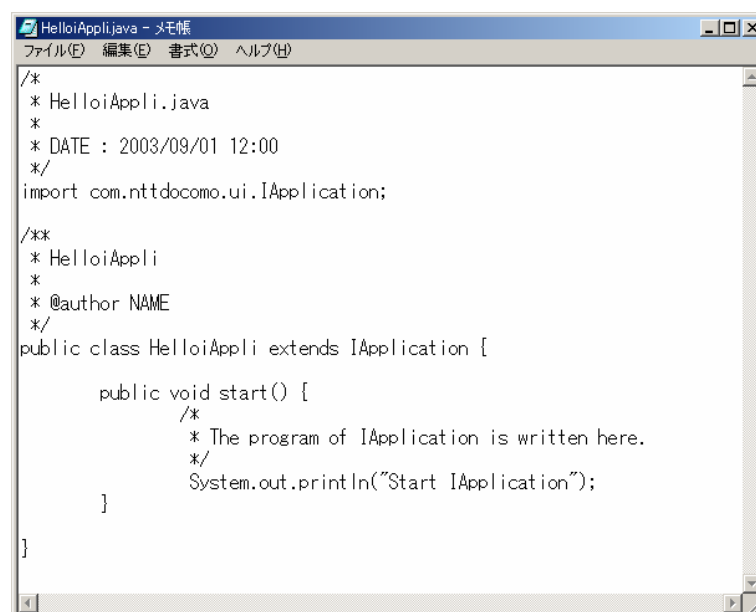


図 5-11 エディタで開かれたソースコード

5.5.3. 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。
2. 「ソースエディタの設定画面」が開くので、「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（図 5-12）。

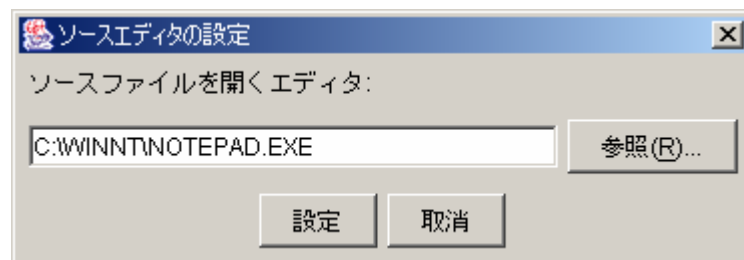


図 5-12 ソースエディタの設定画面

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。

5.6. 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 5-13）。

ソースコードのコンパイルと事前検証を行い、JAR ファイルと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^[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

（「5.3. 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）。

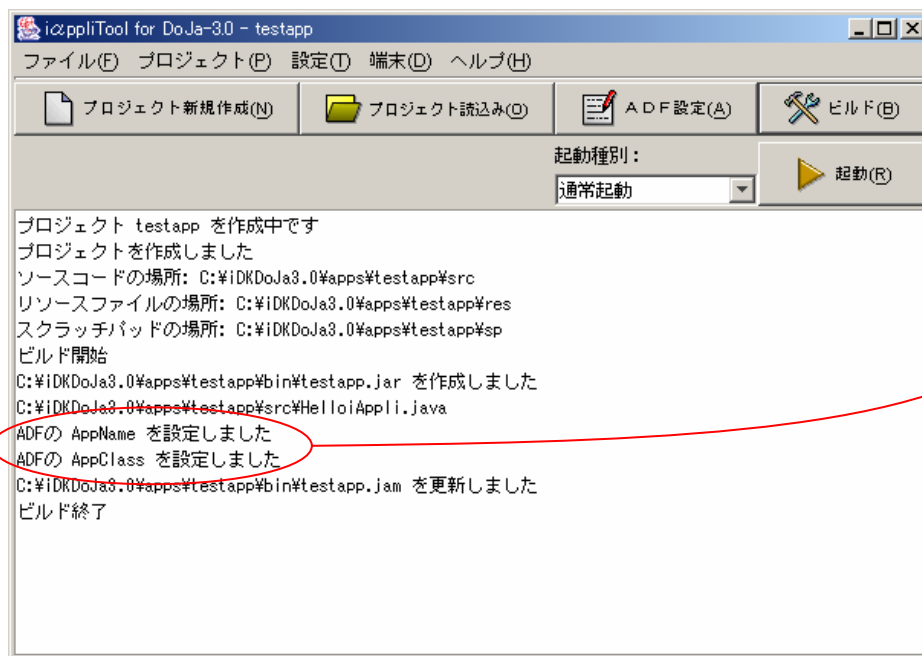


図 5-13 プロジェクトのビルド

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 5-14）。

アプリケーションを起動するために必要な ADF 項目を入力し、「設定」ボタンを押します（「5.3. 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）。

（図 5-9 の例の場合、1.で必須項目が自動的に設定されるので、特に設定する必要はありません。）

^[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。

図 5-14 ADF の設定

3. 「起動」ボタンを押します。

ビルドしたアプリケーションを選択した端末で起動します（図 5-15）。このとき、エミュレータが以下の状態の時にはアプリケーションを起動できません:

- メーラ（「5.17. メーラ起動」を参照）が起動している
- ブラウザ（「5.18. ブラウザ起動」を参照）が起動している
- コードリーダー（「5.19. コードリーダー起動」を参照）が起動している
- 待ち受け起動以外でアプリケーションが起動している
- 待ち受け起動しているアプリケーションが活性化状態あるいはサスペンド状態になっている（「5.7. 待ち受け起動」を参照）

「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。これらの起動方法に関しては「7. i アプリ連携」で説明します。

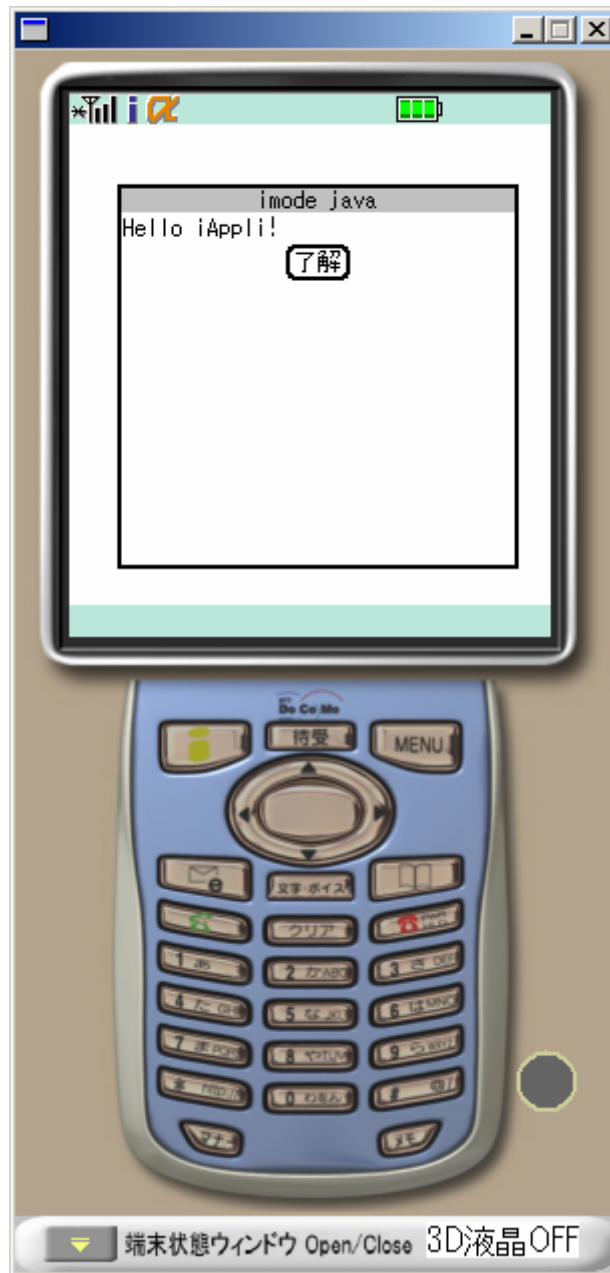


図 5-15 アプリケーション起動

参考

図 5-9 の例で、図 5-15 のような表示をするには、
`import com.nttdocomo.ui.Dialog;`
を追加し、`start()` に
`Dialog dg = new Dialog(Dialog.BUTTON_OK, "imode java");`
`dg.setText("Hello iAppli!");`
`dg.show();`
`terminate();`
を記述します。

5.7. 待ち受け起動

「待ち受け起動」とは、アプリケーションを「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。

5.7.1. 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アプリケーションのことで、`com.nttdocomo.ui.MApplication` クラスを継承している必要があります。表 5-3 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。

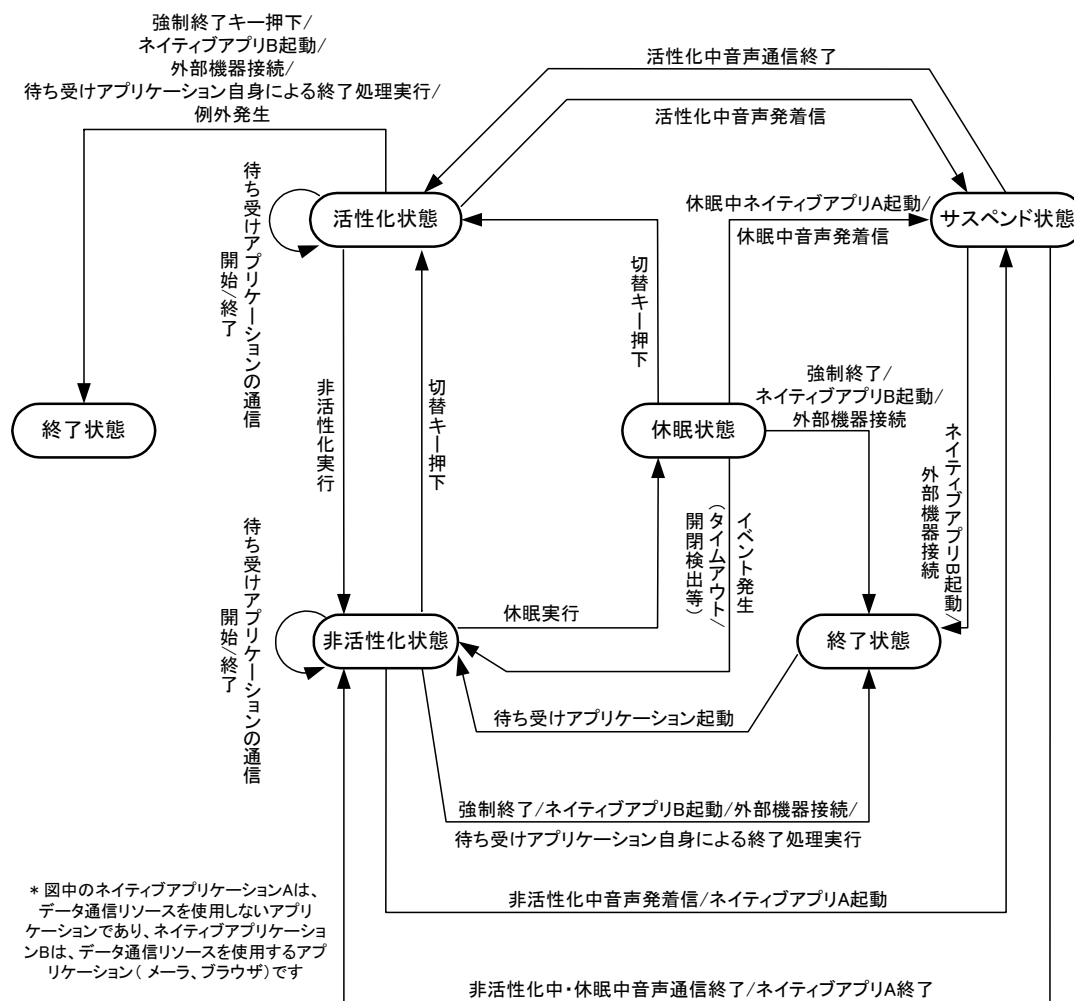
表 5-3 「待ち受け画面」と「待ち受けアプリケーション」の比較

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わ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 5-16 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：待受ボタン

図 5-16 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コードは、「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、簡単に作成する事ができます（「5.5.1. ソースコードの作成」参照）。

5.7.2. 待ち受け起動手順

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 5-17）。

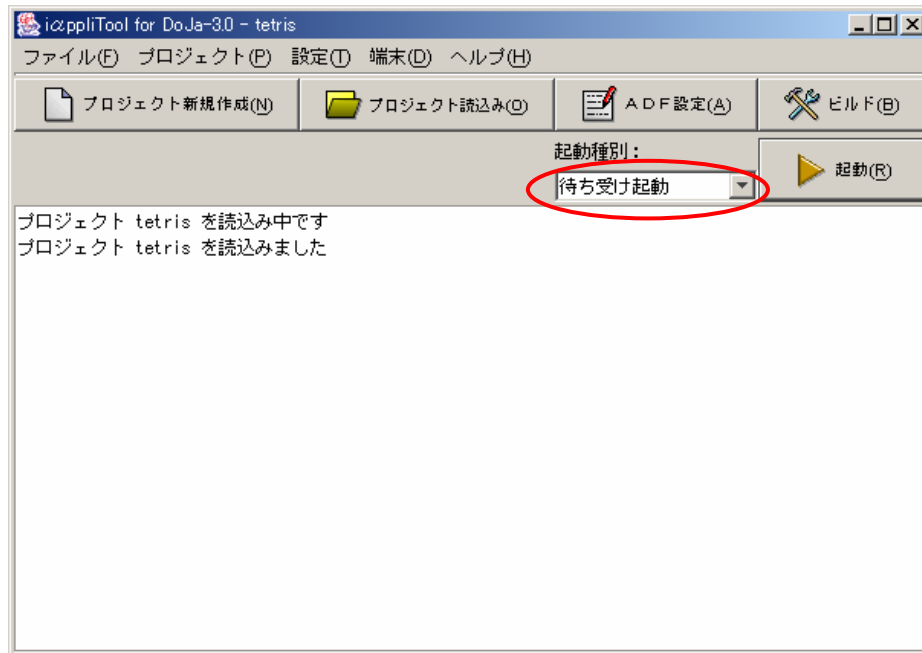


図 5-17 待ち受け起動準備

2. 「起動」ボタンを押します。

アプリケーション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：

- アプリケーション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（「5.3. 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）。
- アプリケーションやメーラ（「5.17. メーラ起動」を参照）、ブラウザ（「5.18. 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 5-4 に示します。

表 5-4 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
MApplicatoion 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、アプリケーション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、アプリケーションを待ち受けアプリケーションとして起動します（図 5-18）。

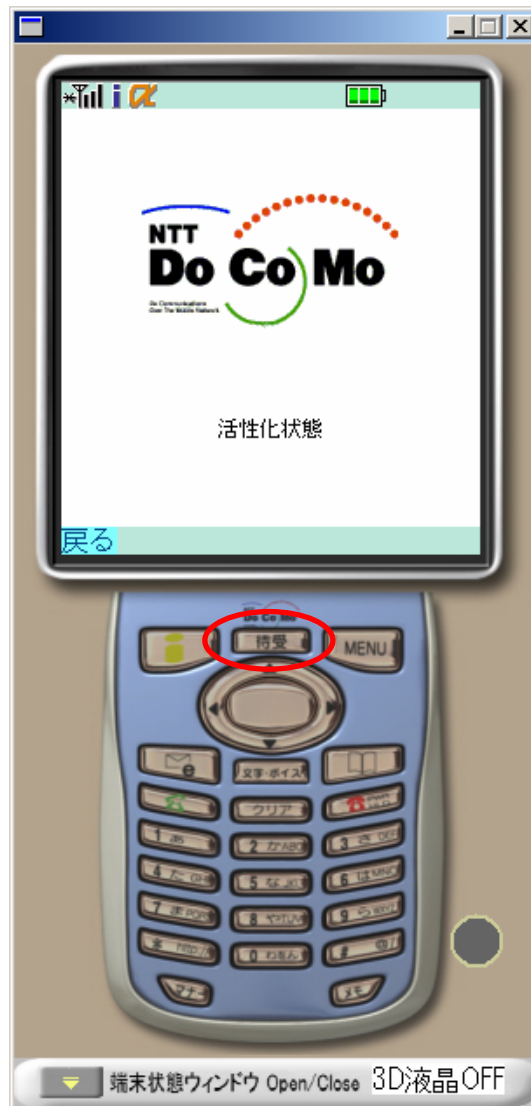


図 5-18 待ち受けアプリケーション起動

3. 待ち受けアプリケーションが「非活性化状態」（起動時）で、「待受」ボタンを押すと「活性状態」になります。
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」以外のアプリケーション
 - メーラ（「5.17. メーラ起動」を参照）
 - ブラウザ（「5.18. ブラウザ起動」を参照）
 - コードリーダー（「5.19. 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以前起動していた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 5-19）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:\iDKDoJa3.0\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 5-19 （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。）

5.8. 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アプリケーションを「iα ppliTool」の「起動」ボタンから直接起動する起動方法の1つですが、「通常起動」とは以下の点のみ異なります：

- 「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（「5.3. 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5-20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」を選択して「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（「5.6. プロジェクトのビルドと通常起動」参照）。

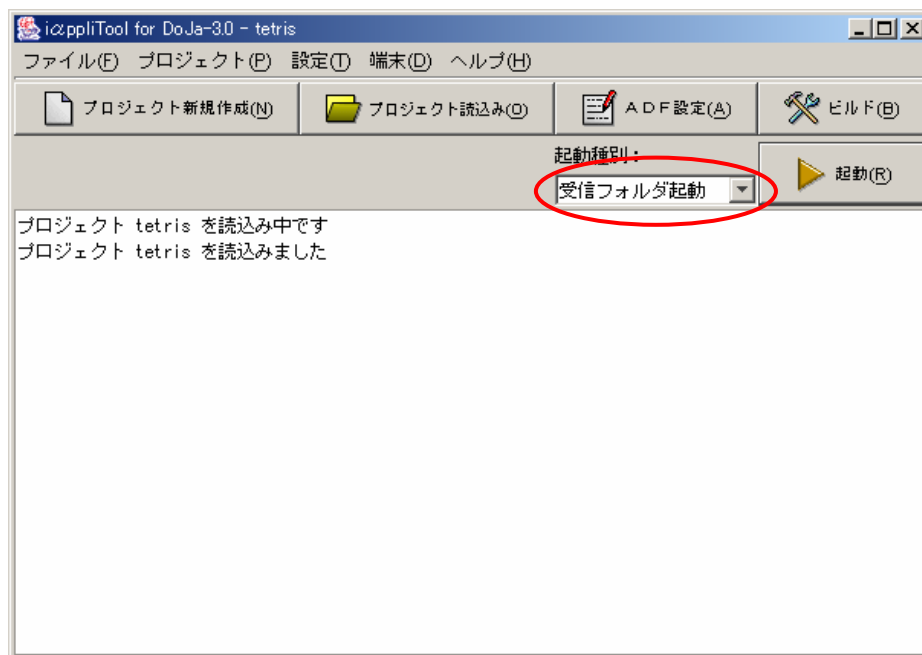


図 5-20 受信フォルダ起動準備

5.9. 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アプリケーションを「iα ppliTool」の「起動」ボタンから直接起動する起動方法の1つですが、「通常起動」とは以下の点のみ異なります：

- 「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（「5.3. 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5-21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」を選択して「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（「5.6. プロジェクトのビルドと通常起動」参照）。

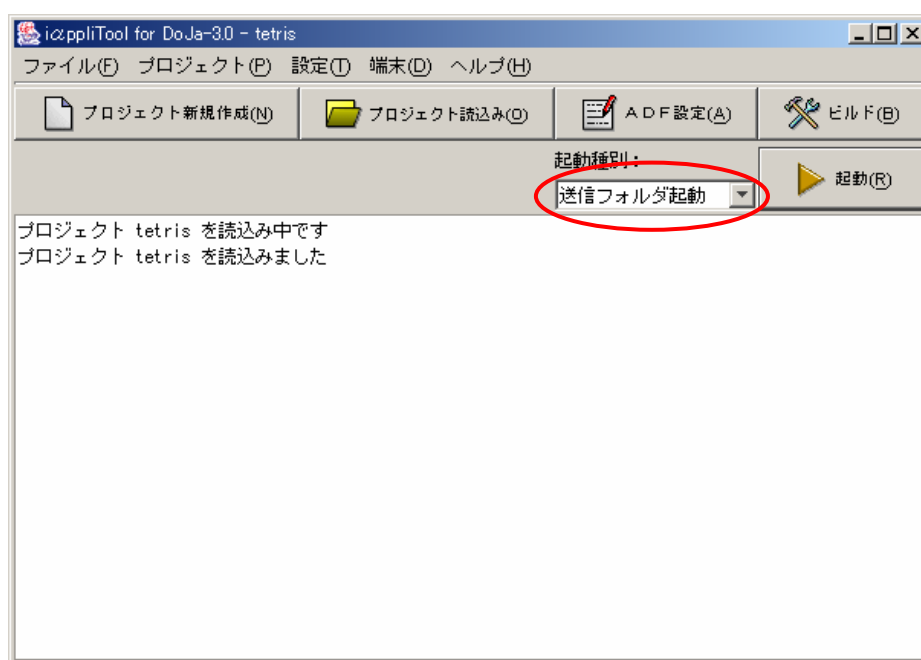


図 5-21 送信フォルダ起動準備

5.10. 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アプリケーションを「iα ppliTool」の「起動」ボタンから直接起動する起動方法の1つですが、「通常起動」とは以下の点のみ異なります：

- 「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（「5.3. 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 5-22 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」を選択して「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（「5.6. プロジェクトのビルドと通常起動」参照）。

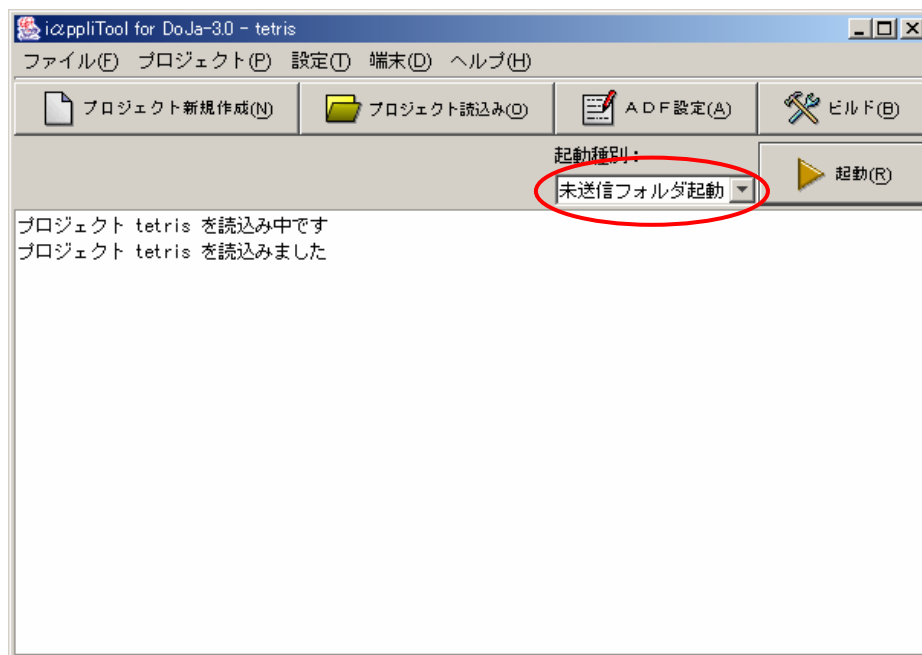


図 5-22 未送信フォルダ起動準備

5.11. 位置情報起動

位置情報起動とは、com.nttdocomo.ui.IApplication クラスを継承しているアプリケーションを「iαppliTool」の「起動」ボタンから直接起動する起動方法の1つですが、「通常起動」とは以下の点のみ異なります：

- 「位置情報起動」の起動タイプは「LAUNCHED_FROM_LOCATION_INFO」です。
- ADF の設定項目「TrustedAPIID」に適切な値を設定し、「トラステッド動作設定」→「位置情報取得APIの許可」をyes に設定した場合のみ起動可能となります（「5.3. 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、「位置情報起動」とは位置メモからの起動に相当するものです。

位置情報起動するには、

1. 位置情報から起動するアプリケーションのプロジェクトを読み込みます。
2. メイン画面のプルダウンから「位置情報起動」を選択します。
（図 5-23）

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（「5.6. プロジェクトのビルドと通常起動」参照）。

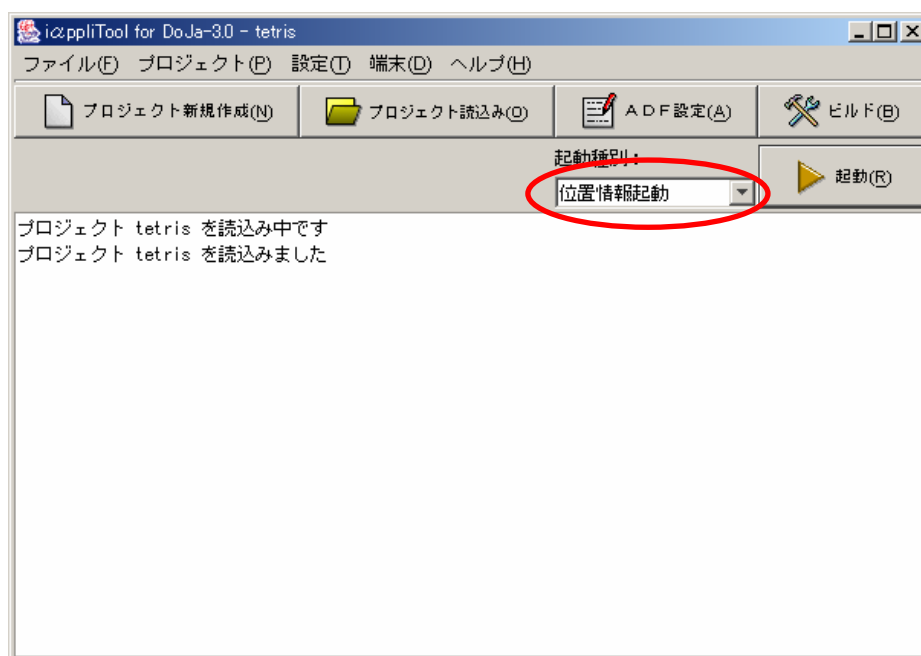


図 5-23 位置情報起動準備

3. 「起動」ボタンを押します。

位置情報確認画面が表示されます（図 5-24）。

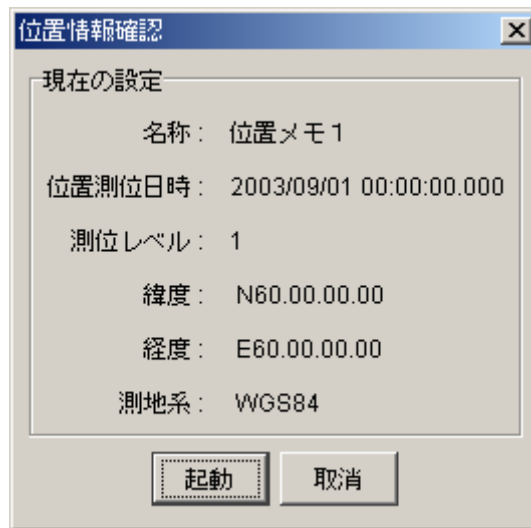


図 5-24 位置情報確認画面

4. 「起動」を押します。

位置情報（登録データ）を利用する旨の確認画面が表示されます（図 5-25）。

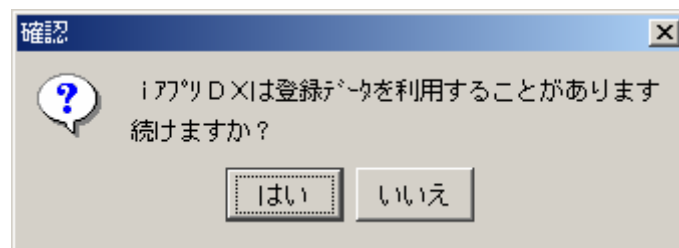


図 5-25 登録データ利用確認画面

5. 「はい」を押すとアプリケーションが起動します。

5.12. 位置情報画像起動

位置情報画像起動とは、com.nttdocomo.ui.IApplication クラスを継承しているアプリケーションを「iα ppliTool」の「起動」ボタンから直接起動する起動方法の1つですが、「通常起動」とは以下の点のみ異なります：

- 「位置情報画像起動」の起動タイプは「LAUNCHED_FROM_LOCATION_IMAGE」です。
- ADF の設定項目「TrustedAPID」に適切な値を設定し、「トラステッド動作設定」→「位置情報取得A P I の許可」を yes に設定した場合のみ起動可能となります（「5.3. 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、「位置情報画像起動」とは位置情報埋め込み画像からの起動に相当するものです。

位置情報画像起動をするには、

1. 位置情報画像から起動するアプリケーションのプロジェクトを読み込みます。
2. メイン画面のプルダウンから「位置情報画像起動」を選択します（図 5-26）。

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（「5.6. プロジェクトのビルドと通常起動」参照）。

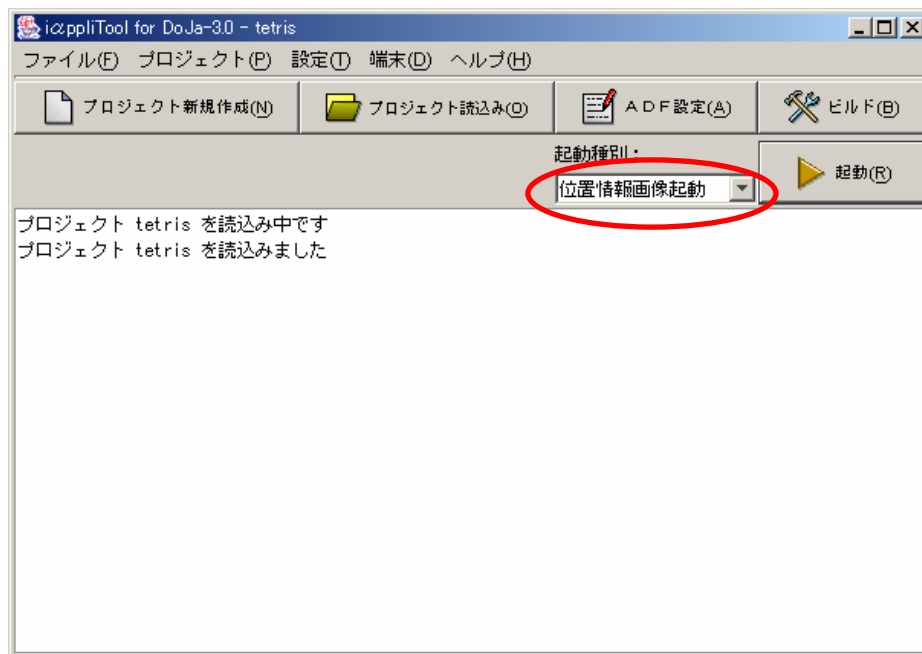


図 5-26 位置情報画像起動準備

3. 「起動」ボタンを押します。

位置情報画像確認画面が表示されます（図 5-27）。

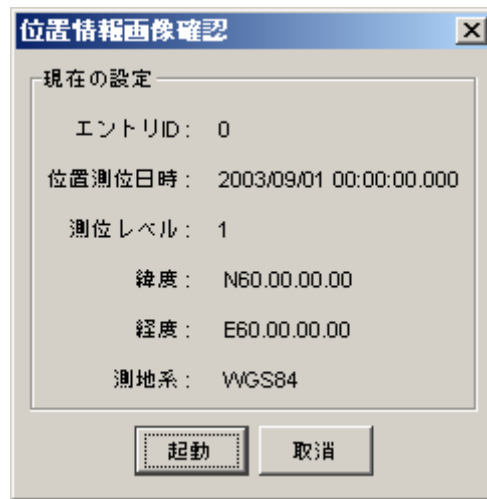


図 5-27 位置情報画像確認画面

4. 「起動」を押します。

位置情報（登録データ）を利用する旨の確認画面が表示されます（図 5-28）。

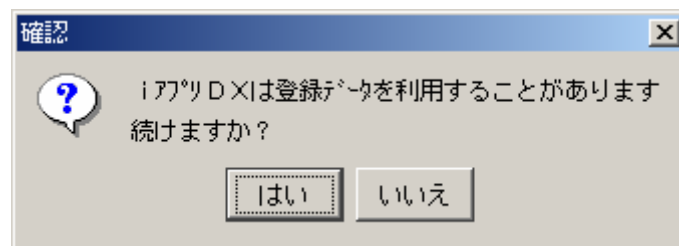


図 5-28 登録データ利用確認画面

5. 「はい」を押すとアプリケーションが起動します。

5.13. 初回起動

初回起動とは、com.nttdocomo.ui.IApplication クラスを継承しているアプリケーションを「iα ppliTool」の「起動」ボタンから直接起動する起動方法の 1 つですが、「通常起動」とは以下の点のみ異なります：

- 「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 5-29 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（「5.6. プロジェクトのビルドと通常起動」参照）。

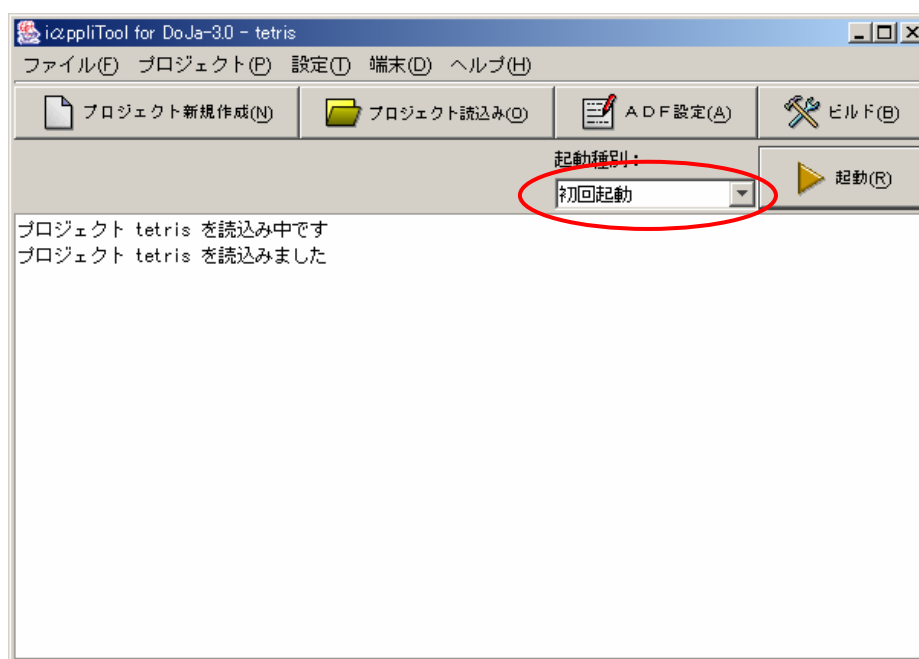


図 5-29 初回起動準備

5.14. 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アプリケーションを「iα ppliTool」の「起動」ボタンから直接起動する起動方法の1つですが、「通常起動」とは以下の点のみ異なります：

- 「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（「5.3. 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 5-30 に示すように、メイン画面のプルダウンから「タイマ起動」を選択して「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（「5.6. プロジェクトのビルドと通常起動」参照）。

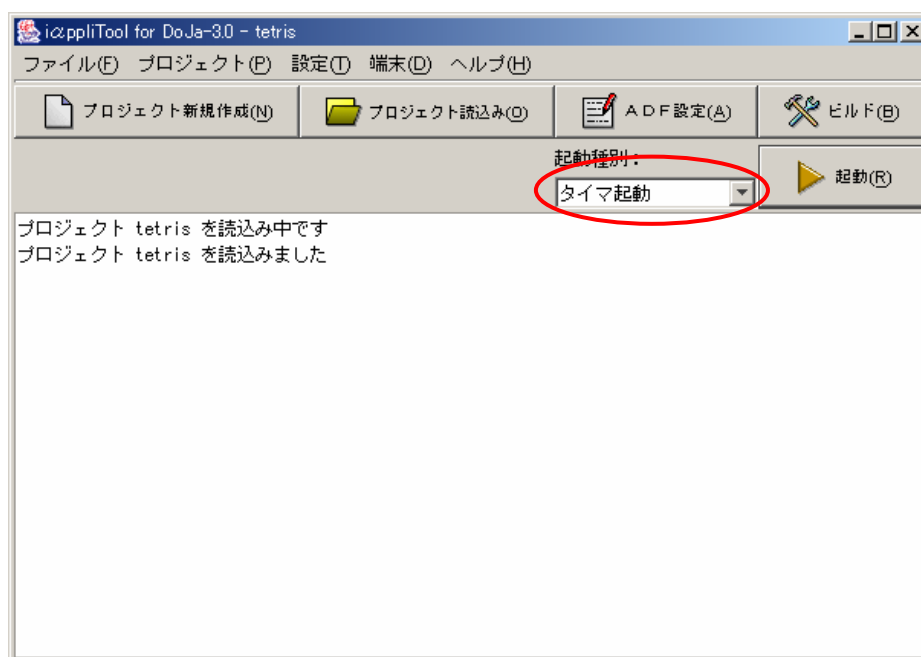


図 5-30 タイマ起動準備

5.15. 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アプリケーションを「iαppliTool」の「起動」ボタンから直接起動する起動方法の1つですが、「通常起動」とは以下の点のみ異なります：

- 「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（「5.3. 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 5-31 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」を選択して「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（「5.6. プロジェクトのビルドと通常起動」参照）。

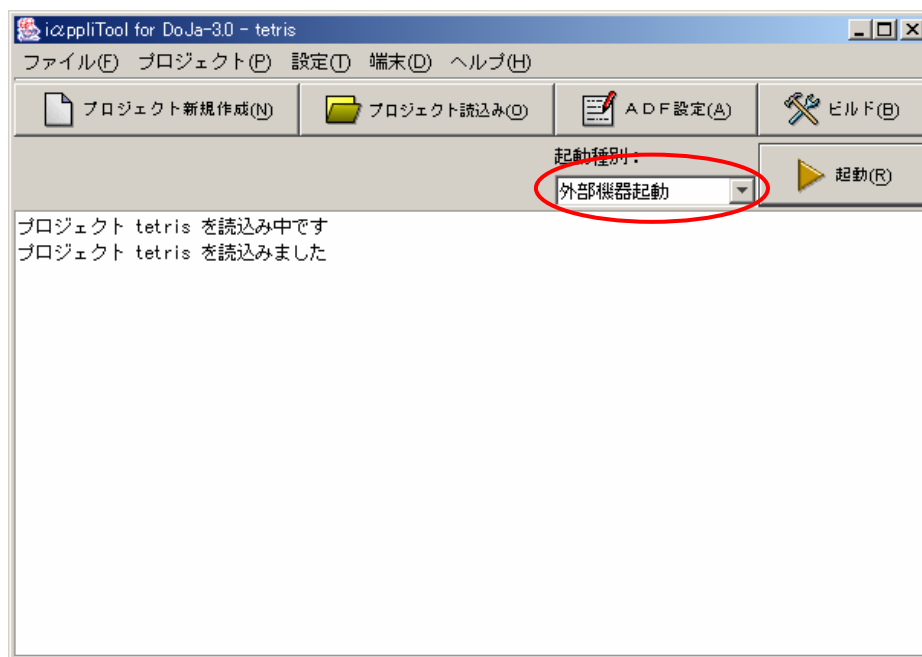


図 5-31 外部機器起動準備

もし、外部機器起動するアプリケーションが位置情報を利用するアプリケーション（トラステッド動作設定の「位置情報取得APIの許可」がyesに設定）であった場合、位置情報を利用する旨の確認画面が表示されますので、「はい」を押してください（図 5-32）。

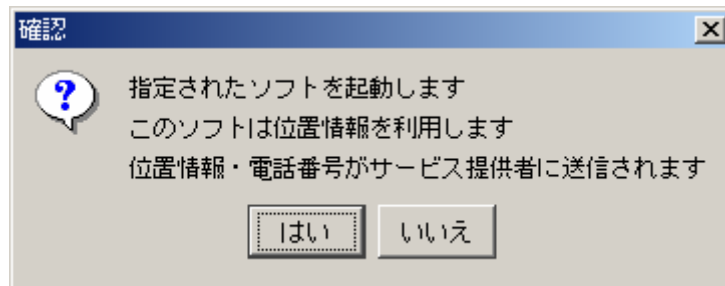


図 5-32 位置情報利用確認画面

5.16. 即起動

「即起動」とは、ダウンロード後にアプリケーション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」で起動されたアプリケーションは、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
- アプリケーションのバージョンアップ
- アプリケーションの保存

上記の条件であった場合には、例外が発生します。

5.16.1. 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 5-33）。

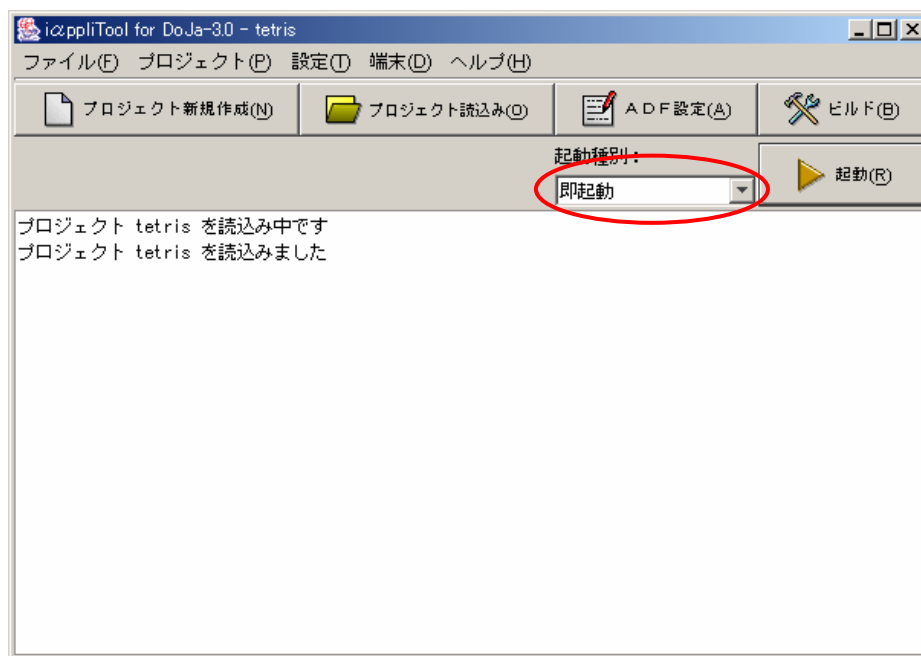


図 5-33 即起動準備

2. 「起動」ボタンを押します。

5.17. メール起動

「メール起動」とは、アプリケーションをメールから起動することです。
「メール起動」の起動タイプは、「LAUNCHED_FROM_MAILER」です。

5.17.1. メール起動手順

1. メイン画面のプルダウンから「メール起動」を選択します（図 5-34）。

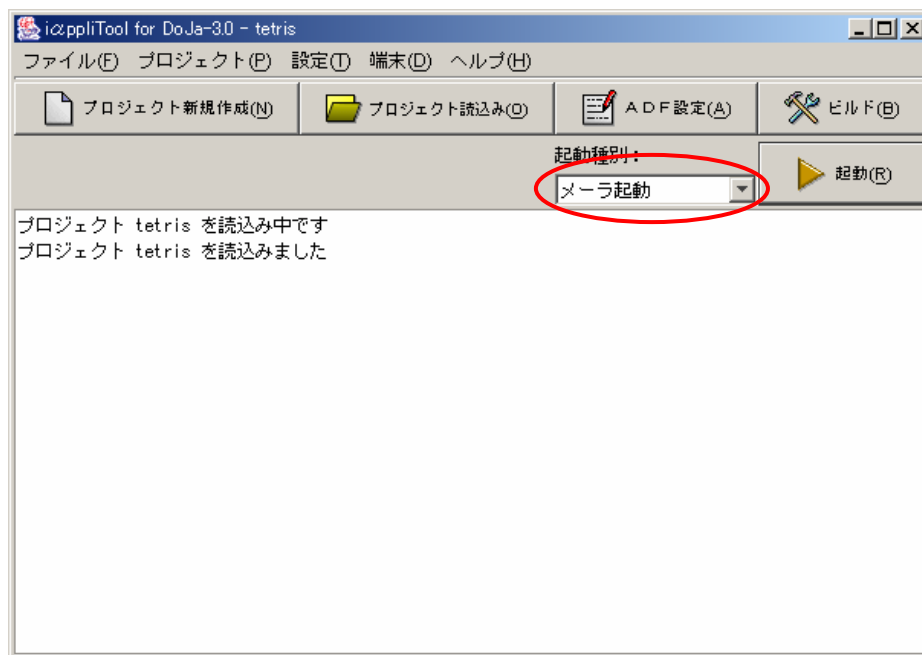


図 5-34 メール起動準備

2. 「起動」ボタンを押します。

メール画面が表示されます（図 5-35）。このとき、エミュレータが以下の状態の時にはメール画面は表示されません:

- 既にメールが起動している。
- ブラウザ（「5.18. ブラウザ起動」を参照）が起動している。
- コードリーダー（「5.19 コードリーダー起動」を参照）が起動している。
- アプリケーションが待ち受け起動以外で起動している。（「5.6. プロジェクトのビルドと通常起動」、「5.13. 初回起動」、「5.14. タイマ起動」および「5.15. 外部機器起動」を参照）
- アプリケーションが待ち受け起動している。（活性化状態あるいはサスペンド状態のとき）（「5.7. 待ち受け起動」を参照）

図 5-35 の上方には、受信メールの一覧が表示されます（「iα ppli Development Kit for DoJa-3.0」を「C:\iDKDoJa3.0」にインストールした場合、
「C:\iDKDoJa3.0\lib\mail」に置かれている「*.mail」ファイル（*は0、1、2、・・・）のヘッダ部分が表示されます（「*.mail」ファイルの記述方法は、
「8.1. 「*.mail」ファイル」を参照））。

受信メールの一覧からメールを選択すると、図 5-35 の下方に選択したメールの内容が表示されます。

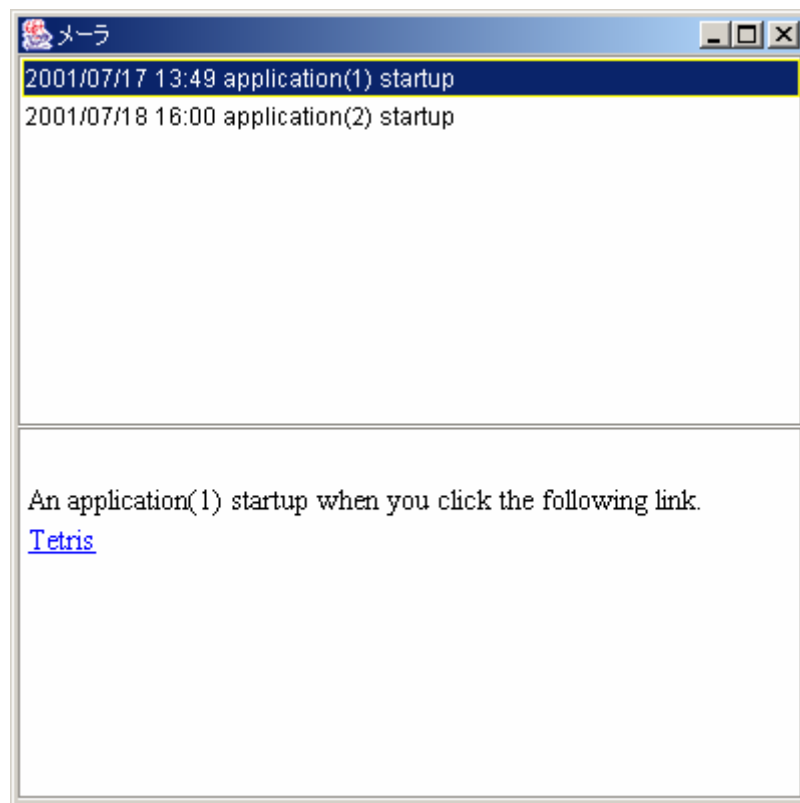


図 5-35 メール画面

3. 図 5-35 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。

アプリケーション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（「5.3. 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。
- 「*.mail」ファイルの ADF の項目をキーとしてダウンロードアプリケーションの設定（「JAM.properties」ファイル）から得られるローカルの ADF の名前（*.jam）が指定のディレクトリに置かれていること（「*.mail」ファイルの記述方法は、「8.1. 「*.mail」ファイル」を参照。ダウンロードアプリケーションの設定方法は、「4.2. ダウンロードアプリケーションの設定」を参照）。
- アプリケーションがデフォルトの格納先（「C:\iDKDoJa3.0」にインストールした場合、「C:\iDKDoJa3.0\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、アプリケーション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、アプリケーションをメーラ起動するための確認ダイアログが表示され（図 5-36）、「はい」ボタンを押すとアプリケーションが起動します（図 5-38）。

メーラから待ち受けアプリケーションを起動した場合は、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 5-36 起動確認ダイアログ

もし、外部機器起動するアプリケーションが位置情報を利用するアプリケーション（トラステッド動作設定の「位置情報取得APIの許可」がyesに設定）であった場合、位置情報を利用する旨の確認画面が表示されますので、「はい」を押してください（図 5-37）。

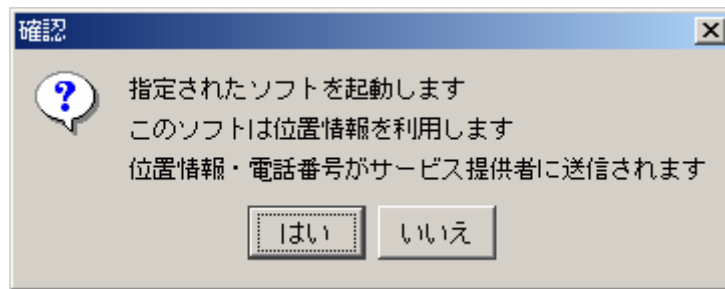


図 5-37 位置情報利用確認画面



図 5-38 メーラ起動されたアプリケーション

4. メーラ起動されたアプリケーションの終了方法によりその後自動起動するアプリケーションが異なります:

- メーラ起動されたアプリケーションを正常終了（アプリケーションが `UIApplication.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。

- メーラ起動されたアプリケーションを強制終了（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。

5.18. ブラウザ起動

「ブラウザ起動」とは、アプリケーション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。

5.18.1. ブラウザ起動手順

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 5-39）。

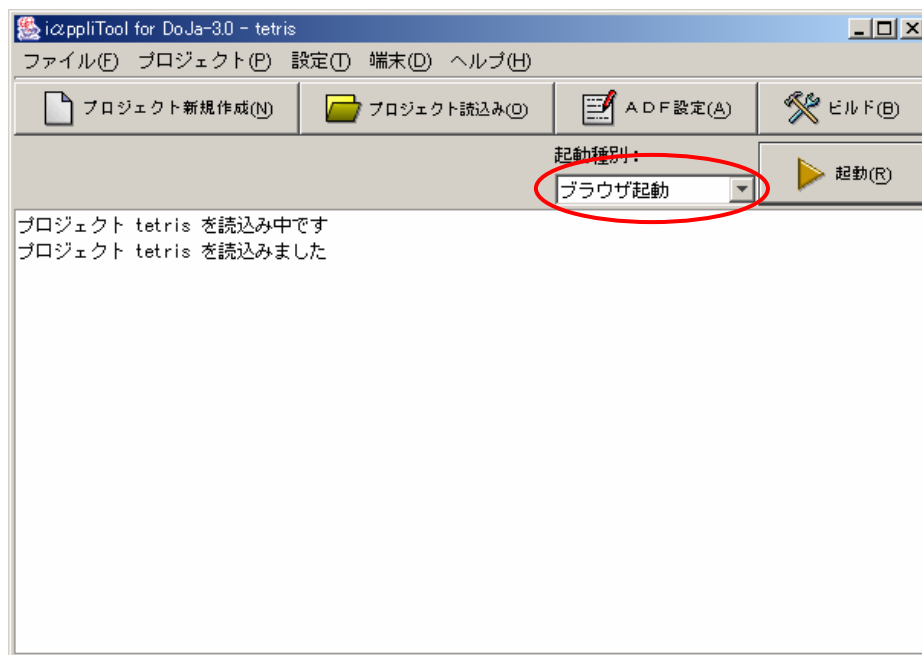


図 5-39 ブラウザ起動準備

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 5-40）。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（「5.17 メーラ起動」を参照）が起動している。
- コードリーダー（「5.19 コードリーダー起動」を参照）が起動している。
- アプリケーションが待ち受け起動以外で起動している。（「5.3. アプリケーションディスクリプタファイル（ADF）の編集」、「5.13. 初回起動」、「5.14. タイマ起動」および「5.15. 外部機器起動」を参照）
- アプリケーションが待ち受け起動している。（活性化状態あるいはサスペンド状態のとき）（「5.7. 待ち受け起動」を参照）

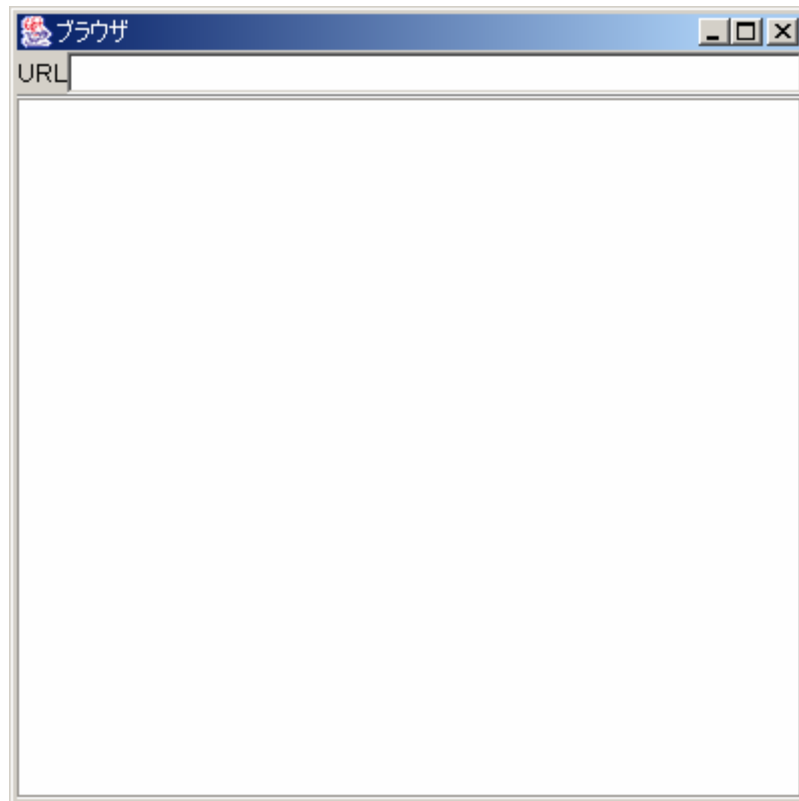


図 5-40 ブラウザ画面

3. 図 5-40 の上方の URL フィールドにブラウザ起動するための URL（この URL で指定した HTML ファイルの記述方法は「8.2. 「HTML」 ファイル」を参照）を入力し、リターンキーを押します（図 5-41）。



図 5-41 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:/iDKDoJa3.0/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、「file:」で指定できます。
その場合、URL に「**file:C:/iDKDoJa3.0/lib/sample.html**」と指定します。

4. 図 5-41 のリンク（青い文字）をクリックします。

アプリケーション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（「5.3. 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 ファイルの OBJECT タグの中に設定されている data の URL をキーとしてダウンロードアプリケーションの設定（「JAM.properties」ファイル）から得られるローカルの ADF の名前（*.jam）が指定のディレクトリに置かれていること（「HTML」ファイルの記述方法は、「8.2. 「HTML」 ファイル」を参照。ダウンロードアプリケーションの設定方法は、「4.2. ダウンロードアプリケーションの設定」を参照）。
- アプリケーションがデフォルトの格納先（「C:\iDKDoJa3.0」にインストールした場合、「C:\iDKDoJa3.0\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、アプリケーション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、アプリケーション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 5-42）、「はい」ボタンを押すとアプリケーションが起動します（図 5-44）。

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は、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。

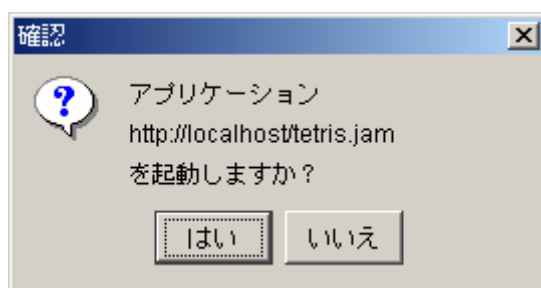


図 5-42 起動確認ダイアログ

もし、外部機器起動するアプリケーションが位置情報を利用するアプリケーション（トラステッド動作設定の「位置情報取得APIの許可」が yes に設定）であった場合、位置情報を利用する旨の確認画面が表示されますので、「はい」を押してください（図 5-43）。

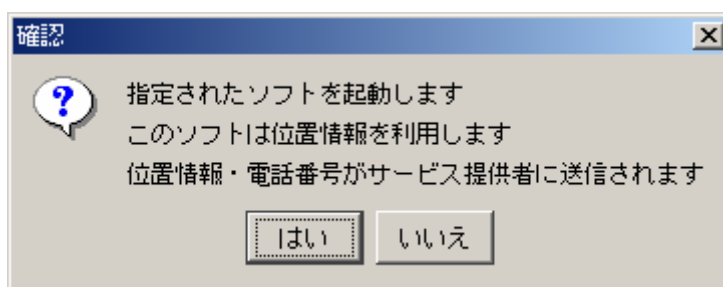


図 5-43 位置情報利用確認画面



図 5-44 ブラウザ起動されたアプリケーション

5. ブラウザ起動されたアプリケーションの終了方法によりその後自動起動するアプリケーションが異なります:

- ブラウザ起動されたアプリケーションを正常終了（アプリケーションが `UIApplication.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アプリケーションを強制終了（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。

5.19. コードリーダー起動

「コードリーダー起動」とは、アプリケーション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。

5.19.1. コードリーダー起動手順

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 5-45）。

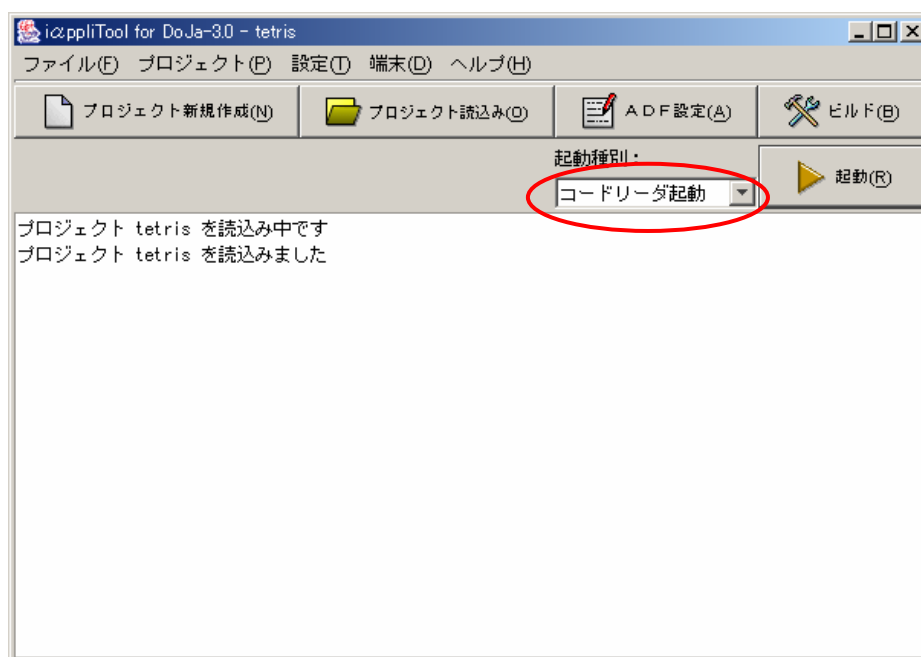


図 5-45 コードリーダー起動準備

2. 「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 5-46）。

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（「5.17. メーラ起動」を参照）が起動している。
- ブラウザ（「5.18. ブラウザ起動」を参照）が起動している。
- アプリケーションが待ち受け起動以外で起動している。（「5.3. アプリケーションディスクリプタファイル（ADF）の編集」、「5.13. 初回起動」、「5.14. タイマ起動」および「5.15. 外部機器起動」を参照）
- アプリケーションが待ち受け起動している。（活性化状態あるいはサスペンド状態のとき）（「5.7. 待ち受け起動」を参照）

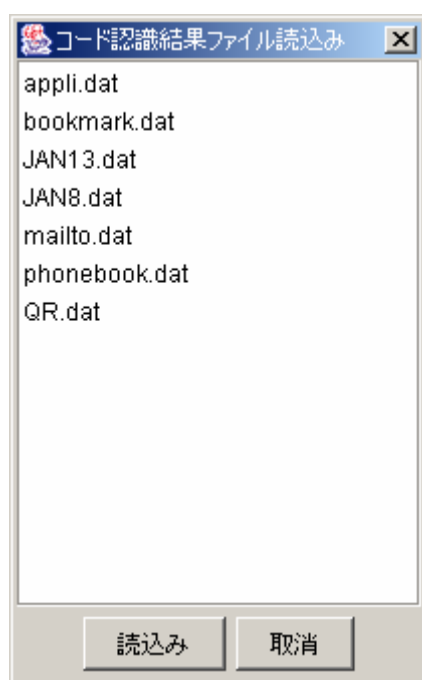


図 5-46 コード認識結果ファイル読み込み画面

3. 「図 5-46」 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。

アプリケーション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:」あるいは、「Code:起動コマンド」が設定されていること（「5.3. 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）。
- コード認識結果ファイルに指定された i アプリ連携機能の識別子「ADFURL:」をキーとしてダウンロードアプリケーションの設定（「JAM.properties」ファイル）から得られるローカルの ADF の名前（*.jam）が指定のディレクトリに置かれていること（「コード認識結果ファイル」の記述方法は、「8.8. コード認識結果ファイル」を参照。ダウンロードアプリケーションの設定方法は、「4.2. ダウンロードアプリケーションの設定」を参照）。
- アプリケーションがデフォルトの格納先（「C:\iDKDoJa3.0」にインストールした場合、「C:\iDKDoJa3.0\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、アプリケーション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 5-47）。

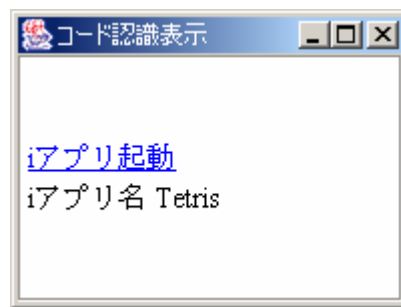


図 5-47 コード認識表示画面

4. 図 5-47 のリンク（青い文字）をクリックします。

アプリケーション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」ボタンを押すとアプリケーションが起動します（図 5-48）。



図 5-48 コードリーダー起動されたアプリケーション

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。

5.19.2. CodeReader の実行手順

「iα ppli Development Kit for DoJa-3.0」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 5-49）。

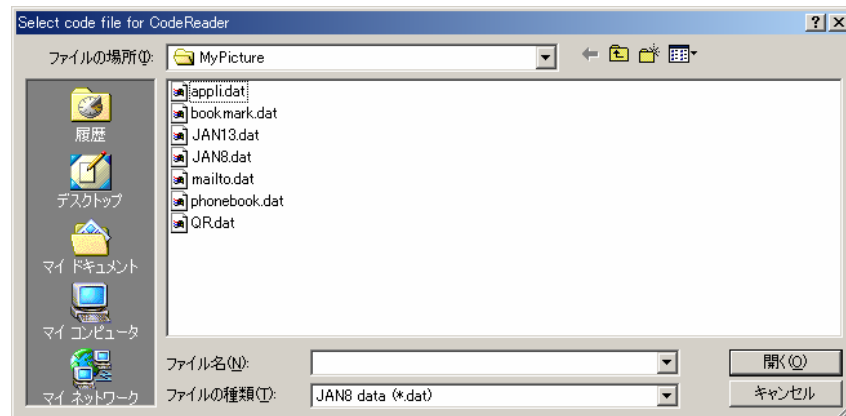


図 5-49 コード認識結果ファイル 指定ダイアログ

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。

5.20. 各種起動方法による画面遷移

これまで、「iαppliTool」でのアプリケーションの起動方法には、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
- 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
- 位置情報起動
- 位置情報画像起動
- 初回起動
- タイマ起動
- 外部機器起動
- 即起動
- 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 5-50 に示します。

「iαppliTool」実行中、画面に表示されているアプリケーションは、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。

- 通常アプリが起動していない
- 待ち受けアプリケーションが起動していない、
または、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である
- 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

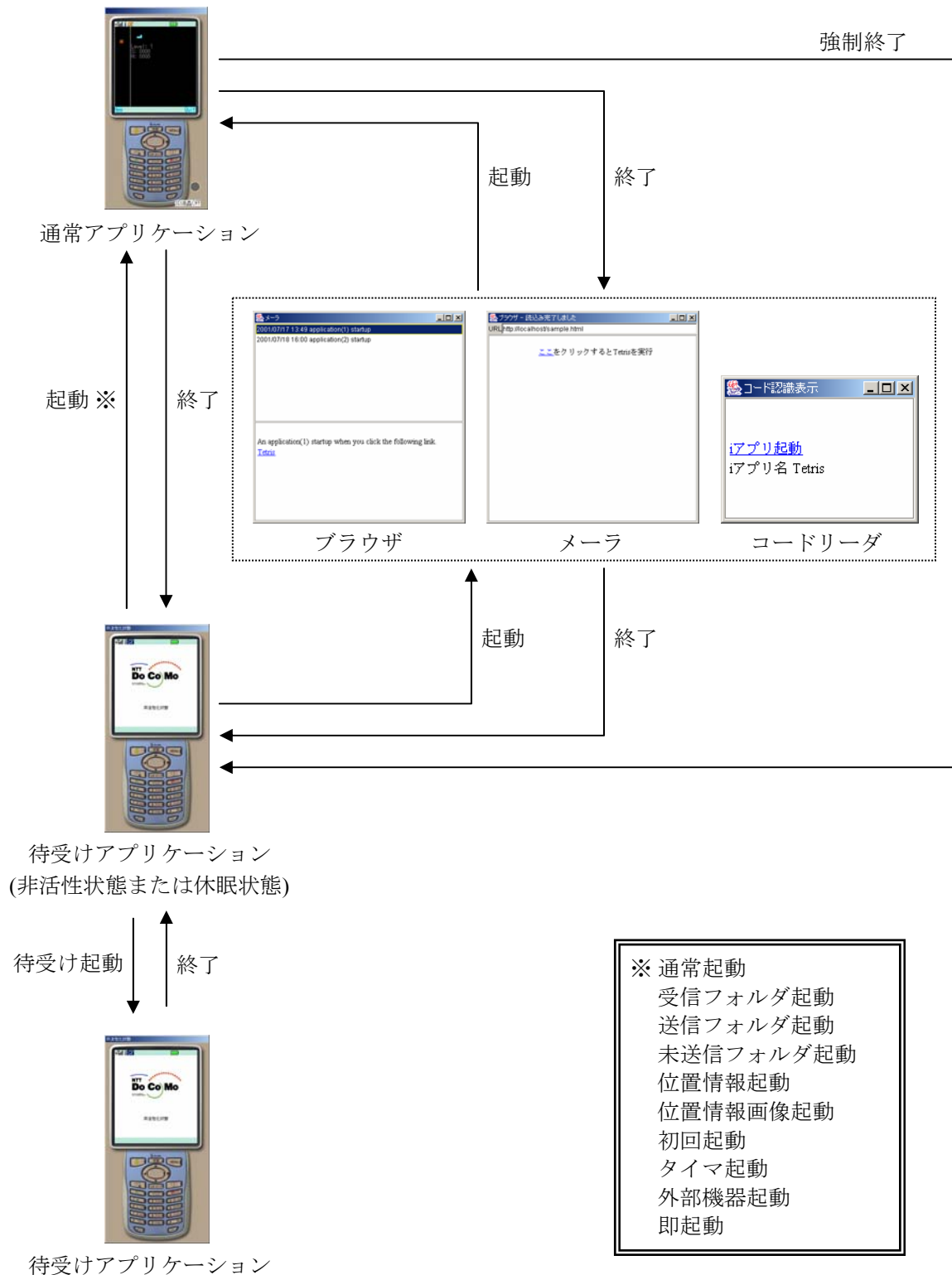


図 5-50 「iαppliTool」における画面遷移

6. コンソールと「iα ppliTool」の終了

- コンソール

起動中のアプリケーションからの標準エラー出力は、コンソールに表示されます。^[6]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。

- 「iα ppliTool」の終了

「iα ppliTool」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。

6.1. デバッグトレースの利用

高度な使い方として、C:\iDKDoJa3.0\bin\doja_g.exe を実行する事により、コンソールにKVMのデバッグトレースを出力する事ができます。

（「iα ppli Development Kit for DoJa-3.0」を C:\iDKDoJa3.0 にインストールした場合）

デバッグトレースを実行するには、doja_g.exe を実行し、メニューから「設定」→「デバッグトレース設定...」を起動してください（図 6-1）。

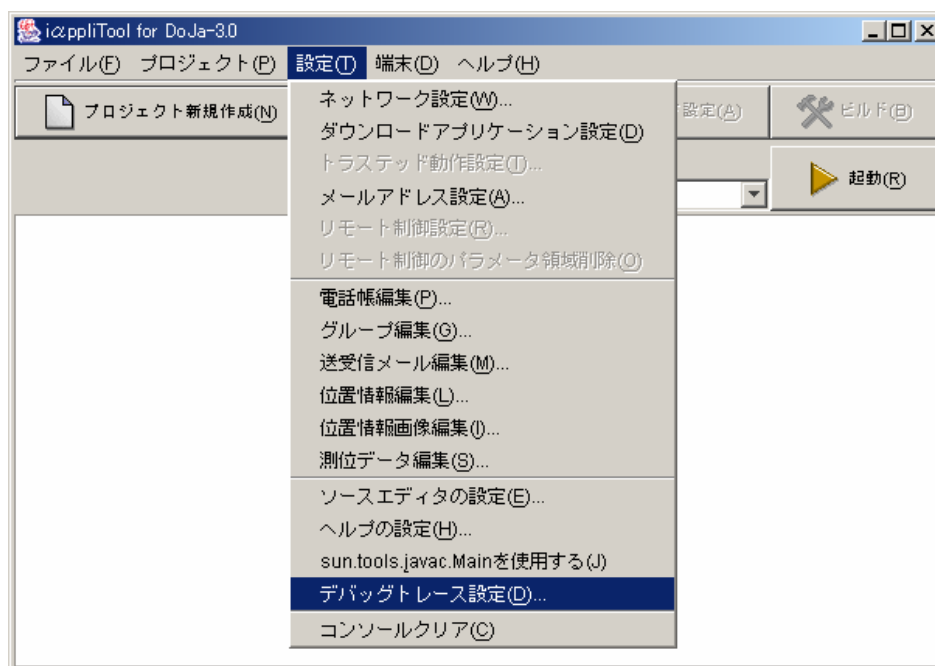


図 6-1 デバッグトレースの設定

^[6]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。

1. デバッグトレース設定画面（図 6-2）が表示されるので、デバッグトレースの出力と種類を選び、「設定」を押します。
選択できるデバッグトレースの出力は以下のとおりです。

- 「無効」 デバッグトレースを出力しません。
- 「有効（全て）」 全てのトレースを出力します。
- 「有効（選択）」 トレースの種類を「デバッグトレース種類」から選択できます。
複数選択したい場合は、[Ctrl]キーを押しながら選択してください。

【トレースの種類】

```

traceallocation.....trace memory allocation
tracegc.....trace garbage collection
tracegcverbose.....trace garbage collection, more verbose
traceclassloading.....trace class loading
traceclassloadingverbose.....trace class loading, more verbose
traceverifier.....trace class file verifier
tracestackmaps.....trace the behavior of stack maps
tracebytecodes.....trace bytecode execution
tracemethods.....trace method calls
tracemethodsverbose.....trace method calls, mode verbose
tracestackchunks.....trace the allocation of new stack chunks
traceframes.....trace stack frames
traceexceptions.....trace exception handling
traceevents.....trace the behavior of the event system
tracemonitors.....trace the behavior of monitor objects
tracethreading.....trace the behavior of the multithreading system
tracenetworking.....trace the network access
    
```

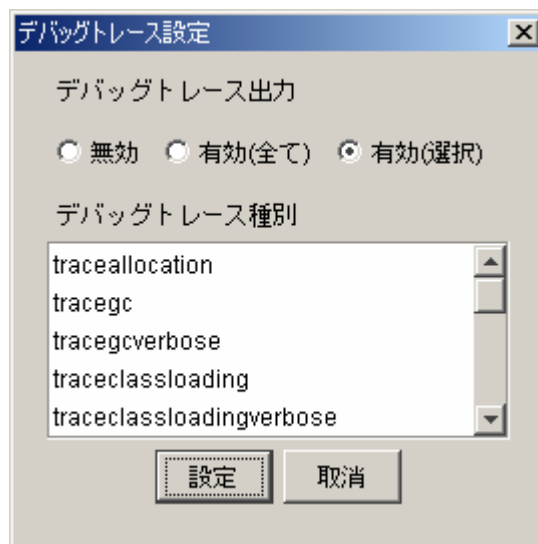


図 6-2 デバッグトレース設定画面

2. トレースしたいプロジェクトを読み込んで起動します。
(5.2. 既存プロジェクトの読み込み を参照してください。)
3. トレース結果がコンソールに出力されます。

7. iアプリ連携

「i アプリ連携」とは、アプリケーションから別のアプリケーション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、起動元の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、起動元の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。

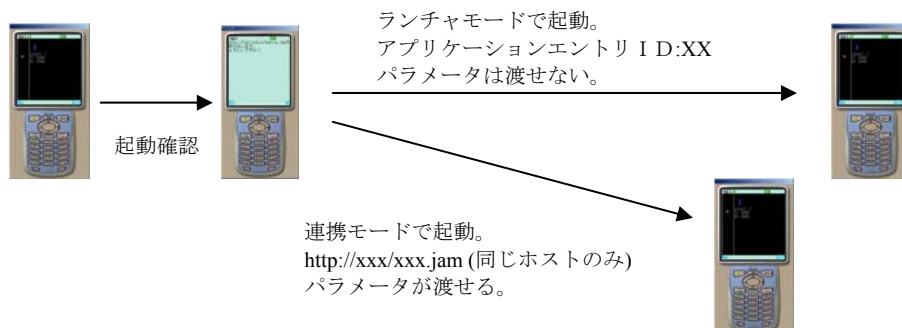


図 7-1 i アプリ連携の実行

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」（「4.3 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（「5.3. 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として、ダウンロードアプリケーションの設定（「JAM.properties」ファイル）から得られるローカルの ADF の名前（*.jam）が指定のディレクトリに置かれていること（ダウンロードアプリケーションの設定方法は、「4.2. ダウンロードアプリケーションの設定」を参照）
- 起動先のアプリケーションがデフォルトの格納先（「C:\idKDoJa3.0」にインストールした場合、「C:\idKDoJa3.0\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。

8. 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:

「iα ppli Development Kit for DoJa-3.0」を「C:\iDKDoJa3.0」にインストールした場合、

- 「*.mail」ファイル:
C:\iDKDoJa3.0\lib\mail\
 - 「JAM.properties」ファイル:
C:\iDKDoJa3.0\lib\
 - 「HTML」ファイル:
Web サーバの適切な場所
 - 着信履歴ファイル:
C:\iDKDoJa3.0\lib\phonehistory\phonehistory.lst
- ここに、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:\iDKDoJa3.0\apps\アプリケーション名\res\アプリケーション名.properties
- メールファイル:
C:\iDKDoJa3.0\lib\mail\メールアドレス 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
- マルチメディアファイル:
C:\iDKDoJa3.0\lib\Multimedia\xxx.gif
C:\iDKDoJa3.0\lib\Multimedia\xxx.jpg
- コード認識結果ファイル:
C:\iDKDoJa3.0\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。

- 位置情報設定ファイル
C:\iDKDoJa3.0\lib\location\location_information\
 - location_information
- 測位データ設定ファイル
C:\iDKDoJa3.0\lib\location\survey_data\survey_data
- 仮想 FeliCa データファイル
C:\iDKDoJa3.0\apps\アプリケーション名<アプリケーション名>.fc

8.1. 「*.mail」ファイルの記述方法

「*.mail」ファイルは、「5.17. メーラ起動」で使します。

「*.mail」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://*****"	起動させるアプリケーションの ADF の URL
11	"A"="B"	アプリケーションに渡すパラメータ
12	"C"="D"	アプリケーションに渡すパラメータ

8.2. 「HTML」ファイルの記述方法

「HTML」ファイルは、「5.18. ブラウザ起動」で使します。

「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="App1"	OBJECT タグ
4	data="http://*****"	起動させるアプリケーションの URL
5	type="application/x-jam">	
6	<PARAM NAME="A" VALUE="B">	
7	<PARAM NAME="C" VALUE="D">	
8	</OBJECT>	
9	Download	ブラウザからアプリを連携起動。 ista="#[OBJECT タグ]" href="[OBJECT タグに対応した HTML の URL]"
10	</BODY>	
11	</HTML>	

8.3. 「JAM.properties」ファイルの記述方法

ダウンロードアプリケーションを設定します（「4.2. ダウンロードアプリケーションの設定」を参照）。

「JAM.properties」ファイルの記述方法は、以下のとおりです:

1	http://サーバ名/iappli.jam = iappli.jam	サーバ名と起動したいアプリケーションの ADF の名前
2	http://サーバ名/mappli.jam = mappli.jam	
:	:	

「http\://(ダウンロード元の URL)=(ローカルにある ADF の名前)」と記述します。

ファイルを複数許可する場合は、1 行ずつ記述します。

コロン(:)を区切り記号ではなく、文字として認識させるために必ず「http」の後に「\」をつけてください。

ランチャモードでのアプリ連携 (IApplication#launch(LAUNCH_AS_LAUNCHER)) を実行する場合、起動先アプリケーションの ID は、「JAM.properties」ファイルに記述した順に 1 から指定します。

※ ここに指定するアプリケーションは、デフォルトの格納先（「C:\iDKDoJa3.0」にインストールした場合、「C:\iDKDoJa3.0\apps」）で作成されている必要があります。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。

8.4. 着信履歴ファイル

アプリケーション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、アプリケーションから取得できるのは、新着履歴ですので、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、アプリケーションからこの文字列がそのまま取得できます。

8.5. 「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	adfUrl = http://*****	起動させるアプリケーションの ADF の URL
2	sentMailMax = 50	送信/未送信メールの最大保存件数
3	receivedMailMax = 200	受信メールの最大保存件数
4	phoneBookEntry = 10	電話帳エントリの最大保存件数
5	launchType = 0	アプリケーション読み込み時の起動タイプ
6	bookmarkEntry = 10	ブックマークの最大保存件数
7	groupEntry = 10	グループの最大保存件数
8	useChat = false	アプリケーションが「チャット機能」を扱う場合、これを true に変更します。 未設定の場合 false になります。 (注) チャット機能 (LAUNCH_CHAT) が有効の場合、 スケジューラ起動 (LAUNCH_SCHEDULER) は使えません。
9	remoteParameter =	リモート制御の受信パラメータ
10	remotePushedTime =	リモート制御の受信時間
11	launchTime =	アプリケーション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します。

8.6. メールファイル

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（「4.9. 送受信メールの編集」を参照）。

8.7. 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:\iDKDoJa3.0** にインストールした場合、**C:\iDKDoJa3.0\lib\Multimedia**になります。
ここにアプリケーション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、アプリケーションからは実際に保存されません。）

「iα ppli Development Kit for DoJa-3.0」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。

8.8. 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。

「iα ppli Development Kit for DoJa-3.0」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（8.7）に作成しておきます。

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	13桁の数字
--------------	--------

3) QR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QR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 ～ 10 ・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 26 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-ADV:[12345678901234567890123456];	TV 発信用電話番号 (複数指定可)
	《TV 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 (※)

MEBK:	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];	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

「iα ppli Development Kit for DoJa-3.0」は、KToolBar から「コードリーダー起動」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 アプリを起動する事ができます (5.19 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:ﾄﾞｺﾓ, ﾀﾚﾝ;TEL:090XXXXXXX;TEL:03XXXXXXX;EMAIL:XXX@docomo.ne.jp;;

8.9. 位置情報設定ファイル

位置情報設定ファイルは、位置情報の内容を格納するファイルです。
位置情報設定ファイルは、次のように 1 行で設定します。各カラム間カンマ区切りで指定します。

名称	位置測位日時	測位レベル	緯度(※1)	経度(※1)	測地系
位置情報の名称	ミリ秒で測定した、測位時刻と 1970 年 1 月 1 日 午前 0 時 (GMT) との差	1 ～ 3 の数値	[-90 度, 90 度]	[-180 度, 180 度]	0 (WGS84) 固定

(※1) Location#getString(Location.PREFIX_DIRECTION, LocationProvider.UNIT_DMS)により定義されているフォーマット
接頭辞は符号【(+)/- (+の時は表記無し)】・方位【N/S/W/E】両方使用できます。

【記述例】

```
位置情報 1,1057222031375,1,N60.52.45.45,W10.05.55.31,0
```

8.10. 位置情報画像設定ファイル

位置情報画像設定ファイルは、位置情報画像に埋め込まれるデータの内容を格納するファイルです。
位置情報画像設定ファイルは、次のように 1 行で設定します。各カラム間カンマ区切りで指定します。

エントリ ID	位置測位日時	測位レベル	緯度(※1)	経度(※1)	測地系
数値	ミリ秒で測定した、測位時刻と 1970 年 1 月 1 日 午前 0 時 (GMT) との差	1 ～ 3 の数値	[-90 度, 90 度]	[-180 度, 180 度]	0 (WGS84) 固定

(※1) Location#getString(Location.PREFIX_DIRECTION, LocationProvider.UNIT_DMS)により定義されているフォーマット
接頭辞は符号【(+)/- (+の時は表記無し)】・方位【N/S/W/E】両方使用できます。

【記述例】

```
0,1057222031375,1,N60.52.45.45,W10.05.55.31,0
```

8.11. 測位データ設定ファイル

測位データ設定ファイルは、LocationProvider クラスで取得される測位データを設定するファイルです。

測位データ設定ファイルは、次のように 1 行で設定します。各カラム間はカンマ区切りで指定します。

測地系	緯度(※1)	経度(※1)	予備	測位レベル	測位所要時間(※2)
0 (WGS84) 固定	[-90 度, 90 度]	[-180 度, 180 度]	0	1 ~ 3 の数値	0 ~ 整数

(※1) Location#getLatitudeString(Location.PREFIX_DIRECTION, LocationProvider.DEGREE)により定義されているフォーマット
接頭辞は符号【(+)/- (+の時は表記無し)】・方位【N/S/W/E】両方使用できます。

(※2) LocationProvider#getLocation() 実行時、測位データ取得までに要する時間です。

【記述例】

```
0,60.879291,-10.098696,0,1,20
```

8.12. 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ワを使用します（「9. 仮想 FeliCa データファイルビューワ」を参照）。

8.13. ファイルの記述具体例

「http://xxx.yyy.zzz」の「tetris.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
 - 起動したいアプリケーション (Tetris) の ADF の「LaunchByMail」に「any」を設定します。
 - 「*.mail」ファイルに「C:\iDKDoJa3.0\lib\mail\0.mail」を用います。
 - 「0.mail」ファイルの ADF には、「ADF="http://xxx.yyy.zzz/tetris.jam"」と記述します。
 - 「JAM.properties」ファイルには、「http://xxx.yyy.zzz/tetris.jam = tetris.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
 - 起動したいアプリケーション (Tetris) の ADF の「LaunchByBrowser」に「any」を設定します。
 - 「HTML」ファイルに「C:\iDKDoJa3.0\lib\sample.html」を用います。
 - 「sample.html」の OBJECT タグの data には、「data="http://xxx.yyy.zzz/tetris.jam"」と記述して、「HTML」ファイルの<A>タグに<OBJECT>タグを指定した「HTML」ファイルの URL、<OBJECT>タグにダウンロードして起動するアプリケーションの ADF を記述します。
 - 「JAM.properties」ファイルには、「http://xxx.yyy.zzz/tetris.jam = tetris.jam」と記述します。
 - ブラウザ起動で、URL に「file:c:/iDKDoJa3.0/lib/sample.html」と指定して起動します。
- コードリーダー起動
 - 起動したいアプリケーション (Tetris) の ADF の「AllowPushBy」に「Code:」を設定します。
 - コード認識結果ファイルに「C:\iDKDoJa3.0\lib\multimedia\sample1.dat」を用います。
 - 「sample1.dat」ファイルの起動するアプリケーション (ADFURL:) には、「ADFURL:http://xxx.yyy.zzz/tetris.jam」と記述します。
 - 「JAM.properties」ファイルには、「http://xxx.yyy.zzz/tetris.jam = tetris.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。

9. 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ことができます。

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「プロジェクト」→「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 9-1）。

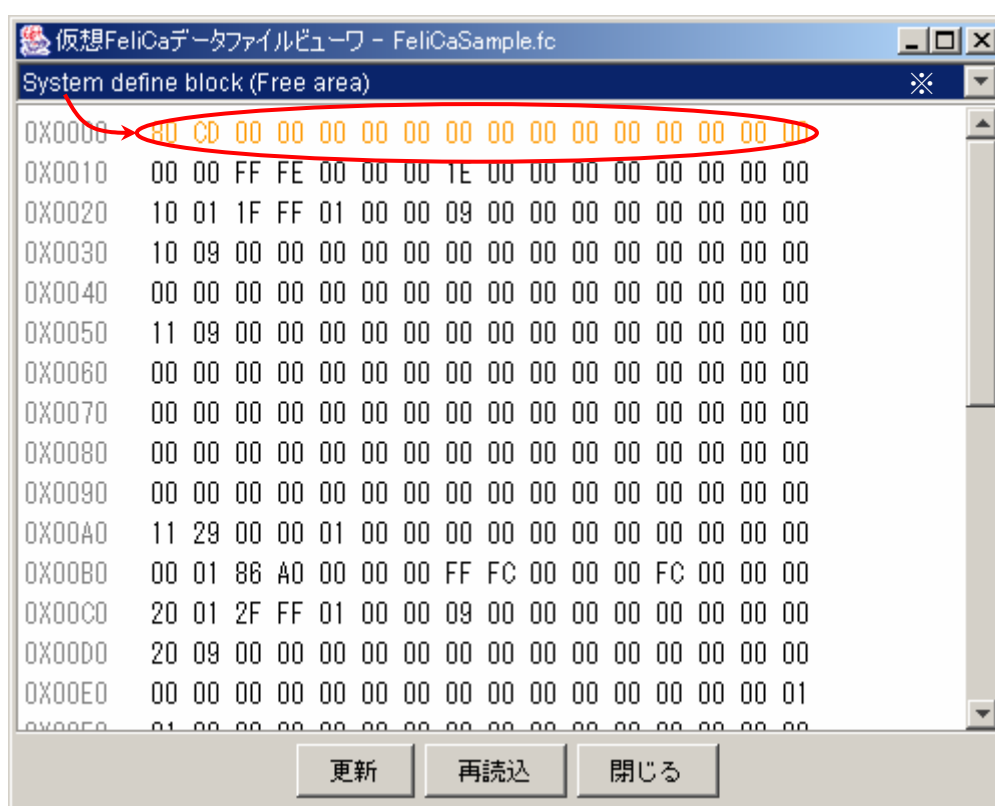


図 9-1 仮想 FeliCa データファイルビューワ画面

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、「8. 12. 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。
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。



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

5. 「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。
6. 「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。
7. 「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。

エミュレータは、仮想 FeliCa データファイルを以下のディレクトリに格納します:

C:\iDKDoJa3.0 にインストールした場合

C:\iDKDoJa3.0\apps\<プロジェクト名>\<プロジェクト名>.fc

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 (「5.3. 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 (「5.3. 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 データファイルへ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、アプリケーション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。

9.1. 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。

9.2. ブロックの構成について

仮想 FeliCa データファイルビューワでは各ブロックが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。

10. スクラッチパッド管理

本章では、「iα ppliTool」に含まれている、スクラッチパッド管理ツールについて説明します。

10.1. 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 進数でダンプする非常にシンプルなツールです。

スクラッチパッド管理ツールを使用すれば、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できます。また、スクラッチパッドを削除することもできます。

10.2. 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、文字列以外にアプリケーションで使用可能な基本タイプのデータを格納できるため、通常の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:

「iα ppli Development Kit for DoJa-3.0」を C:\iDKDoJa3.0 にインストールした場合、
“C:\iDKDoJa3.0\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>.scr です。

10.3. 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」→「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 10-1)。

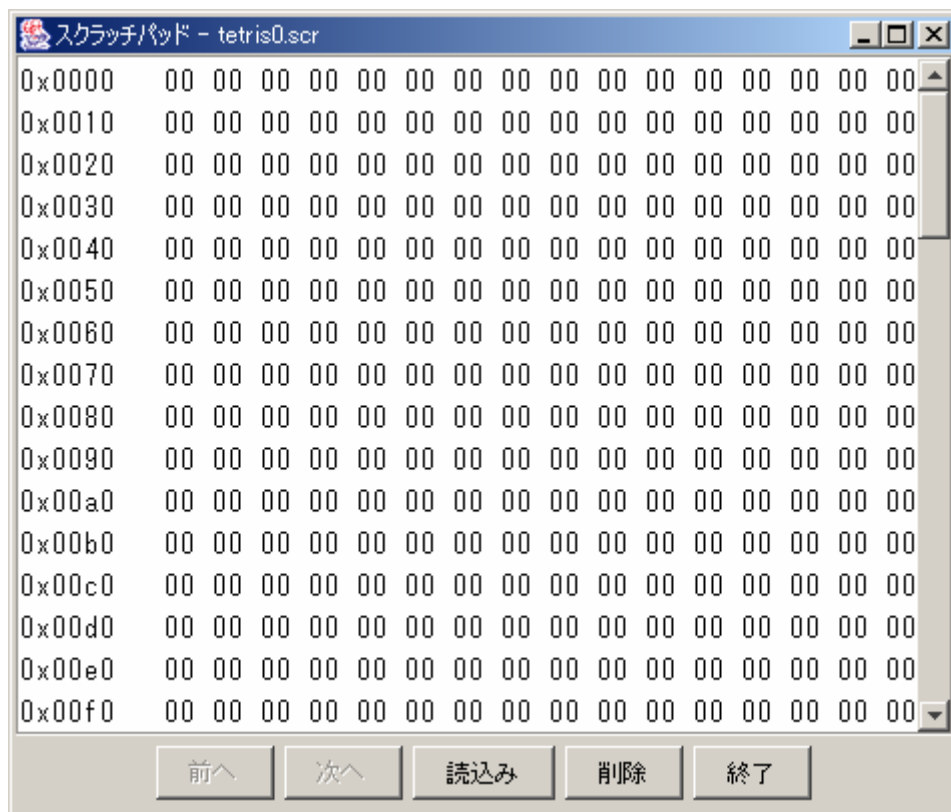


図 10-1 スクラッチパッド管理ツール

10.4. スクラッチパッドの内容表示

図 9-1 に示すように、スクラッチパッド管理ツールを起動すると、現在アクティブな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。ただし、内容の編集はできません。

また、他のプロジェクトのスクラッチパッドの内容を表示するには

1. 「読み込み」ボタンを押します。

「スクラッチパッドファイル選択」ダイアログが表示されます（図 10-2）。

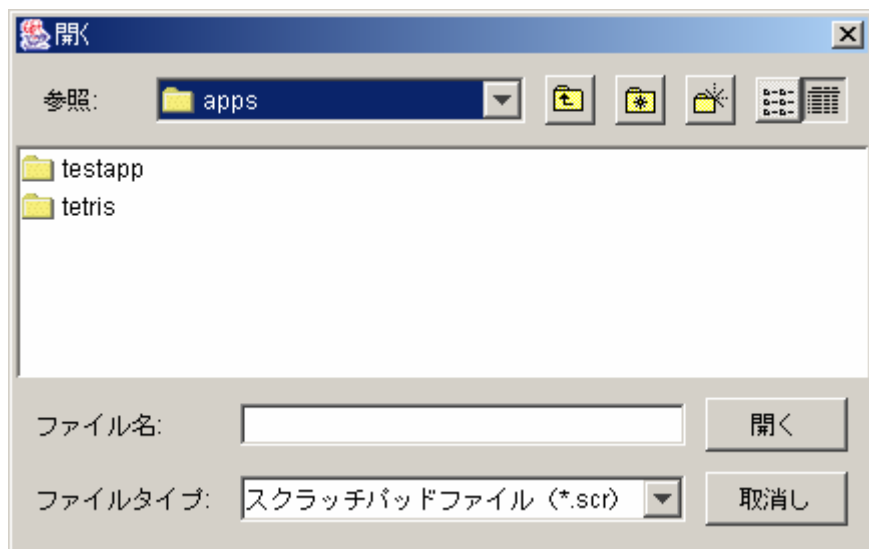


図 10-2 スクラッチパッドファイル選択ダイアログ

2. 内容を表示するスクラッチパッドファイルを選択し、「開く」ボタンを押します。

図 10-1 のスクラッチパッドの内容表示エリアに、スクラッチパッドの内容が表示されます。

10.5. スクラッチパッドの削除

スクラッチパッドを削除するには、図 10-1 の「削除」ボタンを押します。

現在、内容が表示されているスクラッチパッド（ファイル）が削除されます。

また、メニューから「プロジェクト」→「スクラッチパッドの削除」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削除することもできます（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）。

10.6. スクラッチパッド管理ツールの終了

スクラッチパッド管理ツールを終了するには、図 10-1 の「終了」ボタンを押します。

スクラッチパッド管理ツール画面が閉じます。

10.7. 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（「5.3. 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。

11. 証明書の管理

「iαppliTool」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、「iαppliTool」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 11-1)。

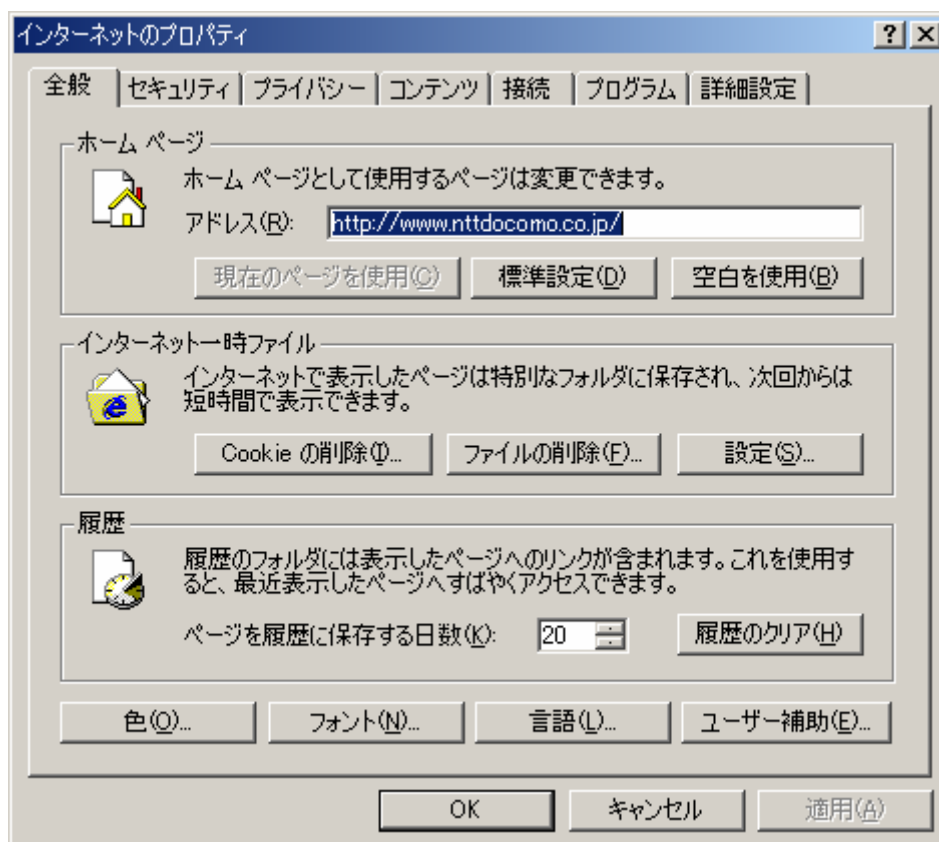


図 11-1 インターネットのプロパティ

2. 図 11-1 の「コンテンツ」タブを選択します（図 11-2）。

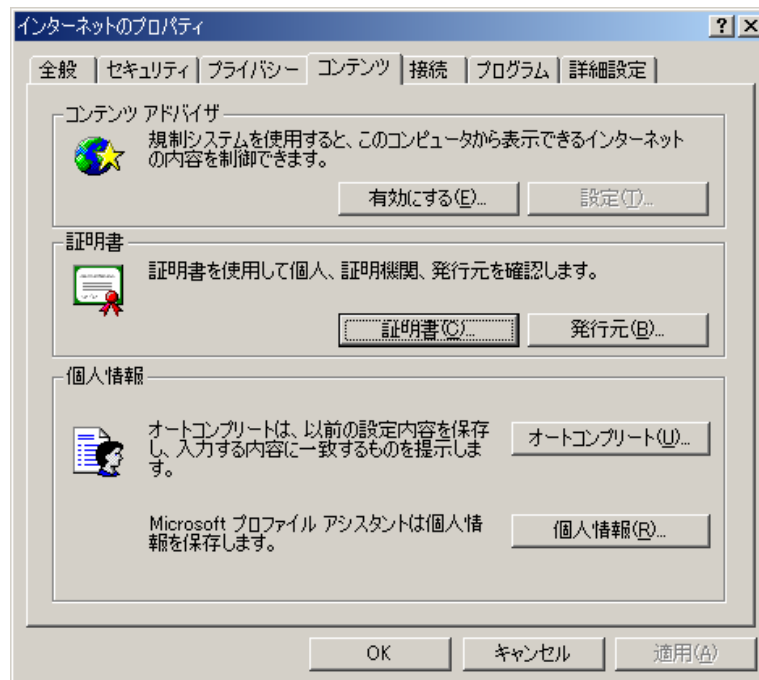


図 11-2 インターネットのプロパティ（コンテンツ）

3. 図 11-2 の「証明書（C）...」ボタンを押します（図 11-3）。

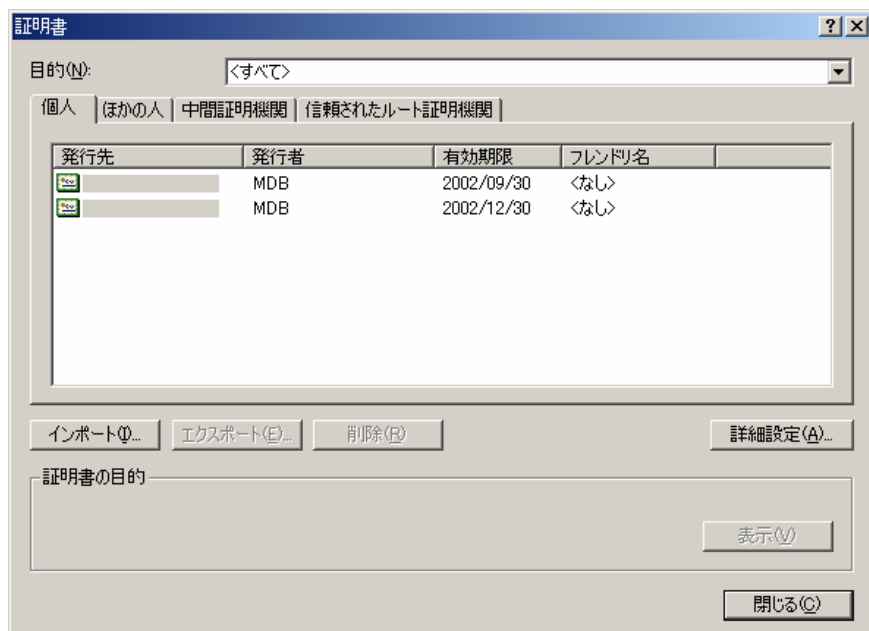


図 11-3 証明書画面

4. 図 11-3 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。

12. OBEX の利用

「iα ppli Development Kit for DoJa-3.0」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:\iDKDoJa3.0」にインストールした場合、「c:\iDKDoJa3.0\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。

Sun ONE Studio 4, Community Edition での設定は「13.4.1. 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。

13. Sun ONE Studio 4, Community Edition の利用

13.1. はじめに

この章では、Sun ONE Studio 4 と「Sun ONE Studio Module for iαppli」の拡張モジュールについて説明します。

Sun ONE Studio 4, Community Edition は、旧 Forte for java です。

13.1.1. Sun ONE Studio 4, Community Edition について

Sun ONE Studio 4, Community Edition に「Sun ONE Studio Module for iαppli」^[7]をインストールする事で、Sun ONE Studio 4 からエミュレータを直接起動することができます。「Sun ONE Studio Module for iαppli」は、i アプリ環境向けアプリケーションのコンパイルや実行を行うための Sun ONE Studio 4 の拡張モジュールです。

Sun ONE Studio 4, Community Edition のダウンロードや、Sun ONE Studio 4 の使用とインストールに関する詳細なドキュメントは、Sun ONE Studio 4 の Web サイト

<http://jp.sun.com/software/sundev/jde/> を参照してください。

注意： Sun ONE Studio 4, Community Edition をインストールする際に、

「C:\Program Files\s1studio\ce」がデフォルトのインストール場所になっていますが、「iαppli Development Kit」と同様に、インストール場所には、「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。

デフォルトでは、「C:\Program Files\」にスペースを含んでいますので、必ずインストール場所を変更してください。

また、古いモジュールや他のプロファイルのモジュールが入っている場合は上書きされます。

「Sun ONE Studio Module for iαppli」は、DoJa-3.5 との共存はできません。

13.2. 「Sun ONE Studio Module for iαppli」のインストール

本節では、Sun ONE Studio 4 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。

13.2.1. 始める前に

Sun ONE Studio 4 で「Sun ONE Studio Module for iαppli」を使用するには、Java 2 Platform, Standard Edition SDK (J2SE SDK) と Sun ONE Studio 4, Community Edition をインストールしてから、「Sun ONE Studio Module for iαppli」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
(<http://java.sun.com/j2se/>) からダウンロードできます。
 - Sun ONE Studio 4 は、次の Web サイト
(<http://jp.sun.com/software/sundev/jde/>) からダウンロードできます。
- 既に旧 Forte for java がインストールされている場合は、最初にアンインストールしてから Sun ONE Studio 4, Community Edition をインストールしてください。

^[7] 「Sun ONE Studio Module for iαppli」は、Sun ONE Studio 4, Mobile Edition、Sun ONE Studio 4, Enterprise Edition for Java および Forte for Java 3.0 には対応しておりません。

- Sun ONE Studio 4, Community Edition は、「スペース（空白）」及び「2 バイト文字（全角文字）」を含まないディレクトリにインストールされていなくてはなりません。

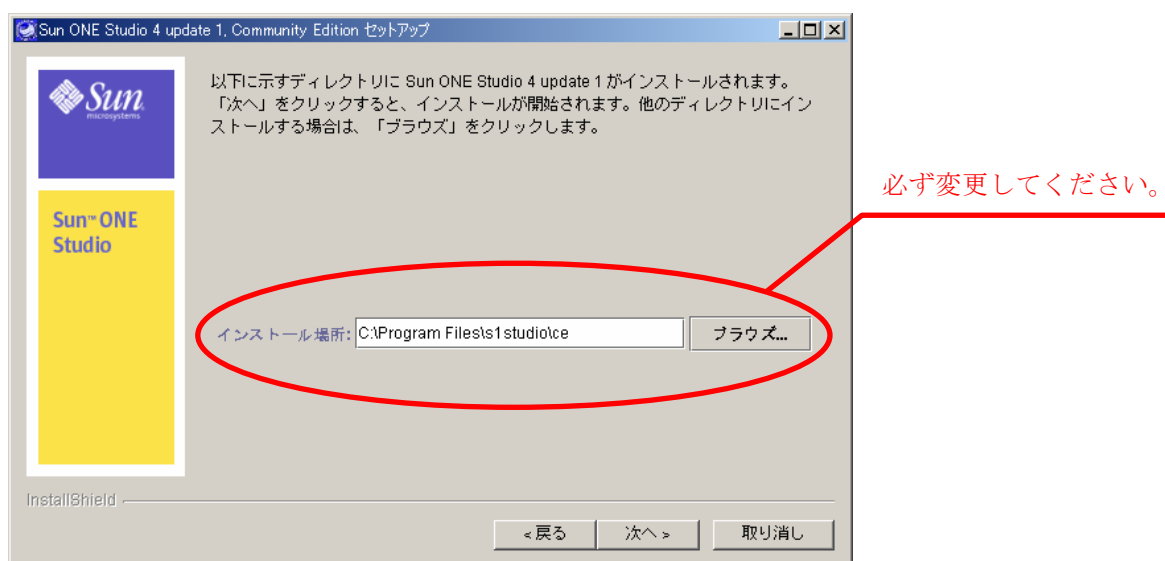


図 13-1 Sun ONE Studio インストール時の注意

13.2.2. 「Sun ONE Studio Module for iα ppli」のインストール

「Sun ONE Studio Module for iα ppli」をインストールするには、「2.2. インストールの手順」のセットアップタイプの選択画面（図 2-5）で、「カスタム」を選択し、機能の選択画面（図 2-6）で、「Sun ONE Studio(TM) 4, CE モジュール」をチェックします。

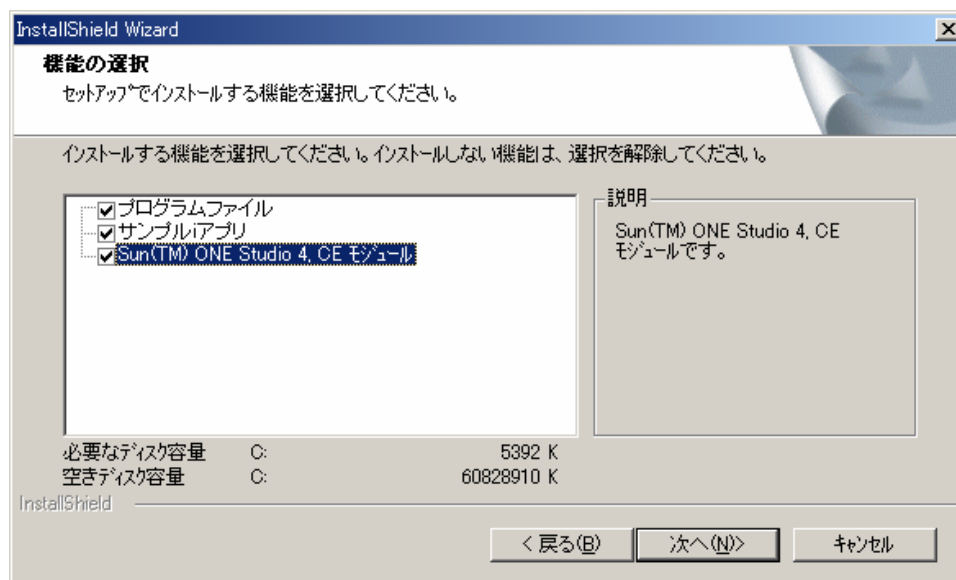


図 13-2 Sun ONE Studio Module のインストール

コンピュータにインストールされている Sun ONE Studio 4, CE が自動的に検出され、モジュールがインストールされます。

インストールプログラムが Sun ONE Studio 4 を検出できなかった場合は、「Sun ONE Studio インストールディレクトリの指定」画面にて次の手順を実行します。

1. 「参照」をクリックし、「ディレクトリの選択」ダイアログを開きます。
2. Sun ONE Studio 4 のインストール先のディレクトリを選択し、「OK」をクリックします。
3. 「次へ」をクリックし、次の画面に進みます。
これで、「Sun ONE Studio Module for iα ppli」が「iα ppliTool」とともにインストールされます。

注意：iα ppli モジュールを、Sun から提供される MIDP 向け開発環境「J2ME Wireless Toolkit」の提供する Sun ONE Studio モジュール (kjava) がインストールされている Sun ONE Studio 環境で使用する場合は kjava モジュールを無効にしてから使用してください。また、kjava モジュールを使用する場合は iα ppli モジュールを無効にしてから使用してください。モジュールを無効にする方法については、Sun ONE Studio のドキュメントを参照してください。

I D E で開発後、実際の端末へアップロードする際には、なるべくエミュレータの「iα ppliTool」を使って作成してください。（「iα ppliTool」については、「1.1.1. 「iα ppliTool」開発環境」を参照してください。）

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。

- パッケージの作成
- アプリケーション・ディスクリプタ・ファイル（ADF）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
- jar ファイルの内容の設定
- アプリケーションの構築と実行
- デバッグの実行

13.3.1. パッケージの作成

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. 「エクスプローラ[ファイルシステム]」から **sampledir** ファイルシステムを右クリックして、メニューから「新規」→「フォルダ」を選択します。

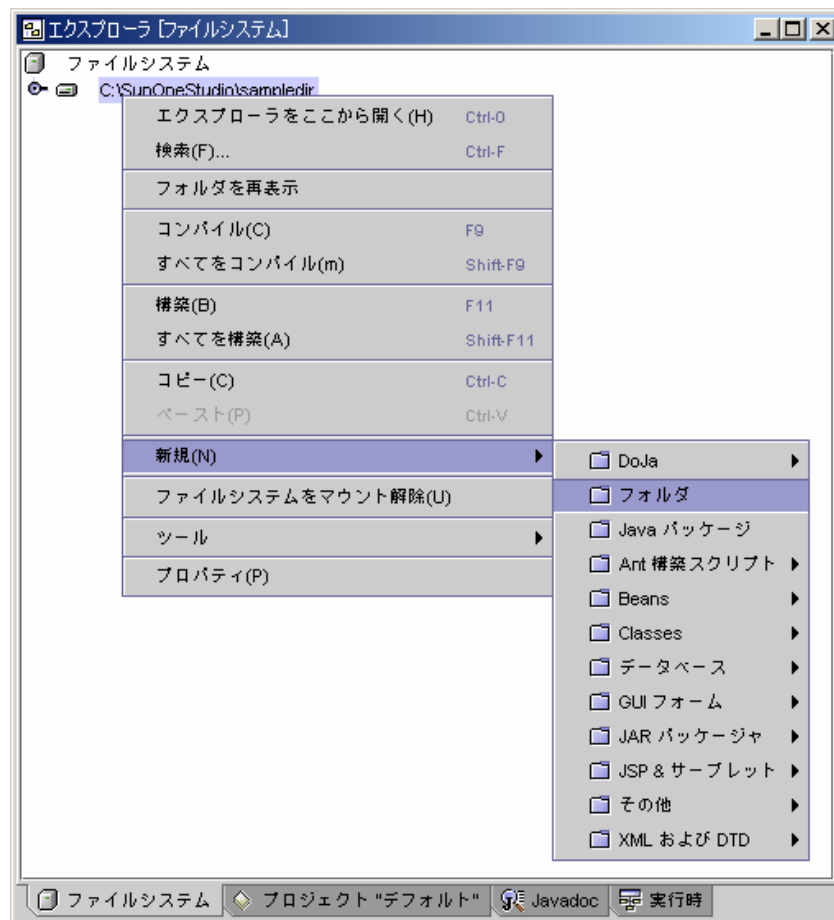


図 13-3 エクスプローラウィンドウ

2. 「新規オブジェクト - フォルダ」ウィザードが表示されるので、パッケージ名を入力し、「完了」をクリックします。
たとえば、パッケージ名を **hello** と入力します。
すると、エクスプローラに **hello** という名前のフォルダが追加されます。

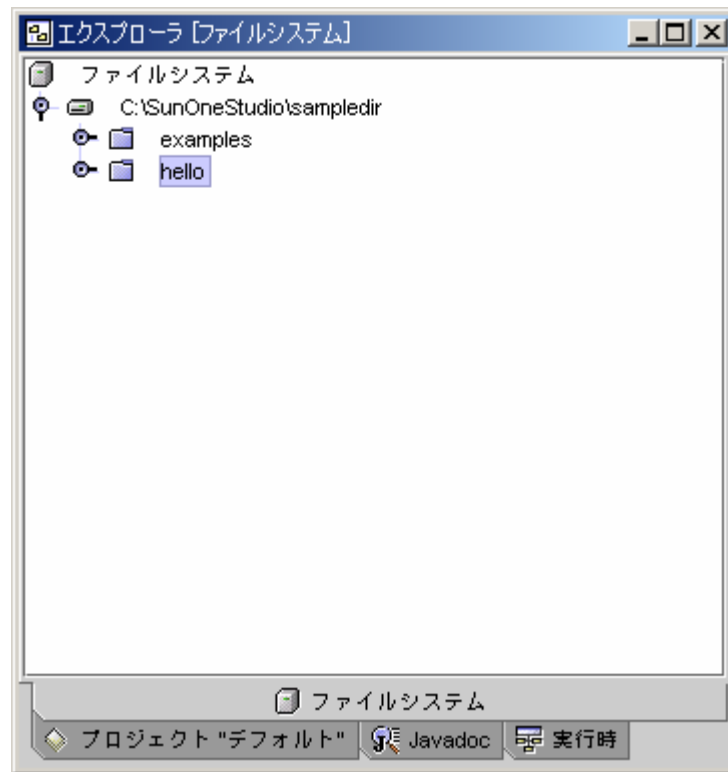


図 13-4 エクスプローラで新しいパッケージを作成

13.3.2. アプリケーション・ディスクリプタ・ファイル(ADF)の作成

ADF を作成するためには次の手順に従います。

ADF の詳細については、「5.3. 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。

1. 「エクスプローラ[ファイルシステム]」からアプリケーションを収容するために作成したパッケージを右クリックします。
2. 前の例で作成したパッケージ (hello) を右クリックし、メニューを表示させます。

3. 「新規」 → 「DoJa」 → 「DOJA」を選択します。

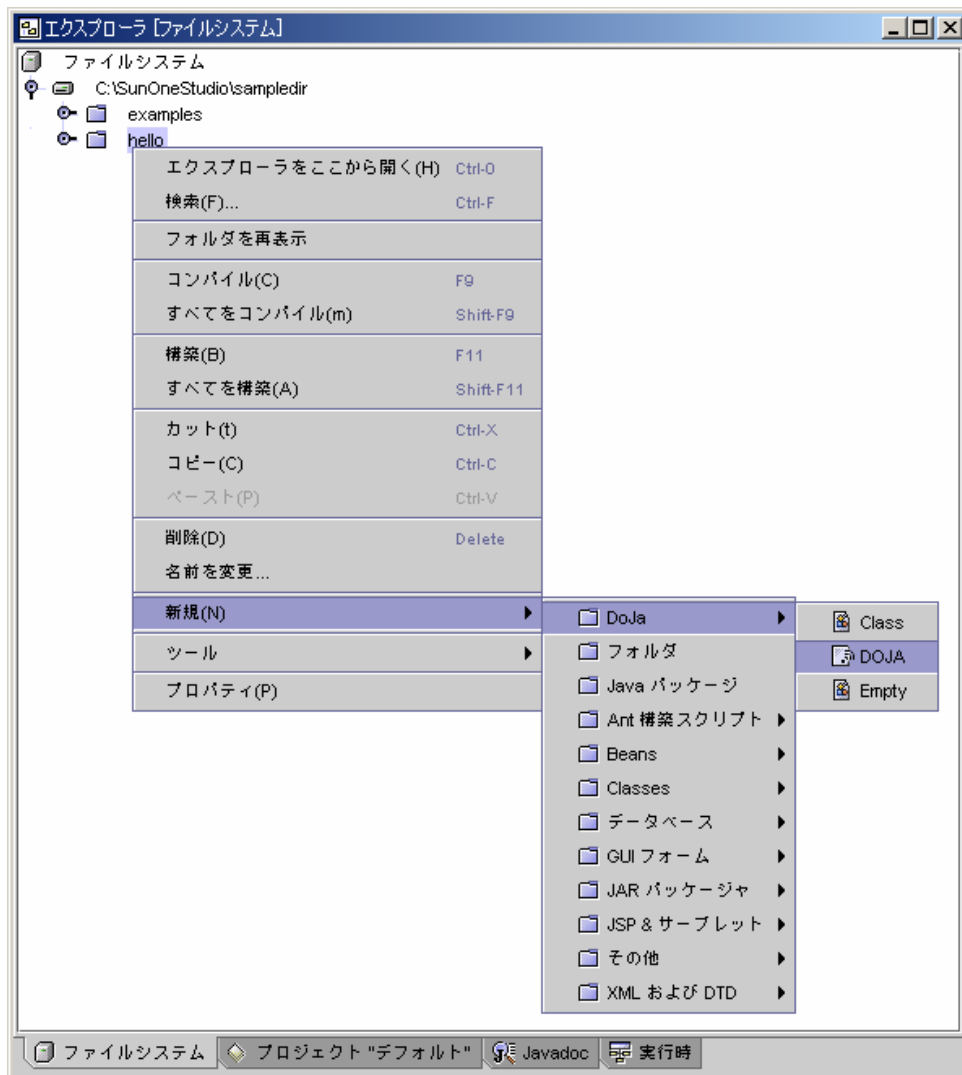


図 13-5 ADF の新規作成

4. 「新規オブジェクト - DOJA」ウィザードが起動されるので、名前に作成する ADF ファイルの名前を指定して「完了」をクリックします。
たとえば hello と入力します。

- ADF 設定ダイアログが表示されるので、右上の「×」で閉じます。
パッケージ (hello) の下に ADF ファイル (hello) が作成されます。

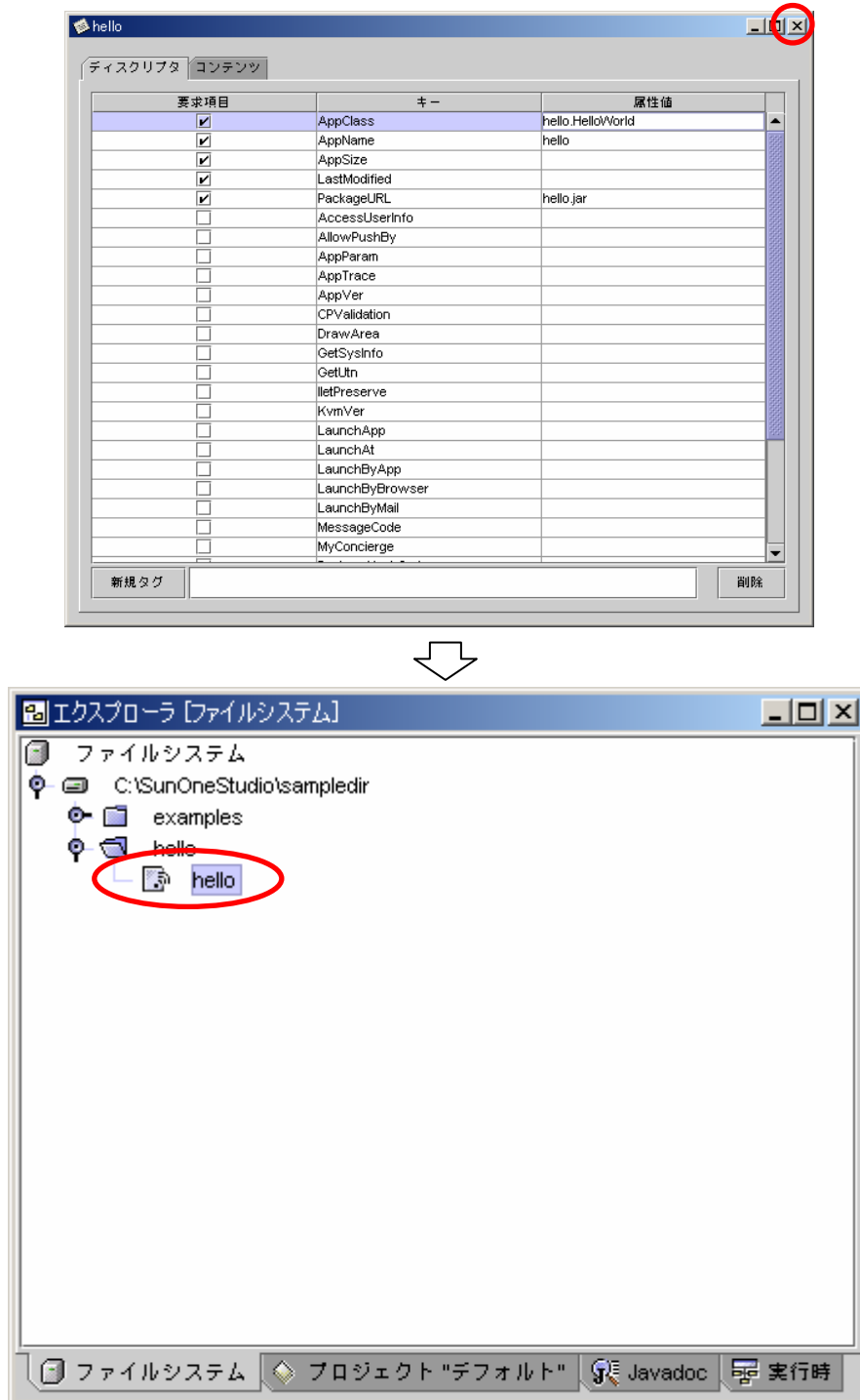


図 13-6 ADF ファイルが追加されたエクスプローラ画面

13.3.3. 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. パッケージ (hello) を右クリックし、「新規」 → 「DoJa」 → 「Class」を選択します。

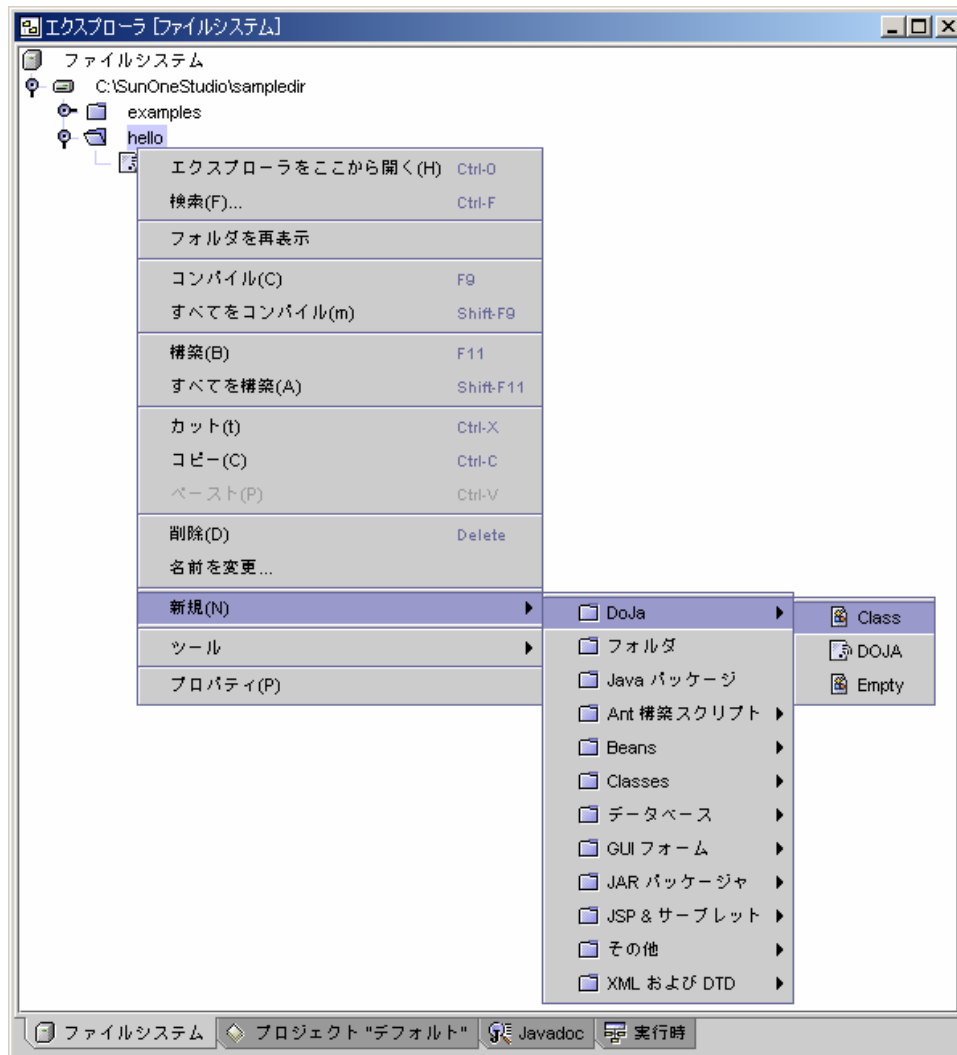


図 13-7 ソースファイルの作成

2. 「新規オブジェクト-Class」ウィザードが起動されるので、名前にアプリケーションのソースファイル名を指定して「完了」をクリックします。
たとえば HelloWorld と入力します。

3. ソースファイルエディタが開き、最初はテンプレートが表示されます。
また、パッケージ（hello）の下にソースファイル（HelloWorld）が作成されます。

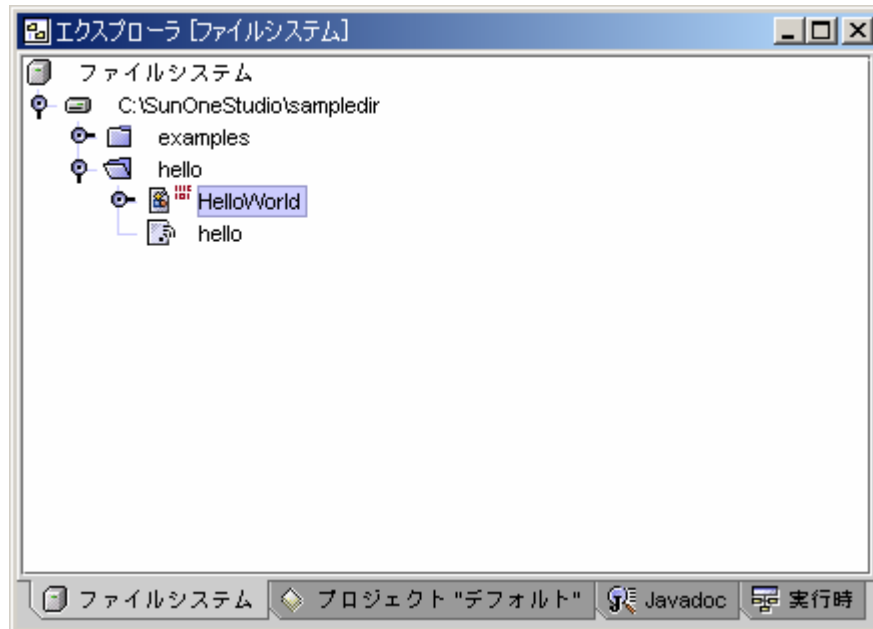


図 13-8 オブジェクトが追加されたエクスプローラ画面

4. ソースファイルを編集します。
たとえば、ソースファイル中の `public void start() {` の後に `System.out.println("HelloWorld");` を追加します。

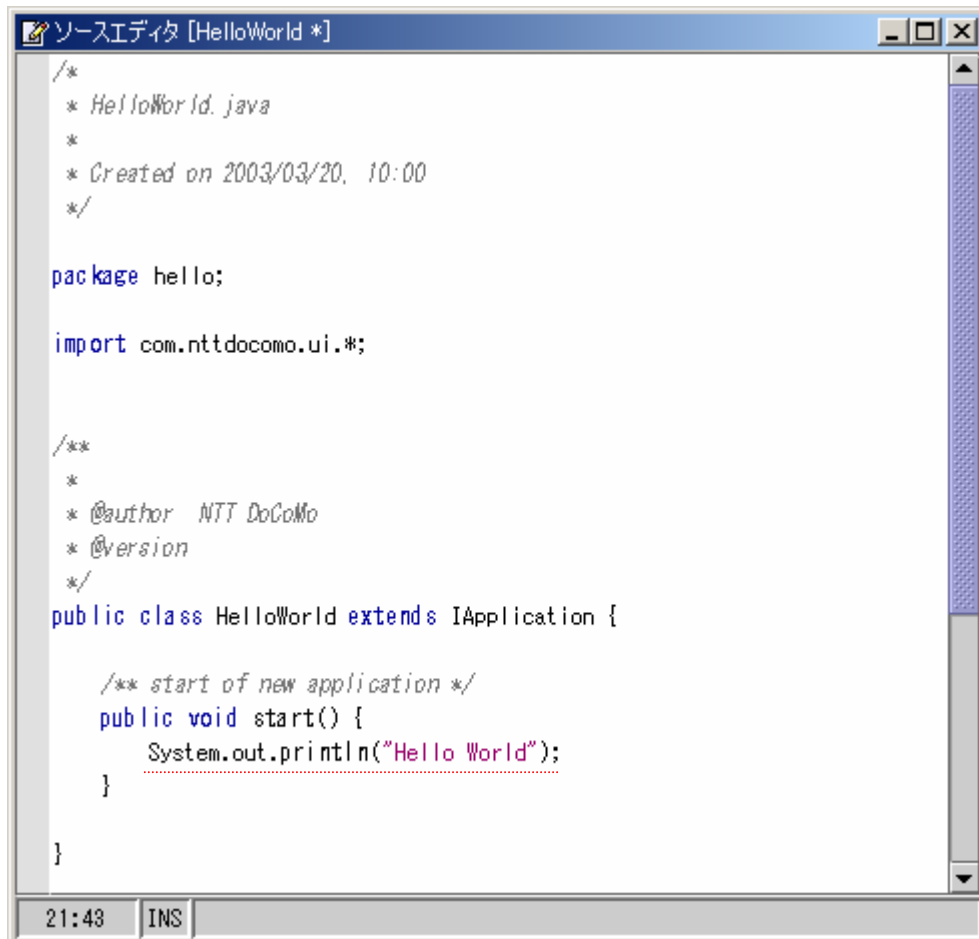


図 13-9 ソースファイルの編集

5. ソースファイルを保存します。
「ファイル」 → 「保存」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。

注意：アプリケーションの起動クラスの名前は、アプリケーション・ディスクリプタ・ファイルの AppClass キーで指定した名前と同じにする必要があります。

13.3.4. アプリケーション・ディスクリプタ・ファイル(ADF)の設定

ADFの設定をするには、次の手順に従います。

1. 「エクスプローラ[ファイルシステム]」で ADF hello を選択し、右ボタンをクリックしてメニューを表示させて、「開く」を選択します。



図 13-10 ADF のオープン

2. ADF 設定ダイアログが表示されるので、必要に応じて各キーの属性値を変更してください。
3. AppClass は、アプリケーションのメインクラスで、新規のプロジェクトが作成されたときのデフォルト値は、アプリケーションのパッケージ名になっています。このフィールドは、実際のメインクラス名 (start() を実装したクラス) に変更してください。
例えば、hello.HelloWorld に変更します。

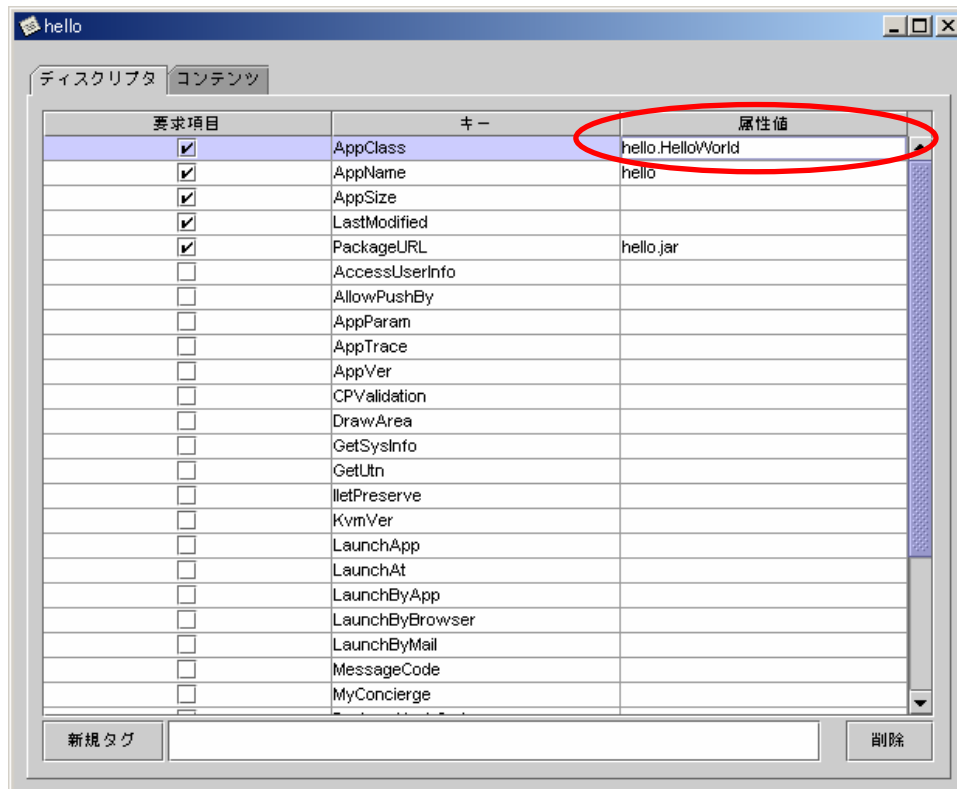


図 13-11 ADF 設定画面

各 ADF の設定項目については、「5.3. アプリケーションディスクリプタファイル (ADF) の編集 表 5-1 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。

13.3.5. トラストッド動作の設定

トラストッド動作の設定をするには、次の手順に従います。

1. 「ツール」 → 「オプション」 を選択して、「オプション」 ウィンドウを表示します。
2. 「分類されない」 → 「iα ppli 設定」 → 「Packager 設定」 をクリックすると、ウィンドウの右側にプロパティが表示されます。
3. 「上級」 タブをクリックして、トラストッドの動作設定の属性値を必要に応じて変更してください。

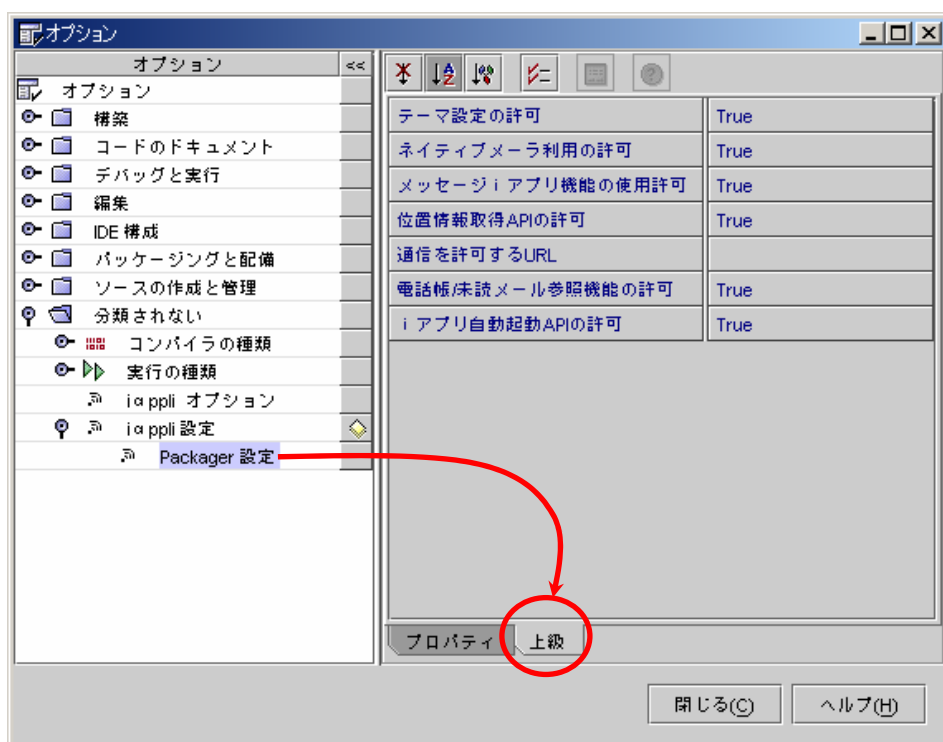


図 13-12 トラストッド動作の設定オプション

各トラストッド動作設定の設定項目については、「4.3. トラストッド動作の設定」を参照してください。

13.3.6. jar ファイルの内容の設定

ここでは、jar ファイルの内容を指定する方法について説明します。

jar に追加したいディレクトリを選択すると、この情報がアプリケーションのビルドで使用されます。

1. 「エクスプローラ[ファイルシステム]」で ADF hello を選択し、右ボタンをクリックしてメニューを表示させて、「開く」を選択します。

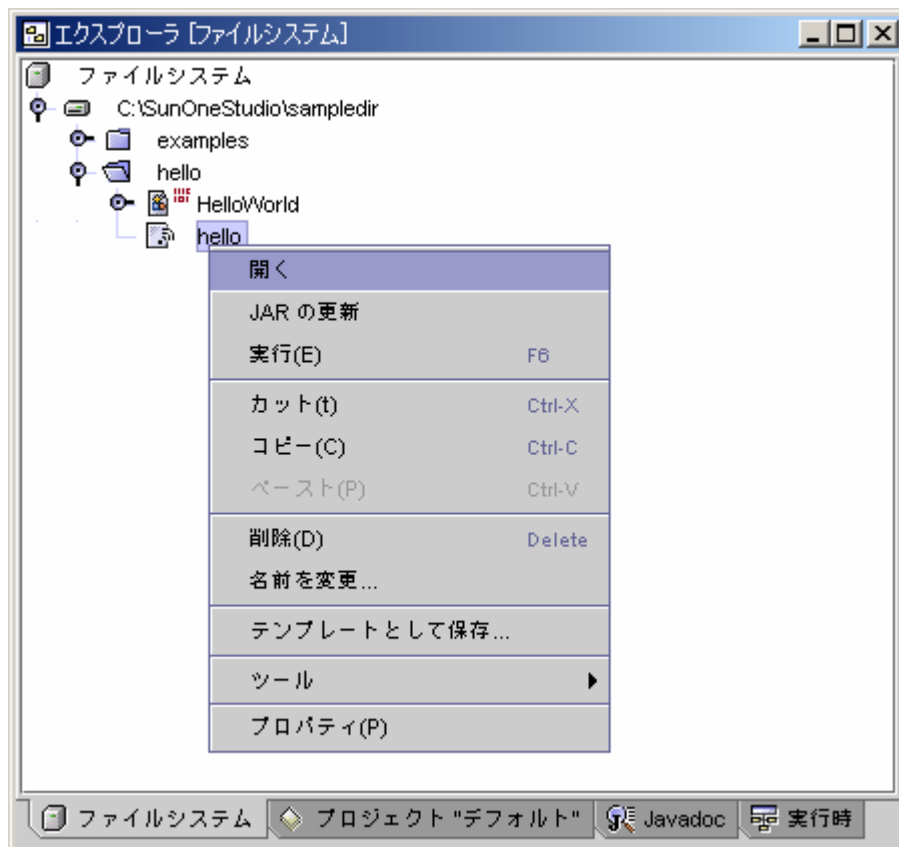


図 13-13 ADF のオープン

2. ADF 設定ダイアログが表示されるので、「コンテンツ」タブをクリックします。

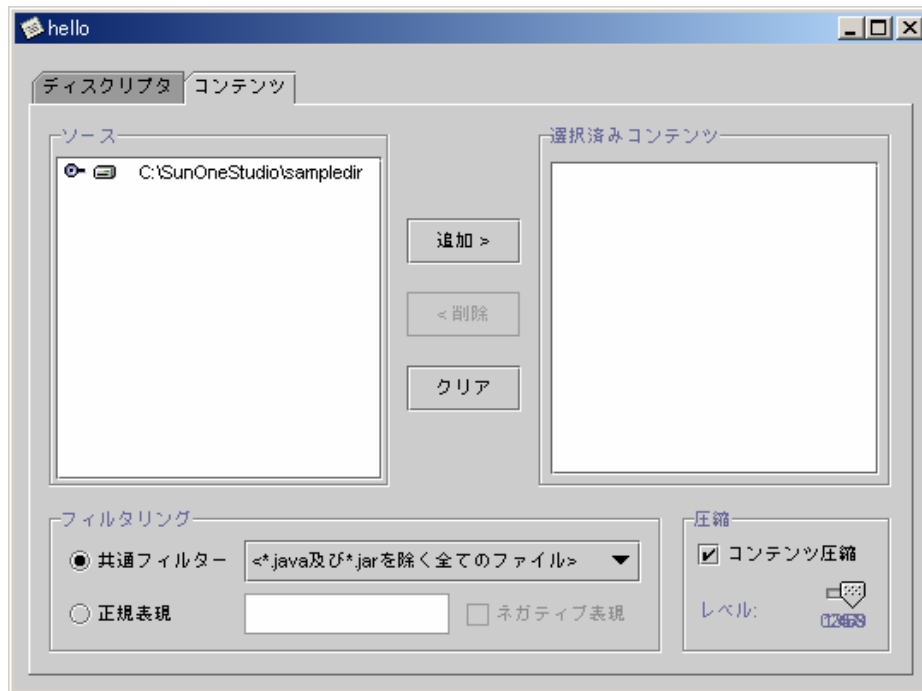


図 13-14 コンテンツの指定

3. クラスファイルを jar ファイルに転送するため、ソースファイルのあるディレクトリ（ここでは hello ディレクトリ）を選択して「追加」を押します。

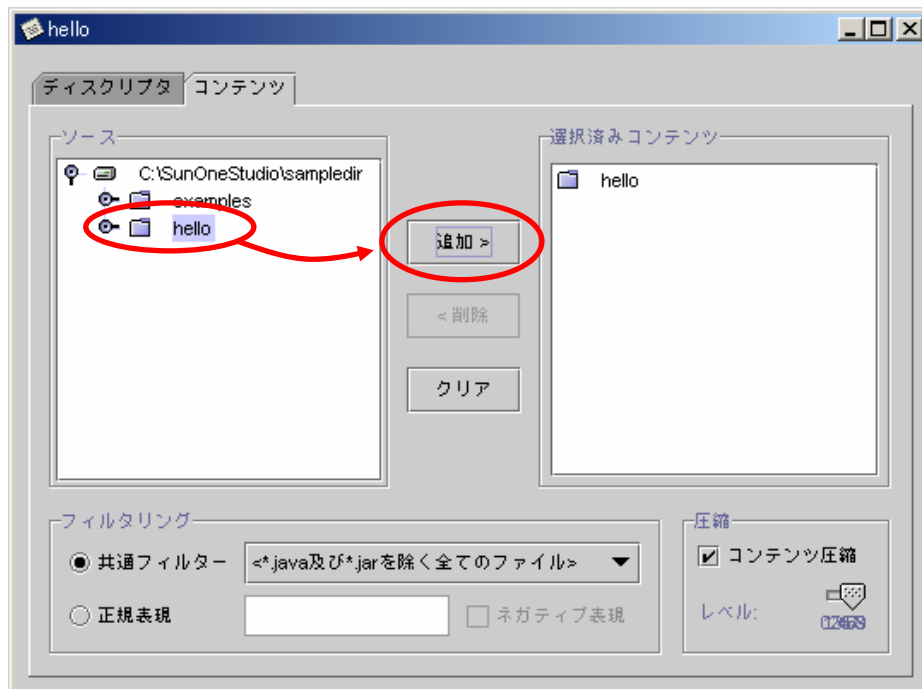


図 13-15 コンテンツを Jar ファイルに転送

13.3.7. 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. 「エクスプローラ[ファイルシステム]」で、ADF を選択してから右ボタンをクリックし、「実行」を選択します。

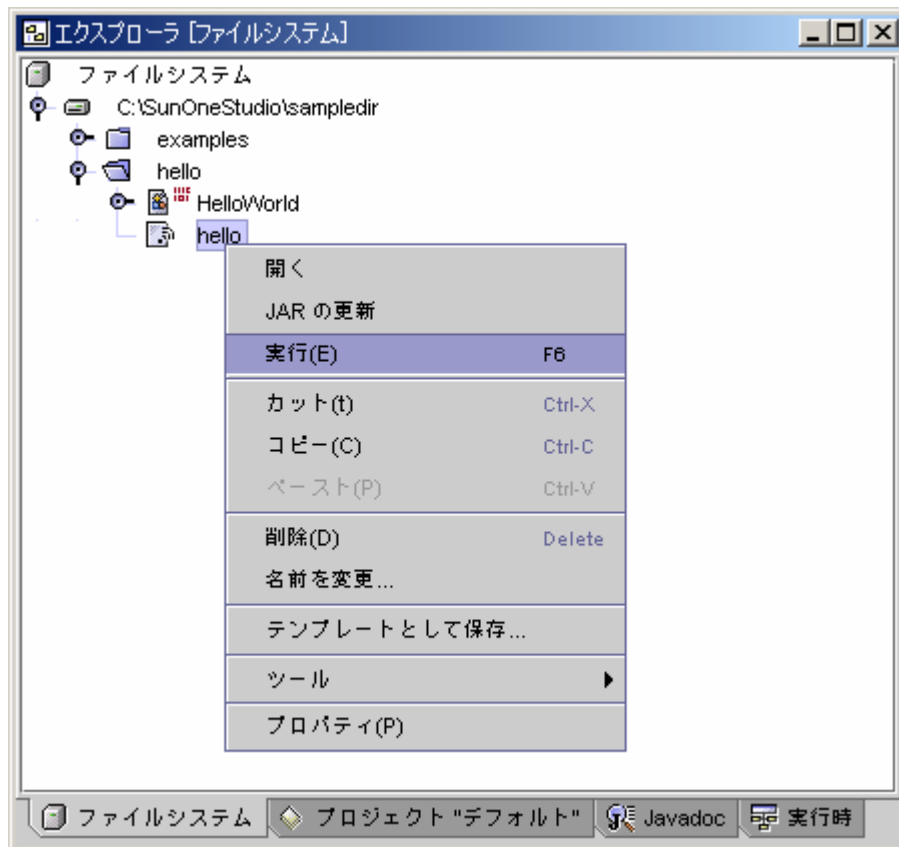


図 13-16 アプリケーションの実行

2. アプリケーション helloWorld がビルドされ、端末ウィンドウが立ち上がって実行されます。
実行結果は、出力ウィンドウの hello タブに Hello World と表示されます。

13.3.8. 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。^[8]

1. まず最初にエミュレータをデバッガモードで実行するようにします。
「ツール」→「オプション」を選択して、「オプション」ウィンドウを表示します。
2. 「分類されない」→「iα ppli 設定」→「Packager 設定」をクリックすると、ウィンドウの右側にプロパティが表示されるので、「デバッグしない」を「False」に変更して、「閉じる」をクリックします。

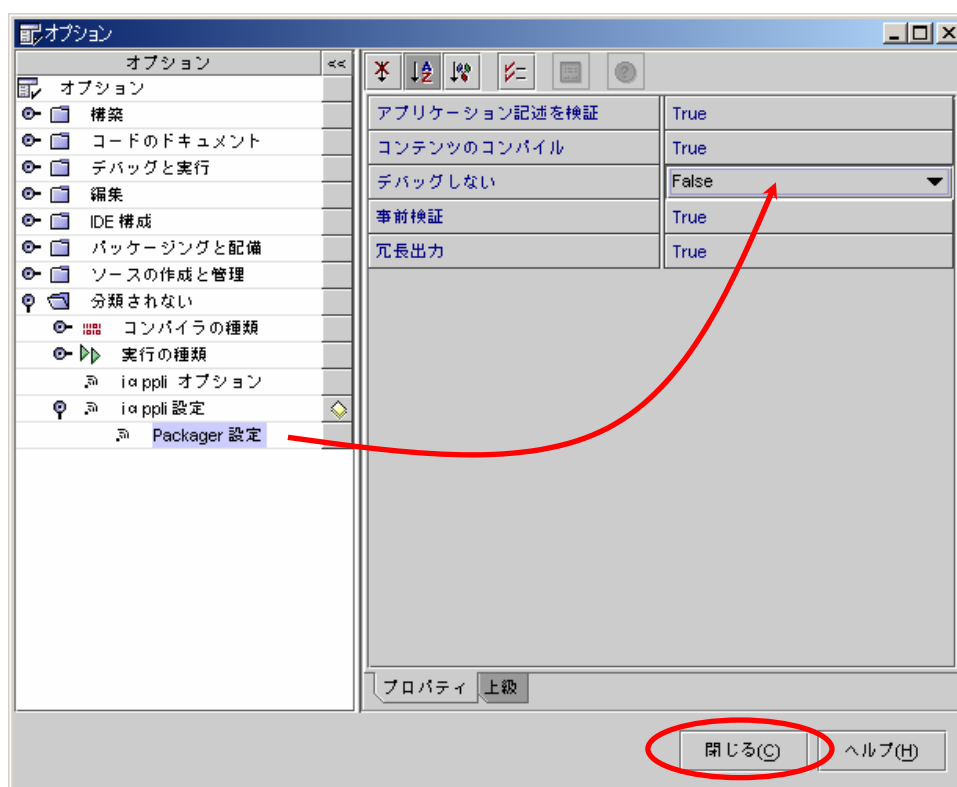


図 13-17 デバッグ実行の有効化

デバッグ操作を止めたい場合は、「デバッグしない」を「True」に戻してください。

^[8] 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。

3. 「編集」タブをクリックして、ソースエディタを開きます。
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。

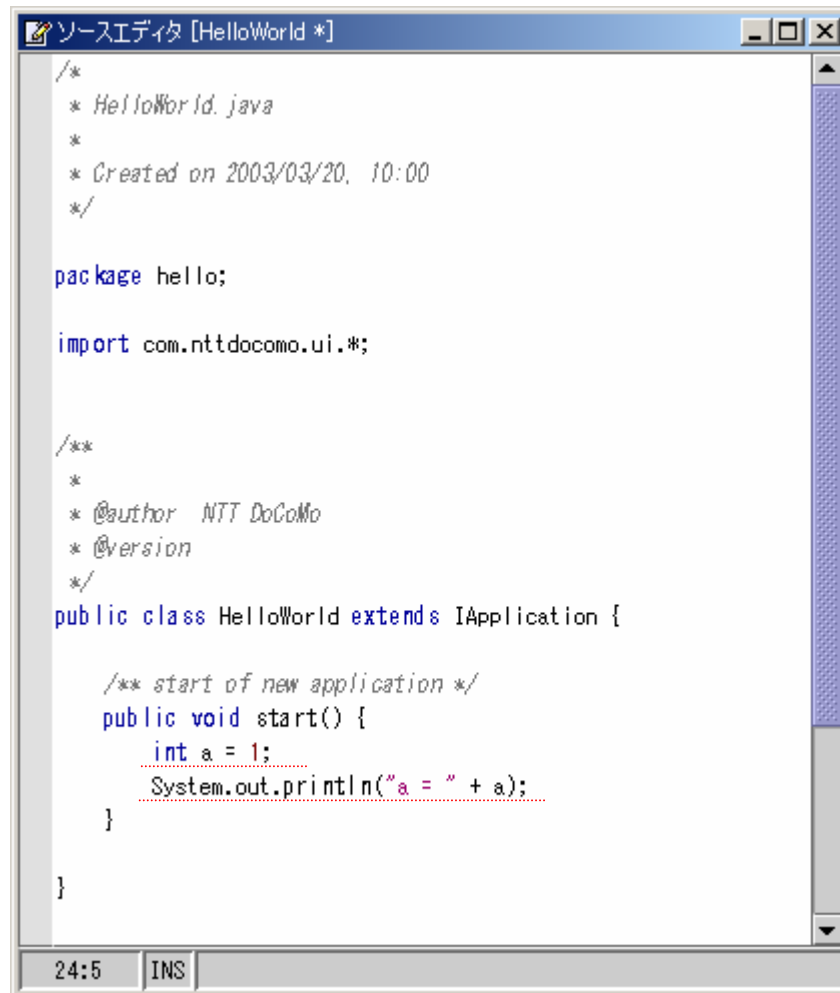


図 13-18 変数の表示

4. 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。

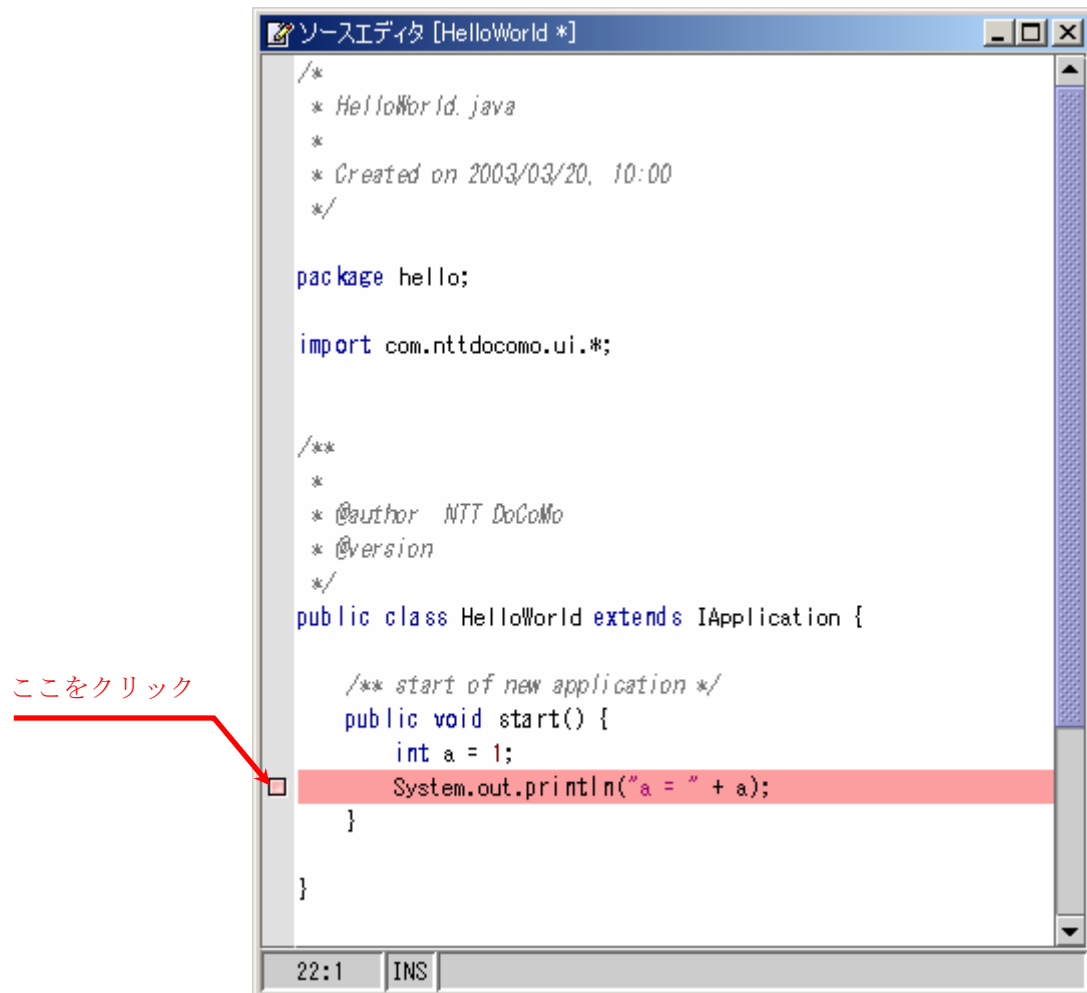


図 13-19 ブレークポイントの設定

5. 「エクスプローラ[ファイルシステム]」で、ADF を選択してから右ボタンをクリックし、「実行」を選択します。
端末ウィンドウは開かず、実行[実行の表示]で止まったままになります。

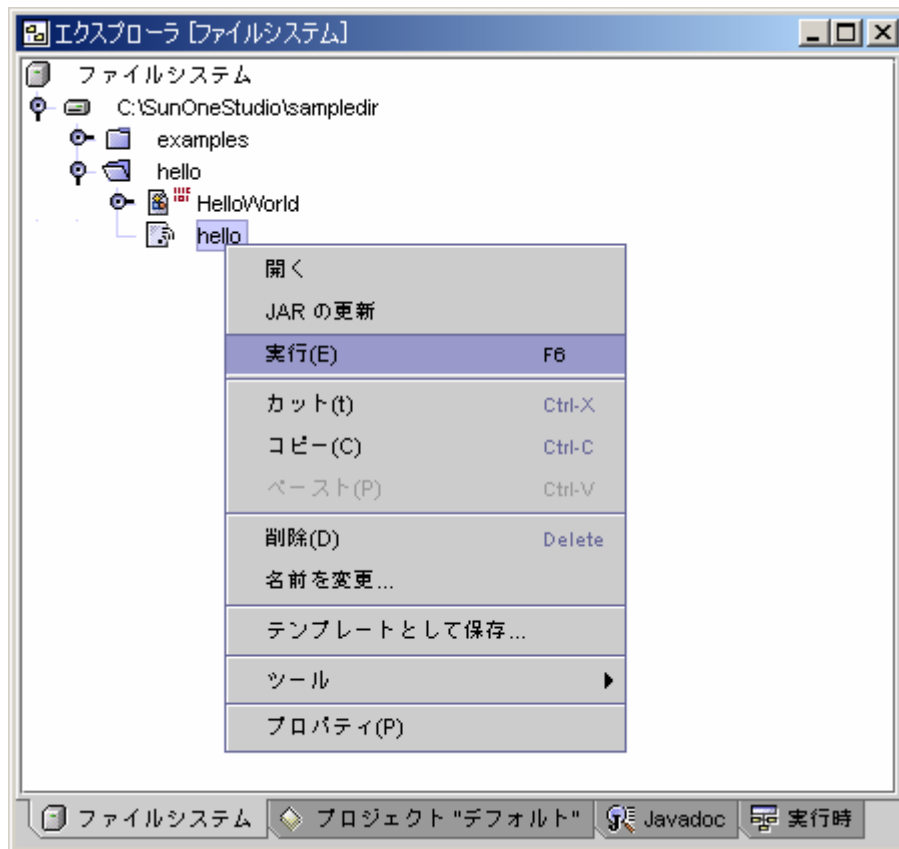


図 13-20 アプリケーションの実行

6. 「デバッグ」 → 「接続」 を選択します。
7. 「VMに接続」 ダイアログが表示されるので、
 デバッガの種類を 「デフォルトデバッガ(JPDA)」
 コネクタを 「SocketAttach(ソケットでその他のVMに接続します。)」
 トランSPORTを 「dt_socket」
 ホストを 「127.0.0.1」
 ポートを 「8000」
 に設定し「了解」をクリックします。
 (この値は、一度設定するとそのまま保持されます。)

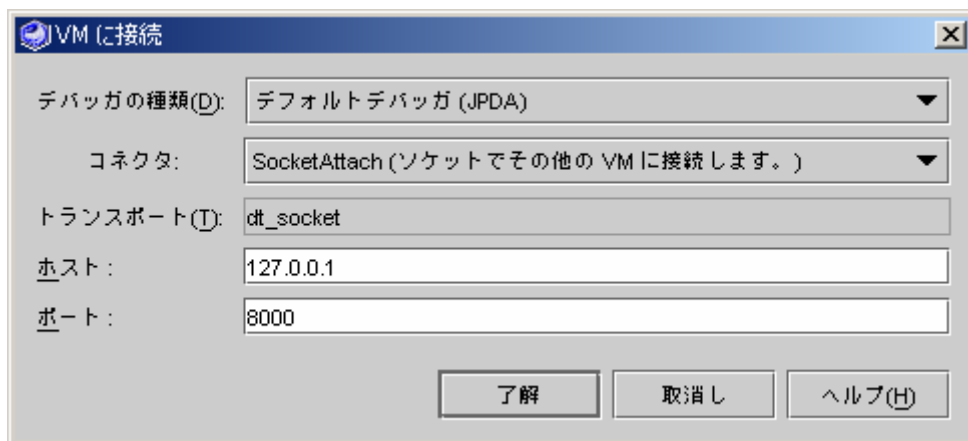


図 13-21 デバッグ接続画面

8. デバッガウィンドウ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。

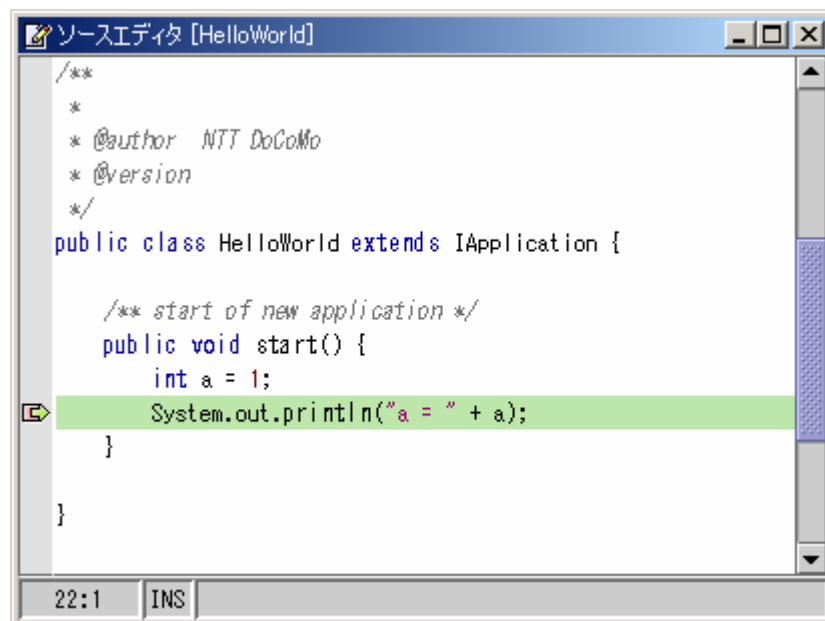
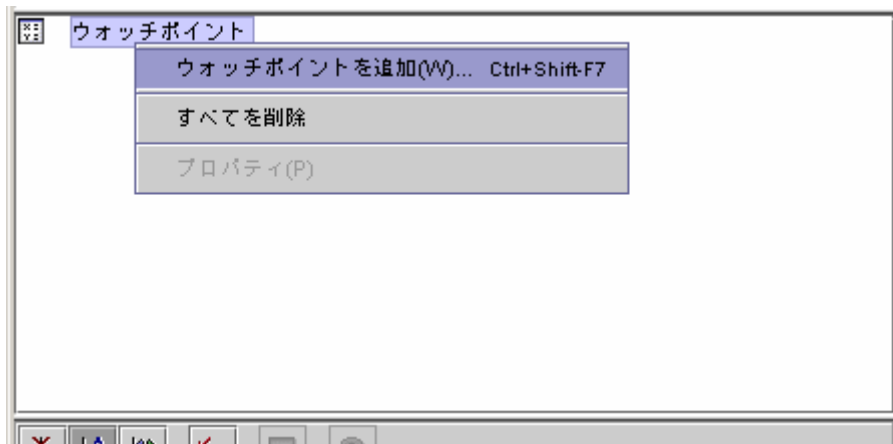


図 13-22 ブレークポイントでの処理の中断

9. デバッガウィンドウのウォッチポイントで、「ウォッチポイント」を右クリックし、「ウォッチポイントを追加」を選択します。



10. 「ウォッチポイントを追加」ダイアログが表示されるので、変数名 (a) を入力し、「了解」をクリックします。

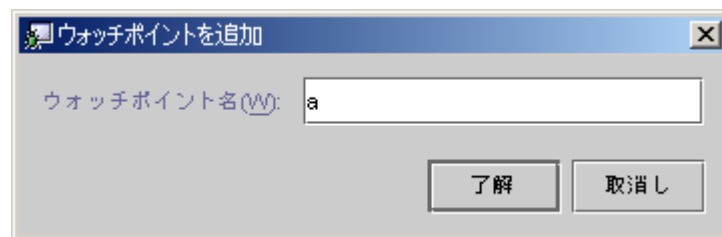


図 13-23 ウォッチしたい変数の追加

11. ウォッチポイントにブレークポイント時点の変数の内容が表示されます。



図 13-24 変数の値の表示

12. ウォッチポイントに表示された変数名（a=1）をダブルクリックして、変数の値を変更します。
たとえば、a の値を 5 に変更します。



図 13-25 変数の値の変更

13. 「デバッグ」 → 「継続」で中断した位置から処理を再開させます。
変数が変更された値で処理が再開されます。
実行結果は、出力ウィンドウの hello タブに a = 5 と表示されます。
このようにして、アプリケーション内で様々なパターンをテストできます。
14. 「デバッグ」 → 「完了」でデバッグを終了します。

13.4. エミュレータの設定

この節では、エミュレータの設定方法について説明します。

- デバイスの設定
- プロキシサーバの設定
- ログメッセージのトレース

13.4.1. デバイスの設定

「Sun ONE Studio Module for iα ppli」では、エミュレータのデバイス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. メインメニューバーから「ツール」→「オプション」を選択して、「オプション」ウィンドウを表示します。
2. 「オプション」ウィンドウで「分類されない」→「実行の種類」→「iα ppli Emulator」を右クリックし、次に「Bean をカスタマイズ」をクリックします。

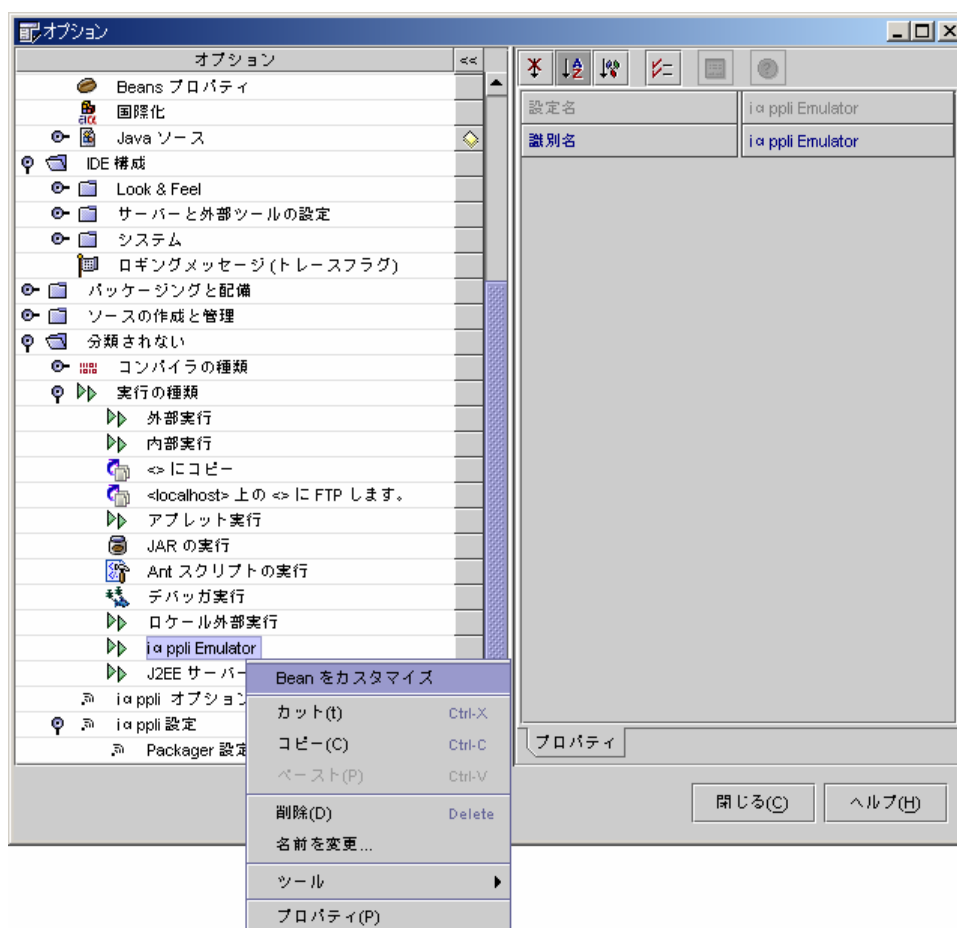


図 13-26 エミュレータの設定

3. 「Beanをカスタマイズ」画面が表示されるので、「カスタマイザ」ボタンをクリックします。

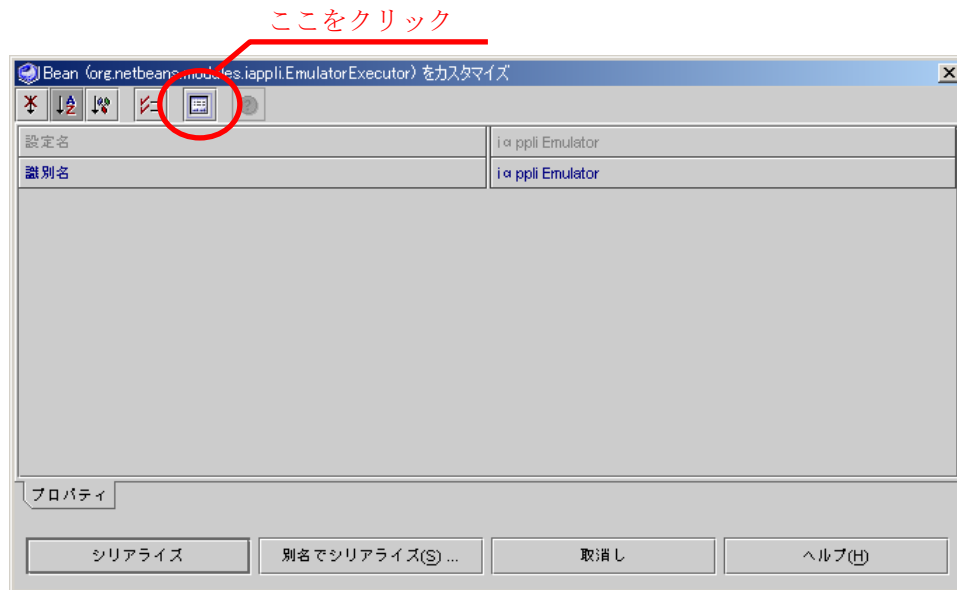


図 13-27 Beanをカスタマイズの設定画面

4. 「カスタマイザダイアログ」(図 13-28)が表示されます。左側の欄には設定項目が、右側の欄には現在の設定値がそれぞれ表示されます。

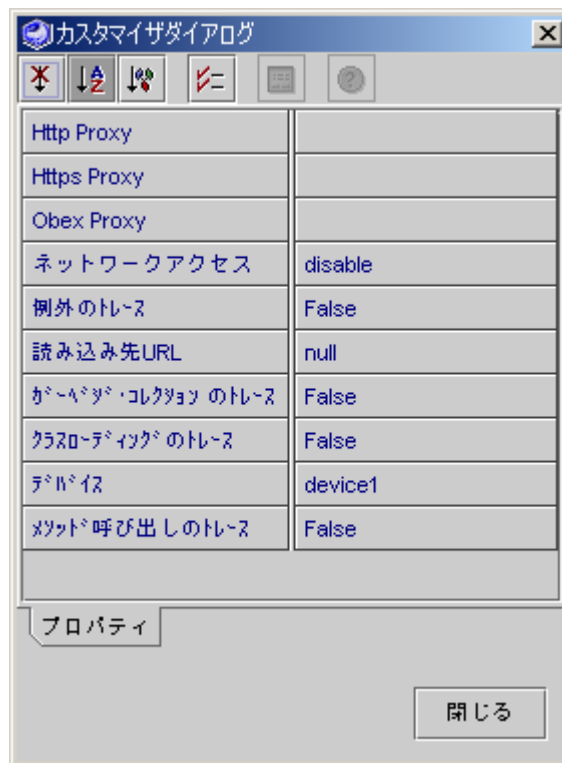


図 13-28 エミュレータの設定画面

5. 変更したい設定項目の値を選択します。

6. 選択項目がある設定項目の場合には下矢印が表示され、設定可能な値がドロップダウンメニューに表示されます。値を記述する設定項目の場合は入力欄が編集可能状態になります。
7. 必要な設定項目をクリックします。
たとえば、デバイスを変更する場合は、デバイスの値をクリックし、下矢印をクリックして、新しい値を選択します（図 13-29）。

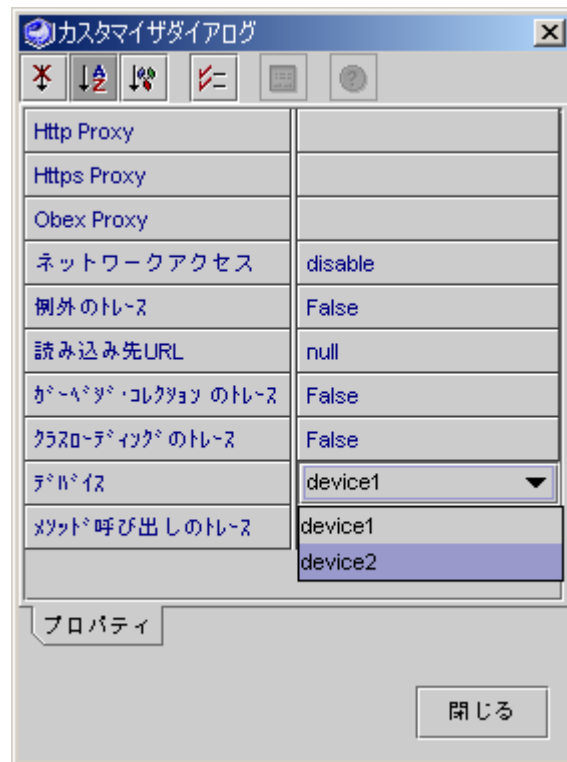


図 13-29 デバイスの変更

8. エミュレータ設定項目は表 13-1 のとおりです。

表 13-1 エミュレータの設定項目

設定項目	説明	デフォルト
デバイス	ターゲットデバイスの名前。デバイスプロパティファイルの名前と同じです。	device1
読み込み先 URL	エミュレータは、アプリケーションがこの URL からダウンロードされたものとみなします。アプリケーションは、このフィールドに指定されたホストとポートだけとしか通信できません。 このフィールドの形式は、ADF の PackageURL タグの形式と同じです。 このフィールドが「*」なら、アプリケーションは、どのサーバとでも通信できます。この設定はエミュレータ上で Java アプリケーションの開発とテストを行う場合にのみ使用できます。	なし

	このフィールドがブランクなら、アプリケーションは、ネットワークを全く使用できません。	
ネットワークアクセス	ネットワークアクセスの設定は、エミュレータが Java アプリケーションからのネットワーク接続を許可するかどうかを管理します。ネットワークアクセスの設定を有効にするには、アプリケーション・ディスクリプタ・ファイルの "UseNetwork" キーに "http" が指定されていなければならないことに注意して下さい。 アプリケーション・ディスクリプタ・ファイルの "UseNetwork" キーが定義されていない場合、NetworkAccess に設定された値にかかわらずネットワーク接続を行おうとすると Java アプリケーションは終了します。 NetworkAccess の設定には、以下のような値を設定できます。 ・ enable - Java アプリケーションからのネットワーク接続を許可します。この接続は読み込み先 URL に設定された値に基づきコントロールされます。 ・ disable - Java アプリケーションからのネットワーク接続は許可されません。Java アプリケーションがネットワーク接続を行おうとした場合は SecurityException がスローされます。	disable
HttpProxy	アプリケーションが Web サーバとの通信に使用する HTTP プロキシサーバのアドレスとポート。形式は <address>:<port> です。	なし
HttpsProxy	アプリケーションが Web サーバとの通信に使用する HTTPS プロキシサーバのアドレスとポート。形式は <address>:<port> です。	なし
Obex	アプリケーションが 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（ホスト名）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）に接続を試みを確認する試みをします。なお、本設定は、ObexClientConnection クラスの connect メソッドで意味をなします（「12. OBEX の利用」を参照）。	なし
ガベージコレクションのトレース	ガベージコレクション情報を表示します。	オフ
クラスローディングのトレース	クラスローディング情報を表示します。	オフ
メソッド呼び出しのトレース	メソッド呼び出しを表示します。	オフ
例外のトレース	例外を表示します。	オフ

13.4.2. プロキシサーバの設定

エミュレータのもとで動作する Java アプリケーションが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(たとえば、Web サーバがファイアウォールの反対側にある場合など)、エミュレータの設定でプロキシサーバの情報を指定する必要があります。

アプリケーションは HTTP か HTTPS プロトコルを使って Web サーバに接続しますが、同じアプリケーション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 のプロキシサーバのアドレスとポートは設定画面(図 13-30)で入力します。

これらの設定値がわからない場合は、システム管理者に問い合わせして下さい。

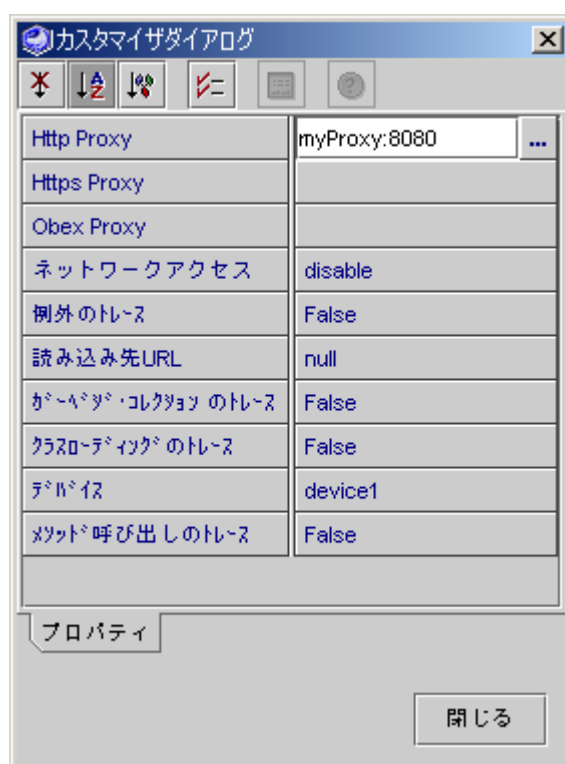


図 13-30 プロキシの設定

13.4.3. ログメッセージのトレース

次の 4 種類のログメッセージをトレースできます。

- ガベージコレクション
- クラスローディング
- メソッド呼び出し
- スローされた例外

エミュレータの設定画面 (図 13-28) で、該当するエントリをオンにしてください。

14. Eclipse の利用

14.1. はじめに

この章では、Eclipse と「Eclipse Module for iαppli」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 Module for iαppli」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 Module for iαppli」は、i アプリ環境向けアプリケーションのコンパイルや実行を行うための Eclipse の拡張モジュールです。

「Eclipse Module for iαppli」は、Eclipse2.1.1 で開発しています。

そのため「Eclipse Module for iαppli」は、Eclipse2.1.X で使われる事を推奨します。

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も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 と共存させたい場合は、別の Eclipse にインストールしてください。

14.2. 「Eclipse Module for iαppli」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 Module for iαppli」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 Module for iαppli」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 (<http://java.sun.com/j2se/>) からダウンロードできます。
- Eclipse と、その Language Pack は、次の Web サイト (<http://www.eclipse.org/>) からダウンロードできます。

14.2.2. 「Eclipse Module for iαppli」のインストール

「Eclipse Module for iαppli」をインストールするには、2.2. インストールの手順のセットアップタイプの選択画面（図 2-5）で、「カスタム」を選択し、機能の選択画面（図 2-6）で、「Eclipse2.1.X プラグイン」をチェックします。

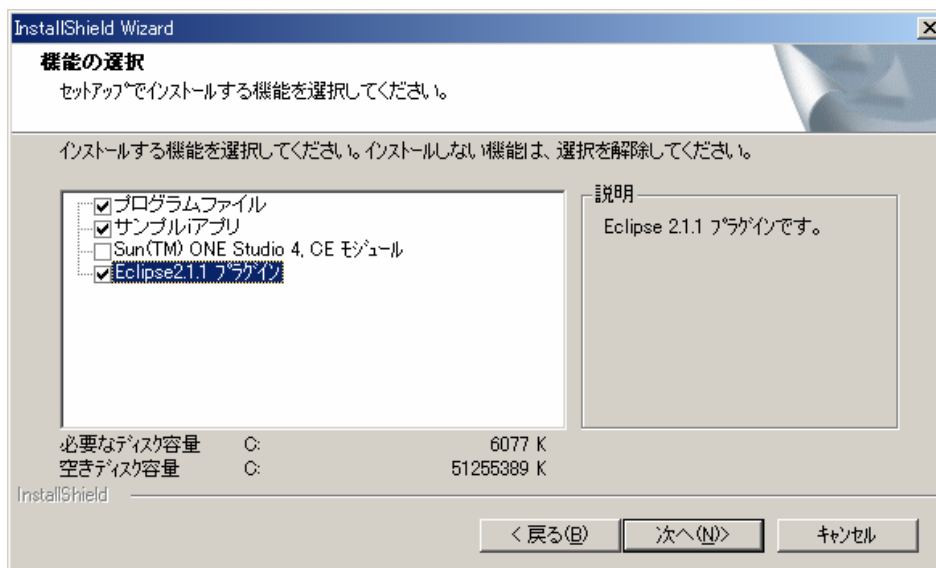


図 14-1 Eclipse Module のインストール

Eclipse は、インストーラが無いので、どこにあるかを指定する必要があります。
Eclipse モジュールをチェックして、次へをクリックすると、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：IDE で開発後、実際の端末へアップロードする際には、なるべくエミュレータの「iαppliTool」を使って作成してください。（「iαppliTool」については、1.1.1 「iαppliTool」開発環境 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
- プロジェクトの作成
- ソースファイルの作成
- プロジェクトメニューのカスタマイズ
- 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
- 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-3.0」のインストール先を設定しておく必要があります。

「Eclipse Module 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-3.0」のインストール先は、「C:/iDKDoJa3.0」になっています。Eclipse に「iα ppli Development Kit for DoJa-3.0」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。

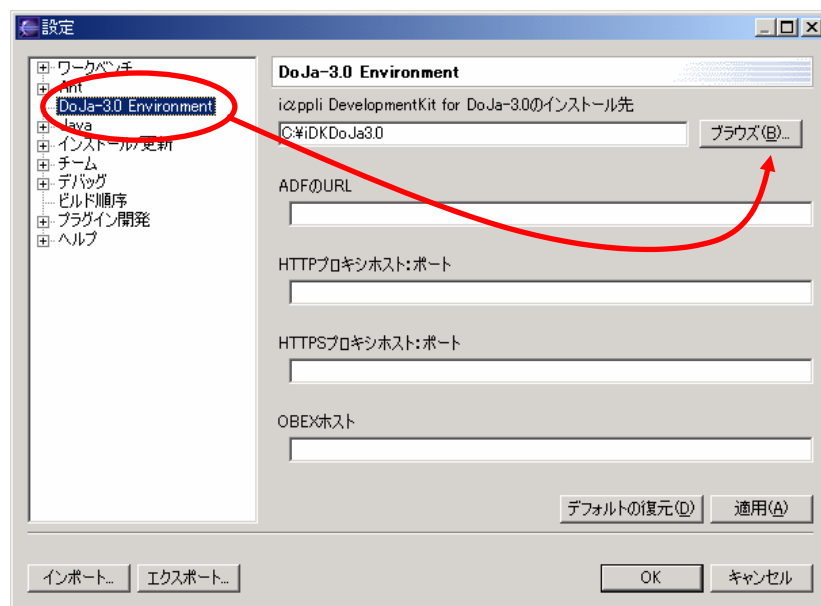


図 14-2 Eclipse 設定ウィンドウ

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-3.0 Environment」を選択すると、「iα ppli Development Kit for DoJa-3.0」のインストール先を設定する画面になります。
3. 「ブラウズ(B)...」をクリックして「iα ppli Development Kit for DoJa-3.0」のインストール先を設定し、「OK」をクリックします。

14.3.2. 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-3.0プロジェクト」を選択し、「次へ」をクリックします。

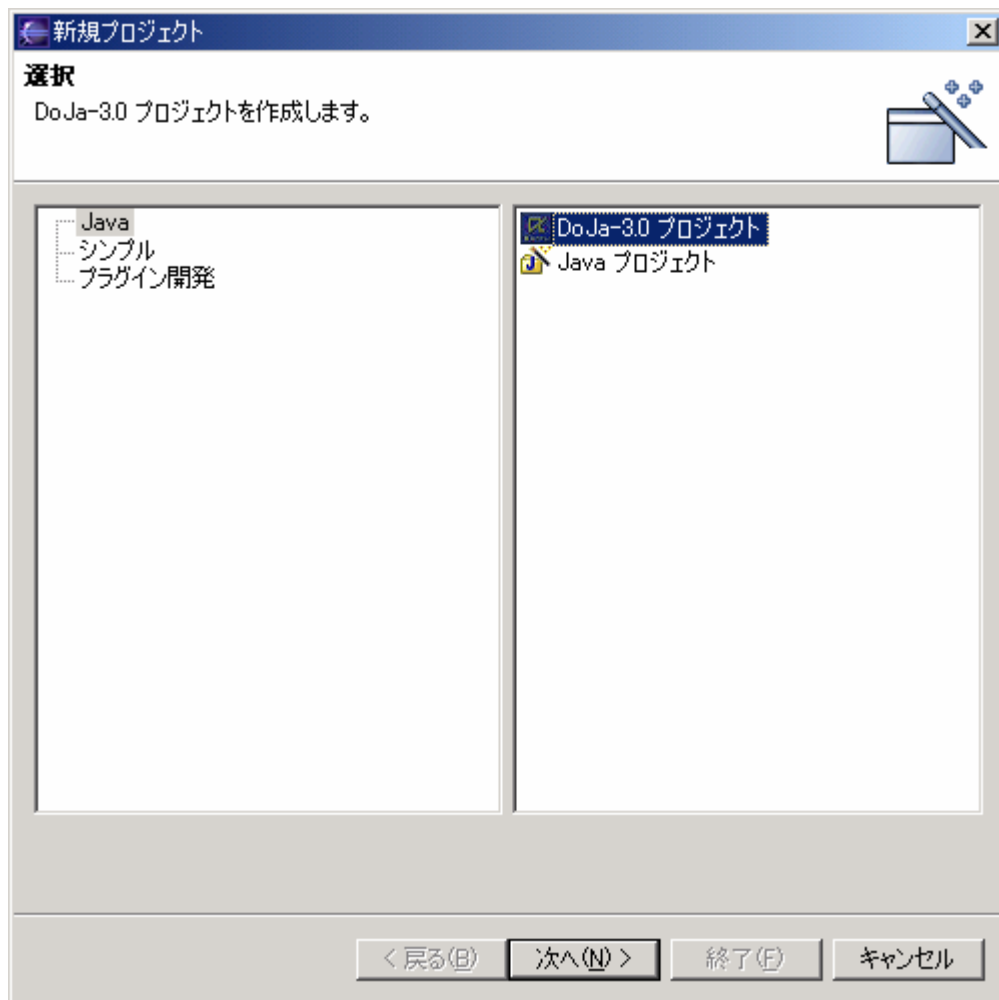


図 14-3 新規プロジェクトウィンドウ

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」をクリックします。
たとえば、hello と入力します。

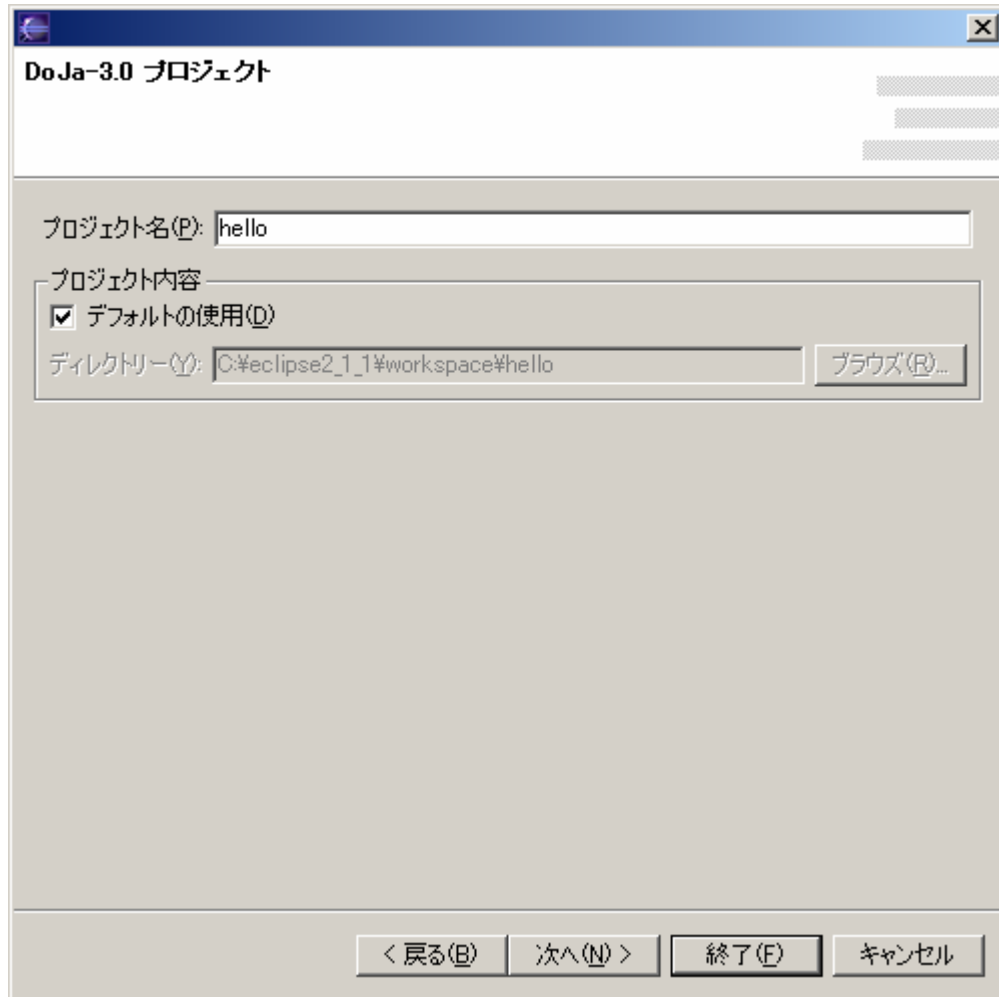


図 14-4 プロジェクト名設定ウィンドウ

- 「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-3.0」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」をクリックします。
(ここでは、何も変更せずに「終了」をクリックします。)

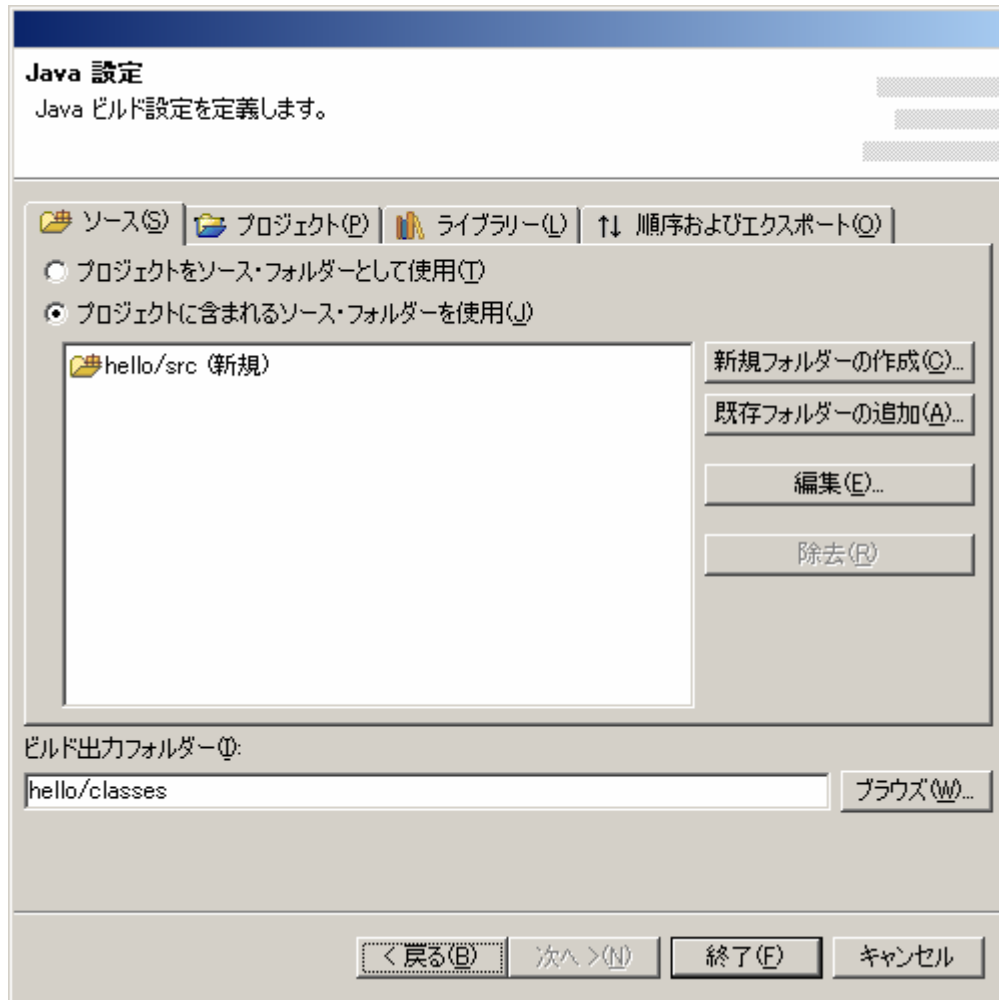


図 14-5 Java 設定ウィンドウ

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。

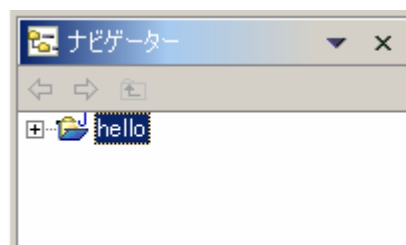


図 14-6 ナビゲーターウィンドウ

14.3.3. 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. ウィンドウ左の「Java パースペクティブ」タブをクリックします。

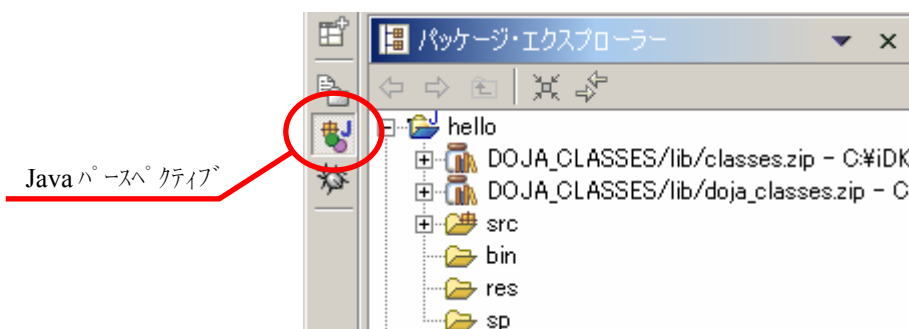


図 14-7 パッケージ・エクスプローラーウィンドウ

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、「src」をクリックして、マウスの右ボタンから「新規」→「クラス」をクリックします。
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」をクリックします。
たとえばパッケージに hello、名前に HelloWorld と入力します。

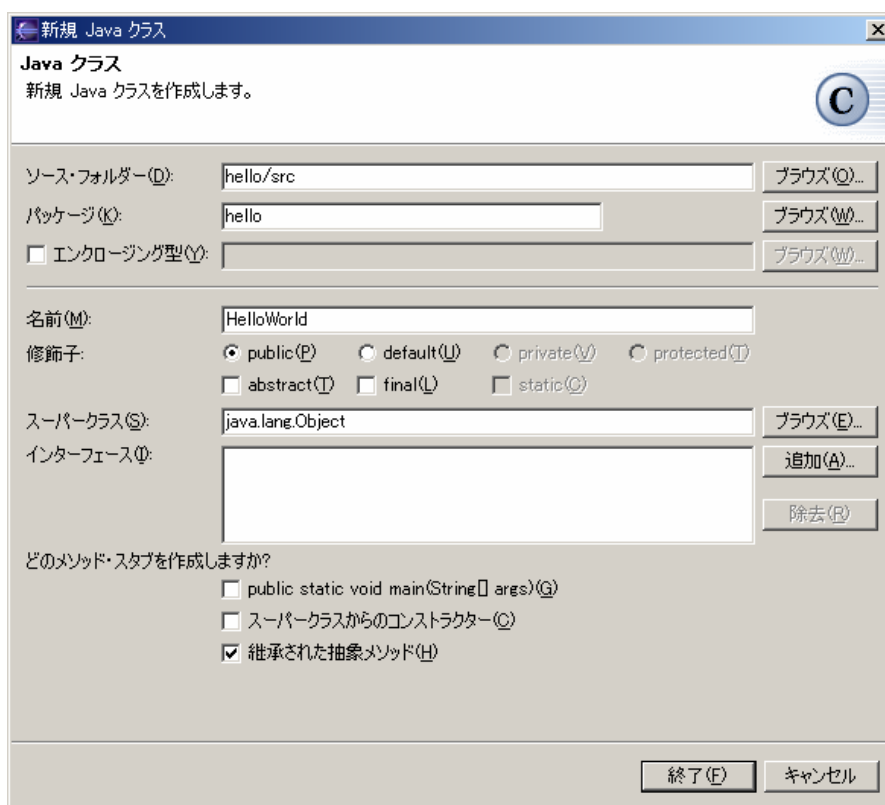


図 14-8 新規クラス設定ウィンドウ

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 { の後に
 public void start() {
 System.out.println("Hello World");
 }
 }
 を追加します。

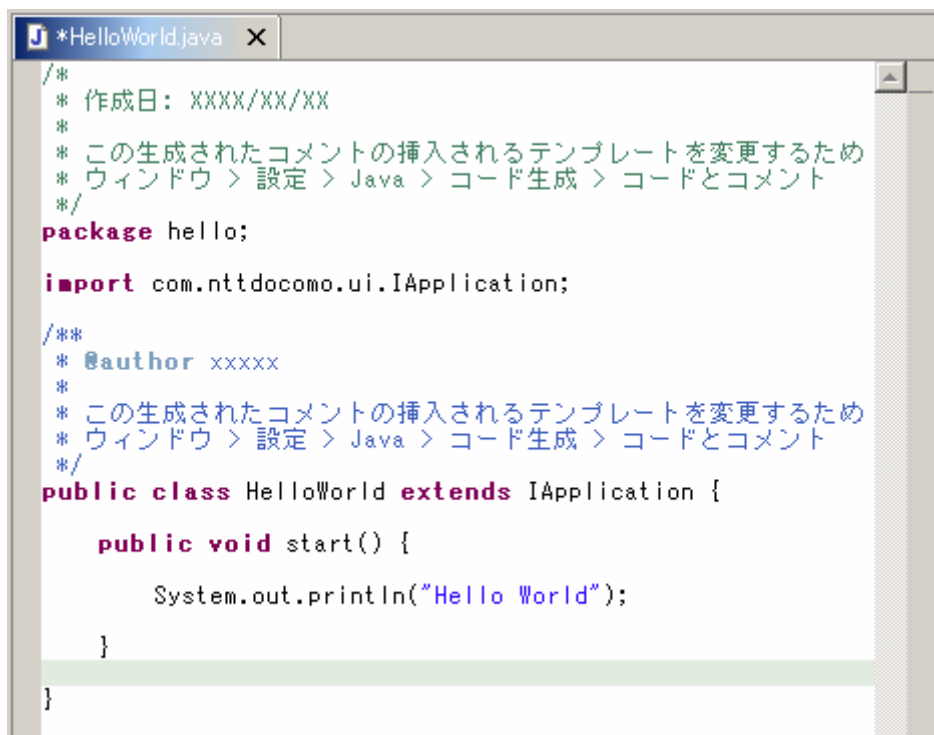


図 14-9 ソースファイルの編集

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ので修正してください。

14.3.4. プロジェクトメニューのカスタマイズ

ADF 設定 と トラストド動作設定をする場合には、次の手順が必要になります。

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. 「その他」→「DoJa-3.0 設定」をチェックします。
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。

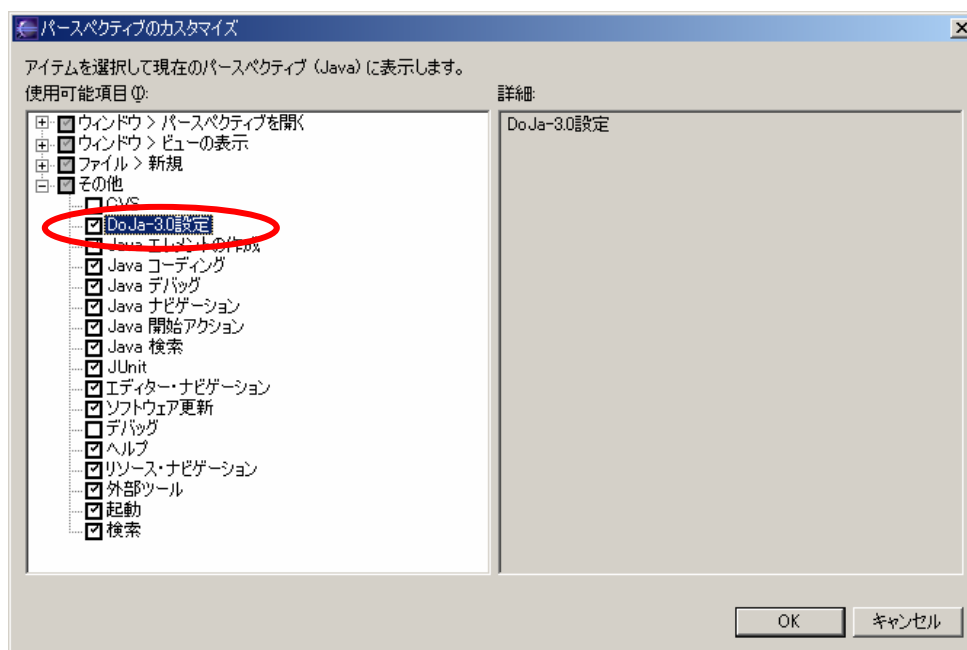


図 14-10 プロジェクトメニューのカスタマイズ

3. パッケージ・エクスプローラーで「プロジェクト」をクリックすると、
「プロジェクト」→「DoJa-3.0 設定」のメニューを選択できるようになります。

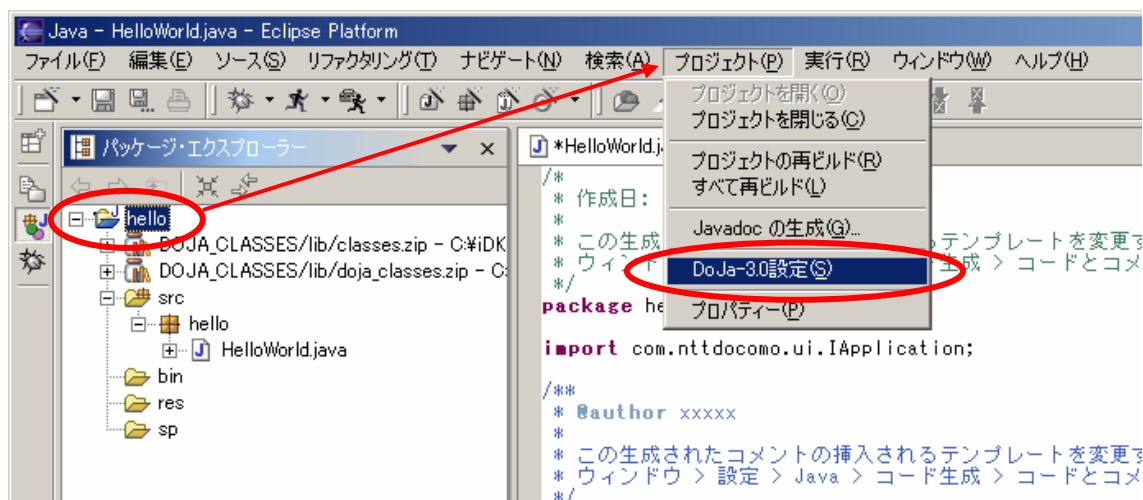


図 14-11 DoJa-3.0 設定メニュー

14.3.5. 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・エクスプローラーからプロジェクト (hello) をクリックして、「プロジェクト」→「DoJa-3.0 設定」を選択します。
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。

ADF キーの AppName と AppClass は必ず設定してください。

たとえば、AppName を hello に AppClass を hello.HelloWorld に変更します。

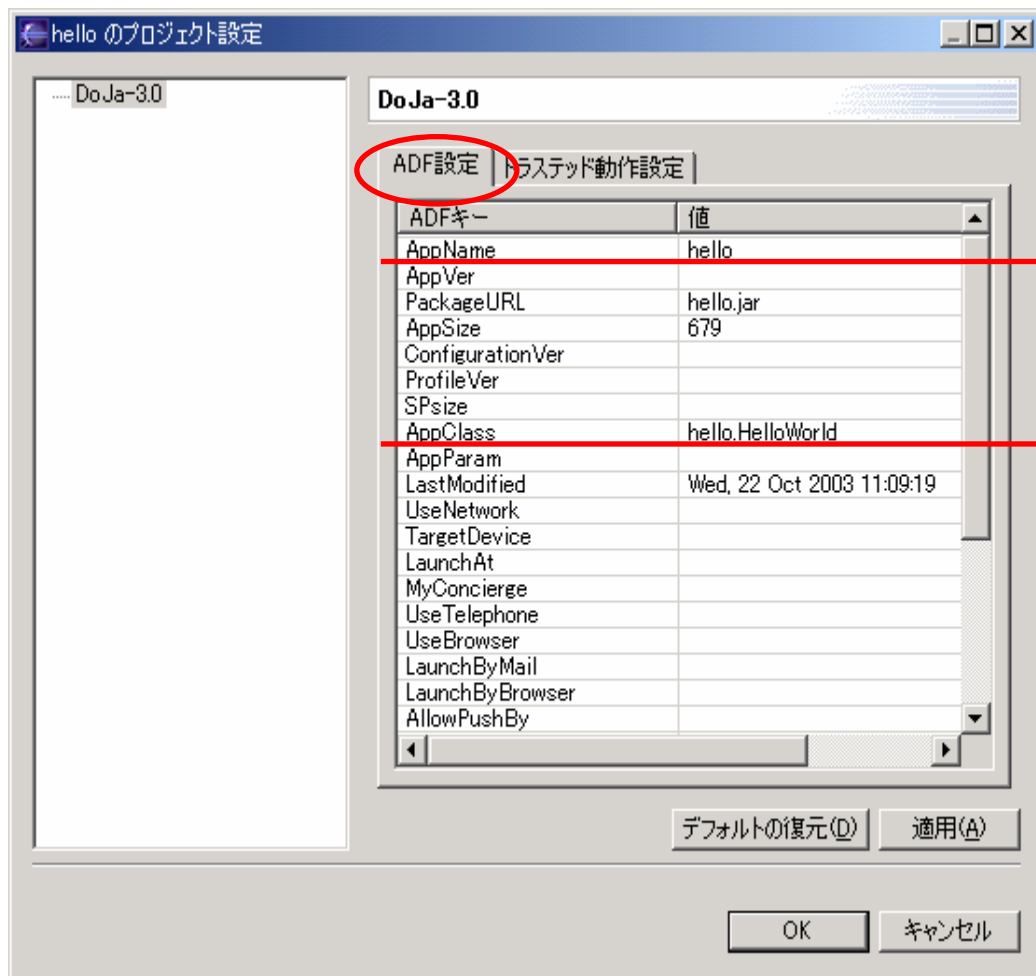


図 14-12 プロジェクト設定-ADF 設定 ウィンドウ

各 ADF の設定項目については、「5.3. アプリケーションディスクリプタファイル (ADF) の編集 表 5-1 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。

14.3.6. トラストッド動作の設定

トラスト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-3.0 設定」をクリックします。
2. プロジェクト設定ウィンドウが表示されるので、「トラストッド動作設定」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。

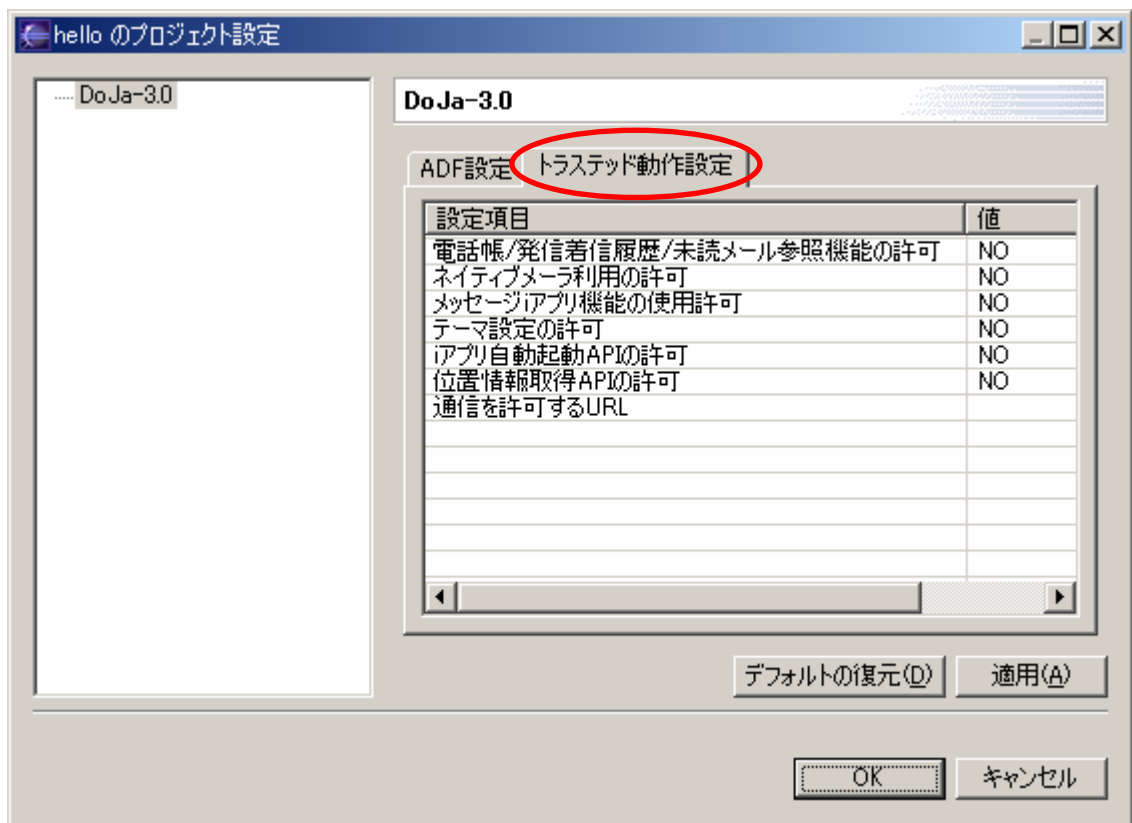


図 14-13 プロジェクト設定-トラストッド動作設定ウィンドウ

各トラストッド動作設定の設定項目については、「4.3. トラストッド動作の設定」を参照してください。

14.3.7. 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」をクリックします。
2. 起動構成ウィンドウが表示されるので、起動構成の「DoJa-3.0 アプリケーション」をマウスでダブルクリック、もしくは、右クリックから「新規」を選択します。
3. 「New_configuration」が追加されるので、これを選択します。
「New_configuration」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. 「Trace Setting」で、出力したいトレースログを選択します。
8. 「実行」をクリックします。

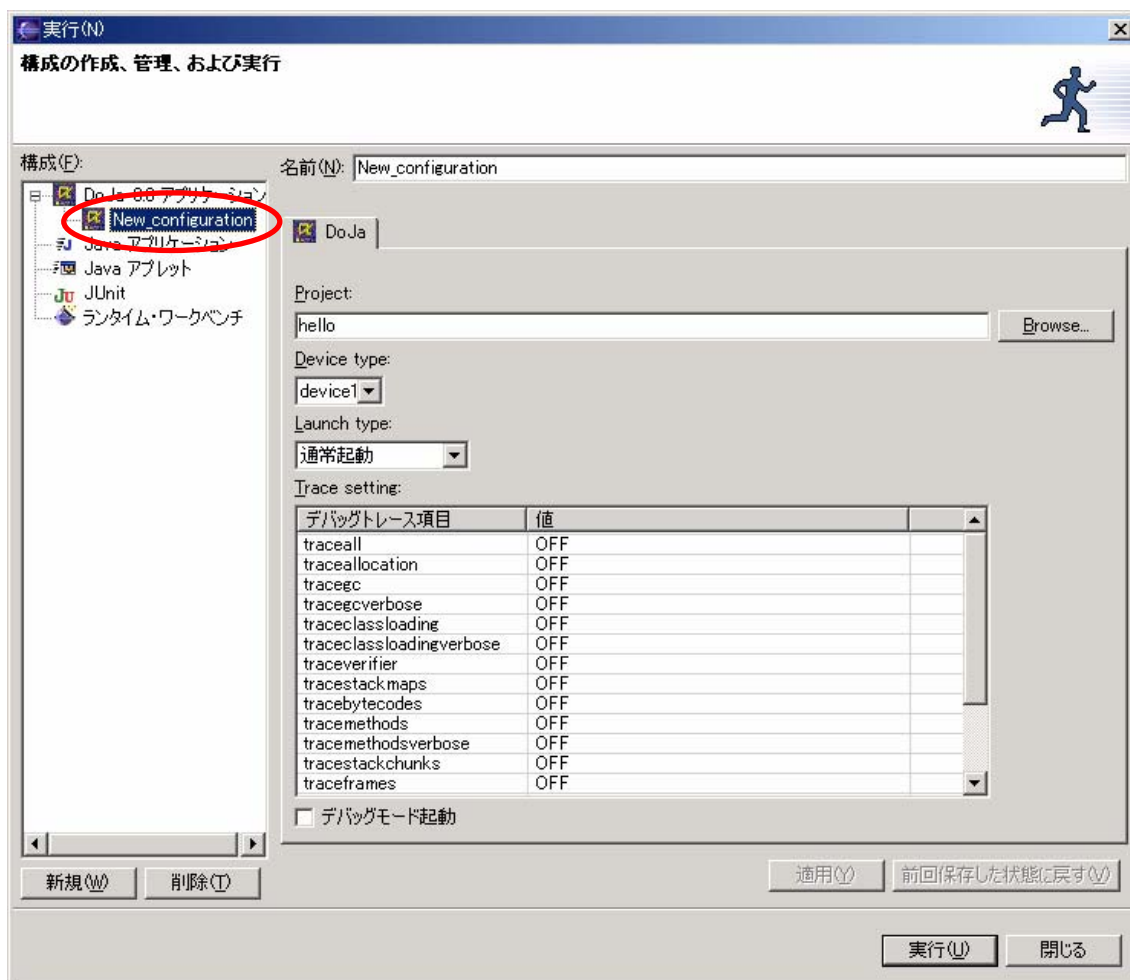


図 14-14 起動構成ウィンドウ

9. 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。
10. 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 14-15）。

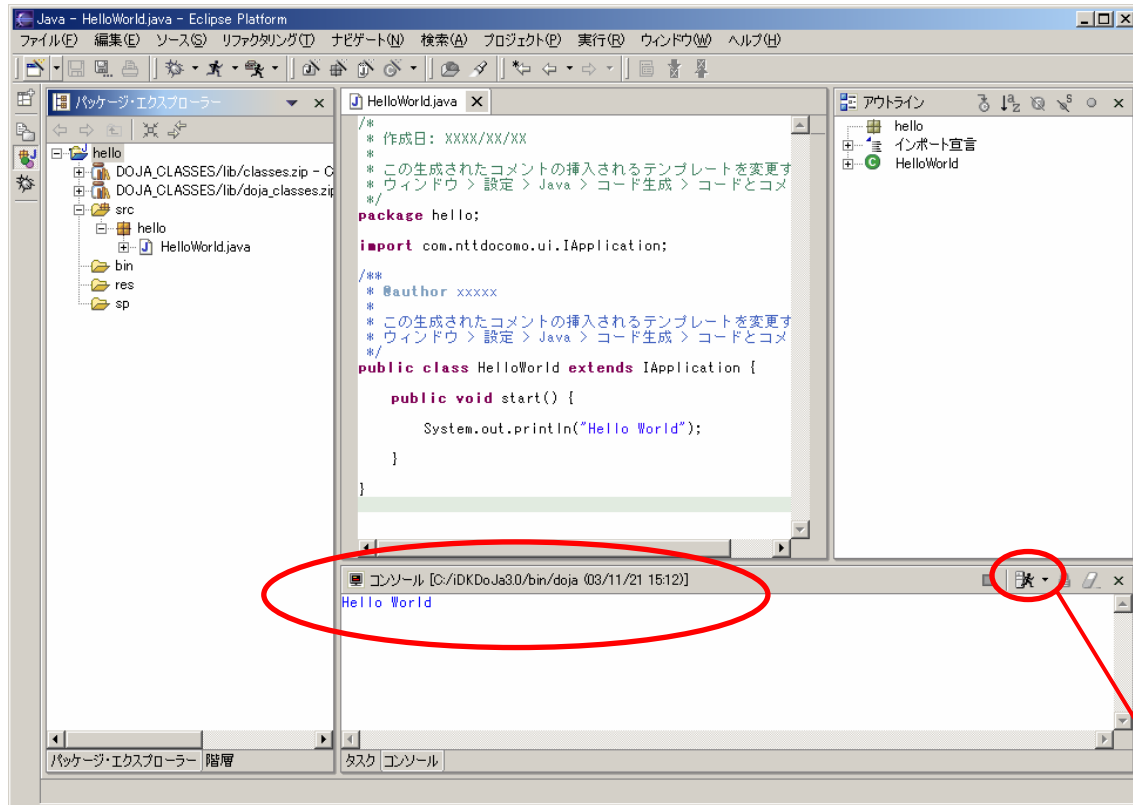


図 14-15 コンソールに出力された内容の表示

表示されない場合は、ここから別のコンソールを選択してください。

14.3.8. 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。

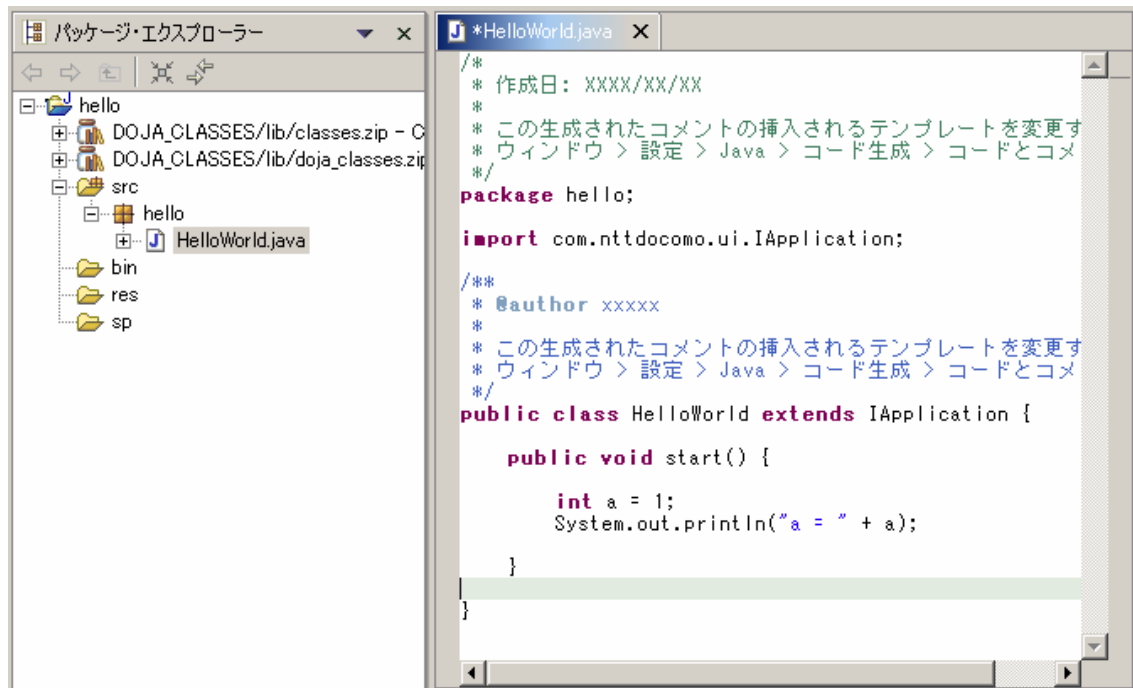


図 14-16 変数の表示

^[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 14-17）。

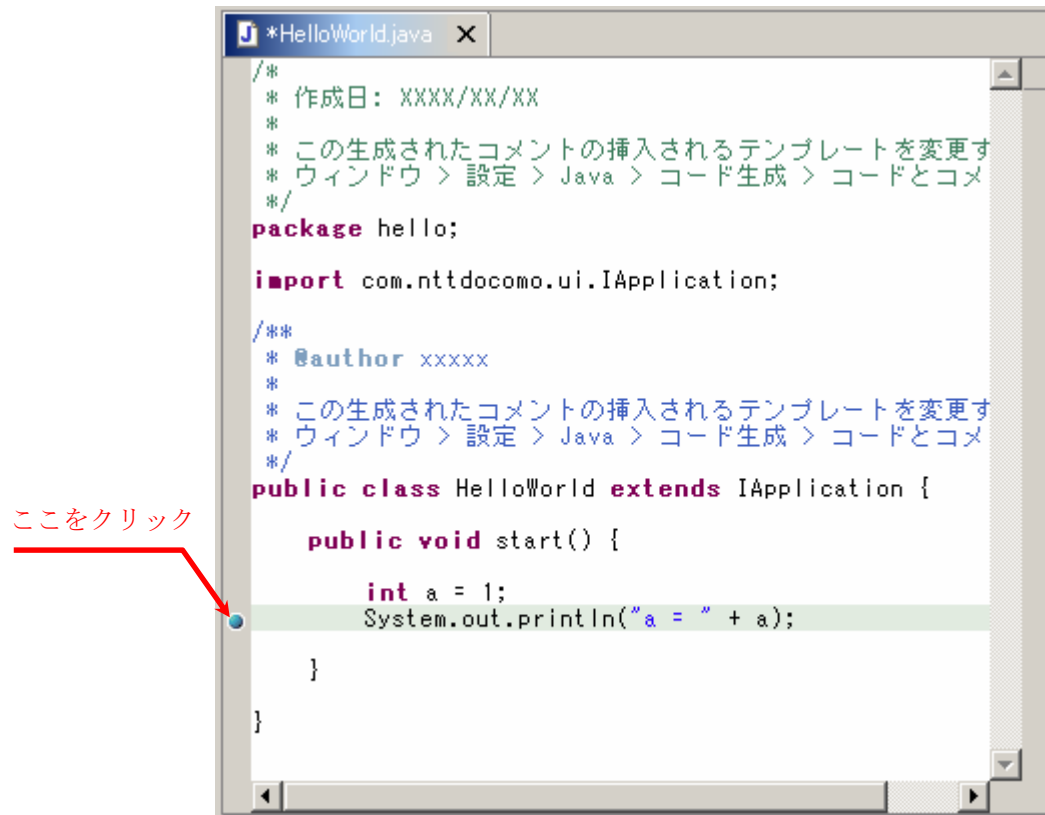


図 14-17 ブレークポイントの設定

3. 「実行」 → 「実行」をクリックします。

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」にチェックを入れて、「実行」をクリックします。

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。

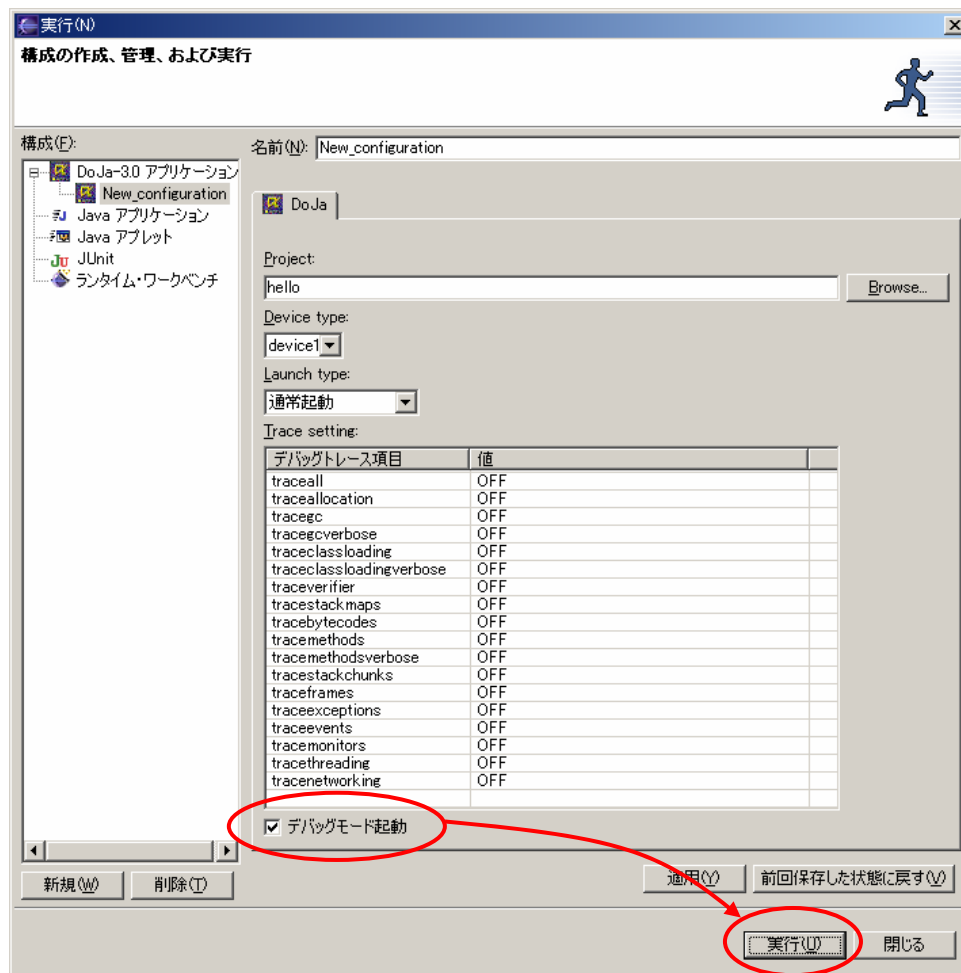


図 14-18 デバッグモードでの実行

5. 端末ウィンドウは開かず、実行が止まったままになります（図 14-19）。

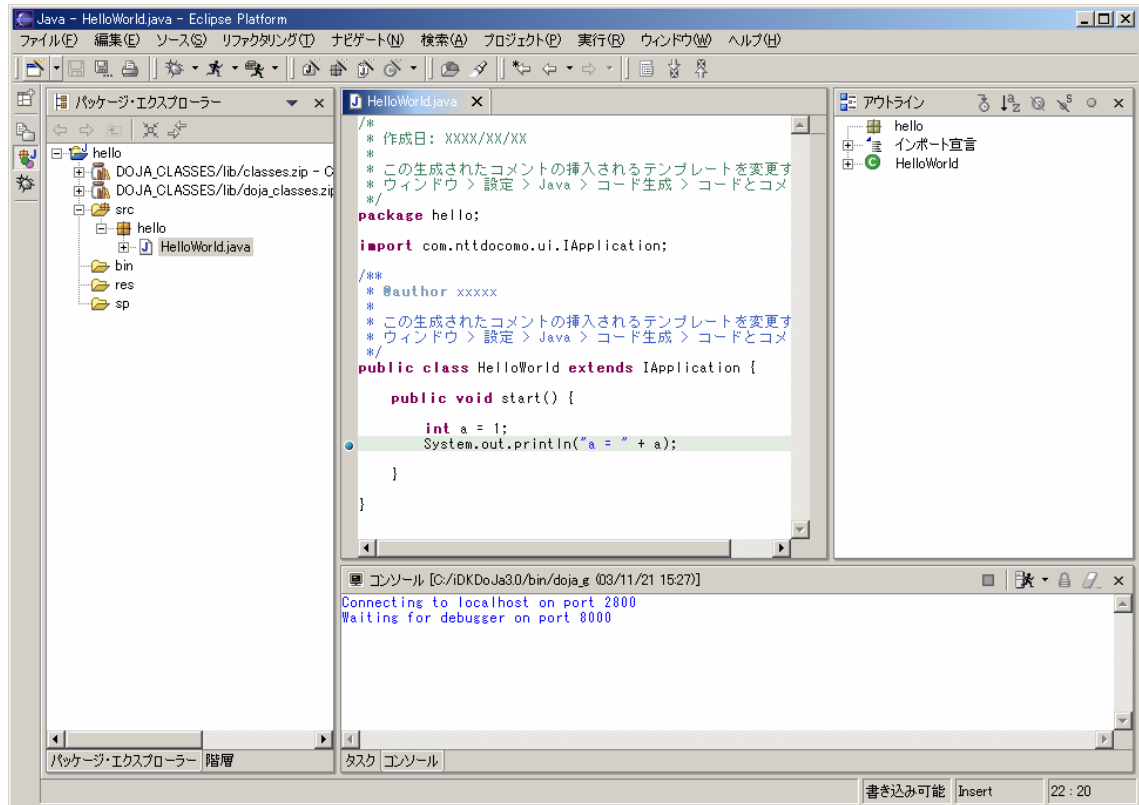


図 14-19 デバッグ実行

6. 「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、右クリックから「新規」を選択します。
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事ができます。
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」をクリックします（図 14-20）。
（この値は、一度設定するとそのまま保持されます。）

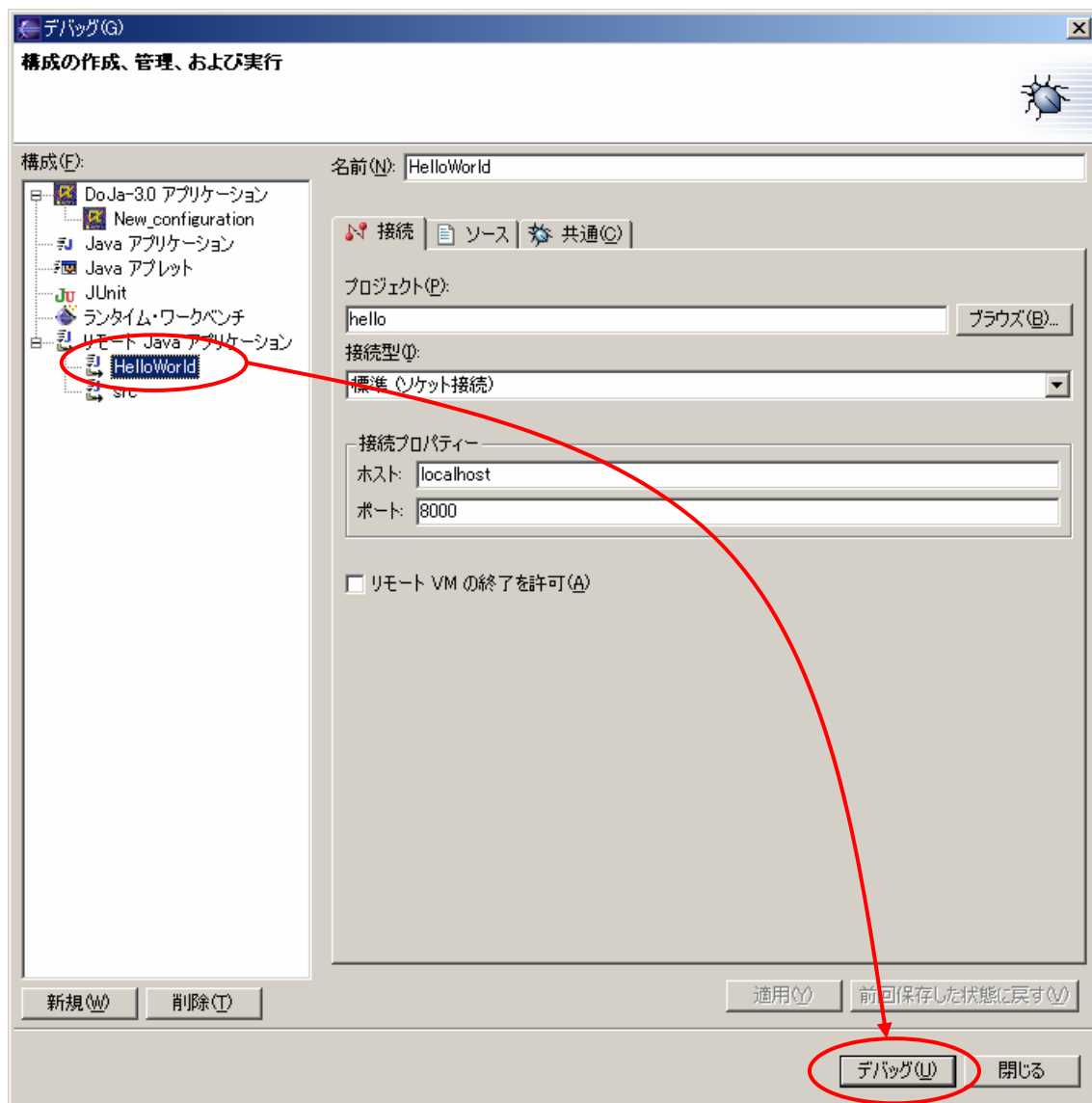


図 14-20 VMへのデバッグ接続

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 14-21）。

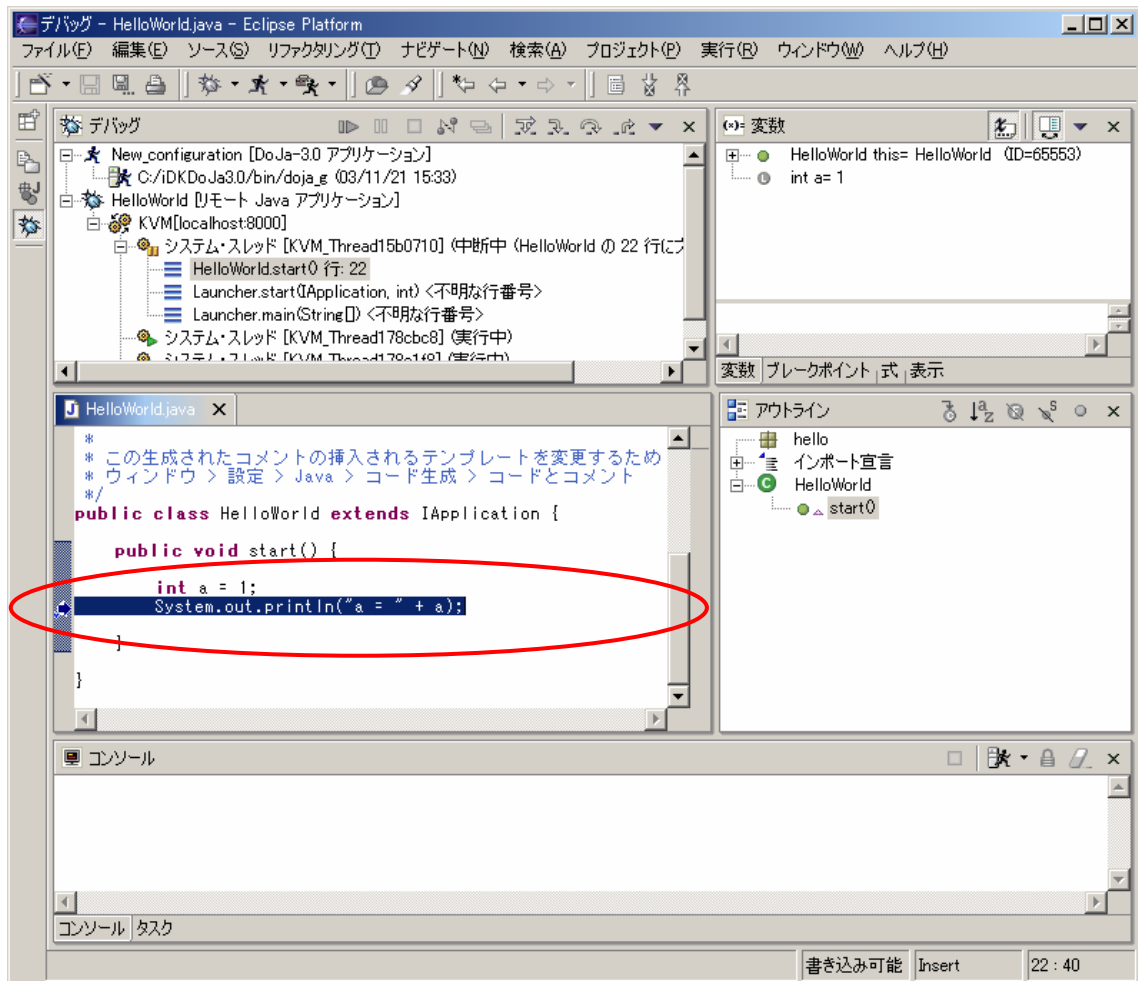


図 14-21 ブレークポイントでの処理の中断

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、「変数の追加」を選択します（図 14-22）。

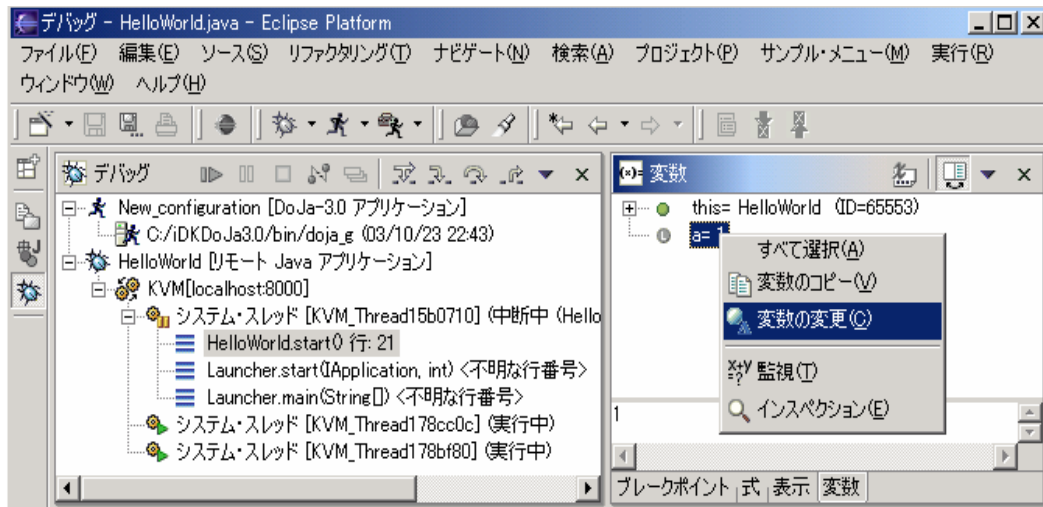


図 14-22 変数の変更

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 14-23）。

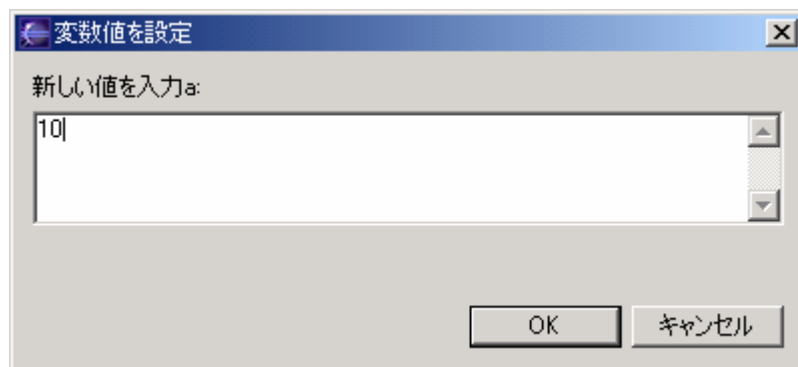


図 14-23 変数値の設定

13. 「実行」→「再開」で中断した位置から処理を再開させます。
変数に変更された状態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。

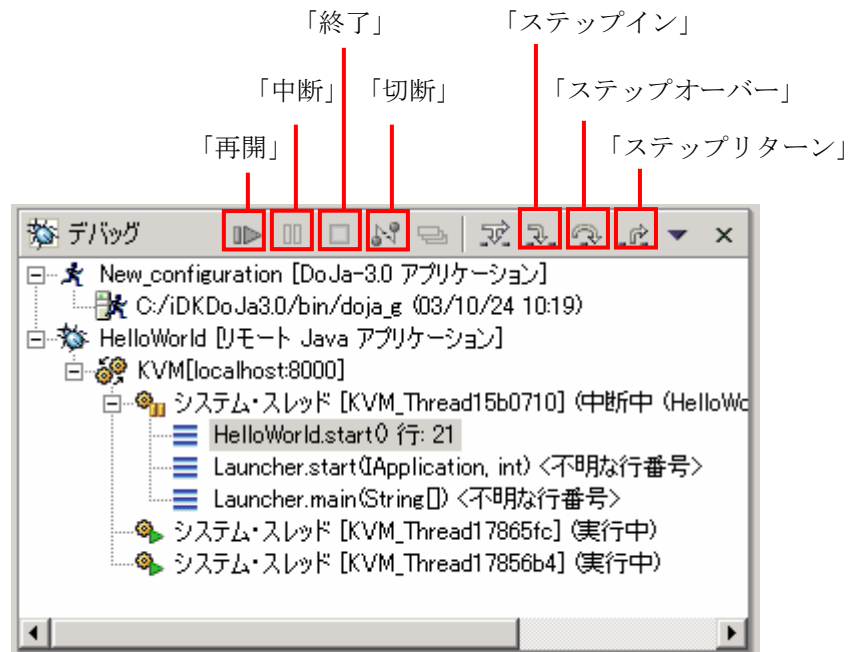


図 14-24 デバッグビューのボタン

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。

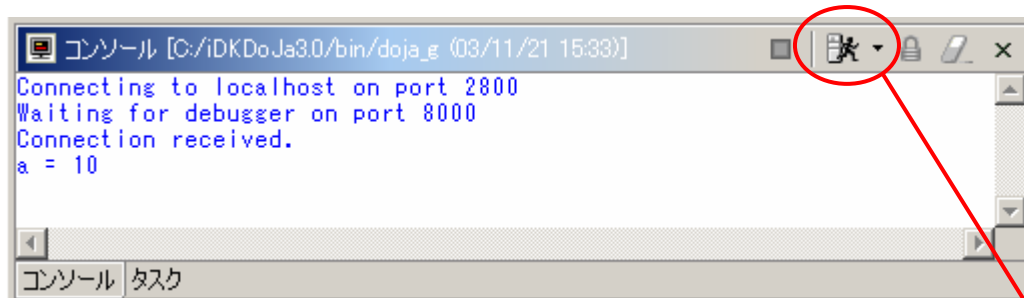


図 14-25 コンソールウィンドウ

表示されない場合は、ここから
別のコンソールを選択してください。

15. 「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. 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（図 14-26）。

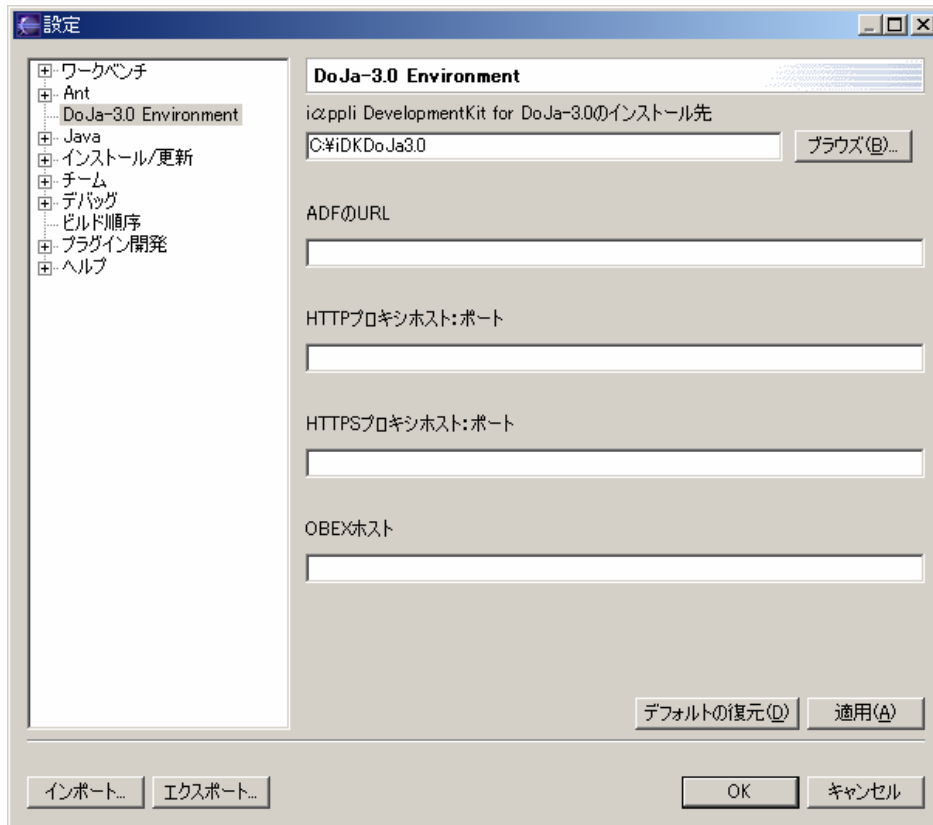


図 14-26 ネットワーク設定ウィンドウ

ネットワークの各設定項目の詳細については、「4.1 ネットワークの設定」を参照してください。

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 アプリオプシオン 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. 指紋センサ・ナビゲーションの設定

F 用 i アプリオプシオン API `FingerSensor` の表示領域を指定できます。

`fingersensor` = 352, 274, 55, 52, 1

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 = 271
lcd_system_height = 15
lcd_system_width = 36, 12, 13, 11, 20, 25, 25, 17, 36, 15, 15, 20, 18
num_cameras = 1
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
```

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]
```

ボタンの位置はエミュレータウィンドウからの継続になります。

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 = 120, 320
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
```