

Oracle9i Lite for Windows CE/Pocket PC

開発者ガイド

リリース 5.0.2.1.0

2003 年 4 月

部品番号 : J07299-01

ORACLE®

Oracle9i Lite for Windows CE/Pocket PC 開発者ガイド, リリース 5.0.2.1.0

部品番号 : J07299-01

原本名 : Oracle9i Lite Developer's Guide, Release 5.0.2.1.0 for Windows CE/Pocket PC

原本部品番号 : B10345-01

Copyright © 2001, 2002, Oracle Corporation. All rights reserved.

Printed in Japan.

制限付権利の説明

プログラム（ソフトウェアおよびドキュメントを含む）の使用、複製または開示は、オラクル社との契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当プログラムのリバース・エンジニアリング等は禁止されております。

このドキュメントの情報は、予告なしに変更されることがあります。オラクル社は本ドキュメントの無謬性を保証しません。

* オラクル社とは、Oracle Corporation（米国オラクル）または日本オラクル株式会社（日本オラクル）を指します。

危険な用途への使用について

オラクル社製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションを用途として開発されておりません。オラクル社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、日本オラクル株式会社と開発元である Oracle Corporation（米国オラクル）およびその関連会社は一切責任を負いかねます。当プログラムを米国国防総省の米国政府機関に提供する際には、『Restricted Rights』と共に提供してください。この場合次の Notice が適用されます。

Restricted Rights Notice

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

このドキュメントに記載されているその他の会社名および製品名は、あくまでその製品および会社を識別する目的にのみ使用されており、それぞれの所有者の商標または登録商標です。

目次

はじめに	xv
------------	----

1 概要

1.1	はじめに	1-2
1.2	Oracle9i Lite のアプリケーション・モデルおよびアーキテクチャ	1-4
1.2.1	Oracle Lite RDBMS	1-6
1.2.2	Mobile SQL (MSQL)	1-6
1.2.3	Mobile Sync	1-7
1.2.4	Mobile サーバー	1-7
1.2.5	Message Generator and Processor (MGP)	1-8
1.2.6	Mobile サーバー・リポジトリ	1-9
1.3	Mobile Development Kit	1-10
1.3.1	サポート対象の Windows CE オペレーティング・システム	1-12
1.3.2	Mobile Development Kit のディレクトリ構造	1-12
1.4	アプリケーション開発	1-13
1.4.1	Oracle Lite ユーティリティ	1-13
1.4.2	開発ツール	1-14
1.4.3	開発インタフェース	1-14
1.5	開発過程	1-16
1.5.1	開発システムの構成	1-17
1.5.2	パブリケーション・アイテムの作成	1-17
1.5.3	Windows CE デバイスの設定	1-19
1.5.4	Windows CE デバイスとの同期	1-19
1.5.5	Windows CE 用のモバイル・アプリケーションの開発	1-20
1.5.6	デバイスでのテスト	1-20

1.5.7	アプリケーションのパッケージ化	1-20
1.6	制限事項	1-20

2 Windows CE 用のオフライン・モバイル・アプリケーションの構築：チュートリアル

2.1	概要	2-2
2.1.1	作業の準備	2-2
2.2	アプリケーションの開発	2-3
2.2.1	Oracle Lite でのデータベース・オブジェクトの作成	2-3
2.2.2	アプリケーション・コードの記述	2-6
2.2.3	アプリケーションのコンパイル	2-13
2.3	アプリケーションのパッケージ化およびパブリッシュ	2-14
2.3.1	パッケージ・ウィザードを使用したアプリケーションの定義	2-14
2.3.2	Oracle データベース・サーバーへのアプリケーション接続の定義	2-18
2.3.3	スナップショットの定義	2-20
2.3.4	アプリケーションのパブリッシュ	2-24
2.4	アプリケーションの管理	2-26
2.4.1	Mobile サーバーの起動	2-27
2.4.2	Mobile サーバーのコントロール・センターの起動	2-27
2.4.3	新規ユーザーの作成	2-28
2.4.4	アプリケーションのプロパティの設定	2-30
2.4.5	ユーザーに対するアプリケーションへのアクセス権の付与	2-31
2.4.6	Message Generator and Processor (MGP) の起動	2-32
2.5	Pocket PC でのアプリケーションの実行	2-33
2.5.1	Pocket PC 用 Mobile クライアントのインストール	2-33
2.5.2	Transport アプリケーションおよびデータのインストールと同期	2-36

3 同期

3.1	概要	3-2
3.1.1	同期の概念	3-2
3.2	同期プロセス	3-3
3.2.1	高速リフレッシュ同期	3-3
3.2.2	完全リフレッシュ同期	3-5
3.2.3	暗号化されたデータベースの同期	3-6
3.3	CE デバイス上の Mobile クライアント	3-6

3.3.1	ActiveSync の概要	3-6
3.3.2	Mobile サーバーからのファイルのインストール	3-6
3.3.3	データ・ソース名の作成	3-6
3.3.4	フラッシュ・メモリー・カードへのデータベースの格納	3-7
3.3.5	ActiveSync との同期	3-8
3.4	RAS トランスポートの構成	3-11
3.4.1	NT RAS の構成	3-11
3.4.2	デバイスの RAS User アカウントの作成	3-13
3.4.3	Windows デスクトップの RAS 設定	3-13
3.4.4	Windows デスクトップの設定	3-14
3.5	デバイスへのサンプル・アプリケーションのインストール	3-14
3.6	同期のテスト	3-14
3.7	Mobile Sync Application Program Interface (API)	3-16
3.7.1	Java インタフェース	3-16
3.7.2	COM インタフェース	3-24
3.7.3	C/C++ インタフェース	3-30
3.7.4	選択同期	3-41

4 ActiveX Data Objects for Windows CE

4.1	概要	4-2
4.2	機能	4-2
4.3	ActiveX Data Objects for Windows CE のオブジェクト	4-4
4.4	Connection オブジェクトのメソッド	4-5
4.4.1	Open	4-5
4.4.2	BeginTrans	4-6
4.4.3	CommitTrans	4-7
4.4.4	RollbackTrans	4-8
4.4.5	Close	4-8
4.5	Connection オブジェクトのプロパティ	4-8
4.5.1	ConnectionString	4-9
4.5.2	IsolationLevel	4-9
4.6	Recordset オブジェクトのメソッド	4-10
4.6.1	AddNew	4-11
4.6.2	CancelUpdate	4-12
4.6.3	Close	4-14

4.6.4	Delete	4-14
4.6.5	GetRows	4-15
4.6.6	Move	4-17
4.6.7	MoveFirst、MoveLast、MoveNext、MovePrevious	4-19
4.6.8	Open	4-19
4.6.9	Supports	4-22
4.6.10	Update	4-25
4.7	Recordset オブジェクトのプロパティ	4-27
4.7.1	AbsolutePosition	4-28
4.7.2	BOF、EOF	4-29
4.7.3	CursorType	4-30
4.7.4	EditMode	4-30
4.7.5	LockType	4-31
4.7.6	RecordCount	4-32
4.7.7	Source	4-32
4.8	Field オブジェクトのプロパティ	4-33
4.8.1	ActualSize	4-34
4.8.2	Attributes	4-34
4.8.3	DefinedSize	4-36
4.8.4	Name	4-37
4.8.5	Type	4-38
4.8.6	UnderlyingValue	4-40
4.8.7	Value	4-40
4.9	Fields コレクション	4-40
4.10	SQL およびデータベースの参照	4-42
4.11	エラー・メッセージ	4-42
4.11.1	ActiveX Data Objects のエラー	4-43
4.11.2	SQL エラー	4-44

5 パッケージ・ウィザードの使用

5.1	パッケージ・ウィザードの概要	5-2
5.2	パッケージ・ウィザードの起動	5-2
5.3	プラットフォームの選択	5-4
5.4	新規アプリケーションの命名	5-4
5.4.1	プラットフォーム・ファイルの検索	5-6

5.5	アプリケーション・ファイルのリスト表示	5-7
5.5.1	ソート	5-9
5.5.2	フィルタ	5-9
5.6	データベース情報の入力	5-10
5.7	レプリケーション用スナップショットの定義	5-11
5.7.1	新規スナップショットの作成	5-14
5.7.2	スナップショットのインポート	5-17
5.7.3	スナップショットの編集	5-18
5.8	アプリケーションの完了	5-20
5.8.1	アプリケーション・ファイル	5-21
5.8.2	JAR ファイルの作成	5-21
5.8.3	SQL ファイルの作成	5-22
5.8.4	パッケージ・ウィザードの再起動	5-22
5.8.5	アプリケーションのパブリッシュ	5-22
5.8.6	アプリケーションの編集	5-23

6 Consolidator

6.1	Consolidator API を使用したパブリッシュ・サブスクライブ・モデル	6-2
6.1.1	パブリッシュ・サブスクライブ・モデルの使用方法	6-3
6.1.2	Consolidator の機能の比較	6-4
6.2	Consolidator を使用したサンプル・プログラム Sample11.java の定義	6-6
6.2.1	Sample11.java	6-7
6.2.2	標準 JDBC を使用した必須表の作成	6-10
6.2.3	Mobile サーバーへの接続	6-11
6.2.4	パブリケーションの作成	6-11
6.2.5	パブリケーション・アイテムの作成	6-13
6.2.6	パブリケーションに対するクライアント・サブスクリプション・パラメータの定義	6-16
6.2.7	パブリケーション・アイテム索引の作成	6-17
6.2.8	パブリケーションへのパブリケーション・アイテムの追加	6-18
6.2.9	ユーザーの作成	6-21
6.2.10	ユーザーの削除	6-22
6.2.11	パブリケーションへのユーザーのサブスクライブ	6-23
6.2.12	サブスクリプションのインスタンス化	6-23
6.3	Consolidator のその他の標準機能	6-24
6.3.1	クライアント・デバイス・データベースの DDL 操作	6-25

6.3.2	パスワードの変更	6-25
6.3.3	シーケンスの作成	6-25
6.3.4	クライアントのシーケンスのパーティション化	6-26
6.3.5	リモート・データベース・リンクのサポート	6-28
6.4	Consolidator をカスタマイズするための拡張機能	6-32
6.4.1	MyCompose を使用した構成フェーズのカスタマイズ	6-32
6.4.2	Sync Discovery API	6-42
6.4.3	マップ表のパーティション API	6-47
6.4.4	AlterPublicationItem を使用したパブリケーション・アイテムの変更	6-50
6.4.5	複数表パブリケーション（ビュー）の高速リフレッシュおよび更新操作	6-51
6.4.6	仮想主キー	6-54
6.4.7	パブリケーション・アイテムの間合せのキャッシング	6-56
6.4.8	ユーザー定義の PL/SQL プロシージャのバインド	6-57
6.4.9	レプリケーションをカスタマイズするためのキュー・インタフェース	6-58
6.4.10	Null Sync コールアウト	6-63
6.4.11	更新可能パブリケーション・アイテムの外部キー制約	6-63
6.4.12	構成と適用を使用する前後に行うコールバックのカスタマイズ	6-65
6.4.13	DML 操作のコールバックのカスタマイズ	6-66
6.4.14	制限選択条件	6-69
6.5	同期エラーと競合	6-69
6.5.1	バージョンング	6-69
6.5.2	ウィニング・ルール	6-69
6.5.3	エラー・キューを使用した競合の解決	6-70
6.6	Oracle サーバーとクライアント間でのデータ型のマッピング	6-71
6.6.1	Oracle Lite データ型	6-71

A システム・カタログ・ビュー

A.1	Oracle Lite データベースのカタログ・ビュー	A-2
A.1.1	ALL_COL_COMMENTS	A-3
A.1.2	ALL_CONSTRAINTS	A-3
A.1.3	ALL_CONS_COLUMNS	A-4
A.1.4	ALL_INDEXES	A-5
A.1.5	ALL_IND_COLUMNS	A-5
A.1.6	ALL_OBJECTS	A-6
A.1.7	ALL_PARTITIONS	A-6

A.1.8	ALL_SEQUENCES	A-7
A.1.9	ALL_SYNONYMS	A-7
A.1.10	ALL_TABLES	A-8
A.1.11	ALL_TAB_COLUMNS	A-9
A.1.12	ALL_TAB_COMMENTS	A-11
A.1.13	ALL_USERS	A-11
A.1.14	ALL_VIEWS	A-12
A.1.15	CAT	A-12
A.1.16	COLUMN_PRIVILEGES	A-12
A.1.17	DATABASE_PARAMETERS	A-13
A.1.18	DUAL	A-13
A.1.19	TABLE_PRIVILEGES	A-14
A.1.20	USER_OBJECTS	A-14

B Oracle Lite ユーティリティ

B.1	CE デバイスでのコマンドライン・ユーティリティの実行	B-2
B.1.1	Compaq 社の iPaq	B-2
B.1.2	HP 社の Jornada と Casio	B-2
B.2	CREATEDB	B-2
B.3	DECRYPDB	B-3
B.4	ENCRYPDB	B-4
B.4.1	暗号化された Oracle Lite データベースとの同期	B-7
B.5	Mobile SQL	B-7
B.5.1	データベース・アクセス	B-7
B.5.2	Mobile SQL の起動	B-7
B.6	ODBINFO	B-8
B.7	REMOVEDB	B-10

C Windows CE 用 MSQLE

C.1	概要	C-2
C.2	「接続」 ページ	C-2
C.3	「表」 ページ	C-3
C.4	「ビュー」 ページ	C-4
C.5	「順序」 ページ	C-5
C.6	「SQL」 ページ	C-6

C.7	「ツール」 ページ	C-6
-----	-----------------	-----

用語集

索引

図目次

1-1	Oracle9i Lite の配布アーキテクチャ	1-5
1-2	Java 開発モデル	1-14
1-3	Active Data Objects 開発モデル	1-16
2-1	新しいプロジェクトの作成	2-6
2-2	eMbedded Visual Basic でオープンされた Transport アプリケーション・プロジェクト	2-8
2-3	アプリケーションの選択	2-15
2-4	プラットフォームの選択	2-16
2-5	「アプリケーション」 パネル	2-17
2-6	アップロードするファイルの選択	2-18
2-7	「データベース」 パネル	2-19
2-8	「データベースへの接続」 ウィンドウ	2-20
2-9	表のインポート	2-21
2-10	WinCE の「スナップショット」 パネル	2-21
2-11	PACKAGES 表の「スナップショットの編集」 パネル	2-22
2-12	ROUTES 表の「スナップショットの編集」 パネル	2-24
2-13	「アプリケーションの定義の完了」 ダイアログ・ボックス	2-25
2-14	「アプリケーションのパブリッシュ」 ダイアログ・ボックス	2-25
2-15	「アプリケーション」 ウィンドウ	2-28
2-16	「ユーザー・プロパティ」 ウィンドウ	2-29
2-17	アプリケーションのプロパティの設定	2-30
2-18	アプリケーションへのアクセス権の付与	2-31
2-19	「MGP コントロール」 ウィンドウ	2-32
2-20	Pocket PC 用 Mobile クライアントのインストール	2-34
2-21	Pocket PC 用 Mobile クライアントのインストール	2-35
2-22	mSync の実行	2-36
3-1	高速リフレッシュ同期	3-3
3-2	アップロードおよびダウンロード・フェーズ	3-4
3-3	適用および構成フェーズ	3-5
3-4	「同期モード」 のオプション	3-8
3-5	ActiveSync の「mSync」 アイコン	3-9
3-6	「mSync プロバイダ設定」 画面	3-9
3-7	「mSync プロキシ設定」 画面	3-10
3-8	「Mobile Sync」 画面	3-15
5-1	「ようこそ」 パネル	5-3
5-2	プラットフォームの選択画面	5-4
5-3	「アプリケーション」 パネル	5-5
5-4	「ファイル」 パネル	5-8
5-5	「フィルタ」 パネル	5-9
5-6	「データベース」 パネル	5-10
5-7	「スナップショット」 パネル	5-11
5-8	「新規スナップショット」 ウィンドウ	5-14
5-9	索引の作成	5-16
5-10	「データベースへの接続」 ウィンドウ	5-17

5-11	「表」 ウィンドウ	5-18
5-12	「スナップショットの編集」 ウィンドウ	5-19
5-13	「アプリケーションの定義の完了」 ウィンドウ	5-21
5-14	「アプリケーションのパブリッシュ」 ウィンドウ	5-22
C-1	「接続」 ページ	C-2
C-2	「MSQL はデータベースに接続されていません」 メッセージ	C-3
C-3	「表」 ページ	C-4
C-4	「ビュー」 ページ	C-5
C-5	「順序」 ページ	C-5
C-6	「SQL」 ページ	C-6
C-7	「ツール」 ページ	C-7

表目次

1-1	サポートされている言語とサブディレクトリ記述子	1-12
1-2	サポートされているチップセットとサブディレクトリ記述子	1-13
2-1	アプリケーション開発用コンピュータの要件	2-2
2-2	PACKAGES 表	2-4
2-3	ROUTES 表	2-4
2-4	TRUCKS 表	2-5
2-5	Pocket PC Transport アプリケーションの設定	2-17
2-6	「データベース」パネル・ウインドウの説明	2-19
2-7	「データベースへの接続」ウインドウの説明	2-20
2-8	「アプリケーションのパブリッシュ」ウインドウの説明	2-26
2-9	Mobile サーバーのコントロール・センターの「ログイン」ウインドウの説明	2-27
2-10	「ユーザー・プロパティ」ウインドウの説明	2-29
2-11	「mSync」ウインドウの説明	2-36
3-1	「mSync プロバイダ設定」画面のオプション	3-10
3-2	「mSync プロキシ設定」画面のオプション	3-11
3-3	Mobile Sync Client のパラメータ	3-14
3-4	Sync クラス・コンストラクタ	3-17
3-5	Sync クラスのパブリック・メソッド	3-17
3-6	SyncException コンストラクタのパラメータ	3-18
3-7	SyncException クラスのパブリック・メソッド	3-18
3-8	SyncOption コンストラクタ	3-19
3-9	SyncOption のパブリック・メソッドのパラメータ	3-19
3-10	Java インタフェースの SyncParam 設定	3-20
3-11	TransportParam パラメータ	3-22
3-12	SyncProgressListener の抽象メソッド	3-23
3-13	SyncProgressListener インタフェースの定数	3-23
3-14	ISync インタフェースのパラメータ	3-25
3-15	ISyncOption のパブリック・メソッド	3-26
3-16	ISyncOption のパブリック・プロパティ	3-26
3-17	COM インタフェースの SyncParam 設定	3-27
3-18	COM インタフェースの TransportParam パラメータ	3-28
3-19	ISyncProgressListener の抽象メソッド	3-29
3-20	ISyncProgressListener の定数	3-29
3-21	ocSessionInit のパラメータ	3-30
3-22	ocSessionTerm のパラメータ	3-31
3-23	ocSaveUserInfo のパラメータ	3-31
3-24	ocDoSynchronize のパラメータ	3-32
3-25	ocSetTableSyncFlag のパラメータ	3-33
3-26	ocGetPublication のパラメータ	3-34
3-27	ocEnv 構造体のフィールドのパラメータ	3-36
4-1	ADOCE のオブジェクト型	4-2
4-2	Connection オブジェクトのメソッド	4-5
4-3	Connection.Open のパラメータ	4-5

4-4	Connection オブジェクトのプロパティ	4-8
4-5	Recordset オブジェクトのメソッド	4-10
4-6	AddNew のパラメータ	4-11
4-7	Delete のパラメータ	4-14
4-8	GetRows のパラメータ	4-15
4-9	Move のパラメータ	4-17
4-10	Open のパラメータ	4-20
4-11	Supports のパラメータ	4-23
4-12	Update のパラメータ	4-25
4-13	Recordset オブジェクトのプロパティ	4-27
4-14	BOF および EOF のパラメータ	4-30
4-15	編集モードの戻り値	4-31
4-16	ロック・モードの戻り値	4-31
4-17	Field オブジェクトのプロパティ	4-33
4-18	Attributes の戻り値	4-35
4-19	Type の戻り値	4-38
4-20	OLADOCE ではサポートされていない Microsoft ADOCE システム	4-42
4-21	ADOCE のエラー	4-43
4-22	SQL エラー	4-44
4-23	Oracle Lite 内部のネイティブ・エラーへの SQL エラーのマッピング	4-45
5-1	「ようこそ」パネルのオプション	5-3
5-2	「アプリケーション」パネルのオプション	5-5
5-3	「ファイル」パネルのオプション	5-8
5-4	「データベース」パネルのオプション	5-10
5-5	スナップショット・パラメータ	5-12
5-6	「新規スナップショット」ウィンドウのオプション	5-14
5-7	「スナップショットの編集」ウィンドウのオプション	5-19
5-8	「アプリケーションのパブリッシュ」ウィンドウのオプション	5-23
6-1	パブリッシュ・サブスクライブ・モデルの要素	6-2
6-2	Consolidator の基本機能の比較	6-4
6-3	Consolidator の拡張機能の比較	6-5
6-4	CreatePublication のパラメータ	6-12
6-5	クライアントの格納タイプ	6-12
6-6	CreatePublicationItem のパラメータ	6-14
6-7	SetSubscriptionParameter のパラメータ	6-16
6-8	CreatePublicationItemIndex のパラメータ	6-17
6-9	AddPublicationItem のパラメータ	6-19
6-10	createUser のパラメータ	6-21
6-11	dropUser のパラメータ	6-22
6-12	CreateSubscription のパラメータ	6-23
6-13	InstantiateSubscription のパラメータ	6-24
6-14	setPassword のパラメータ	6-25
6-15	CreateSequencePartition のパラメータ	6-27
6-16	リモート・データベース・リンク用の CreatePublicationItem のパラメータ	6-29
6-17	CreateDependencyHint のパラメータ	6-30

6-18	RemoveDependencyHint のパラメータ	6-31
6-19	needCompose のパラメータ	6-34
6-20	doCompose のパラメータ	6-35
6-21	init のパラメータ	6-36
6-22	destroy のパラメータ	6-36
6-23	getPubItemDMLTableName のビュー構造のパラメータ	6-37
6-24	getBaseTablePK のパラメータ	6-38
6-25	baseTableDirty のパラメータ	6-39
6-26	getBaseTableDMLLogName のパラメータ	6-39
6-27	getBaseTableDMLLogName のビュー構造のパラメータ	6-39
6-28	getMapView のビュー構造のパラメータ	6-40
6-29	RegisterMyCompose のパラメータ	6-41
6-30	DeRegisterMyCompose のパラメータ	6-41
6-31	getDownloadInfo のパラメータ	6-42
6-32	DownloadInfo クラスの access メソッド	6-43
6-33	PublicationSize クラスの access メソッド	6-44
6-34	PartitionMap のパラメータ	6-48
6-35	AddMapPartition のパラメータ	6-49
6-36	AlterPublicationItem のパラメータ	6-51
6-37	ParentHint のパラメータ	6-52
6-38	PrimaryKeyHint のパラメータ	6-53
6-39	CompleteRefresh パラメータ	6-54
6-40	CreateVirtualPKColumn のパラメータ	6-55
6-41	DropVirtualPKColumn のパラメータ	6-55
6-42	EnablePublicationItemQueryCache のパラメータ	6-56
6-43	DisablePublicationItemQueryCache のパラメータ	6-57
6-44	キュー・インタフェース作成のパラメータ	6-60
6-45	CreateQueuePublicationItem のパラメータ	6-62
6-46	Mobile DML 操作のパラメータ	6-66
6-47	ExecuteTransaction のパラメータ	6-70
6-48	PurgeTransaction のパラメータ	6-71
6-49	Oracle Lite データ型	6-72
A-1	ALL_COL_COMMENTS のパラメータ	A-3
A-2	ALL_CONSTRAINTS のパラメータ	A-3
A-3	ALL_CONS_COLUMNS のパラメータ	A-4
A-4	ALL_INDEXES のパラメータ	A-5
A-5	ALL_IND_COLUMNS のパラメータ	A-5
A-6	ALL_OBJECTS のパラメータ	A-6
A-7	ALL_PARTITIONS のパラメータ	A-6
A-8	ALL_SEQUENCES のパラメータ	A-7
A-9	ALL_SYNONYMS のパラメータ	A-7
A-10	ALL_TABLES のパラメータ	A-8
A-11	ALL_TAB_COLUMNS のパラメータ	A-9
A-12	ALL_TAB_COMMENTS のパラメータ	A-11
A-13	ALL_USERS のパラメータ	A-11

A-14	ALL_VIEWS のパラメータ	A-12
A-15	CAT のパラメータ	A-12
A-16	COLUMN_PRIVILEGES のパラメータ	A-12
A-17	DATABASE_PARAMETERS のパラメータ	A-13
A-18	DUAL のパラメータ	A-13
A-19	TABLE_PRIVILEGES のパラメータ	A-14
A-20	USER_OBJECTS のパラメータ	A-14
B-1	ツールとユーティリティ	B-1
B-2	DECRYPDB のリターン・コード	B-4
B-3	ENCRYPDB のリターン・コード	B-5
B-4	ODBInfo のパラメータ	B-8

はじめに

『Oracle9i Lite for Windows CE/Pocket PC 開発者ガイド』では、Oracle Lite とそのコンポーネントを紹介します。このガイドでは、Oracle Lite データベースの開発、配布およびメンテナンスの方法を説明します。

内容は次のとおりです。

第 1 章「概要」

Oracle Lite データベースおよび Mobile Development Kit for Windows CE の概要を説明します。

第 2 章「Windows CE 用のオフライン・モバイル・アプリケーションの構築: チュートリアル」

Oracle9i Lite ADOCE for Pocket PC を使用して、Visual Basic アプリケーションを構築する方法をチュートリアル方式で説明します。

第 3 章「同期」

同期、トランスポートの構成、および Mobile Sync Client のプログラミング・インタフェースについて説明します。

第 4 章「ActiveX Data Objects for Windows CE」

Oracle Lite ActiveX Data Objects for Windows CE について説明します。

第 5 章「パッケージ・ウィザードの使用」

パッケージ・ウィザードを使用してアプリケーションを作成および配布する方法を説明します。

第 6 章「Consolidator」

Consolidator および Resource Manager API を使用した、パブリッシュ・サブスクライブ・モデルのカスタマイズ手法など、高度な項目について説明します。

付録 A 「システム・カタログ・ビュー」

システム・カタログ内のオブジェクト型について説明します。

付録 B 「Oracle Lite ユーティリティ」

Oracle Lite で使用できるデータベースのツールとユーティリティについて説明します。

付録 C 「Windows CE 用 MSQL」

Windows CE 用 Mobile SQL (MSQL) について説明します。

この章では、Oracle9i Lite について紹介し、Mobile Development Kit およびそのコンポーネントを使用した Windows CE 向けアプリケーション開発過程の概要を説明します。

内容は次のとおりです。

- 1.1 項「はじめに」
- 1.2 項「Oracle9i Lite のアプリケーション・モデルおよびアーキテクチャ」
- 1.3 項「Mobile Development Kit」
- 1.4 項「アプリケーション開発」
- 1.5 項「開発過程」
- 1.6 項「制限事項」

1.1 はじめに

Oracle9i Lite により、多くのモバイル・ユーザーを対象としたオフラインのモバイル・データベース・アプリケーションの開発、配布、管理が容易になります。オフラインのモバイル・アプリケーションとは、サーバーと常時接続しなくても、モバイル・デバイスで実行できるアプリケーションです。オフラインのデータベース・アプリケーションを使用するには、モバイル・デバイス上にローカル・データベースが存在する必要があります。このデータベースの内容は、エンタープライズ・データ・サーバーに格納されているデータのサブセットです。アプリケーションによってローカル・データベースに加えられた変更は、サーバーのデータと一致するように随時調整されます。モバイル・データベースとエンタープライズ・データベース間の変更を調整するために使用するテクノロジーは、データ同期と呼ばれます。

オフラインのモバイル・データベース・アプリケーションは、様々な方法で開発できます。モバイル・プラットフォーム専用のネイティブ C または C++ アプリケーションを開発する方法が最も一般的です。これらのアプリケーションは、Open Database Connectivity (ODBC) インタフェース、Active Data Object for CE (ADOCE)、または ODBC 上に構築された他のインタフェースを使用してデータベースにアクセスします。オフラインのモバイル・データベース・アプリケーションを開発する別の方法として、Java および Java Database Connectivity (JDBC) インタフェースを使用する方法もあります。Oracle9i Lite では、Web-to-Go と呼ばれるサプレット・ベースの Web モデルを使用してオフラインのモバイル・データベース・アプリケーションを開発する方法も提供されています。Web-to-Go アプリケーションは、Windows 32 プラットフォームでのみ実行できます。

この章では、Windows CE システムのみを対象としたアプリケーション開発とデータベースの同期を説明します。Windows 32 については、『Oracle9i Lite for Windows 32 開発者ガイド』を参照してください。Palm については、『Oracle9i Lite Developer's Guide for Palm』を参照してください。Web-to-Go の詳細は、『Oracle9i Lite for Web-to-Go 開発者ガイド』を参照してください。

アプリケーションの開発が終了したら、それを配布する必要があります。アプリケーションの配布で重要なのは、エンド・ユーザーがアプリケーションを簡単にインストールして使用できるようにサーバー・システムを設定することです。Oracle9i Lite アプリケーションのサーバー・システムの中核部は、モバイル・アプリケーションが配布される Mobile サーバーです。配布は、次の 5 つの手順で構成されます。

1. 必要なレベルのパフォーマンス、拡張性、セキュリティ、可用性、接続性を実現させるためのサーバー・システムの設計。Oracle9i Lite では、consp perf ユーティリティなどのデータ同期のパフォーマンスをチューニングするためのツールが提供されます。また、拡張性を持たせるための容量の見積りに使用できる、ベンチマーク・データも提供されます。認証、認可、暗号化などのセキュリティ対策は、適切な基準を使用してサポートされます。また、可用性および拡張性は、ロード・バランシング、キャッシング、Oracle9i Application Server (Oracle9iAS) および Oracle9i データベース・サーバーの透過的なスイッチオーバー・テクノロジーによりサポートされます。

2. サーバーへのアプリケーションのパブリッシュ。これは、アプリケーションのすべてのコンポーネントを **Mobile** サーバーにインストールすることです。**Oracle9i Lite** では、**Mobile** サーバーにアプリケーションをパブリッシュするために使用するパッケージ・ウィザードと呼ばれるツールが提供されます。
3. モバイル・ユーザーへのアプリケーションのプロビジョニング。このプロビジョニングには、どのデータのサブセットを使用してどのアプリケーションをどのユーザーに提供するか の決定も含まれます。**Oracle9i Lite** では、ユーザーの作成、ユーザーへのアプリケーションの実行権限の付与、ユーザーのデータ・サブセットの定義などを行うコントロール・センターと呼ばれるツールが提供されます。**Java API** を使用してプロビジョニングを行うこともできます。
4. 実際の配布環境での機能およびパフォーマンスのテスト。モバイル・アプリケーション・システムは、様々なモバイル・デバイスのクライアント・テクノロジー（オペレーティング・システム、フォーム・ファクタなど）、様々な接続オプション（LAN、無線 LAN、携帯電話、ワイヤレス・データなどのテクノロジー）、様々なサーバー構成オプションなどが関与する複合システムです。展開前に実際の環境で行うシステムのテストおよびパフォーマンスのチューニングにかかわる手段はありません。パフォーマンスの問題のほとんどの原因が、データのサブセット化問合せのパフォーマンスにあるため、このパフォーマンスのチューニングは特に注意して行う必要があります。
5. モバイル・デバイスでのアプリケーションの初期インストール方法（アプリケーションの配布）の決定。初期インストールには、**Oracle9i Lite** クライアント、アプリケーション・コードおよび初期データベースのインストールが含まれます。モバイル・デバイスにアプリケーションを初期インストールする場合に必要なデータ量は、非常に膨大になる可能性があります。そのため、モバイル・デバイスとサーバーの間に信頼性のある高速の接続を使用するか、またはオフライン・インスタンス化と呼ばれる手法を使用する必要があります。オフライン・インスタンス化の場合、モバイル・デバイスにアプリケーションをインストールするために必要なものは、すべて **CD** または **フロッピー・ディスク** に収められ、ユーザーに郵送されます。ユーザーは、このメディアを使用して、デスクトップ・マシンを使用してデバイスにアプリケーションをインストールします。**Oracle9i Lite** では、オフライン・インスタンス化用のツールが提供されます。

配布後、拡張または不具合解消のために、アプリケーションとデータ・スキーマの両方が変更されることもあります。アプリケーションの更新およびデータ・スキーマの展開は、**Oracle9i Lite** の **Mobile** サーバーが管理します。管理者は、アプリケーションとデータの再パブリッシュのみを行う必要があります。**Mobile** サーバーは、旧バージョンのアプリケーションまたはデータを持つ **Mobile** クライアントを自動的に更新します。

Oracle9i Lite のインストールでは、Mobile サーバーまたは Mobile Development Kit をインストールするオプションが提供されます。アプリケーション開発では、開発に使用するマシンに Mobile Development Kit をインストールする必要があります。ただし、このマニュアルで後述する開発例では、Mobile サーバーが実行されている必要があります。そのため、サンプルのアプリケーションをシステムで再作成する場合は、Mobile サーバーを別のマシンにインストールすることをお勧めします。Mobile サーバーをインストールする場合は、Oracle データベースのインスタンスが実行されている必要があります。これには、既存のテスト用データベースを使用してもかまいません。Mobile サーバーは、メタデータをこのデータベースに格納します。

1.2 Oracle9i Lite のアプリケーション・モデルおよびアーキテクチャ

Oracle9i Lite のアプリケーション・モデルでは、各アプリケーションは、パブリケーションを使用してデータ要件を定義します。パブリケーションは、データベース・スキーマのようなもので、1 つ以上のパブリケーション・アイテムを持ちます。パブリケーション・アイテムは、パラメータ化されたビュー定義と同様、バインド変数を指定した SQL 問合せを使用してデータのサブセットを定義します。このようなバインド変数は、サブスクリプション・パラメータまたはテンプレート 変数と呼ばれます。

サブスクリプションとは、各メンバー・パブリケーション・アイテムの各サブスクリプション・パラメータに値が指定されているパブリケーションです。サブスクリプションは、ユーザーに対してアプリケーションのプロビジョニングを行うために必要で、ユーザー用のデータのサブセットを定義します。

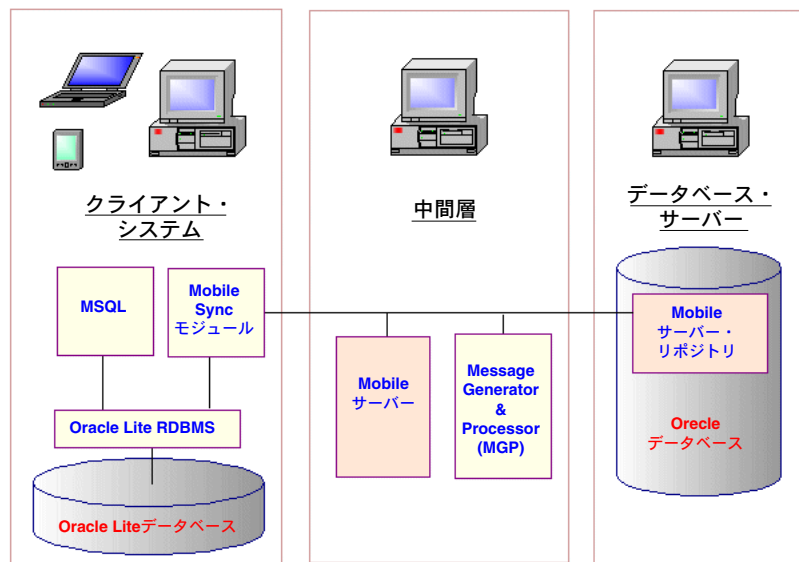
ユーザーが Mobile サーバーに初めてログインする際、Mobile サーバーはそのユーザーに対してプロビジョニングされたサブスクリプションごとに、クライアント・マシン上に Oracle Lite データベースを作成します。次に、Mobile サーバーは、サブスクリプションに含まれているパブリケーション・アイテムごとにこのデータベースにスナップショットを作成し、パブリケーション・アイテムに関連付けられた SQL 問合せ（変数をすべてバインド）を実行してサーバー・データベースから取り出したデータを、そのスナップショットに移入します。インストール後、Oracle Lite データベースは、エンド・ユーザーに対し透過的に機能し、最低限のチューニングと管理のみが必要となります。

ユーザーがアプリケーションを使用している際に Oracle9i Lite データベースに加えられた変更は、スナップショットにより取得されます。Mobile サーバーへの接続が使用可能になっている時点では、ユーザーは変更を Mobile サーバーと同期させることができます。Oracle9i Lite Mobile Sync アプリケーション（msync）を直接使用するか、またはアプリケーションから Mobile Sync API をプログラムでコールすることにより、ユーザーは同期を開始することができます。Mobile Sync アプリケーションは、Mobile サーバーと通信し、クライアント・マシンで行われた変更をアップロードします。次に、Mobile サーバーがすでに準備したクライアントについての変更をダウンロードします。

Mobile サーバーと同じ層で実行される Message Generator and Processor (MGP) と呼ばれるバックグラウンド・プロセスは、アップロードされたすべての変更を多くのモバイル・ユーザーから定期的に収集し、それをサーバー・データベースに適用します。次に、MGP は、各モバイル・ユーザーに送信する必要がある変更を準備します。次回モバイル・ユーザーが Mobile サーバーと同期をとる際に、これらの変更がクライアントにダウンロードされ、クライアント・データベースに適用されるため、この手順は非常に重要です。

図 1-1 に、Oracle9i Lite アプリケーションの配布アーキテクチャを示します。

図 1-1 Oracle9i Lite の配布アーキテクチャ



注意： 図には示されていませんが、Web-to-Go クライアントには、Lightweight HTTP Listener が追加コンポーネントとして存在します。

1.2.1 Oracle Lite RDBMS

Oracle Lite RDBMS は、ラップトップ・コンピュータ、ハンドヘルド・コンピュータ、PDA およびその他の情報機器専用に作成された、フットプリントの小さい、Java 対応の、安全なリレーショナル・データベースです。Oracle Lite RDBMS は、Windows 98/NT/2000/XP、Windows CE/Pocket PC、Palm Computing、EPOC および Embedded Linux で稼働します。Oracle Lite RDBMS では、JDBC、ODBC および OKAPI プログラミング・インタフェースと呼ばれるオブジェクト指向インタフェースが提供されているため、Java、C/C++、Visual Basic などの様々なプログラミング言語でデータベース・アプリケーションを作成できます。このようなデータベース・アプリケーションは、ユーザーがデータベース・サーバーから切断されている状態でも使用できます。

Mobile Development Kit をインストールすると、Oracle Lite RDBMS と、『Oracle9i Lite for Windows 32 開発者ガイド』の付録 C およびこのマニュアルの付録 B にリストされているすべてのユーティリティが、開発用マシンにインストールされます。本番システムでは、Oracle9i Lite アプリケーションが Mobile サーバーによりインストールされると、クライアント・マシンには、RDBMS、Mobile Sync および Mobile SQL アプリケーションのみがインストールされます。

1.2.2 Mobile SQL (MSQL)

MSQL は、ラップトップおよびハンドヘルド・コンピュータ上で Oracle Lite データベースを作成、アクセスおよび操作できるようにするインタラクティブ・ツールです。MSQL を使用すると、次のことができます。

- 表やビューなどのデータベース・オブジェクトの作成
- 表の表示
- SQL 文の実行

MSQL は、Mobile Development Kit のインストール時にインストールされます。MSQL は、アプリケーション・インストールの一部として Mobile サーバーによってもインストールされます。Windows 32 プラットフォーム用 MSQL は、Oracle SQL*Plus ツールと同様のコマンドライン・ツールです。Windows CE および Palm 用 MSQL は、Graphical User Interface (GUI) をサポートします。

Oracle Lite データベースのフォーマットは、Windows 32 および Windows CE と同様です。Windows 32 上のスナップショットは、Windows 32 MSQL コマンドラインを使用して作成およびテストできます。その後、データベースを Windows CE プラットフォームにコピーできます。デバイス上にあるデータベースは、Windows CE MSQL を使用して操作します。

1.2.3 Mobile Sync

Mobile Sync は、モバイル・デバイスに常駐するフットプリントの小さいアプリケーションです。Mobile Sync を使用すると、ハンドヘルド・コンピュータ、デスクトップ・コンピュータおよびラップトップ・コンピュータと Oracle データベース間のデータを同期させることができます。Mobile Sync は、Windows 98/NT/2000/XP、Windows CE/Pocket PC、Palm Computing、EPOC および Embedded Linux 上で稼働します。

Mobile Sync は、Oracle Lite データベース内のスナップショットを、対応する Oracle データ・サーバーのデータと同期させます。これらのスナップショットは、Mobile サーバーによって、モバイル・アプリケーションに関連付けられたパブリケーション・アイテムからユーザーごとに作成されます。Mobile サーバーは、同期プロセスの調整も行います。

Mobile Sync アプリケーションは、サポートされるプロトコルのいずれか（HTTP など）を使用して、Mobile サーバーと通信します。Mobile Sync アプリケーションは、モバイル・ユーザーからコールされると、ユーザー情報を収集してから、Mobile サーバーを使用してユーザーを認証します。次に、Oracle Lite データベースに加えられた変更をスナップショットの変更ログから収集し、Mobile サーバーにアップロードします。その後、ユーザーについての変更を Mobile サーバーからダウンロードし、Oracle Lite データベースに適用します。

この基本機能の他に、Mobile Sync アプリケーションには、送信データを暗号化、復号化および圧縮する機能もあります。

Mobile Development Kit をインストールすると、Mobile Sync アプリケーションも開発用マシンにインストールされます。また、Mobile Sync はアプリケーションのインストールの一部として、Mobile サーバーによってもクライアント・マシンにインストールされます。

スナップショットは、実表やビューとは異なり、SQL 文を使用して Oracle Lite データベース内に作成することはできません。スナップショットは、アプリケーションに関連付けられたパブリケーション・アイテムから導出されるサブスクリプションに基づき、Mobile サーバーによってのみ作成されます。この機能は、この章の後半および第 4 章で説明します。

1.2.4 Mobile サーバー

Mobile サーバーは、次の機能を提供する中間層サーバーです。

- アプリケーションのパブリッシュ
- アプリケーションのプロビジョニング
- アプリケーションのインストールおよび更新
- データ同期

Mobile サーバーには、Resource Manager および Consolidator と呼ばれる 2 つのモジュールがあります。Resource Manager は、アプリケーションのパブリッシュ、プロビジョニングおよびインストールを行います。Consolidator は、データおよびアプリケーションの同期を行います。

アプリケーションのパブリッシュとは、モバイル・ユーザーに対してアプリケーションをプロビジョニングできるように、**Mobile** サーバーにアプリケーションをアップロードすることです。アプリケーションの開発を終了すると、パッケージ・ウィザードと呼ばれる開発ツールを使用して、アプリケーションを **Mobile** サーバーにパブリッシュできます。

アプリケーションのプロビジョニングとは、ユーザーのサブスクリプションの作成およびアプリケーション実行権限のユーザーへの割当てのことです。アプリケーションのプロビジョニングには、次の方法も使用できます。

- コントロール・センターと呼ばれる管理ツールを使用すると、ユーザーおよびグループを作成できます。また、サブスクリプション・パラメータに値を割り当てることにより、ユーザーのサブスクリプションを作成できます。さらに、アプリケーションを使用する権限をユーザーまたはグループに付与することができます。
- **Resource Manager API** を使用すると、前述の作業をプログラムにより実行できます。

エンド・ユーザーは、モバイル・アプリケーションを次の 2 つの手順でインストールします。まず、モバイル・ユーザーとして、**Mobile** サーバーのセットアップ・ページを参照し、使用するプラットフォーム用のセットアップ・プログラムを選択します。セットアップ・プログラムは、**Windows 32** プラットフォームでのみ稼働します。**Windows 32** ベースのクライアント・システムの場合は、セットアップ・プログラムを直接 **Windows 32** システムにダウンロードし実行して、**Oracle9i Lite** クライアントをセットアップできます。**Windows CE** および **Palm** デバイスの場合は、先に **Windows 32** デスクトップにセットアップ・プログラムをダウンロードし、実行する必要があります。その後で、**ActiveSync** (**Windows CE** の場合) または **Hot Sync** (**Palm** の場合) を使用して、デバイスに **Oracle9i Lite** クライアントをインストールします。

次に、モバイル・デバイスで **Mobile Sync (msync)** コマンドを実行します。このコマンドを実行すると、ユーザー名とパスワードを入力するように要求されます。**Mobile Sync** アプリケーションは、**Mobile** サーバーの **Consolidator** モジュールと通信し、プロビジョニングされたアプリケーションおよびデータをダウンロードします。

アプリケーションおよびデータのインストールが終了すると、アプリケーションを使用できます。定期的に **Mobile Sync** またはカスタム・コマンドを使用して、ローカル・データベースをサーバー・データベースと同期させます。この同期により、変更したすべてのアプリケーションも更新されます。

1.2.5 Message Generator and Processor (MGP)

Mobile サーバーの **Consolidator** モジュールは、変更内容をクライアント・データベースからサーバーにアップロードし、関連するサーバーの変更内容をクライアントにダウンロードします。ただし、**Consolidator** モジュールは変更内容の調整は行いません。変更内容の調整と、変更することによって発生する競合の解決は、**MGP** が行います。**MGP** は、一定の間隔でサイクルを開始するように制御できるバックグラウンド・プロセスとして稼働します。

MGP の各サイクルは、適用と構成の 2 つのフェーズで構成されています。

適用フェーズ

適用フェーズでは、MGP は前回の適用フェーズ以降、ユーザーがアップロードした変更を収集し、サーバー・データベースに適用します。変更をアップロードした各ユーザーについて、MGP は、各サブスクリプションの変更を 1 つのトランザクションで適用します。トランザクションが失敗すると、MGP はログ・ファイルに理由を書き込み、エラー・ファイルに変更を保存します。

構成フェーズ

適用フェーズが終了すると、MGP は構成フェーズに進み、各クライアントにダウンロードする必要のある変更を準備します。

サーバー・データベースへの変更の適用

データ同期は非同期で行われるため、モバイル・ユーザーは予期しない結果を受け取る場合があります。その典型的なケースは、ユーザーがレコードを更新する際に、サーバーに接続している他のユーザーもそのレコードを更新した場合です。この場合、同期が一巡しても、ユーザーはサーバーの変更内容を受け取らない可能性があります。

この状況は、ユーザーが行う変更とサーバー・データベースに加えられる変更との間で調整がとられていないために発生します。MGP の次のサイクルで、サーバー・データベースとの間で変更が調整され、調整により発生した競合が解決されます。その後、クライアントに変更内容をダウンロードするために、新しいレコードが準備されます。ユーザーが再度同期をとると (2 回目)、サーバーの変更内容を反映するレコードがユーザーに送信されます。サーバーの変更内容とクライアントの変更内容の間に競合が発生した場合、競合解決ポリシーの定義に基づいて、サーバーの変更内容またはクライアントの変更内容のいずれかを反映したレコードが、ユーザーに送信されます。

1.2.6 Mobile サーバー・リポジトリ

Mobile サーバー・リポジトリ (リポジトリと略します) には、Mobile サーバーの実行に必要なすべての情報が含まれています。通常、この情報は、アプリケーション・データと同じデータベースに格納されます。唯一の例外は、アプリケーション・データがリモート・インスタンスに存在し、DBLink を使用するこのリモート・インスタンスに対するシノニムが、Mobile サーバーに定義されている場合です。

MSQL を使用してリポジトリを操作することはできますが、参照にのみ使用してください。リポジトリを更新する場合は、Mobile サーバーのコントロール・センターまたは Mobile サーバーの Resource Manager API を使用する必要があります。

1.3 Mobile Development Kit

Oracle9i Lite を使用してオフライン・アプリケーションを開発する前に、アプリケーション開発用マシンに Oracle9i Lite Mobile Development Kit をインストールする必要があります。Mobile Development Kit をインストールする方法は、『Oracle9i Lite for Windows NT/2000/XP インストールレーションおよび構成ガイド』を参照してください。

Oracle9i Lite Mobile Development Kit には、次のコンポーネントが含まれています。

- Oracle Lite RDBMS – 軽量の組込みリレーショナル・データベース管理システム
- パッケージ・ウィザードアプリケーションを Mobile サーバーにパブリッシュするツール
- Mobile Sync – トランザクション同期エンジン
- Windows CE プラットフォーム固有の実行可能ファイルおよびライブラリ

C または C++ 開発ツールを Mobile Development Kit (Windows CE 用) と組み合わせて使用すると、Oracle Lite データベースを対象とする Windows CE 用モバイル・アプリケーションを開発し、パッケージ・ウィザードを使用してこれらのアプリケーションを Mobile サーバーにパブリッシュできます。Mobile サーバーをインストールする方法は、『Oracle9i Lite for Windows NT/2000/XP インストールレーションおよび構成ガイド』を参照してください。

アプリケーションを Mobile サーバーにパブリッシュすると、コントロール・センターを使用してモバイル・ユーザーに対してアプリケーションのプロビジョニングを行うことができます。プロビジョニングでは、特定のユーザーを対象としたアプリケーションに必要なデータのサブセット化に使用するサブスクリプション・パラメータの値を指定する必要があります。アプリケーションのプロビジョニングがすでに終了しているユーザーは、Mobile サーバーにログインし、ユーザーのデバイス上でアプリケーションを実行するために必要なすべての設定を行うように Mobile サーバーに要求できます。

Mobile Development Kit は、<ORACLE_HOME>/Mobile/SDK ディレクトリにインストールされます。この bin ディレクトリには、次のものが含まれます。

- Oracle Lite RDBMS および Mobile SQL (msql.exe) などのコンポーネント (付録 B 「Oracle Lite ユーティリティ」を参照)。Mobile SQL は Java で作成されているため、使用する前に Java Runtime Environment (JRE) 1.3 をシステムにインストールする必要があります。JDK 1.3 がインストールされている場合、JRE はすでにマシンにインストールされています。
- Mobile Sync、実行可能ファイル (msync.exe)、COM インタフェースおよび COM インタフェース用 Java ラッパ。
- パッケージ・ウィザード (wtgpack.exe)
- ODBC データ・ソースの作成に使用する ODBC Data Source Administrator (odbcad32.exe)。

サンプルのディレクトリ <ORACLE_HOME>%Mobile%SDK%Examples には、サンプル・アプリケーションがいくつか含まれています。このマニュアルの 2.8 項で、サンプル・プログラムの内容と実行方法を説明します。ソース・コードを調べ、サンプルを実行して、Oracle9i Lite の様々な機能を理解しておく必要があります。

<ORACLE_HOME>%Mobile%SDK%OLDB40 ディレクトリには、**polite.odb** という初期データベースが含まれています。

Mobile Development Kit をインストールすると、Mobile Development Kit の bin ディレクトリを含むように環境変数 PATH がインストーラによって設定されます。Windows 32 マシンでコマンド・プロンプトを使用して、次のようなクイック・テストが行えます。

コマンド・プロンプトで次のとおり入力します。

```
msql system/manager@jdbc:polite:polite
```

```
...
```

```
SQL>create table test (c1 int, c2 int);
```

```
Table created
```

```
SQL>insert into test values(1,2)
```

```
1 row(s) created
```

```
SQL>select * from test;
```

```
C1 | C2
```

```
----+----
```

```
1 | 2
```

```
SQL>rollback;
```

```
Rollback completed
```

```
SQL>exit
```

注意： Oracle Lite database for Windows CE データベース・ファイルは、デスクトップ上の Oracle Lite データベース・ファイル (。**ODB** ファイル) と互換性があります。この 2 つの環境間でデータベース・ファイルをコピーできます。ActiveSync は、ファイルをコピーするために使用されることがあります。

また、Oracle Lite database for Windows CE データベース・ファイル (。**ODB** ファイル) はフラッシュ・メモリーに格納できるため、格納領域を拡大できます。フラッシュ・メモリーに常駐する Oracle Lite データベースへのアクセスもサポートされます。詳細は、第 3 章「同期」の 3.3.4 項「フラッシュ・メモリー・カードへのデータベースの格納」を参照してください。

1.3.1 サポート対象の Windows CE オペレーティング・システム

Oracle Lite for Windows CE は、現在、Pocket PC 2000 と呼ばれる Pocket PC 3.0 オペレーティング・システム、Pocket PC 2002、および HandheldPC 2000 オペレーティング・システム（Windows CE 2.11 に対応）とも呼ばれる HandheldPC Pro（HPC Pro）をサポートします。

1.3.2 Mobile Development Kit のディレクトリ構造

Mobile Development Kit 内の Windows CE のファイルは、次のような構造に基づいて wince ディレクトリ内に分散されます。

<Form Factor>%<Language>%<Chipset>

変数の意味は、次のとおりです。

- フォーム・ファクタ文字列名ー「Pocket PC」または「HPC Pro」。
- 言語ーサポートされている言語の 1 つ。
- チップセッ トーサポートされているチップセット・タイプの 1 つ。

1.3.2.1 サポートされている言語

Oracle Lite for Windows CE は、[表 1-1](#) にリストされている言語をサポートします。

表 1-1 サポートされている言語とサブディレクトリ記述子

言語	サブディレクトリ記述子
米語	us
日本語	ja
中国語	cn
韓国語	ko
台湾語	tw

1.3.2.2 サポートされているチップセット

Oracle Lite for Windows CE は、[表 1-2](#) にリストされているチップセットとフォーム・ファクタをサポートします。

表 1-2 サポートされているチップセットとサブディレクトリ記述子

チップセット	Pocket PC	HPC Pro	サブディレクトリ記述子
SH3	○	○	sh3
SH4	×	○	sh4
MIPS	○	○	mips
ARM	○	○	arm
X86	×	×	x86

1.4 アプリケーション開発

Windows CE デバイス用のアプリケーションは、最初にデスクトップ上で開発およびテストし、その後デバイスにダウンロードしてさらにテストするのが一般的です。

1.4.1 Oracle Lite ユーティリティ

次に示す、Oracle Lite データベースのその他のツールとユーティリティを Windows CE デバイス上で使用できます。

- CREATEDB
- REMOVEDB
- ODBINFO
- ENCRYPDB
- DECRYPDB
- Mobile SQL (MSQL)

これらのユーティリティの詳細は、[付録 B](#) を参照してください。

1.4.2 開発ツール

Mobile Development Kit for Windows CE は、次の開発ツールを使用した開発に対して認証されています。

- Visual C++ 6.0 (Windows CE 対応)
- Visual Basic 6.0
- Microsoft eMbedded Visual Tools バージョン 3.0

Microsoft eMbedded Visual Tools を使用すると、C++ と Visual Basic の両方でアプリケーションを開発できます。また、Pocket PC に固有のアプリケーションを作成するためにも、このツールが必要です。

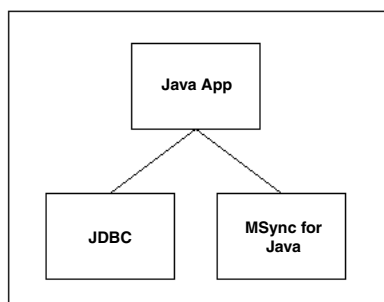
1.4.3 開発インタフェース

Oracle Lite のアプリケーションは、いくつかの異なるインタフェースで開発できます。

1.4.3.1 JDBC

Java Database Connectivity (JDBC) インタフェースは、Java アプリケーション用に、ODBC に似た SQL データベース・インタフェースを提供する Java クラス・セットを指定します。Java Development Kit (JDK) の主要部分の一部である JDBC は、リレーショナル・データベースに対するオブジェクト・インタフェースを提供します。Oracle Lite データベースは、JDBC (Type-2) ドライバを通じて JDBC をサポートします。このドライバは、[図 1-2](#) に示すとおり、JDBC コールを解釈して Oracle Lite データベースに渡します。Mobile Development Kit には JDK 1.3.1 が必要です。

図 1-2 Java 開発モデル



詳細は、<ORACLE_HOME>%Mobile%SDK%Examples%Java ディレクトリを参照してください。

1.4.3.2 ODBC

ODBC ドライバ・マネージャは、Windows CE ではサポートされていません。Visual C++ ODBC アプリケーションで Oracle Lite データベースをデータベースとして使用するには、次の手順を実行する必要があります。

- Microsoft eMbedded Visual Tools 3.0 または Microsoft Visual C++ 6.0 および Windows CE Toolkit for Visual C++ をインストールします。
- アプリケーションを作成する際、コンパイル時フラグ `-D SQL_NOUNICODEMAP` を指定します。
- `<ORACLE_HOME>%Mobile%SDK%wince%platform%Lib` ディレクトリにある `olod2040.LIB` ライブラリにアプリケーションをリンクします。（`platform` は MIPS、SH-3、SH-4、ARM、x86 です。）

1.4.3.3 オブジェクト・カーネル API (OKAPI)

OKAPI は、Oracle Lite オブジェクト・カーネルへのアプリケーション・プログラミング・インタフェース (API) です。OKAPI は、次のデータベース機能をサポートします。

- 実行時のクラスの作成とクラス情報へのアクセス
- オブジェクト識別情報とナビゲーションに基づく直接オブジェクト・アクセス
- オブジェクトのクラスタ化とグループ化
- クラスおよびそのサブクラスに対する問合せ
- オブジェクトのネーミングとオブジェクト間の関連
- バイナリ・ラージ・オブジェクト (BLOB) データ
- トランザクションおよびクラッシュ・リカバリ

詳細は、『Oracle9i Lite (C および C++) オブジェクト・カーネル API リファレンス』を参照してください。

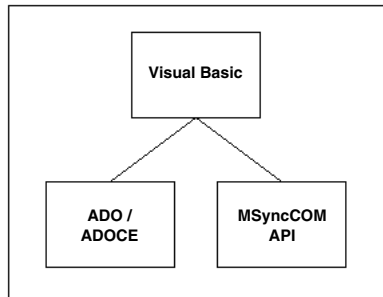
1.4.3.4 Mobile Sync API モジュール

この API を使用すると、アプリケーションは同期プロセスをプログラムによって制御できます。アプリケーションは Mobile Sync API の関数を呼び出して、同期プロセスを開始し、Mobile Sync API により生成されるエラー・メッセージを獲得します。Mobile Sync API は、COM アプリケーション、Java アプリケーションおよびネイティブの C/C++ アプリケーションをサポートします。詳細は、[3.7 項「Mobile Sync Application Program Interface \(API\)」](#)を参照してください。

1.4.3.5 Active Data Objects for Windows CE (ADOCE)

ActiveX Data Objects は、プログラミング・インタフェースです。これを使用すると、[図 1-3](#) に示すとおり、Visual Basic のような高水準言語で作成されたアプリケーションで Oracle Lite データベースの機能にアクセスできます。この開発インタフェースの詳細は、[第 4 章](#)を参照してください。

図 1-3 Active Data Objects 開発モデル



1.5 開発過程

この項では、Mobile Development Kit for Windows CE の開発過程の概要を説明します。この処理に含まれる手順は、次のとおりです。

1. 開発システムの構成
2. パブリケーション・アイテムの作成
3. Windows CE デバイスの設定
4. Windows CE デバイスとの同期
5. Windows CE 用のモバイル・アプリケーションの開発
6. デバイスでのテスト
7. アプリケーションのパッケージ化

1.5.1 開発システムの構成

Mobile Development Kit による開発を開始するには、Windows で実行する開発システムに次のコンポーネントをインストールする必要があります。

- Mobile サーバー
- Mobile Development Kit
- Windows CE 用統合開発環境 (IDE)
 - Microsoft Visual C++ 6.0 および Windows CE Toolkit for Visual C++、または
 - Microsoft eMbedded Visual Toolkit 3.0

サーバーおよび開発環境のインストールおよび構成の詳細は、ActiveSync を使用する場合は [3.3 項](#)、HTTP トランスポートを使用する場合は [3.4 項](#) を参照してください。

1.5.2 パブリケーション・アイテムの作成

次の方法を使用すると、パブリケーション・アイテムを作成できます。パブリケーション・アイテムは、同期が発生したときにモバイル・データベース上に自動的に表を作成します。パブリケーション・アイテムの作成には、次のような方法があります。

1. [宣言によるパブリケーション・アイテムの作成](#) – パッケージ・ウィザードを使用してパブリケーション・アイテムを作成します。これがお薦めの方法です。
2. [プログラムによるパブリケーション・アイテムの作成](#) – Consolidator および Resource Manager API を使用して、パブリケーション・アイテムをプログラムによって作成します。

1.5.2.1 宣言によるパブリケーション・アイテムの作成

この方法では、Mobile サーバーに含まれているパッケージ・ウィザードを利用します。アプリケーションを Mobile サーバーのリポジトリにパブリッシュするときにパブリケーション・アイテムが自動的に作成されるため、プログラミングは不要です。開発者は同期処理を実行する必要があります。この処理により、パッケージ・ウィザードに提供されるスナップショット（データ・サブセット）情報に基づいて、Mobile データベースのパブリケーション・アイテム・オブジェクト（表や索引など）が自動的に作成されます。パッケージ・ウィザードは、スナップショット定義を含むアプリケーションをパッケージ化して Mobile サーバー・リポジトリにパブリッシュするために使用します。Mobile サーバーは、スナップショット以外のアプリケーション情報が実装されていない場合でも、データの同期に必要な同期環境を作成します。

パッケージ・ウィザードの図形ツールは、開発者が Mobile データベースを作成する際に使用できる、安全でエラーが発生しにくいツールです。このツールを使用するには、実際のアプリケーション・プログラミングを開始する前に、次の手順を実行する必要があります。

- アプリケーションのパッケージ化
- Oracle データベース・サーバーでのデータベースの作成
- Mobile サーバー・リポジトリへのアプリケーションのパブリッシュ
- Mobile クライアントの設定
- アプリケーションおよびデータの配布

Mobile サーバーのアーキテクチャは、集中サーバーを使用して Mobile アプリケーションを管理および配布できるように設計されています。使用の手順については、[第 5 章](#)で説明しています。

1.5.2.2 プログラムによるパブリケーション・アイテムの作成

データベースを作成しデータを移入する 2 つ目の方法は、Consolidator API を使用してプログラムによってパブリケーション・アイテムを作成する方法です。パブリケーションには、パブリケーション・アイテム、サブスクリプションなどのデータ同期オブジェクトが含まれます。Consolidator API を起動する前に、データベース・スキーマが必要になります。分散データベース・スキーマを作成するには、次の基本手順が必要です。

- パブリケーションの作成
- パブリケーション・アイテムの作成
- ユーザー ID の作成
- サブスクリプションの作成

パブリケーションの作成

パブリケーションは、表のサブセット化定義や索引などのメタデータを含むテンプレート・グループです。パブリケーションは、Consolidator API を使用して作成できます。この API には、パブリッシュ・サブスクライブ・モデルを実装する Java 関数が含まれています。これらの関数は、Java プログラムの内部から標準のファンクション・コールとしてコールできます。

パブリケーション・アイテムの作成

パブリケーション・アイテムは、クライアントがアクセスできるデータ・サブセットを指定する SQL の SELECT 文です。パブリケーション・アイテムは、通常クライアント・デバイス上のレプリカ表に対応します。パブリケーション・アイテムは、Consolidator API を使用して作成できます。この API には、パブリッシュ・サブスクライブ・モデルを実装する Java 関数が含まれています。これらの関数は、Java プログラムの内部から標準のファンクション・コールとしてコールできます。

ユーザー ID の作成

各クライアントはユーザー ID により識別されます。開発の目的上、データ・サブスクリプションを特定ユーザーに対応付けるために、ユーザー ID は Resource Manager API を使用して作成する必要があります。

サブスクリプションの作成

サブスクリプションは、ユーザーをパブリケーションにリンクします。サブスクリプションは、Consolidator API を使用して作成できます。この API には、パブリッシュ・サブスクライプ・モデルを実装する Java 関数が含まれています。これらの関数は、Java プログラムの内部から標準のファンクション・コールとしてコールできます。Java を使用してパブリケーションおよびサブスクリプションを作成する方法は、[6.2.4 項](#)を参照してください。

重要： プログラムによるパブリケーション・アイテムの作成は、Java に関する高度なスキルが必要です。また、Consolidator と Resource Manager の両方の API を使用するため、労力を必要とする作業です。[第 5 章](#)で説明されているパッケージ・ウィザードの使用をお勧めします。これを使用すると段階ごとに順を追って作業する必要がありますが、容易に作業できます。

1.5.3 Windows CE デバイスの設定

開発過程の 3 番目の手順は、開発システムに Oracle Lite データベースを作成するためのデータをレプリケートします。同期を実行する前に、Mobile Development Kit for Windows CE デバイスのランタイム・ライブラリをインストールしておく必要があります。

Mobile サーバーと Windows CE デバイスの間のトランスポートを構成した後で、Mobile クライアントをインストールし構成する必要があります。同様に、Mobile サーバーと同期する前に、ユーザーおよび接続パラメータをデバイスに入力して保存する必要があります。

構成およびインストールの詳細は、[3.3 項](#)または [3.4 項](#)を参照してください。

1.5.4 Windows CE デバイスとの同期

開発過程の 4 番目の手順には最初の同期サイクルが含まれ、このサイクルで Oracle のデータベース・サーバーのデータを、開発システム上の Oracle Lite データベースと同期させます。サンプルの Oracle Lite データベースは、Mobile Development Kit に含まれているサンプル・ファイルをインストールしたときに Windows 開発システム上に作成されます。

サンプル・ファイルのインストールの詳細は、『Oracle9i Lite for Windows NT/2000/XP インストールेशनおよび構成ガイド』を参照してください。

1.5.5 Windows CE 用のモバイル・アプリケーションの開発

開発過程の 5 番目の手順はアプリケーションの開発です。デバイス上でサーバー・データを同期した後、Windows CE IDE を使用して Windows 開発システム (PC) 上でアプリケーションを開発します。Windows CE アプリケーションを Windows CE デバイスに配布および実行する前に、そのアプリケーションのテストを試みる必要があります。

開発ツールおよびインタフェースの詳細は、[1.4.2 項](#)および [1.4.3 項](#)を参照してください。

1.5.6 デバイスでのテスト

開発過程の 6 番目の手順では、実際の Windows CE Computing 携帯端末でアプリケーションをテストします。実際のデバイスで Windows CE アプリケーションをテストするには、開発者が手動でインストールと構成の作業を実行する必要があります。開発者は、ActiveSync を使用して、ランタイム・ライブラリとアプリケーション・ライブラリを手動で Windows CE Computing 携帯端末にインストールします。『Oracle9i Lite 管理者およびデプロイ・ガイド』には、ランタイム・ライブラリとアプリケーション・ライブラリをより簡単にインストールする方法が説明されています。

1.5.7 アプリケーションのパッケージ化

開発過程の 7 番目の手順には Windows CE Computing アプリケーションを開発およびテストした後、そのアプリケーションをパッケージ化する必要があります。パッケージ・ウィザードを使用すると、アプリケーション・ファイルおよび同期ロジックを含んだ自己完結型のパッケージを作成できます。Mobile サーバー管理者は、このパッケージ・ウィザードまたはコントロール・センターを使用して、Windows CE Computing アプリケーションを Mobile サーバー・リポジトリにパブリッシュします。

パッケージ・ウィザードの使用の詳細は、[第 5 章](#)を参照してください。

1.6 制限事項

今回のリリースには、次のような制限があります。

- 現時点では、Java のストアド・プロシージャとトリガーは、Windows CE ではサポートされていません。
- CREATEDB のような Oracle Lite ツールは、実行中にメッセージ・ボックスを表示し、出力ファイルを **ANSIOUT.TXT** というファイルに格納します。「スタート」メニューの「ファイル名を指定して実行」コマンドを使用して、引数を指定できます。ツールを終了すると、メッセージ・ボックスは自動的に消えます。「OK」ボタンをクリックすると、処理が即時に終了します。

Windows CE 用のオフライン・モバイル・アプリケーションの構築：チュートリアル

この章では、Oracle9i Lite ADOCE interface for Pocket PC を使用して Visual Basic アプリケーションを構築する方法を説明します。このインタフェースでは、Oracle9i Lite を使用して Pocket PC 用のオフラインのモバイル・アプリケーションを実装することができます。このインタフェースは、オフラインのモバイル・アプリケーションを構築、配布および管理するための完全なフレームワークを提供します。Oracle9i Lite は、ODBC、JDBC、ADOCE などの業界標準インタフェースに対応しているため、様々な Pocket PC 用のアプリケーション・モデルをサポートします。内容は次のとおりです。

- [2.1 項「概要」](#)
- [2.2 項「アプリケーションの開発」](#)
- [2.3 項「アプリケーションのパッケージ化およびパブリッシュ」](#)
- [2.4 項「アプリケーションの管理」](#)
- [2.5 項「Pocket PC でのアプリケーションの実行」](#)

2.1 概要

この章では、Pocket PC のサンプル・アプリケーションを使用して、オフラインのモバイル・アプリケーション全体の実装過程を順番に説明します。Pocket PC のサンプル・アプリケーションを使用すると、Pocket PC Windows CE アプリケーションの作成、配布および管理ができます。

Pocket PC のサンプル・アプリケーションは、運送業界や物流業界の配送担当者の標準的な作業に基づいています。このような担当者の日常業務には、梱包物の集荷と配送が含まれます。配送担当者は、発送センターを出る前に、当日納品する梱包物の完全リストおよび梱包物の配送先情報を自分の Pocket PC に取り込みます。トラック運転手も梱包物の集荷および配送に関する情報を携えるため、配送担当者は、オフラインで作業をし、梱包物の集荷および配送ステータスを自身の Pocket PC で更新できます。配送担当者は、更新した情報をワイヤレス・ネットワークを介して、発送センターで稼働している中央サーバーと同期させることができます。

2.1.1 作業の準備

このチュートリアルでは、Pocket PC のアプリケーション開発に使用するデスクトップ PC 上に Mobile サーバーがインストールされていることを前提とします。オフラインのモバイル・アプリケーションの開発過程を開始する前に、開発用コンピュータとクライアント・デバイスが、次の要件を満たすことを確認する必要があります。

2.1.1.1 アプリケーション開発用コンピュータの要件

開発用コンピュータに、次のコンポーネントを構成しインストールする必要があります。

表 2-1 に、モバイル・アプリケーション開発用コンピュータの構成およびインストール要件を示します。

表 2-1 アプリケーション開発用コンピュータの要件

要件	説明
Windows NT/2000/XP でのユーザー・ログイン	Windows NT/2000/XP コンピュータにログインするユーザーには、「管理者」権限が必要です。
Java コンポーネントがインストール済	Java Development Kit 1.3.1 以上
Oracle コンポーネントがインストール済	Oracle9i Lite Mobile サーバー (Oracle9i Lite CD-ROM) Oracle9i Lite Mobile Development Kit (Oracle9i Lite CD-ROM)
Pocket PC コンポーネントがインストール済	Microsoft ActiveSync3.5 以上 Microsoft eMbedded Visual Toolkit 3.x

2.1.1.2 クライアント・デバイスの要件

クライアント・デバイスをデスクトップに接続してから、Pocket PC 用の Oracle9i Lite クライアントを、デバイスにインストールする必要があります。デバイスに Mobile クライアントをインストールする方法の詳細は、[2.5.1 項「Pocket PC 用 Mobile クライアントのインストール」](#)を参照してください。

2.2 アプリケーションの開発

この項では、Mobile Development Kit for Pocket PC を使用して、Pocket PC Transport アプリケーションを開発しテストします。Pocket PC Transport アプリケーションは、eMbedded Visual Basic で作成します。

Pocket PC Transport アプリケーションを開発しテストするには、次の作業を実行する必要があります。

1. Oracle Lite でデータベース・オブジェクトを作成します。
2. アプリケーション・コードを記述します。
3. アプリケーションをコンパイルします。

2.2.1 Oracle Lite でのデータベース・オブジェクトの作成

Pocket PC Transport アプリケーションにより、Oracle サーバー・データベースにデータベース・オブジェクトが自動的に作成されます。配置時、Mobile サーバーにより Oracle9i Lite データベースが必須表およびデータとともにクライアント・デバイス内に作成されます。

2.2.1.1 Pocket PC Transport アプリケーションのデータベース・オブジェクト

Pocket PC Transport アプリケーションでは、次のデータベース・オブジェクトを使用します。

1. PACKAGES 表
2. ROUTES 表
3. TRUCKS 表

表 2-2 に、梱包物に関するすべての情報が格納できる梱包物の関数を示します。

表 2-2 PACKAGES 表

列	説明
DID	梱包物 ID です。
DDSC	梱包物の説明です。
DWT	梱包物の重量です。
DSTR	配送先（町）です。
DCTY	配送先（市）です。
DST	配送先（州）です。
DRTNR	ルート番号です。
DRTNM	ルート名です。
DESN	署名です。
DSTS	梱包物のステータスです。
TID	車両番号です。
PRTY	優先順位です。
PTNO	地点番号です。
TIND	「D」は配送、「P」は集荷を表します。

表 2-3 に、ルートに関するすべての情報が格納できるルートの関数を示します。

表 2-3 ROUTES 表

列	説明
ROUTE_NO	ルート番号（主キー）です。
ROUTE_NM	ルート名です。
EST_TIME	予定時刻です。

表 2-4 に、車両の可用性ステータスおよび行先に関するすべての情報が格納できる車両の関数を示します。

表 2-4 TRUCKS 表

列	説明
TRUCK_NO	車両番号（主キー）です。
TRUCK_STATUS	車両のステータスです。
ALERT_ADDRESS	警告の送り先となる携帯電話またはポケットベルの番号です（ポータル・ユーザー・インタフェース）。
DRIVER_ID	運転手の ID です。

データベース・オブジェクトの作成手順

- 1. Oracle データベース・サーバーでは、「master」スキーマが使用できます。「master」スキーマが使用可能になっていない場合は、コマンド・プロンプト・ウィンドウに次のコマンドを入力します。

```
> sqlplus system/manager@webtogo.world

SQL> create user master identified by master;

SQL> grant connect,resource to master;
```

- 2. 次のコマンドを入力して、Oracle データベース・サーバーにデータベース・オブジェクトを作成します。

```
> cd <ORACLE_HOME>%mobile%sdk%wince%samples%tutorial%Transport

> sqlplus master/master@webtogo.world @create_all.sql
```

注意： 前述のコマンドを入力してデータベース・オブジェクトを作成する場合は、「webtogo.world」と「@create_all.sql」の間に必ず空白を入れる必要があります。

2.2.2 アプリケーション・コードの記述

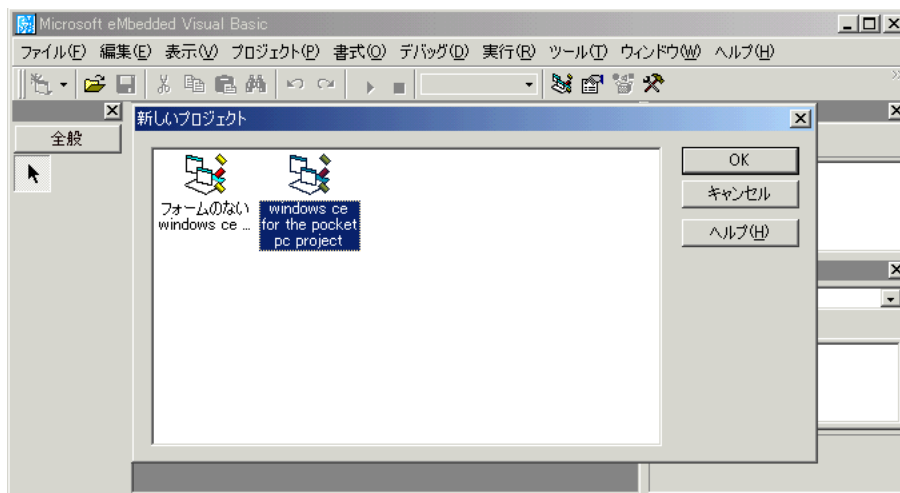
Pocket PC Transport アプリケーションの eMbedded VB コードは、サンプル・アプリケーションですぐに使用できます。次の項では、Transport アプリケーションに記述されているコードを説明します。

2.2.2.1 新しいプロジェクトの作成

eVT を使用すると、新しいアプリケーション・プロジェクトを作成できます。「ファイル」メニューの「新しいプロジェクト」を選択します。「新しいプロジェクト」ダイアログ・ボックスで「windows ce for the pocket pc project」を選択します。新しいプロジェクトがデフォルトのフォームで作成されます。コントロールをドラッグ・アンド・ドロップでき、コントロールのイベントまたはフォームにコードを記述できます。詳細は、eMbedded Visual Basic ヘルプを参照してください。

図 2-1 に、「新しいプロジェクト」ダイアログ・ボックスを示します。

図 2-1 新しいプロジェクトの作成



2.2.2.2 既存プロジェクトのオープン

eMbedded Visual Basic 統合開発環境 (IDE) を使用して、**transport.ebp** プロジェクト・ファイルを開きます。このプロジェクト・ファイルは、
<ORACLE_HOME>%mobile¥sdk¥wince¥samples¥tutorial¥Transport ディレクトリにあります。「transport.ebp」をダブルクリックすることによっても、プロジェクト、関連するユーザー・インタフェースおよびコードを eMbedded VB IDE でオープンできます。

2.2.2.3 ADOCE を使用した Oracle Lite への接続

ActiveX Data Objects は、Visual Basic アプリケーションがデータベース機能にアクセスできるようにするプログラミング・インタフェースです。これは COM インタフェース標準に基づくオブジェクト指向プログラミング・インタフェースで、データベース・エンジンにアクセスする機能を実現します。Oracle Lite ActiveX Data Objects は、eMbedded Visual Basic プログラミング環境で Oracle Lite データベースにアクセスできるようにするために、Oracle Lite から提供されるプログラミング・インタフェースです。

Oracle Lite ActiveX Data Objects for Windows CE コントロールには、次のオブジェクトがあります。

- [Connection](#)
- [Recordset](#)
- [Field](#)
- [Fields](#)

Connection

CreateObject 文を使用して、Connection オブジェクトを作成し、DSN エントリに基づく接続をアクティブにする必要があります。このオブジェクトは、デバイスの ¥ORACE ディレクトリの下にある ODBC.txt ファイルで使用できます。新規の Recordset オブジェクトを作成したが、接続がオープンしていない場合は、デフォルトの polite DSN がコールされ、Recordset がデフォルトの polite.odc データベースに自動的に接続されます。

Recordset

CreateObject 文を使用して、Recordset オブジェクトを新規作成する必要があります。レコードセットをオープンして表からレコードを返すために、SQL 文を実行します。これは、レコードセットの Open 文に表の名前を指定することにより実現します。

Field

Field オブジェクトは、直接作成できません。既存のレコードセットのコンテキストにのみ存在するためです。

Fields

Fields コレクションには、Recordset の各列の Field オブジェクトが 1 つずつ含まれます。名前または索引によって、特定のフィールドを参照できます。Fields コレクションは Count プロパティをサポートします。

次のコード例に、Fields コレクションを使用して MyTable 表についてのすべてのフィールド名を取得する方法を示します。

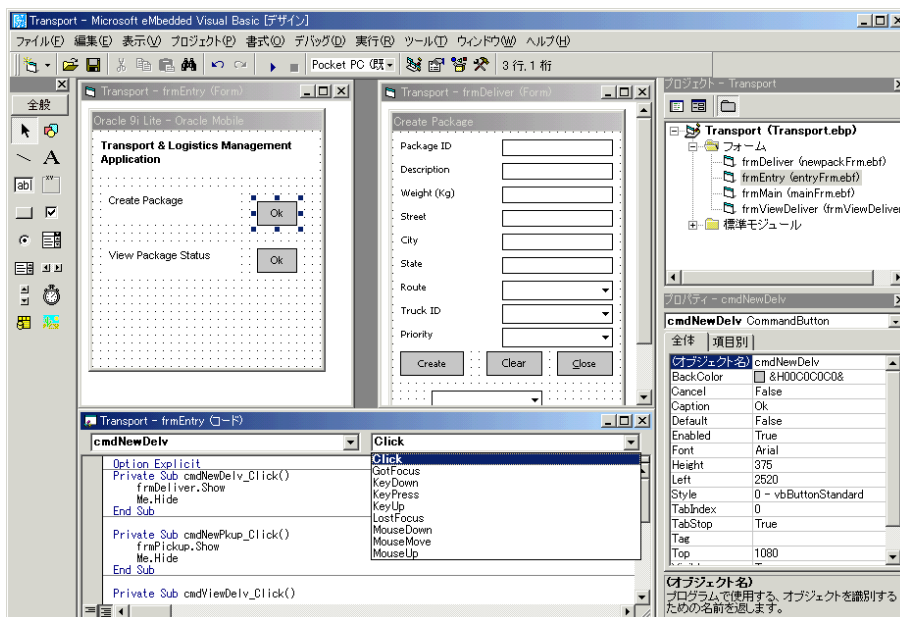
```
Dim rs, n
Set rs = CreateObject(oladoce.recordset)
```

```
rs.open "MyTable", dbconn
for n = 0 to rs.Fields.Count -1
Msgbox rs.Fields(n).Name
Next
```

2.2.2.4 コントロール用のイベント・ハンドラ・サブルーチンの記述

図 2-2 に、eMbedded Visual Basic でオープンされた Transport アプリケーション・プロジェクトを示します。

図 2-2 eMbedded Visual Basic でオープンされた Transport アプリケーション・プロジェクト



データベース接続のオープン (Main())

データベース接続をオープンするには、「dbconn」という名前の Connection オブジェクトを宣言する必要があります。このオブジェクトには、サンプルの Transport アプリケーションのプロジェクト・レベルの有効範囲が含まれています。このオブジェクトは、modTransport モジュールの「General - Declarations」セクションにナビゲートして、コード・ウィンドウの最上部で宣言する必要があります。

次の文により、「dbconn」という名前の Connection オブジェクトを宣言できます。

```
'Variable to hold the Lite Database Connection

Public dbconn
```

前述の文を宣言した後、Main サブルーチンでアプリケーションを実行し、「dbconn」を Oracle Lite ADOCE Connection オブジェクトに設定する必要があります。DSN 名「Transport」を使用して、データベースへの接続をオープンします。DSN 名は、パッケージ・ウィザードに記載され、Mobile サーバーは、この DSN 名を使用することにより、クライアント・デバイスでこの名前についてのエントリを作成することができます。詳細は、[2.3.2 項「Oracle データベース・サーバーへのアプリケーション接続の定義」](#)を参照してください。

次の文により、「dbconn」を Oracle Lite ADOCE Connection オブジェクトに設定できます。

```
Set dbconn = CreateObject("oladoce.Connection")

dbconn.connect ("Transport")

... ..
```

Add(addCmd_Click)

Transport アプリケーションの「Create Package」フォームに梱包物の情報を入力した後、「作成」ボタンをクリックします。この段階で、次の操作を実行する必要があります。

- a. オブジェクトを宣言し、それを Oracle Lite Recordset オブジェクトに設定します。
- b. PACKAGES 表をオープンして、新規レコードを追加します。adOpenKeySet および adLockOptimistic という 2 つのパラメータを、レコードセットの Open メソッドに指定する必要があります。Recordset オブジェクトのメソッドのパラメータの詳細は、[第 4 章「ActiveX Data Objects for Windows CE」](#)を参照してください。

```
rsnewpack.open "PACKAGES", dbconn, adOpenkeyset,
adLockOptimistic

... ..
```

次の文を使用して、新しい行を受け入れるようにレコードセットを準備し、レコードセットをオープンした後、addnew 文を実行できます。この文を指定した後、テキスト・フィールドの値をレコードセットの値に割り当て、Update 文を実行して Oracle Lite データベースに加えられた変更を更新できます。

```
rsnewpack.addnew

rsnewpack.fields("DID").Value = packidTxt.Text

rsnewpack.fields("DDSC").Value = descptxt.Text

rsnewpack.fields("DWT").Value = wtTxt.Text
```

```
rsnewpack.fields("DSTR").Value = delstreetTxt.Text

rsnewpack.fields("DCTY").Value = delCityTxt.Text

rsnewpack.fields("DST").Value = delstateTxt.Text

rsnewpack.fields("DSGN").Value = ""

rsnewpack.fields("DSTS").Value = "NEW"

If priorityCmb.ListIndex > 0 Then

    rsnewpack.fields("PRTY").Value =
priorityCmb.List (priorityCmb.ListIndex)

Else

    rsnewpack.fields("PRTY").Value = "NORMAL"

End If
```

次のフィールドは、オフライン・アプリケーションでサンプル・データとしてのみ使用しますが、同期プロセスでも使用します。

```
rsnewpack.fields("PTNO").Value = "-1"

rsnewpack.fields("TIND").Value = "P"
.....
```

- c. 前述の操作を完了し、次の文を使用してレコードセットをクローズします。

```
rsnewpack.close
```

Update(cmdUpdate_Click)

Transport アプリケーションで「Update Status」ボタンをクリックした場合は、レコードセットをオープンする必要があります。このレコードセットは、メイン画面の下の「View Package Status」セクションに表示されます。レコードを更新するには、次の文を指定して、更新が必要なレコードを問い合わせる必要があります。

```
rsnewpack.open "SELECT * FROM PACKAGES WHERE DID = '" &
txtpackID.Text & "'", dbconn, adOpenkeyset, adLockOptimistic
```


問合せを行い、レコードが見つかった後で、テキスト・フィールドの値をレコードセット・フィールドに割り当てる必要があります（テキスト・フィールドの値が空ではない場合）。ここで、次の **Update** 文を指定することにより、レコードを更新できます。

```
If rsnewpack.fields("DSTS").Value = "NEW" Then

    Me.lblStatus.Caption = "COMPLETED"

    rsnewpack.fields("DSTS").Value = "COMPLETED"

Else

    Me.lblStatus.Caption = "NEW"

    rsnewpack.fields("DSTS").Value = "NEW"

End If

rsnewpack.UPDATE
```

Update 文を指定した後、close 文を指定してレコードセットをクローズし、このタスクに割り当てられたシステムのメモリーを解放することができます。レコードセットをクローズするには、Recordset を Nothing に設定する必要があります。

Query(cmdGetPack_Click)

レコードセットの問合せを行うには、梱包物 ID の値として DID が指定されているレコードセットをオープンする必要があります。次の文を使用すると、レコードセットの問合せができます。

```
txtPackID.Enabled = False

cmdGetPack.Enabled = False

rsnewpack.open "SELECT * FROM PACKAGES WHERE DID = '" & txtPackID.Text & "'",
dbconn, adOpenForwardOnly, adLockOptimistic

updateFields

rsnewpack.close

rsnewpack.open "SELECT * FROM PACKAGES ORDER BY DID", dbconn, adOpenkeyset,
adLockOptimistic

Me.prevCmd.Enabled = True

Me.nxtCmd.Enabled = True
```

Navigate(prevCmd_Click and nxtCmd_Click)

次の2つのプロシージャにより、結果セット全体をナビゲートすることができます。Query プロシージャで使用されるレコードセットと同じモジュール・レベルのレコードセットを使用できます。レコードセットの始め (BOF) より前またはレコードセットの終わり (EOF) より後にレコードセットを移動しようとする、エラーが発生します。したがって、これらのプロシージャで `rsnewpack.BOF` および `rsnewpack.EOF` ファイルをチェックすることにより、この2つのイベントを取得できます。

1. prevCmd_Click()

```
.....  
    If Not rsnewpack.bof Then  
  
        rsnewpack.moveprevious  
  
        updateFields  
  
    Else  
  
        rsnewpack.movefirst  
  
        prevCmd.Enabled = False  
  
    End If  
  
.....
```

2. nxtCmd_Click()

```
prevCmd.Enabled = True  
  
    If Not rsnewpack.EOF Then  
  
        rsnewpack.movenext  
  
        updateFields  
  
    Else  
  
        rsnewpack.movelast  
  
        nxtCmd.Enabled = False  
  
    End If
```

注意： グローバルなモジュール・レベルのレコードセット (rs) は、PACKAGES 表に対して変更動作（追加、更新、削除など）が行われるたびに、自動的にクローズされ、オープンされます。この動作により、レコードセットが更新され、Oracle Lite の PACKAGES 表内の実際の値と同期がとられた状態が保持されます。変更が行われるたびに、グローバルなモジュール・レベルのレコードが自動的にオープンおよびクローズしない場合、ナビゲートまたは問合せを行うと、「カーソル状態が無効です」というエラー・メッセージが表示されます。

アプリケーション・コードの参照

アプリケーション・コードおよびユーザー・インタフェースは、次の場所で参照できます。

<ORACLE_HOME>%mobile%sdk%wce%samples\tutorial\Transport

2.2.3 アプリケーションのコンパイル

アプリケーションをデバイスにインストールするには、CAB ファイルを作成する必要があります。CAB ファイルは、アプリケーションのパブリッシュ・フェーズで Mobile サーバー・リポジトリにアップロードされます。CAB ファイルは、eMbedded Visual Basic IDE を使用して作成できます。

CAB ファイルを作成する前に、eMbedded Visual Basic IDE から実行可能ファイル .vb をコンパイルして作成する必要があります。

2.2.3.1 実行可能ファイル .vb の作成

.vb ファイルを作成するには、「ファイル」メニューから「... の作成」オプションを選択します。この段階で、実行可能ファイル .vb を保存するためのファイル名を指定する必要があります。 .vb ファイルは、次のディレクトリに保存できます。

<ORACLE_HOME>%mobile%sdk%wce%samples\tutorial\Transport\Transport.vb

2.2.3.2 CAB ファイルの作成

CAB ファイルを作成するには、eMbedded Visual Basic IDE で「ツール」メニューの下「リモートツール」オプションから「アプリケーション インストール ウィザード」を選択します。

1. アプリケーションのプロジェクト・ファイル .ebp を開き、
<ORACLE_HOME>%Mobile%Sdk%wce%samples\tutorial\Transport\Transport.ebp を値として入力します。
2. 前述の項で作成し保存した、.vb ファイルのディレクトリ名を入力します。

3. CAB ファイルを格納するためのディレクトリ名を入力します。たとえば、C:\Transportinstall のように入力します。
4. CAB ファイルの作成対象となる必須プロセッサ（SH3、SH4、ARM）を選択します。
5. アプリケーション インストール ウィザードにデフォルト値が表示されます。「次へ」をクリックします。
6. アプリケーション インストール ウィザードの「会社名」フィールドを除くすべてのフィールドの値として「Transport」を入力します。「会社名」フィールドに「Oracle」と入力します。
7. 「インストール プログラムの作成」をクリックします。

アプリケーション インストール ウィザードにより、選択されたプロセッサの CAB ファイルが作成され、C:\Transportinstall\CD1 ディレクトリに保存されます。

2.2.3.3 CAB ファイルからのアプリケーションのインストール

アプリケーションをパッケージ化しパブリッシュした後は、デバイスにアプリケーションをダウンロードしてインストールできます。次の項では、アプリケーションをパッケージ化し、パブリッシュする方法を説明します。

2.3 アプリケーションのパッケージ化およびパブリッシュ

この項では、アプリケーションをパッケージ化し、Mobile サーバーにパブリッシュできるように準備する方法を説明します。アプリケーションをパッケージ化しパブリッシュするには、次の作業を実行する必要があります。

1. パッケージ・ウィザードを使用してアプリケーションを定義します。
2. Oracle データベース・サーバーへのアプリケーション接続を定義します。
3. スナップショットを定義します。

2.3.1 パッケージ・ウィザードを使用したアプリケーションの定義

パッケージ・ウィザードを使用すると、Transport アプリケーションを選択して定義することができます。

2.3.1.1 新規アプリケーションの作成

パッケージ・ウィザードを使用すると、Pocket PC アプリケーションを作成または変更し、そのアプリケーションを Mobile サーバーにパブリッシュできます。パッケージ・ウィザードの使用の詳細は、[第 5 章](#)を参照してください。

ここでは、Pocket PC アプリケーションを作成または変更する方法を説明します。パッケージ・ウィザードを通常モードで起動すると、Pocket PC Transport アプリケーションを選択して定義することができます。

パッケージ・ウィザードを通常モードで起動するには、次の手順を実行します。

1. コマンド・プロンプト・ウィンドウを開き、次のとおり入力します。

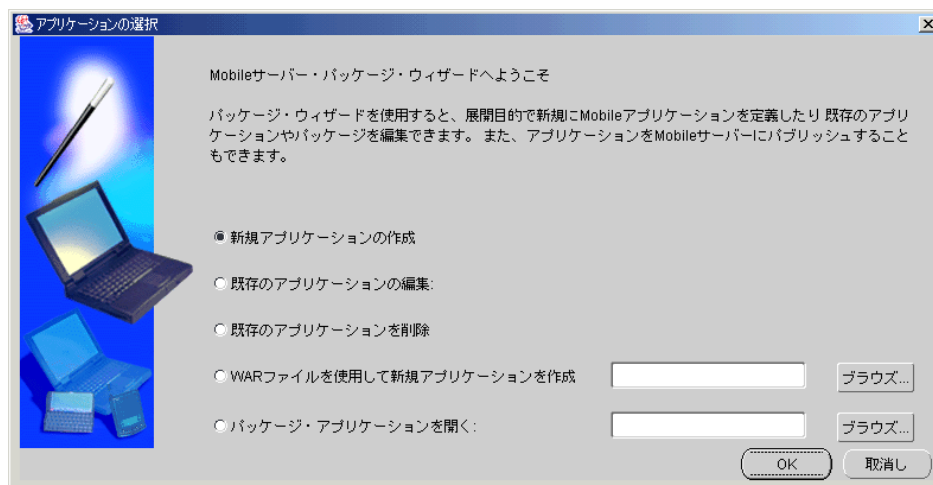
```
> cd <ORACLE_HOME>%mobile%sdk%bin
```

```
> wtgpack
```

パッケージ・ウィザードの「**アプリケーションの選択**」ダイアログ・ボックスに「ようこそ」パネルが表示されます。

2. 「**新規アプリケーションの作成**」を選択します。

図 2-3 アプリケーションの選択



3. 「プラットフォームの選択」ダイアログ・ボックスが表示されます。このダイアログ・ボックスで、アプリケーションのプラットフォームを指定できます。

- a. 「使用可能プラットフォーム」のリストに、対応するプロセッサのタイプと言語とともにデバイス・タイプのリストが表示されます。「Handheld PC」または「Pocket PC」と指定の言語（英語、中国語、日本語、韓国語など）を選択します。適切なプロセッサのタイプ（SH3、SH4、ARM、MIPS など）を選択し、下矢印ボタンをクリックすると、選択したプラットフォームが「選択済プラットフォーム」リストに表示されます。ターゲット・デバイスによっては、複数のプラットフォームを選択できます。「**次へ**」をクリックします。「**アプリケーション**」パネルが表示されます。

注意：「使用可能プラットフォーム」ダイアログ・ボックスでは、「Win32 ネイティブ」や「Palm」など、他のサポートされるプラットフォームをアプリケーションに指定できます。この章では、「Handheld PC」または「Pocket PC」をプラットフォームとして選択しています。

図 2-4 に、Pocket PC アプリケーションで使用できるすべてのプラットフォームを示します。

図 2-4 プラットフォームの選択



4. 「アプリケーション」パネルに、Pocket PC Transport アプリケーションの設定を入力します。次の表に示されているとおりにデータを入力します。

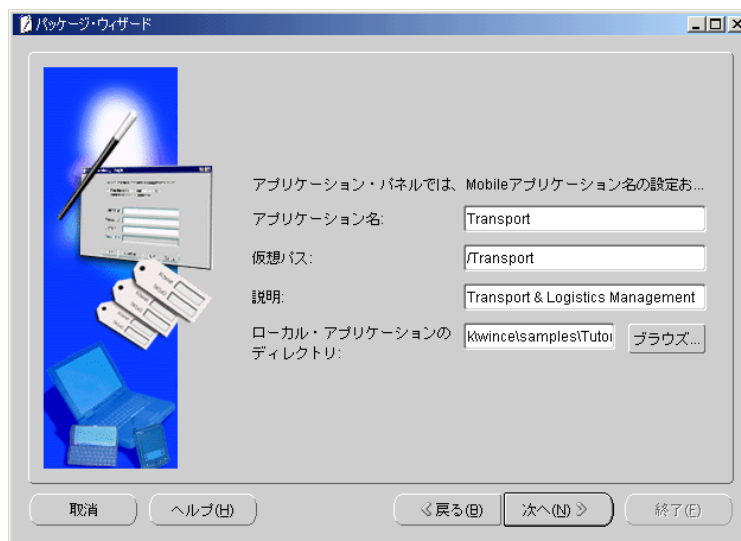
表 2-5 に、「アプリケーション」パネルの関連するフィールドに指定する必要があるアプリケーションの設定値を示します。

表 2-5 Pocket PC Transport アプリケーションの設定

フィールド	値
アプリケーション名	Transport
仮想パス	/Transport
説明	Transport and Logistics Management
ローカル・アプリケーションのディレクトリ	<ORACLE_HOME>\¥Mobile¥Sdk¥wince¥samples¥tutorial¥transport

図 2-5 に、「アプリケーション」パネルを示します。

図 2-5 「アプリケーション」パネル



5. 「次へ」をクリックします。「ファイル」パネルが表示されます。

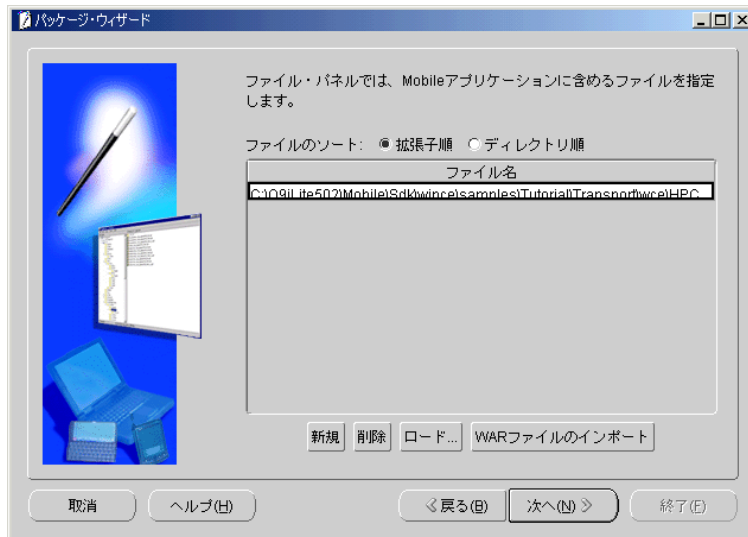
「ファイル」パネルでは、前述のパネルに指定された「ローカル・アプリケーションのディレクトリ」に基づいてダウンロードする必要があるすべてのファイルが自動的に検索されます。正しい CAB ファイルを、対応するディレクトリに確実にコピーする必要があります。

たとえば、Transport.Arm 1100 (4K) v3.00.cab ファイルを Arm ディレクトリにコピーします。このディレクトリのフルパスは、次のとおりです。

```
<ORACLE_HOME>%Mobile%Sdk%wince%samples\tutorial\Transport%wce%\Pocket_PC%us%\arm
```

図 2-6 に、「ファイル」パネルを示します。

図 2-6 アップロードするファイルの選択

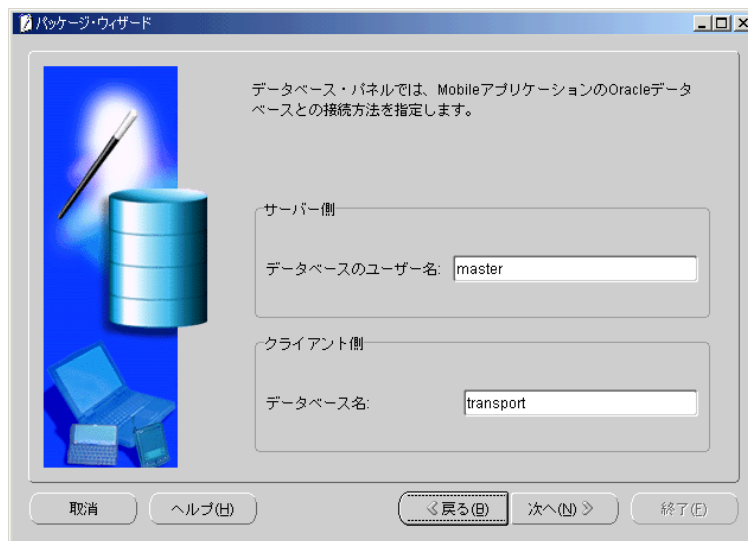


2.3.2 Oracle データベース・サーバーへのアプリケーション接続の定義

パッケージ・ウィザードの「データベース」パネルを使用すると、Oracle データベース・サーバーに対して Transport アプリケーション接続情報を定義できます。

図 2-7 に、「データベース」パネルを示します。

図 2-7 「データベース」 パネル



「サーバー側」の「データベースのユーザー名」フィールドは、Transport アプリケーションについてのサーバー側のデータベース・オブジェクトが含まれているユーザー・スキーマを指します。「クライアント側」の「データベース名」フィールドは、Oracle Lite データベース・ファイルのデータ・ソース名 (DSN) を指します。この名前はデバイス上で自動的に作成されます。

Transport アプリケーションの Oracle データベース・サーバーへ接続情報を定義するには、次の手順を実行します。

1. 次の表に示されているとおりに値を入力します。

表 2-6 に、「データベース」パネルに入力する必要がある値を示します。

表 2-6 「データベース」パネル・ウインドウの説明

フィールド	値
サーバー側のデータベースのユーザー名	master
クライアント側のデータベース名	transport

2. 「次へ」をクリックします。「スナップショット」パネルが表示されます。

2.3.3 スナップショットの定義

「スナップショット」パネルは、同期を必要とするデータベース表を定義します。このパネルでは、Transport アプリケーションの同期ロジックを定義できます。パッケージ・ウィザードを使用すると、Oracle サーバー・データベースから表定義をインポートすることもできます。

Transport アプリケーションのスナップショットを定義するには、次の手順を実行します。

- 1. 「スナップショット」パネルで、「インポート」をクリックして Oracle サーバー・データベースから表定義をインポートします。「データベースへの接続」ウィンドウが表示されます。次の表に示されているとおりに値を入力します。

表 2-7 に、「データベースへの接続」パネルに入力する必要がある値を示します。

表 2-7 「データベースへの接続」ウィンドウの説明

フィールド	説明	値
ユーザー名	データベース・オブジェクトを持つスキーマ名（データベースのユーザー名）	master
パスワード	スキーマの所有者のパスワード	master
データベースの URL	jdbc:oracle:oci8:@<Oracle データベース・サーバーの TNSNAME>	jdbc:oracle:oci8:@webtogo.world

図 2-8 に、「データベースへの接続」ウィンドウを示します。

図 2-8 「データベースへの接続」ウィンドウ



- 2. 「OK」をクリックします。利用可能な表のリストを示す「表」ウィンドウが表示されます。

図 2-9 に、「表」パネルを示します。

図 2-9 表のインポート



3. 「PACKAGES」、「TRUCKS」、「ROUTES」のいずれかの表を選択します。「追加」をクリックしてから「閉じる」をクリックします。

図 2-10 に、データベース表を含む「スナップショット」パネルを示します。

図 2-10 WinCE の「スナップショット」パネル



- 4. 「PACKAGES」を選択し、「編集」をクリックします。「スナップショットの編集」ダイアログ・ボックスが表示されます。
- 5. クライアント上で行われるスナップショットのリフレッシュの順序を制御するには、「PACKAGES」表の比率を「1」に変更する必要があります。「SQL の生成」チェックボックスの選択を解除します。これは、Oracle データベース・サーバー内にデータベース・オブジェクトをすでに作成したため、データベースを作成するための SQL を作成する必要がないからです。

図 2-11 に、PACKAGES 表の「スナップショットの編集」パネルを示します。

図 2-11 PACKAGES 表の「スナップショットの編集」パネル



6. 「WinCE」タブをクリックします。「クライアントで作成」チェックボックスが選択されていること、および「テンプレート」フィールドに次の SQL 文が表示されていることを確認する必要があります。

```
SELECT * FROM MASTER.PACKAGES
```

注意：「クライアントで作成」チェックボックスが選択されていることを確認してください。「クライアントで生成」チェックボックスが選択されていない場合、クライアント側の Oracle Lite データベース内に対応するスナップショットは作成されません。

7. 「OK」をクリックします。
8. 「スナップショット」パネルで、「ROUTES」表を選択し「編集」をクリックします。「スナップショットの編集」ダイアログ・ボックスが表示されます。
9. クライアント上で行われるスナップショットのリフレッシュの順序を制御するには、「ROUTES」表の比率を「2」に変更する必要があります。「SQL の生成」チェックボックスの選択を解除します。これは、Oracle データベース・サーバー内にデータベース・オブジェクトをすでに作成したため、データベースを作成するための SQL を作成する必要がないからです。

注意： クライアント上のスナップショットを更新するには、「更新可能」チェックボックスが選択されていることを確認してください。「更新可能」チェックボックスが選択されていない場合、データ同期は常に Oracle データベースから単方向で行われるため、デバイスで行われたすべての変更が失われます。

10. 「WinCE」タブをクリックします。「クライアントで作成」チェックボックスが選択されていること、および「テンプレート」フィールドに次の SQL 文が表示されていることを確認します。

```
SELECT * FROM MASTER.ROUTES
```

図 2-12 に、ROUTES 表の「スナップショットの編集」パネルを示します。

図 2-12 ROUTES 表の「スナップショットの編集」パネル



注意： デスクトップにデータベース・オブジェクトが存在しない場合でも、「スナップショットの編集」パネルの「新規」ボタンを使用して、データベース・オブジェクトを作成することができます。

11. TRUCKS 表について、手順 8 ～ 10 を繰り返します。比率は「3」を使用します。

2.3.4 アプリケーションのパブリッシュ

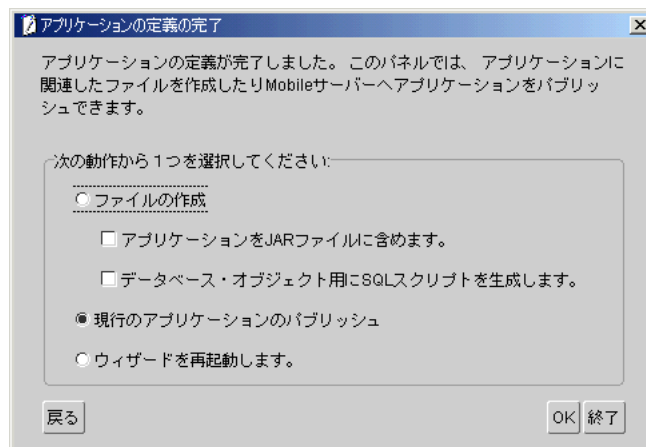
「アプリケーションの定義の完了」ダイアログ・ボックスを使用すると、Pocket PC Transport アプリケーションをパッケージ化しパブリッシュできます。

Transport アプリケーションをパブリッシュするには、次の手順を実行します。

1. 「**現行のアプリケーションのパブリッシュ**」オプションを選択します。

図 2-13 に、「アプリケーションの定義の完了」ダイアログ・ボックスを示します。

図 2-13 「アプリケーションの定義の完了」 ダイアログ・ボックス



2. 「OK」をクリックします。
3. 「アプリケーションのパブリッシュ」 ダイアログ・ボックスが表示されます。
図 2-14 に、「アプリケーションのパブリッシュ」 ダイアログ・ボックスを示します。

図 2-14 「アプリケーションのパブリッシュ」 ダイアログ・ボックス



4. 「アプリケーションのパブリッシュ」ウィンドウに、次の表に示されているとおりに値を入力します。

表 2-8 に、「アプリケーションのパブリッシュ」ウィンドウに入力する必要がある値を示します。

表 2-8 「アプリケーションのパブリッシュ」ウィンドウの説明

フィールド	説明	値
Mobile サーバーの URL	Mobile サーバーが稼働しているマシンの URL または IP アドレスです。	<Mobile サーバー>
Mobile サーバーの ユーザー名	管理者権限を持つ Mobile サーバー・ユーザーの ユーザー名です。	administrator
Mobile サーバーの パスワード	管理者権限を持つ Mobile サーバー・ユーザーの パスワードです。	admin
リポジトリのディレクトリ	このアプリケーションのすべてのファイルが格納される Mobile サーバー・リポジトリ内のディレクトリの名前です。	/Transport
アプリケーションをパブリックにする。	すべてのユーザーがこのアプリケーションを使用できるようにする場合を除き、このチェックボックスは選択しないでください。	選択解除

5. Mobile サーバー・リポジトリにアプリケーションをパブリッシュするには「OK」をクリックします。アプリケーションのパブリッシュ・ステータスを示すダイアログ・ボックスが表示されます。アプリケーションがパブリッシュされるまで待機します。

これで、アプリケーションのパッケージ化またはパブリッシュに必要なすべての開発作業が完了しました。次は、コントロール・センターでアプリケーションの開発を担当しているユーザーに対して、Transport アプリケーションの割当てとプロビジョニングを行います。

2.4 アプリケーションの管理

この項では、Mobile サーバーにパブリッシュしたモバイル・アプリケーションを管理する方法を説明します。アプリケーションを管理するには、次の作業を実行する必要があります。

1. Mobile サーバーを起動します。
2. コントロール・センターを起動します。
3. 新規ユーザーを作成します。
4. アプリケーションのプロパティを設定します。

- 5. ユーザーにアプリケーションへのアクセス権を付与します。
 - 6. MGP を起動します。
- コントロール・センターで実施する作業の詳細は、『Oracle9i Lite 管理者およびデプロイ・ガイド』を参照してください。

2.4.1 Mobile サーバーの起動

Mobile サーバーをスタンドアロン・モードで起動するには、コマンド・プロンプト・ウィンドウに次のコマンドを入力します。

```
> webtogo -d0 mobileadmin/manager@webtogo.world
```

「MobileAdmin」スキーマに指定したパスワードを使用する必要があります。サンプルの Transport アプリケーションの場合、パスワードは「manager」です。

2.4.2 Mobile サーバーのコントロール・センターの起動

ログイン・ユーザー名およびパスワードを使用すると、Mobile サーバーにログインし、Mobile サーバーのコントロール・センターを起動できます。

Mobile サーバーのコントロール・センターを起動するには、次の手順を実行します。

- 1. Web ブラウザを開き、次の URL を入力して、Mobile サーバーに接続します。

http://<mobileserver>/webtogo

注意： mobileserver 変数を、Mobile サーバーのホスト名に置き換える必要があります。

- 2. 次の表に示されているとおりにアカウント情報を入力して、Mobile サーバーの管理者としてログインします。
- 表 2-9 に、Mobile サーバーのコントロール・センターの「ログイン」ウィンドウに入力する必要がある値を示します。

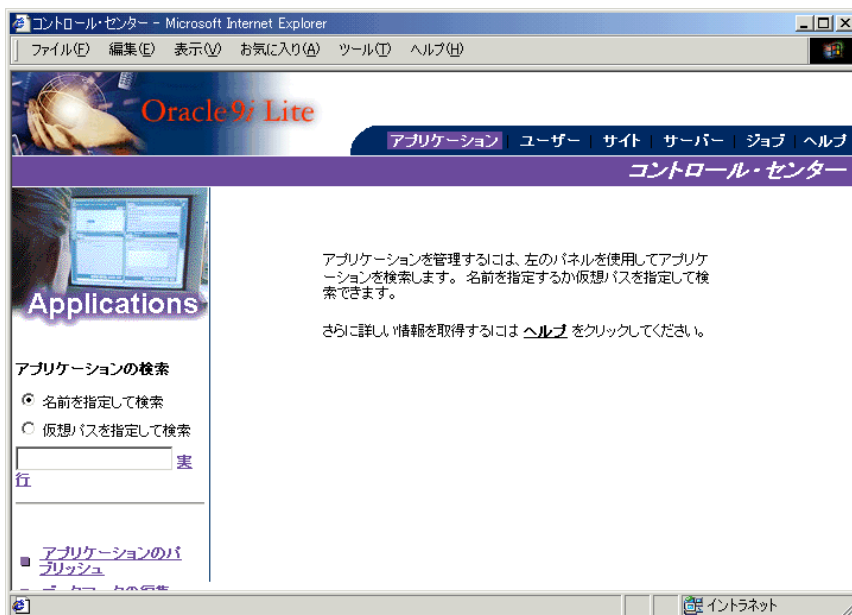
表 2-9 Mobile サーバーのコントロール・センターの「ログイン」ウィンドウの説明

フィールド	値
ユーザー名	administrator
パスワード	admin

3. 「コントロール・センター」をクリックしてコントロール・センターを起動します。新しいブラウザ・ウィンドウに、Mobile サーバーのコントロール・センターの「アプリケーション」ページが表示されます。Mobile サーバーのコントロール・センターで、アプリケーションを名前または仮想パスを指定して検索できます。

図 2-15 に、Mobile サーバーのコントロール・センターの「アプリケーション」ページを示します。

図 2-15 「アプリケーション」ウィンドウ



2.4.3 新規ユーザーの作成

Mobile サーバーの新規ユーザーを作成するには、次の手順を実行します。

1. コントロール・センターで「ユーザー」タブをクリックします。左側のフレームに「ユーザー」メニューが表示されます。
2. 「ユーザーの作成」をクリックします。右側のフレームに「ユーザー・プロパティ」ウィンドウが表示されます。
3. 次の表に示されているとおりにデータを入力します。

4. 「保存」をクリックします。

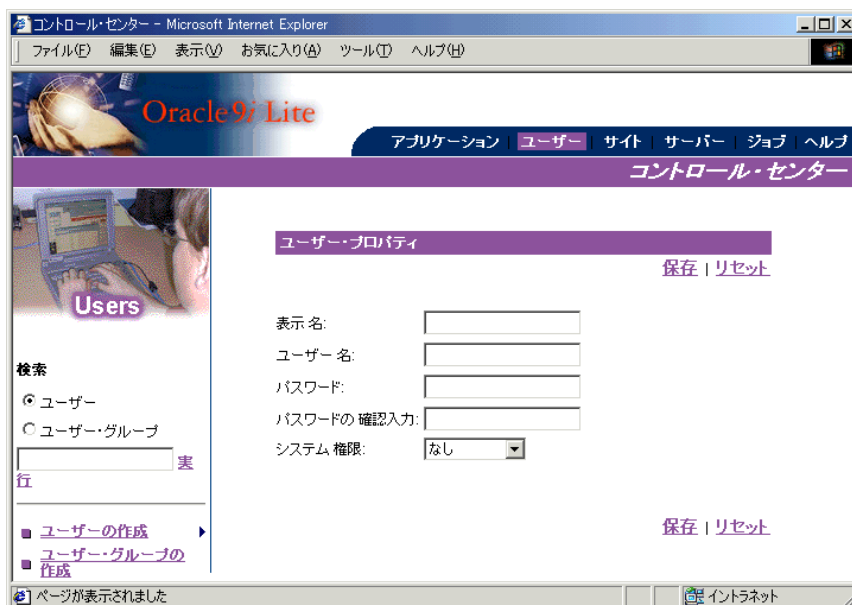
表 2-10 に、「ユーザー・プロパティ」ウィンドウに入力する必要がある値を示します。

表 2-10 「ユーザー・プロパティ」ウィンドウの説明

フィールド	値
表示名	bob
ユーザー名	bob
パスワード	bobhope
パスワードの確認入力	確認のため前述のパスワードを再入力します。
システム権限	「ユーザー」オプションを選択します。

図 2-16 に、コントロール・センターの「ユーザー・プロパティ」ウィンドウを示します。

図 2-16 「ユーザー・プロパティ」ウィンドウ



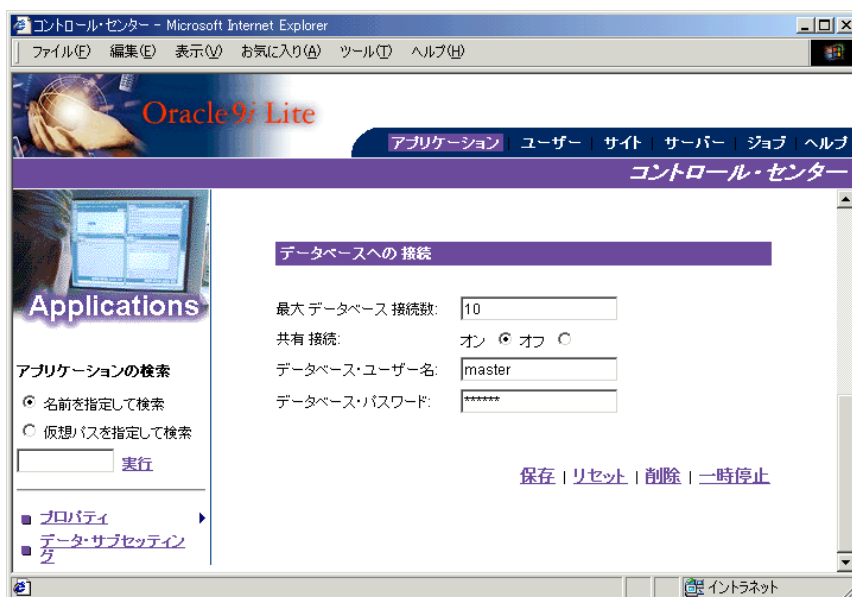
2.4.4 アプリケーションのプロパティの設定

Pocket PC Transport アプリケーションのプロパティを設定するには、次の手順を実行します。

1. コントロール・センターで「**アプリケーション**」タブをクリックします。左側のフレームに「**アプリケーション**」メニューが表示されます。
2. 「**実行**」をクリックします。現在実行可能なアプリケーションのリストが、コントロール・センターに表示されます。「**Transport and Logistics Management**」をクリックします。
3. 左側のフレームの「**プロパティ**」をクリックします。右側のフレームに Transport アプリケーションのプロパティが表示されます。
4. 「**データベース・パスワード**」フィールドに、「master」と入力します。これは、Oracle サーバー・データベースのユーザー・スキーマ「master」のデフォルト・パスワードです。
5. 「**保存**」をクリックします。

図 2-17 に、コントロール・センターの「アプリケーション」ウィンドウの「データベースへの接続」セクションを示します。

図 2-17 アプリケーションのプロパティの設定



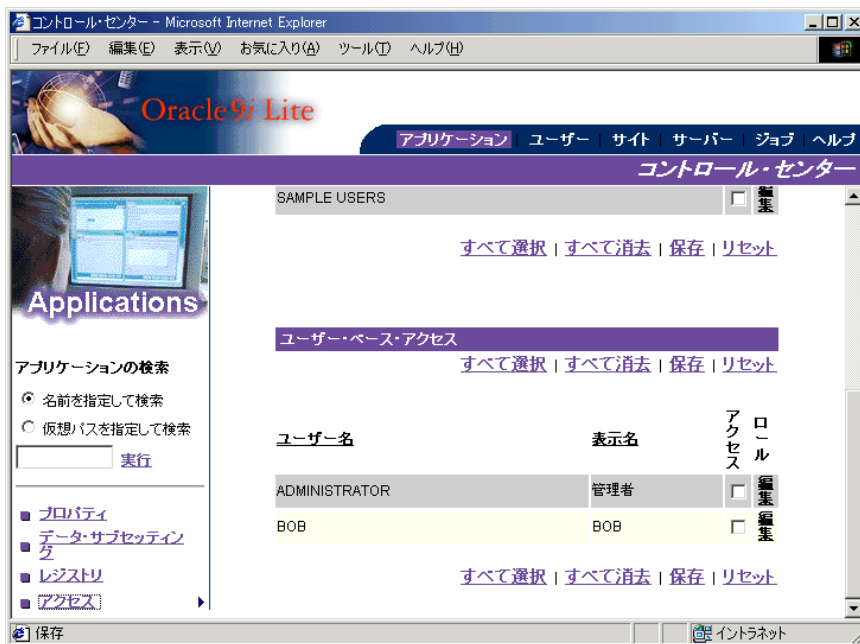
2.4.5 ユーザーに対するアプリケーションへのアクセス権の付与

アプリケーションへのアクセス権をユーザーに付与するには、次の手順を実行します。

1. コントロール・センターの左側のフレームにある「アクセス」をクリックします。コントロール・センターに、リストが2つ表示されます。1つは、アプリケーション・ユーザーのリスト、もう1つはアプリケーション・グループのリストです。Transport アプリケーションへのアクセス権をユーザーまたはユーザーのグループに付与するには、「アクセス」チェックボックスを選択します。
2. BOB というユーザーにアクセス権を付与するには、「ユーザー・ベース・アクセス」リストから「BOB」というユーザー名を検索し、「アクセス」チェックボックスを選択します。
3. 「保存」をクリックします。ユーザー BOB に、Transport アプリケーションへのアクセス権が付与されます。

図 2-18 に、コントロール・センターの「アプリケーション」ウィンドウの「ユーザー・ベース・アクセス」セクションを示します。

図 2-18 アプリケーションへのアクセス権の付与



2.4.6 Message Generator and Processor (MGP) の起動

Oracle9i Lite の非同期レプリケーション・モデルの場合、クライアントはサーバーがペイロードを準備するまで待機することはありません。ペイロードには、同期がとられるデータが含まれています。Mobile サーバーは、常時バックグラウンドで MGP プロセスを実行して、すべての Mobile クライアントに対するペイロードを非同期で準備します。そのため、モバイル・ユーザーが同期プロセスを開始すると、Mobile サーバーはクライアントのペイロードをインキューにアップロードし、対応するアウトキューからクライアントのペイロードを取り込みます。MGP は、インキューおよびアウトキュー内のペイロードを処理し、バックグラウンドで Mobile サーバーによるデータベース操作を実行します。

MGP を起動するには、次の手順を実行します。

1. コントロール・センターで「サーバー」タブをクリックします。
2. 「サーバー」パネルの左側のフレームにある「MGP コントロール」をクリックします。
3. 「開始」ボタンをクリックします。右側のフレームの「現在のモード」に「MGP を起動中」と表示され、「開始」ボタンは「ステータス」ボタンに置き換えられます。

図 2-19 に、「MGP コントロール」ウィンドウを示します。

図 2-19 「MGP コントロール」ウィンドウ



2.5 Pocket PC でのアプリケーションの実行

この項では、作成、テスト、配布、管理が完了したアプリケーションを実行する方法を説明します。アプリケーションを実行するには、次の作業を実行する必要があります。

1. Pocket PC 用 Mobile クライアントをインストールします。
2. Transport アプリケーションをインストールし、同期をとります。

2.5.1 Pocket PC 用 Mobile クライアントのインストール

ブラウザと ActiveSync Application Manager を使用すると、Pocket PC 用 Mobile クライアントをインストールできます。

Pocket PC 用 Mobile クライアントをインストールするには、次の手順を実行します。

1. デスクトップ・ブラウザを開き、次の URL を入力して、Mobile サーバーに接続します。

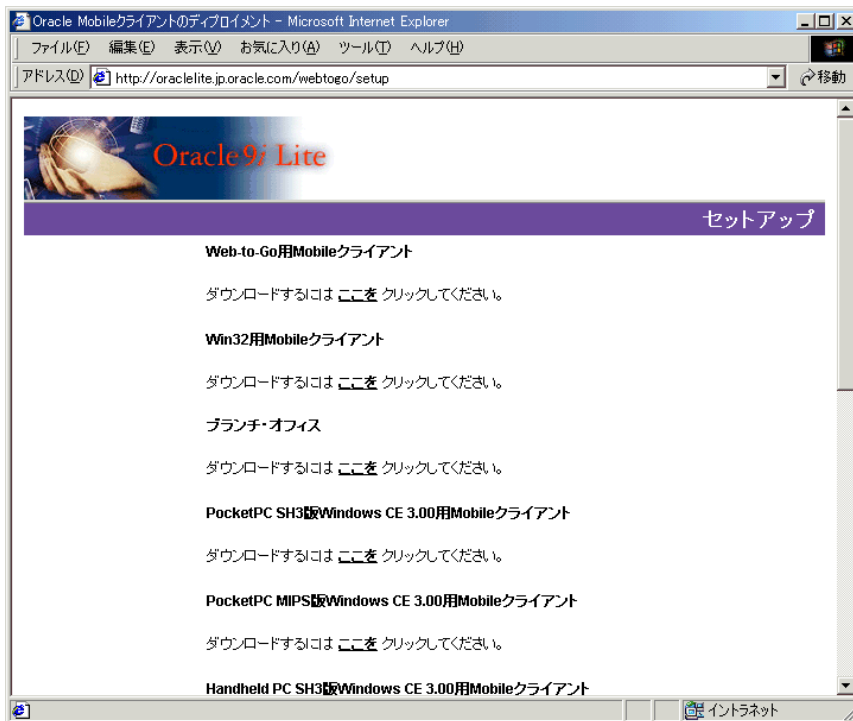
```
http://<MobileServer>/webtogo/setup
```

注意： MobileServer 変数は、Mobile サーバーのホスト名または IP アドレスに置き換える必要があります。

Web ページに、SH3、SH4、ARM プロセッサを搭載した Pocket PC や Handheld PC などのプロセッサが実装された、様々な Windows CE クライアントへのリンクが表示されます。

2. ハイパーリンク「ここを」をクリックして、デバイスに応じた Pocket PC 用 Mobile クライアントのセットアップ・プログラムにアクセスします。たとえば、Pocket PC は、StrongArm プロセッサで稼働します。

図 2-20 Pocket PC 用 Mobile クライアントのインストール



3. 使用しているブラウザが Netscape の場合は、デスクトップ上でセットアップ・プログラムを保存する場所を選択し、「OK」をクリックします。Windows のエクスプローラを開き、「setup.exe」を検索します。「setup.exe」をダブルクリックして、セットアップ・プログラムを実行します。

Internet Explorer を使用している場合は、ブラウザのウィンドウからセットアップ・プログラムを実行します。セットアップ・プログラムを起動すると、インストール先のディレクトリを指定するよう求められます。

4. C:\Temp などのローカル・ディレクトリを選択し、「OK」をクリックします。セットアップ・プログラムにより、すべての必要なコンポーネントがデスクトップ上の指定された場所に自動的にダウンロードされます。
5. これにより、ActiveSync アプリケーションを起動し、デバイスをデスクトップに接続できます。

6. 「ツール」メニューをクリックし、「アプリケーションの追加と削除」を選択します。

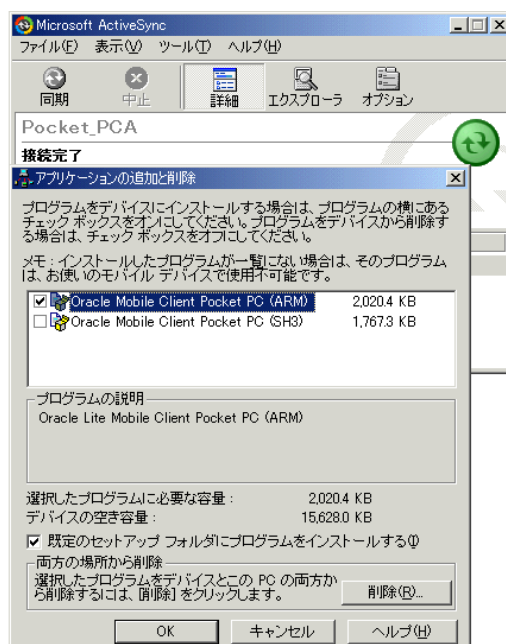
対応するプロセッサを搭載した使用可能な Pocket PC 用 Mobile クライアントのリストが表示されます。

7. デバイスとプロセッサのタイプに一致する Mobile クライアントを選択します。「アプリケーションの追加と削除」ダイアログ・ボックスで「OK」をクリックします。

アプリケーションをデバイスにインストールする処理が開始されます。処理が完了すると、Pocket PC 用 Mobile クライアントが、デバイスの ¥ORACE ディレクトリの下にインストールされます。

図 2-21 に、「アプリケーションの追加と削除」ダイアログ・ボックスを示します。このダイアログ・ボックスで、必要な Pocket PC のデバイス・タイプとプロセッサを選択できます。

図 2-21 Pocket PC 用 Mobile クライアントのインストール



2.5.2 Transport アプリケーションおよびデータのインストールと同期

Transport アプリケーションおよびデータをインストールするには、次の手順を実行します。

- 1. デバイス上の ¥OraCE ディレクトリを検索し、「mSync」アプリケーションをクリックします。
- 2. 「mSync」ダイアログ・ボックスが表示されます。Transport アプリケーションおよびスナップショットをダウンロードするために、次の表に示されているとおりにデータを入力します。

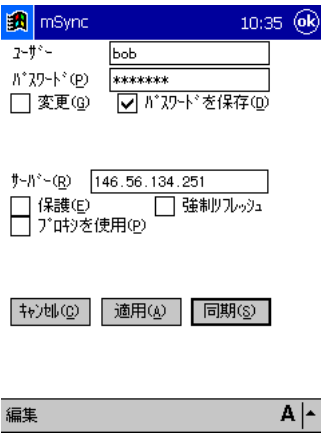
表 2-11 に、「mSync」ダイアログ・ボックスに入力する必要がある値を示します。

表 2-11 「mSync」ウィンドウの説明

名前	値
ユーザー名	bob
パスワード	bobhope（すべて小文字）
「パスワードを保存」チェックボックス	選択します。
サーバー	マシンの名前または IP アドレスです。

図 2-22 に、「mSync」ダイアログ・ボックスを示します。

図 2-22 mSync の実行



3. これらの値を保存するには、「適用」をクリックします。
4. 「同期」をクリックすると、アプリケーションおよびデータがデバイスと同期します。

注意： デバイスがデスクトップと接続されていること、および Mobile サーバーが稼働していることを確認してください。

5. 同期プロセスが完了すると、transport.odt ファイルおよび CAB ファイルが ¥OraCE ディレクトリの下に作成されます。
6. CAB ファイルをクリックして、アプリケーションをインストールします。
7. デバイスの「スタート」メニューを使用すると、「プログラム」メニューでアプリケーションを検索できます。
8. Transport アプリケーションをクリックします。

この章では、Mobile サーバーを使用したレプリケーションについて説明します。内容は次のとおりです。

- 3.1 項「概要」
- 3.2 項「同期プロセス」
- 3.3 項「CE デバイス上の Mobile クライアント」
- 3.4 項「RAS トランスポートの構成」
- 3.5 項「デバイスへのサンプル・アプリケーションのインストール」
- 3.6 項「同期のテスト」
- 3.7 項「Mobile Sync Application Program Interface (API)」

3.1 概要

Mobile サーバーにより、携帯端末上の新規または既存のアプリケーションやデータを Oracle データベースと同期させ共有できます。デバイス上のデータは、Mobile サーバー・エージェントを介して Oracle データベースに直接マップできます。Mobile Development Kit では、クライアントの携帯端末のデータ・サブセット化ポリシーを管理するパブリッシュ・サブスクライブ・モデルが使用されます。デバイス上のデータは、HTTP または ActiveSync によるトランスポートを介して Oracle データベースにマップされます。

最も一般的な同期方法は、高速リフレッシュです。この場合、クライアントにより変更がアップロードされ、クライアントに対する変更がダウンロードされます。バックグラウンド・プロセスは、すべてのクライアントによりアップロードされた変更を定期的に収集し、データベース表に適用します。その後、次の同期時に各クライアントに対して行われるダウンロードに備え、サブスクリプションと呼ばれる事前定義済のパラメータに基づいて、新規データを構成します。

もう 1 つの一般的な同期方法は、完全リフレッシュです。完全リフレッシュでは、パブリケーションのすべてのデータがクライアントにダウンロードされます。たとえば、最初の同期セッションでは、クライアントのすべてのデータは Oracle データベースからリフレッシュされます。パブリケーションにより参照されるすべての表が、クライアント・デバイスに転送されるため、このタイプの同期は時間がかかります。

3.1.1 同期の概念

Mobile サーバーと相互に通信するクライアント・デバイスから Mobile Sync クライアント・モジュールを起動することにより、オフラインのクライアント上の Oracle Lite データベースと Oracle データベース・インスタンス間でデータを同期します。Mobile サーバーは、ユーザー、パブリケーション、パブリケーション・アイテムおよびサブスクリプションなどの同期オブジェクトを使用して、同期セッション時のクライアント操作を処理します。これにより、Oracle Lite の基礎になる同期のパブリッシュ・サブスクライブ・モデルが構成されます。

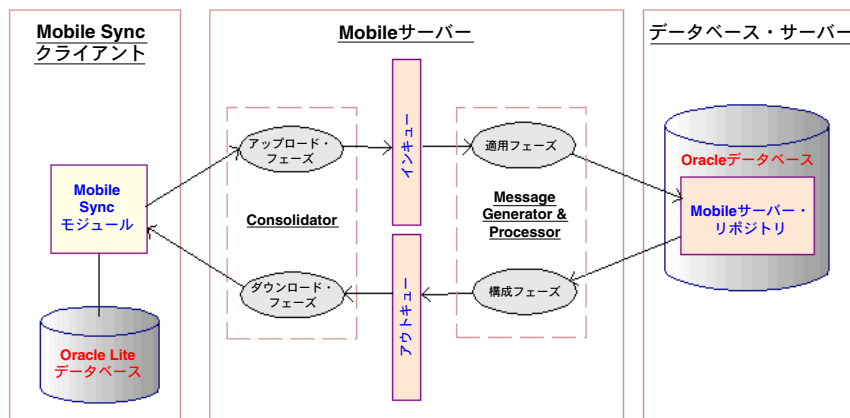
- パブリケーション・アイテムには、Oracle データベース表に対して定義された SQL 問合せが含まれています。パブリケーション・アイテムは、同期させるデータを定めるサブスクリプション・パラメータ（オプション）を指定できるスナップショット定義です。
- パブリケーションは、Mobile サーバー・リポジトリ内のパブリケーション・アイテムのコンテナとなり、クライアント・デバイス上に作成されたスナップショットにマップされます。
- パラメータを使用して、ユーザー依存データのサブセット化変数を定義します。データのサブセット化を行うことにより、パブリケーションをサブスクライブする特定のユーザーについてのデータ要件を詳細化できます。

- パブリケーションは、1人以上のユーザーによってサブスクライブされます。ユーザーおよびサブスクリプションは、Mobile サーバーにより追跡されます。
- Mobile サーバーは、確立されたサブスクリプションを介して、新規データを各クライアントに対して準備します。このデータは、クライアントの同期時にダウンロードされます。データのうち必要なサブセットのみが各クライアントにダウンロードされます。パブリケーションに完全リフレッシュのフラグが設定されている場合は、すべてのデータがクライアントにダウンロードされます。

3.2 同期プロセス

Oracle Lite は、Mobile サーバーが介在する Oracle Lite データベース・クライアントと Oracle データベース・サーバー間の同期については、非同期方法を使用します。これは、Mobile Sync モジュールは MGP とは独立して動作し、他に依存せずに操作を完了することを意味します。

図 3-1 高速リフレッシュ同期



3.2.1 高速リフレッシュ同期

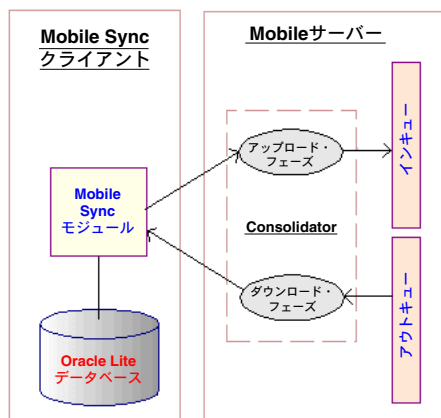
デフォルトの同期方法は、図 3-1 に示すとおり、高速リフレッシュ・モードです。高速リフレッシュは、アップロード・フェーズで変更がアップロードされインキューに格納された後、アウトキューに格納された変更がダウンロードされクライアントに適用される増分リフレッシュです。その間、MGP はインキューを定期的に検索し、そこで検索されたすべての内容を取り出し、適用フェーズでそれをデータベースに適用します。このクライアント、他のクライアント、Oracle データベースのサーバー側のアプリケーションから生成された変更は構成されて、次回のクライアントの同期時までアウトキューに格納されます。

アップロードおよびダウンロード・フェーズは、適用または構成フェーズとは独立して行われます。適用フェーズは、アップロード・フェーズには依存せず、構成フェーズもダウンロード・フェーズには依存しません。いずれの同期セッションにおいても、ダウンロードはアップロード後に発生し、構成は適用後に発生します。

3.2.1.1 クライアントのアップロードおよびダウンロード操作

同期が開始すると、クライアントでは選択したトランスポート・モードを介して Mobile サーバーへの接続がオープンされ、その結果 Mobile サーバーによって Oracle データベース・サーバーへの接続がオープンされます。以降のプロセスを図 3-2 に示します。

図 3-2 アップロードおよびダウンロード・フェーズ



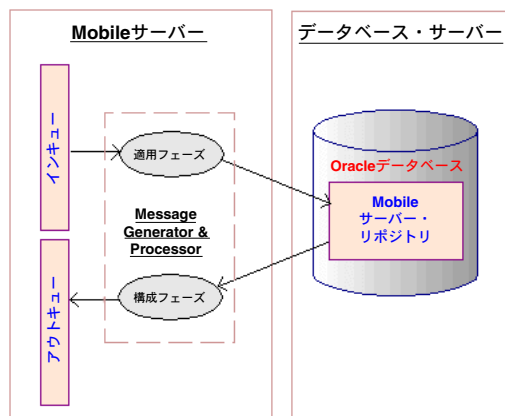
Oracle Lite データベース・レコードへの変更は、挿入、更新または削除が行われるデータ操作言語 (DML) のタイプ別コードとともに蓄積されフラグが設定されます。データは、暗号化および圧縮されて、インキューと呼ばれるオブジェクトを移入するために Mobile サーバーに送信されます。インキューは、同期の際に一時的にデータを格納するために作成される永続データベース・オブジェクトです。

同セッションの間、クライアントのスナップショットは、アウトキューから Oracle Lite データベースにデータを適用することで更新されます。アウトキューがインキューと異なるのは、表ではなく、Oracle データベースの実表に含まれるデータへの参照を含むデータ構造であるという点です。インキューおよびアウトキューでのデータの処理方法のカスタマイズについては、[6.4.9 項「レプリケーションをカスタマイズするためのキュー・インタフェース」](#)を参照してください。

3.2.1.2 Mobile サーバーの適用操作

MGP は、インキューのすべての内容を取り込み、Oracle データベースの実表に適用します。この時点で競合が検出され、解決されます。詳細は、6.5 項「同期エラーと競合」を参照してください。

図 3-3 適用および構成フェーズ



3.2.1.3 Mobile サーバーの構成操作

適用フェーズが終了すると、MGP は実表を再確認して構成し、クライアントにダウンロードする必要がある変更をアウトキューに格納します。適用および構成フェーズを図 3-3 に示します。

注意： MGP は、定期的にアクティブとなり、インキューを参照し、Oracle データベースの実表に変更を適用するバックグラウンド・プロセスです。MGP の構成方法によっては、同期の頻度にかかわらず、クライアントにダウンロードするアウトキューの構成および準備の速度が遅くなる場合があります。変更は、MGP が処理を行うまでインキューに安全に格納され、次の同期時にクライアントにダウンロードされます。

3.2.2 完全リフレッシュ同期

完全リフレッシュの間、クライアントに存在するスナップショット表のすべての内容は、Oracle データベース表からリフレッシュされます。すべての内容がリフレッシュされるため MGP には関係しませんが、このタイプの同期は、時間とシステム・リソースを集中的に使用します。

3.2.3 暗号化されたデータベースの同期

ENCRYPDB と呼ばれるユーティリティを使用して、Oracle Lite データベースを暗号化できます。暗号化されたデータベースを同期するには、Oracle Lite による暗号化されたデータベースの管理方法について理解しておく必要があります。詳細は、[B.4 項「ENCRYPDB」](#)を参照してください。

3.3 CE デバイス上の Mobile クライアント

Mobile クライアントは、Microsoft ActiveSync 3.5 のアプリケーションの追加と削除機能を使用して Mobile サーバーからインストールできます。ActiveSync 3.5 は、同期をとるために必須です。

3.3.1 ActiveSync の概要

ActiveSync によって、Windows ベースのデスクトップ・システムと Windows CE デバイスの間でデータを同期できます。同期プロセスは、デバイスとコンピュータの間の任意のトランスポートを介して発生し、追加、削除、またはその他の改訂操作を含むことができます。

注意： ActiveSync では、デバイス間のピアツーピア同期はできません。

3.3.2 Mobile サーバーからのファイルのインストール

Mobile サーバーから必要ファイルをデバイスに配布するには、使用している Mobile サーバー・インスタンスをブラウザに指定し、**/setup** を追加して、セットアップ画面を表示します。たとえば、Mobile サーバー・インスタンスの名前が **MyMobileServer** の場合、**MyMobileServer/webtogo/setup** と入力します。

ActiveSync を介してデスクトップに接続している Windows CE デバイスに対応するバージョンを選択します。リンクを選択すると、デスクトップに **.cab** ファイルがダウンロードされ、ActiveSync に自動的に登録されます。これで ActiveSync を実行できるようになります。ダウンロードするバージョンをリストから選択して、「OK」を押してください。Mobile サーバー・インスタンスで使用する言語セットのみが、ダウンロードできます。

3.3.3 データ・ソース名の作成

データ・ソース名 (DSN) は、データベースを識別するために使用する ODBC インタフェースおよび JDBC インタフェースの名前です。アプリケーションは、DSN を指定してデータベースに接続します。

Mobile クライアントは、そのクライアントと同期する各パブリケーション・アイテムの DSN を自動的に作成します。ここでは、それらの名前を生成する方法を説明します。

各パブリケーション・アイテムは、クライアント・デバイスの特定のデータベースにマップされます。

各データベースには、関連付けられた DSN があります。データベースの名前と DSN の名前は同じです (例外は、データベース・ファイルの拡張子が .ODB の場合)。アプリケーションをパブリッシュするときに、クライアントに作成される DSN とデータベースの名前を定義できます。パブリケーション・パネルの下で、クライアント・データベース名を設定できます。この文字列は、DSN およびデータベース名として使用されます。この名前を指定しないと、一意の文字列が生成されて、DSN およびデータベース名として使用されます。

たとえば、アプリケーションをパブリッシュしたときにクライアント・データベース名を「Orders」と設定したパブリケーションを作成した場合、初回の同期の後に、クライアント・デバイス上に DSN「Orders」とデータベース・ファイル「Orders.odbc」が作成されていることがわかります。

3.3.4 フラッシュ・メモリー・カードへのデータベースの格納

Oracle Lite は、フラッシュ・メモリー・カードへのデータベース・ファイルの格納をサポートします。コンパクトフラッシュが挿入されていると、ルート・ディレクトリに「メモリカード」という名前のディレクトリが表示されます。このディレクトリは、デバイスにある他の任意のディレクトリと同じように使用できます。このディレクトリ内に移動またはコピーするファイルは、すべてコンパクトフラッシュに格納されます。

初回の同期の前に polite.txt 構成ファイルを修正することで、すべてのデータベースをコンパクトフラッシュに格納するように、Oracle Lite に指示できます。初回の同期の後でこのファイルを修正すると、指定された場所には新しいパブリケーションのみが格納されます。

すべてのデータベースをコンパクトフラッシュに格納するには、初回の同期の前に、次の手順を実行する必要があります。

1. ActiveSync アプリケーション・ツールから、Oracle Lite for Windows CE をインストールします。
2. デバイスでファイル・エクスプローラを実行して、ファイル /ORACE/polite.txt を開きます。
3. セクション [All Databases] で、次のエントリを追加または変更します。

```
DATADIRECTORY=/ メモリ カード :/Oracle:/tmp
```

実際には DataDirectory に任意の有効なディレクトリを設定できます。これにより、すべてのデータベースがそのディレクトリに作成されます。ただし、そのディレクトリは初回の同期の前に作成する必要があります。

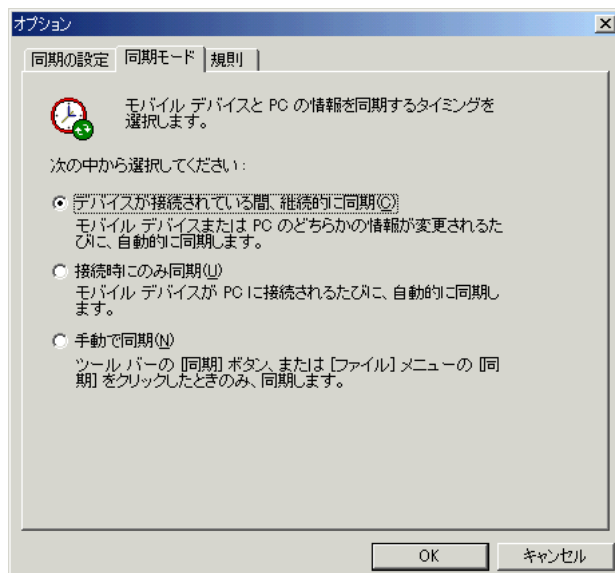
4. 4. mSync.exe を実行して同期します。

初回の同期の最初に、すべてのデータベースが / メモリ カード / ディレクトリに作成され、odbc.txt にあるすべての DSN がそのディレクトリを指していることがわかります。

3.3.5 ActiveSync との同期

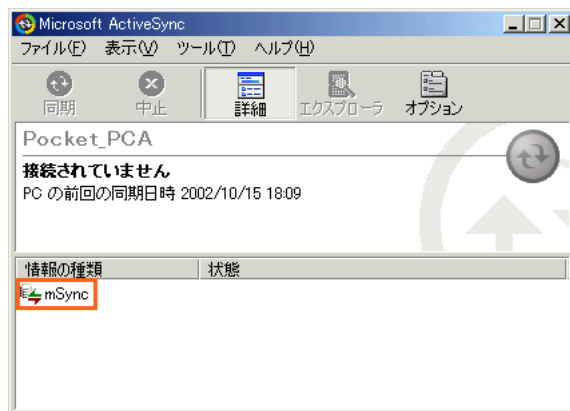
同期を開始する方法はいくつかあり、選択した同期モードによって異なります。同期モードを変更するには、図 3-4 で示す ActiveSync の「オプション」ダイアログに戻り、「同期モード」タブをクリックします。

図 3-4 「同期モード」のオプション



「デバイスが接続されている間、継続的に同期」または「接続時にのみ同期」のどちらかを選択している場合、クレードルにデバイスを置くと同時にすべての有効なプロバイダの同期を開始します。「手動で同期」を選択している場合、ActiveSync メイン・ウィンドウにある「同期」ボタンをクリックしないかぎり同期しません。このボタンは、ボタンバーの左上隅に表示されていて、上向きと下向きの矢印が付いています。

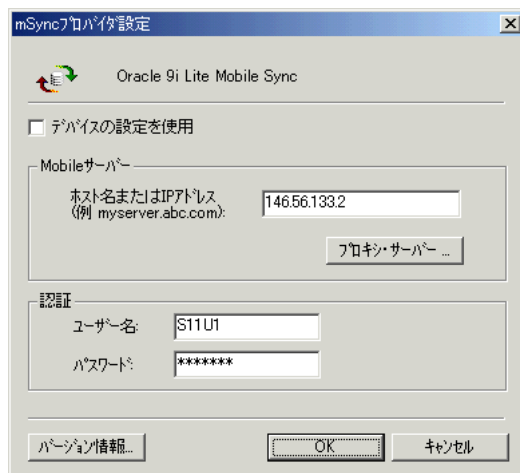
図 3-5 ActiveSync の「mSync」アイコン



3.3.5.1 同期の設定

同期の設定を変更する方法は、2 つあります。mSync が有効になっている場合、図 3-5 に示すとおり、ActiveSync のメイン・ウィンドウで「mSync」アイコンをダブルクリックします。または、ActiveSync の「オプション」ダイアログで「mSync」を選択し、「設定」ボタンをクリックします。いずれの場合も、図 3-6 に示すダイアログが表示されます。

図 3-6 「mSync プロバイダ設定」画面



ダイアログでの設定を表 3-1 に示します。

表 3-1 「mSync プロバイダ設定」画面のオプション

オプション	説明
デバイスの設定を使用	デフォルトで選択されています。これが選択されている場合、ローカルの設定が無効になり、プロバイダはデバイスの同期設定を使用しようとします。この場合、デバイス・プロファイルの設定には、デバイスから mSync.exe を実行し、情報を入力して「適用」をクリックします。同期の前にこの操作を完了していない場合、通常は Mobile サーバーに接続できないというエラーが返されます。 このチェックボックスが選択されていない場合は、ローカル設定のコントロールが有効になります。ローカル設定が、かわりに同期に使用されます。これは、 Windows CE デバイスとの直接モデム接続またはワイヤレス接続とは異なるネットワーク設定をデスクトップ上で使用している場合に役に立ちます。たとえば、オフィス・ネットワークに接続するときにプロキシを使用して、ワイヤレス・モデムを使用するときにはプロキシを使用しないようにできます。
Mobile サーバー：ホスト名または IP アドレス	Mobile サーバーの名前。XXX.XXX.XXX.XXX という形式の IP アドレスも入力できます。空白は許されていません。
プロキシ・サーバー ...	プロキシの設定を変更する場合は、このボタンを選択します。
ユーザー名	Mobile サーバーでの認証に使用されるログイン名。
パスワード	パスワード。

3.3.5.2 mSync プロキシ設定

「mSync プロキシ設定」画面を開くと、図 3-7 に示すオプションを設定できます。

図 3-7 「mSync プロキシ設定」画面



プロキシ設定パラメータを表 3-2 に示します。

表 3-2 「mSync プロキシ設定」画面のオプション

オプション	説明
プロキシ・サーバーを使用	このチェックボックスを選択すると、そのプロバイダに対してプロキシ・サーバーを使用できます。
ホスト名	プロキシ・サーバーのホスト名または IP アドレスです。
ポート番号	プロキシ・サーバーのポート番号です。サービス名も入力できますが、システム・サービス表で有効なサービスにかぎられます。
デフォルトを使用	ブラウザまたはオペレーティング・システムからプロキシの設定をコピーするときは、これを選択します。

3.4 RAS トランスポートの構成

HTTP を使用する Windows CE デバイスのレプリケートには、TCP/IP 通信プロトコルが必要です。ここでは、Windows NT RAS を使用して Windows CE デバイスの TCP/IP 通信を可能にする方法を示します。

3.4.1 NT RAS の構成

Windows NT RAS を使用して Windows CE デバイスと Windows NT サーバーの間の TCP/IP 通信を可能にするには、次の手順を実行します。

1. Windows の「コントロール パネル」で、「ネットワーク」のアイコンをダブルクリックします。
2. 「ネットワーク」ウィンドウの「サービス」タブで、「追加」ボタンをクリックします。
3. 「リモート アクセス サービス」を選択して「OK」ボタンをクリックします。
4. Windows NT の CD-ROM をディスク・ドライブに挿入して、セットアップ・プログラムが既存の Windows NT RAS ファイルを検索する場所を指定します。
5. 「続行」ボタンをクリックします。セットアップ・プログラムが、Windows NT RAS のファイルを適切なディレクトリにコピーします。セットアップ・プログラムがモデムを検出できない場合は、モデムの追加を要求するプロンプトを表示します。
6. 「はい」ボタンをクリックして、モデムをインストールするために必要な情報を入力します。「RAS デバイスの追加」ウィンドウが表示されます。
7. ドロップダウン・リストから「COM1- ダイヤルアップ ネットワーク シリアル ケーブル」を選択し、「OK」ボタンをクリックします。「リモート アクセス セットアップ」ウィンドウが表示されます。

8. 「ポート」の下で、COM1 を選択して「構成」ボタンをクリックします。「ポート使用の構成」ウィンドウが表示されます。
9. 「着信のみ」ラジオ・ボタンを選択して「OK」ボタンをクリックします。
10. 「リモート アクセス セットアップ」ウィンドウで「ネットワーク」ボタンをクリックします。「ネットワークの構成」ウィンドウが表示されます。
11. 「TCP/IP」を選択して「OK」ボタンをクリックします。「RAS サーバー TCP/IP の構成」ウィンドウが表示されます。
12. 「ネットワーク全体」および「静的アドレス プールを使う」を選択します。
13. 複数デバイスの場合、TCP/IP アドレスの範囲を指定できます。範囲を開始する TCP/IP アドレスを「開始アドレス」フィールドに、終了する TCP/IP アドレスを「終了アドレス」フィールドに入力します。範囲は、クライアント数 +1 にしてください。たとえば、50 台のデバイスがある場合、次の範囲を入力します。

Begin: 10.1.0.1

End: 10.1.0.51

重要： 同一ネットワーク上の他のコンピュータによって、選択した範囲の TCP/IP アドレスが使用されていないことを確認する必要があります。

14. 「OK」ボタンをクリックします。
15. 「ネットワークの構成」ウィンドウで、「クリア テキストを含む任意の認証を許可する」を選択して「OK」ボタンをクリックします。「続行」ボタンをクリックします。「セットアップの内部メッセージ」ウィンドウが表示されます。「OK」ボタンをクリックします。
16. 「ネットワーク」ウィンドウで、「閉じる」ボタンをクリックします。「ネットワーク設定の変更」ウィンドウが表示されます。「はい」ボタンをクリックします。
17. システムを再起動してから、Windows の「コントロールパネル」で「サービス」アイコンをクリックします。「サービス」ウィンドウが表示されます。
18. 「Remote Access Server」を選択して「開始」ボタンをクリックします。
19. 「自動」を選択して「OK」ボタンをクリックします。「閉じる」ボタンをクリックします。
20. Windows の「スタート」メニューで「プログラム」→「管理ツール」→「ユーザー マネージャ」を選択します。「ユーザー マネージャ」ウィンドウが表示されます。

3.4.2 デバイスの RAS User アカウントの作成

1. 「ユーザー」メニューで「新しいユーザー」を選択します。「新しいユーザー」ウィンドウが表示されます。
2. 必要なフィールドに、ユーザー名、パスワードおよびパスワードの確認を入力します。
3. 「パスワードを無期限にする」を選択します。
4. 「ダイヤルイン」ボタンをクリックします。「ダイヤルイン情報」ダイアログが表示されます。
5. 「ユーザーにダイヤルインの許可を与える」を選択します。「OK」ボタンをクリックします。
6. 「OK」ボタンをクリックすると、「新しいユーザー」ダイアログが終了します。
7. 「ユーザー マネージャ」画面を終了します。
8. Windows の「スタート」メニューで「プログラム」→「管理ツール」→「リモートアクセス管理」を選択します。「リモートアクセス管理」ウィンドウが表示されます。デバイスにサンプル・アプリケーションをインストールします。
9. 新規 RAS ユーザーにリモート・アクセス権限が付与されたかどうかを確認して、「OK」ボタンをクリックします。

3.4.3 Windows デスクトップの RAS 設定

RAS を構成するときには、次のことを確認する必要があります。

- モデムの設定が「ダイヤルアウトと着信」になっていること。
- Windows CE デバイスがドメインの任意のコンピュータに接続できるオプションを選択していること。
- Windows CE デバイスの静的 IP アドレスが、そのデバイスに接続しているデスクトップと同じサブネット内にあること。同じサブネット内にはない場合は HTTP 接続を作成できないため、レプリケーションが失敗します。
- Windows CE デバイスの終了 IP アドレスがネットワークで使用されていないこと。
- ドメイン・ログオン・サーバーが Windows CE デバイスに対して使用可能であること。
- Mobile Sync Client アプリケーションを実行する前に、Windows CE デバイスの IP アドレスが正しく設定されていること。

3.4.4 Windows デスクトップの設定

Windows デスクトップの設定を構成するときには、次のことを確認する必要があります。

- 「コントロール パネル」のモデムおよびポートの構成で、ボー・レートが正しく設定されていること。また、Windows CE デバイスのボー・レートがデスクトップのボー・レートと一致していること。
- Windows CE サービスをインストールした後で、最新の Windows NT サービス・パックを再インストールしていること。

3.5 デバイスへのサンプル・アプリケーションのインストール

Mobile Development Kit では、<ORACLE_HOME>%Mobile%SDK%wince%samples フォルダにサンプル・アプリケーションが提供されています。各フォルダに含まれている readme.txt ファイルでは、Microsoft eMbedded Visual Tools を使用してサンプルをコンパイルする方法が説明されています。

3.6 同期のテスト

<ORACLE_HOME>%Mobile%SDK%wince% 中の Windows CE のバージョンと使用しているデバイスのプロセッサがあるフォルダをオープンして Mobile クライアント・アプリケーションを開始し、そのフォルダから mSync.exe を開始します。Windows CE のバージョンやデバイスのプロセッサを判断する方法の詳細は、『Oracle9i Lite for Windows NT/2000/XP インストールेशनおよび構成ガイド』を参照してください。

Mobile Sync では、同期のために表 3-3 に示すパラメータが必要です。

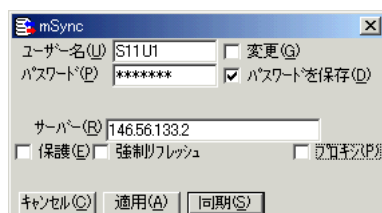
表 3-3 Mobile Sync Client のパラメータ

パラメータ	説明
ユーザー名	Mobile クライアント・ユーザー名です。このフィールドは大 / 小文字を区別しません。
パスワード	Mobile クライアント・パスワードです。このフィールドは大 / 小文字を区別します。
変更	このボックスは空白のままにします。
パスワードを保存	パスワードを保存するには、このチェック・ボックスを選択します。
サーバー	Mobile サーバーの IP アドレスです。
プロキシ	必要であれば、選択します。

SAMPLE11 サンプル・アプリケーションを利用するには、次のユーザー名とパスワードを使用します。ユーザー作成アプリケーションおよびユーザー・スナップショットには、別個のユーザー名とパスワードを指定する必要があります。

- ユーザー名 = S11U1
 - パスワード = MANAGER
1. Mobile Sync Client を起動するには、「mSYNC」アイコンをクリックします。図 3-8 に示す「mSync」画面が表示されます。

図 3-8 「Mobile Sync」画面



2. 必要なフィールドに情報を入力します。「パスワードを保存」チェック・ボックスを忘れないで選択してください。「サーバー」フィールドには、画面名または IP アドレスを含めることができます。
3. 「適用」ボタンを選択します。
4. 「同期」ボタンを選択します。

進行状況バーが表示され、各同期作業の進捗状況（構成中 ...、送信中 ...、受信 ...、処理中 ...）が表示されます。進行状況バーには、各作業の完了に要する時間も表示されます。同期が正常に実行されると、同期成功画面が表示されます。同期に失敗すると、同期が失敗したというメッセージが表示されます。同期に失敗した原因を判断するには、サーバー管理者は Mobile サーバー・ログ・ファイルで追跡情報を表示できます。

3.7 Mobile Sync Application Program Interface (API)

開発者にとって、基盤となる **ocapi.dll** ライブラリを使用して同期を起動するネイティブ・アプリケーションを作成する方法は 3 種類あります。これらの API は、Mobile Sync Client の **msync.exe** により提供されているものとは異なるアプローチで、アプリケーションを開発するために提供されています。

- [3.7.1 項「Java インタフェース」](#)
- [3.7.2 項「COM インタフェース」](#)
- [3.7.3 項「C/C++ インタフェース」](#)

注意： 現在 Sun SPARC Solaris 向けのクライアント側の同期プログラミング・インタフェースは存在しません。これらのインタフェースによるプログラミングには、Windows オペレーティング・システムを使用することを前提とします。

3.7.1 Java インタフェース

Mobile Sync のクライアント側の同期に Java インタフェースを使用することにより、Java で記述されたプログラムで、OCAPI ライブラリによって提供された機能を使用できます。Java インタフェースは、**oracle.lite.msync** パッケージ内にあります。

Java インタフェースにより、次の機能が提供されます。

- ユーザー名、パスワード、サーバーなどのデータを含む、クライアント側のユーザー・プロファイルの設定。
- 同期プロセスの開始。
- 同期プロセスの進行状況の追跡。

Java インタフェースは、**mSync.jar** および **msync_java.dll** の 2 つのファイルで構成されています。Java インタフェースを使用するには、クラスパスに **msync.jar** を含める必要があります。**mSync.jar** ファイルは、¥Orace ディレクトリにあります。**msync_java.dll** は、¥Windows ディレクトリにあります。

Java インタフェースは、4 つの部分に分かれています。

- [3.7.1.1 項「Sync クラス」](#)
- [3.7.1.2 項「SyncException クラス」](#)
- [3.7.1.3 項「SyncOption クラス」](#)
- [3.7.1.6 項「SyncProgressListener インタフェース」](#)

3.7.1.1 Sync クラス

このクラスは、提供されている同期オプションを使用して同期を開始します。コンストラクタのパラメータを[表 3-4](#)に示します。

コンストラクタ

`Sync(SyncOption option)`

表 3-4 Sync クラス・コンストラクタ

パラメータ	説明
option	SyncOption クラスのインスタンスです。同期を行うために必要なすべてのパラメータが含まれます。詳細は、 3.7.1.3 項「SyncOption クラス」 を参照してください。

パブリック・メソッド

パブリック・メソッド `SyncProgressListener` は、プログレス・リスナーをオブジェクトに追加して、同期の進行状況を監視します。

`SyncProgressListener add(ProgressListener listener)`

パブリック・メソッド `SyncProgressListener` のパラメータを[表 3-5](#)に示します。

表 3-5 Sync クラスのパブリック・メソッド

パラメータ	説明
listener	ProgressListener インタフェースを実装するオブジェクトです。同期オブジェクトは、このオブジェクトの <code>progress()</code> 関数をコールして、同期の進行状況を知らせます。
<code>void doSync()</code>	同期セッションを開始し、同期が完了するまでそのスレッドをブロックします。
<code>void abort()</code>	同期セッションを強制終了します。

例

次のコードは、デフォルト設定を使用したセッションの開始方法を示します。

```
try
{
    Sync mySync = new Sync( new SyncOption());
    mySync.doSync();
}
catch ( SyncException e)
{
}
```

```
System.err.println( "Sync Error:"+e.getMessage());
}
```

3.7.1.2 SyncException クラス

このクラスは、同期プロセス中のリカバリ不能なエラーの信号を送ります。
SyncException() は消去オブジェクトを構成します。コンストラクタのパラメータを表 3-6 に示します。

コンストラクタ

```
SyncException()
SyncException( int errorCode,StringerrorMessage)
```

表 3-6 SyncException コンストラクタのパラメータ

パラメータ	説明
errorCode	エラーです。使用可能なコードのリストは、『Oracle9i Lite メッセージ・リファレンス』を参照してください。
errorMessage	追加の情報を提供する、読込み可能なテキストのメッセージです。

パブリック・メソッド

SyncException のメソッドを表 3-7 に示します。

表 3-7 SyncException クラスのパブリック・メソッド

パラメータ	説明
int getErrorCode()	エラー・コードを取得します。
String getErrorMessage()	エラー・メッセージを取得します。

3.7.1.3 SyncOption クラス

SyncOption クラスは、同期プロセスのパラメータを定義するために使用します。このクラスは、手動で構成するか、またはユーザー・プロファイルに保存されているデータまたはロードされたデータを使用して構成することもできます。

コンストラクタ

```
SyncOption()
SyncOption
( String user,
  String password,
  String syncParam,
```

```
String transportDriver,  
String transportParam)
```

SyncOption コンストラクタのパラメータを[表 3-5](#) に示します。

表 3-8 SyncOption コンストラクタ

パラメータ	説明
user	Mobile サーバーが認証のために使用する名前を含む文字列です。
password	ユーザーのパスワードを含む文字列です。
syncParam	同期セッションのパラメータのオプション・リストを定義する文字列です。詳細は、 3.7.1.4 項「Java インタフェースの SyncParam 設定」 を参照してください。
transportDriver	トランスポート・ドライバ名を含む文字列です。現在は、HTTP のみがサポートされています。
transportParam	指定されたドライバが動作するために必要なすべてのパラメータを含む文字列です。詳細は、 3.7.1.5 項「Java インタフェースの TransportParam パラメータ」 を参照してください。

パブリック・メソッド

これらのメソッドは、ユーザー・プロファイルをロードおよび保存します。パブリック・メソッドのパラメータを[表 3-6](#) に示します。

表 3-9 SyncOption のパブリック・メソッドのパラメータ

パラメータ	説明
void load()	同期の最後のユーザーに対してプロファイルがロードされます。
void save()	アクティブ・ユーザーのプロファイルに設定を保存します。
void setUser(String username) String getuser()	現在のユーザーを設定および取得するために使用します。
void setPassword(String password) String getPassword()	パスワードを設定および取得するために使用します。

表 3-9 SyncOption のパブリック・メソッドのパラメータ (続き)

パラメータ	説明
<code>void setSyncParam(String syncParam)</code> <code>String getSyncParam()</code>	同期パラメータを設定および取得するために使用します。
<code>void setTransportDriver(String driverName)</code> <code>String getTransportDriver()</code>	ドライバ名を設定および取得するために使用します。リリース 5.0.2 では、HTTP ドライバがサポートされていません。
<code>void setTransportParam(String transportParam)</code> <code>String getTransportParam()</code>	トランスポート・パラメータを設定および取得します。

例

次のコード例は、デフォルト設定を使用した同期セッションの開始方法を示します。

```
SyncOption opt = new SyncOption
("sam", "lion", "pushonly", "HTTP", "server=server1;proxy=www-proxy.us.oracle.com;proxyP
ort=80");
opt.save();
```

3.7.1.4 Java インタフェースの SyncParam 設定

syncParam は、SyncOption オブジェクトを作成する際に渡される文字列です。同期セッションにサポート・パラメータを指定できます。この文字列は、名前と値のペアで構成されます。たとえば、次のようになります。

```
"name=value;name2=value2;name3=value3, ...;"
```

名前では大 / 小文字は区別されませんが、値では区別されます。使用可能なフィールド名を表 3-10 に示します。

表 3-10 Java インタフェースの SyncParam 設定

名前	値 / オプション	説明
reset	未対応	残りの設定を適用する前に、環境内のすべてのエントリを消去します。
security	SSL CAST5	SSL または CAST5 ストリーム暗号化から、該当する暗号化を選択します。

表 3-10 Java インタフェースの SyncParam 設定（続き）

名前	値 / オプション	説明
pushonly	未対応	この設定は、クライアントからサーバーに変更をアップロードする場合にのみ使用します。データの送信方向がクライアントからサーバーの場合に有効です。
noapps	未対応	新規または更新されたアプリケーションをダウンロードしません。低速な接続または低速なネットワーク上で同期する場合に有効です。
syncDirection	SendOnly	SendOnly は pushonly と同様です。
	ReceiveOnly	ReceiveOnly を使用すると、サーバーに変更が転送されません。
noNewPubs	未対応	この設定は、前回の同期以降に作成された新しいパブリケーションが送信されないようにし、現行パブリケーションのデータの同期のみを行います。
tableFlag [Publication.Item]	enable	enable に設定すると、[Publication.Item] で同期できます。disable は、同期を防止します。
	disable	
fullrefresh	未対応	完全リフレッシュを強制実行します。
clientDBMode	"EMBEDDED	EMBEDDED に設定すると、データベースへのアクセスは、従来の ODBC によって行われます。CLIENT に設定すると、複数のクライアントの ODBC によって行われます。
	"CLIENT	

例 1

次の例では、SSL セキュリティを有効にし、現在の同期セッションのアプリケーションの配布を無効にします。

```
"security=SSL; noapps;"
```

例 2

次の例では、前回のすべての設定をリセットし、Dept 表のアップロードのみをアクティブにします。

```
"reset;pushOnly;tableFlag[TestApp.Emp]=disable;tableFlag[TestApp.Dept]=enable;"
```

3.7.1.5 Java インタフェースの TransportParam パラメータ

TransportParam 文字列の形式は、名前と値のペアとなっている文字列を使用して特定のパラメータを設定します。たとえば、次のようになります。

```
"name=value;name2=value2;name3=value3, ...;"
```

名前では大 / 小文字は区別されませんが、値では区別されます。使用可能なフィールド名を表 3-11 に示します。

表 3-11 TransportParam パラメータ

名前	値	説明
reset	未対応	残りの設定を適用する前に、環境内のすべてのエントリを消去します。
server	サーバー・ホスト名	Mobile サーバーのホスト名または IP アドレスです。
proxy	プロキシ・サーバー・ホスト名	プロキシ・サーバーのホスト名または IP アドレスです。
proxyPort	ポート番号	プロキシ・サーバーのポート番号です。
cookie	Cookie 文字列	トランスポートに使用する Cookie です。

例

次の例では、Mobile Sync エンジンに、ポート 8080 の proxy.oracle.com というプロキシ・サーバーを介して test.oracle.com のサーバーを使用するように指示されます。

```
"server=test.oracle.com;proxy=proxy.oracle.com;proxyPort=8080;"
```

3.7.1.6 SyncProgressListener インタフェース

SyncProgressListener は、同期中に進行状況の更新をトラップできるようにするインタフェースです。

このクラスは、提供されている同期オプションを使用して同期を開始します。メソッドのパラメータを表 3-12 に示します。

メソッド

```
void progress
    (int progressType,
     int completed);
```

表 3-12 SyncProgressListener の抽象メソッド

パラメータ	説明
progressType	表 3-13 に示される定数の 1 つに設定します。
completed	特定の progressType の完了の割合です。

同期の進行状況をレポートする定数の名前を表 3-13 に示します。

表 3-13 SyncProgressListener インタフェースの定数

定数名	進行状況タイプ
PT_INT	同期エンジンが初期化段階であることを示します。current と total の数は 0 に設定されます。
PT_PREPARE_SEND	同期エンジンがサーバーに送信するローカル・データを準備中であることを示します。これには、ローカルに変更されたデータの取得が含まれます。ストリーミング実装の場合、所要時間はさらに短くなります。
PT_SEND	同期エンジンがネットワークにデータを送信中であることを示します。 total は送信対象のバイト数を示し、current は現在までに送信されたバイト数を示します。
PT_RECV	同期エンジンがサーバーからデータを受信中であることを示します。 total は受信対象のバイト数を示し、current は現在までに受信されたバイト数を示します。
PT_PROCESS_RECV	同期エンジンが、サーバーから新しく受信したデータを、ローカルのデータ・ストアに適用中であることを示します。
PT_COMPLETE	同期エンジンが同期プロセスを完了したことを示します。

例

この単純なクラスは、SyncProgressListener を実装します。

```
class myProgressTracker implements SyncProgress Listener;
{
    public void progress
        (int progressType,
         int completed)
        {
            System.out.println( "Status: "+progressType+"="+ completed+"%" );
        } //progress
}
```

3.7.2 COM インタフェース

COM インタフェースは、同期プロセスを開始するアプリケーションをプログラミングする際に使用され、様々な設定を使用可能にします。このインタフェースはモジュール形式で拡張可能であり、**ocapi.dll** をラッパー形式のインタフェースを介して使用できるようにします。このインタフェースは、アプリケーションを **Visual Basic** で作成できるように設計されていますが、他に **VBScript** などの COM インタフェースでサポートされているプログラミング手法も使用可能です。

3.7.2.1 機能およびコンポーネント

COM インタフェースは、次の機能をサポートします。

- 同期プロセスの開始
- 同期プロセスの進行状況の追跡
- ユーザー名、パスワードおよびサーバーなどのデータを含む、クライアント側ユーザー・プロファイルの設定
- 表レベルの同期オプションの割当て
- トランスポートの選択

COM インタフェース API およびサンプルは、
<ORACLE_HOME>%Mobile%SDK%MSyncCOM サブディレクトリにインストールされています。次のクラスは、**mSync.dll** ライブラリにあります。

- [3.7.2.2 項「ISync インタフェース」](#)
- [3.7.2.3 項「ISyncOption インタフェース」](#)
- [3.7.2.6 項「ISyncProgressListener インタフェース」](#)

このインタフェースは、MSync ライブラリに含まれています。ISync インタフェースを使用する場合は、インタフェース名として **MSync.ISync** を使用する必要があります。

COM インタフェース用のオブジェクト・ブラウザ参照を eVB に追加する場合、次の手順を実行します。

1. **msyncCom.dll** ファイルを、CE デバイスの %windows% サブディレクトリから開発用マシンのプロジェクトのディレクトリにコピーします。
2. プロジェクトで、「プロジェクト」>「参照設定」を選択します。
3. 「ブラウザ」をクリックして、新しくコピーした **msyncCom.dll** ファイルを参照として追加します。
4. 有効な参照として、'mSync 1.0 Type Library' を選択します。
5. MSYNC が有効なライブラリとして、ご使用のオブジェクト・ブラウザに表示されます。

3.7.2.2 ISync インタフェース

ISync インタフェースである Msync.ISync を使用すると、同期プロセスをインスタンス化できます。表 3-14 に、ISync インタフェースのフォーマットを示します。

表 3-14 ISync インタフェースのパラメータ

名前	説明
HRESULT doSync()	同期プロセスを開始します。同期プロセスが完了するまで、アクセスはブロックされます。
void abort()	現行の同期を強制終了します。これは、プログレス・リスナーのコールバックからコールできます。
HRESULT setOption(ISyncOption* syncObj)	次の同期時に使用するために、ポインタを SyncOption に設定します。doSync() の前にこの関数がコールされない場合、最後に保存したオプションが使用されます。

例

次の Visual Basic コードは、デフォルト設定を使用した同期セッションの開始方法を示します。

```
Dim sync As Msync.sync
Set sync = CreateObject("MSync.Sync")
sync.DoSync
```

SyncOption を使用しない場合、インタフェースは最後に保存された情報をロードして同期を実行します。

3.7.2.3 ISyncOption インタフェース

ISyncOption クラスの MSync.SyncOption は、同期プロセスのパラメータを定義します。これは手動で構成するか、ロードされたデータまたはユーザー・プロファイルに保存されているデータを使用して構成できます。ISyncOption クラスのパブリック・メソッドを表 3-15 に示します。load を使用する場合は、自動的に transportType パラメータに HTTP が設定されます。

transportType パラメータに HTTP を設定する必要があります。load を使用する場合は、自動的に transportType パラメータに HTTP が設定されます。load を使用しない場合は、このパラメータに HTTP を設定する必要があります。現在このパラメータには、HTTP タイプのみサポートされています。

表 3-15 ISyncOption のパブリック・メソッド

名前	説明
void load()	最後のユーザー同期のプロファイルをロードします。この方法では、transportType パラメータに HTTP が設定されます。
void save()	設定をユーザー・プロファイルに保存します。

ISyncOption クラスのパブリック・プロパティを表 3-16 に示します。

表 3-16 ISyncOption のパブリック・プロパティ

名前	説明
username	ユーザーの名前。
password	ユーザーのパスワード。
syncParam	同期の設定。詳細は、3.7.2.4 項「COM インタフェースの SyncParam 設定」を参照してください。
transportType	使用するトランスポートのタイプ。現在は、HTTP タイプのみサポートされています。デフォルト値は HTTP です。
transportParam	トランスポートのパラメータ。詳細は、3.7.2.5 項「COM インタフェースの TransportParam パラメータ」を参照してください。

例

次の Visual Basic コードは、デフォルト設定を使用した同期セッションの開始方法を示します。

注意： Windows CE では、ISyncOption インタフェース・オブジェクトは、次のように Dim'ed に設定する必要があります。

```
Dim syncOpt as MSync.SyncOption
```

```
Dim syncOpt as MSync.SyncOption
Set syncOpt = CreateObject("MSync.SyncOption")
' Load last sync info
syncOpt.Load
' Change user name to Sam
syncOpt.username = "Sam"
Set sync = CreateObject("MSync.Sync")
' Tell ISync to use this option
sync.setOptionObject (syncOpt)
' Do sync
sync.DoSync
```

3.7.2.4 COM インタフェースの SyncParam 設定

syncParam は、同期セッションにサポート・パラメータを指定するための文字列です。この文字列は、名前と値のペアで構成されます。たとえば、次のようになります。

```
"name=value;name2=value2;name3=value3, ...;"
```

名前では大 / 小文字は区別されませんが、値では区別されます。使用可能なフィールド名を表 3-17 に示します。

表 3-17 COM インタフェースの SyncParam 設定

名前	値またはオプション	説明
reset	未対応	残りの設定を適用する前に、環境内のすべてのエントリを消去します。
security	SSL CAST5	SSL または CAST5 ストリーム暗号化から、該当する暗号化を選択します。
pushonly	未対応	この設定は、クライアントからサーバーに変更をアップロードする場合にのみ使用します。ダウンロードは実行できません。データの送信方向が、クライアントからサーバーの場合に有効です。
noapps	未対応	新規または更新されたアプリケーションをダウンロードしません。低速な接続または低速なネットワーク上で同期する場合に有効です。
syncDirection	SendOnly	SendOnly は pushonly と同様です。
	ReceiveOnly	ReceiveOnly を使用すると、サーバーに変更が転送されません。
noNewPubs	未対応	この設定は、前回の同期以降に作成された新しいパブリケーションが送信されないようにし、現行パブリケーションのデータの同期のみを行います。
tableFlag [Publication.Item]	enable	enable に設定すると、 [Publication.Item] で同期できます。 disable は、同期を防止します。
	disable	
fullrefresh	未対応	完全リフレッシュを強制実行します。
clientDBMode	"EMBEDDED	EMBEDDED に設定すると、データベースへのアクセスは、従来の ODBC によって行われます。CLIENT に設定すると、複数のクライアントの ODBC によって行われます。
	"CLIENT	

例 1

次の例では、SSL セキュリティを有効にし、現在の同期セッションのアプリケーションの配布を無効にします。

```
"security=SSL;noapps;"
```

例 2

次の例では、前回のすべての設定をリセットし、Dept 表のアップロードのみをアクティブにします。

```
"reset;pushOnly;tableFlag[TestApp.Emp]=disable;tableFlag[TestApp.Dept]=enable;"
```

3.7.2.5 COM インタフェースの TransportParam パラメータ

TransportParam 文字列の形式は、名前と値のペアとなっている文字列を使用して特定のパラメータを設定します。たとえば、次のようになります。

```
"name=value;name2=value2;name3=value3, ...;"
```

名前では大 / 小文字は区別されませんが、値では区別されます。使用可能なフィールド名を表 3-18 に示します。

表 3-18 COM インタフェースの TransportParam パラメータ

名前	値	説明
reset	未対応	残りの設定を適用する前に、環境内のすべてのエントリを消去します。
server	サーバー・ホスト名	Mobile サーバーのホスト名または IP アドレスです。
proxy	プロキシ・サーバー・ホスト名	プロキシ・サーバーのホスト名または IP アドレスです。
proxyPort	ポート番号	プロキシ・サーバーのポート番号です。
cookie	Cookie 文字列	トランスポートに使用する Cookie です。

例

次の例では、Mobile Sync エンジンに、ポート 8080 の proxy.oracle.com というプロキシ・サーバーを介して test.oracle.com のサーバーを使用するように指示されます。

```
"server=test.oracle.com;proxy=proxy.oracle.com;proxyPort=8080;"
```


3.7.2.6 ISyncProgressListener インタフェース

ISync は接続ポイント・コンテナを実装して、同期の進行状況情報を追跡できるようにしています。ISyncProgressListener は、ISync インタフェースから更新を返すように実装する必要があります。ISyncProgressListener の抽象メソッドを表 3-19 に示します。

表 3-19 ISyncProgressListener の抽象メソッド

名前	説明
<code>HRESULT progress([in] int progressType, int param1, int param2</code>	新しい進行状況情報が使用可能になった場合に同期エンジンによりコールされます。progressType は、ISyncProgressListener 定数表に定義されている進行状況タイプ定数のいずれかに設定します。詳細は、3.7.1.6 項を参照してください。current は現在までの完了数で、total は最大値です。current が total に等しくなると、その段階が完了します。total および current の単位は、progressType によって異なります。

ISynchProgressListener は、同期中に進行状況の更新をトラップできるようにするインタフェースです。同期の進行状況をレポートする定数の名前を表 3-20 に示します。

表 3-20 ISyncProgressListener の定数

名前	進行状況タイプ
PT_INIT	同期エンジンが初期化段階であることを示します。current と total の数はともに 0 に設定されます。
PT_PREPARE_SEND	同期エンジンが、サーバーに送信するローカル・データを準備中であることを示します。これには、ローカルに変更されたデータの取得が含まれます。ストリーミング実装の場合、所要時間はさらに短くなります。
PT_SEND	同期エンジンがネットワークにデータを送信中であることを示します。 total は送信対象のバイト数を示し、current は現在までに送信されたバイト数を示します。
PT_RECEIVE	同期エンジンがサーバーからデータを受信中であることを示します。 total は受信対象のバイト数を示し、current は現在までに受信されたバイト数を示します。
PT_PROCESS_RECV	同期エンジンが、サーバーから新しく受信したデータを、ローカルのデータ・ストアに適用中であることを示します。
PT_COMPLETE	同期エンジンが同期プロセスを完了したことを示します。

例

次の Visual Basic コードは、イベントのレポート方法を示します。

```
' Define the ISync object with events
Dim WithEvents sync As MSync.sync
' Create the callback.
' The name of the call back is the name of the ISync object (not the class), and
' underscore and then the function name - progress
Private Sub sync_progress(ByVal progressType As Long, ByVal param1 As Long,
ByVal param2 As Long)
    Desc = ""
    ' Decipher the progressType
    Select Case progressType
        Case PT_SEND
            Desc = "Sending data..."
        Case PT_RECV
            Desc = "Receiving..."
    End Select
End Sub
```

3.7.3 C/C++ インタフェース

C/C++ インタフェースは、5つのファンクション・コールと1つの制御構造で構成されます。これらの定義は、<ORACLE_HOME>%Mobile%Sdk%<Form Factor>%Pub ディレクトリ内の **ocapi.h** および **ocapi.dll** にあります。この API を使用すると、アプリケーションはデータベースとの同期をクライアントから開始および監視でき、Mobile サーバーから開始する必要はありません。デフォルトの転送方法は HTTP ですが、他の転送形式が使用できる場合は、それを指定できます。

3.7.3.1 ocSessionInit

同期環境を初期化します。

構文

```
int ocSessionInit( ocEnv *env );
```

ocSessionInit 関数のパラメータを表 3-21 に示します。

表 3-21 ocSessionInit のパラメータ

名前	説明
env	戻される同期環境を保持する ocEnv 構造バッファへのポインタ。

コメント

このコールは、ocEnv 構造体を初期化し、最後の ocSaveUserInfo() コールで保存されたユーザー設定をリストアします。OcEnv 構造体を指すポインタがパラメータとして渡されるので、これをコール側で割り当てる必要があります。ocSessionInit() コールの後、コール側でユーザー設定情報を上書きする場合は、ocSaveUserInfo() をコールします。コール側は、ocEnv 構造体のメモリーを割り当てる必要があります。

3.7.3.2 ocSessionTerm

同期環境を解放し、クリーンアップします。

構文

```
int ocSessionTerm( ocEnv *env );
```

ocSessionTerm 関数のパラメータを表 3-22 に示します。

表 3-22 ocSessionTerm のパラメータ

名前	説明
env	ocSessionInit により戻された環境構造体へのポインタ。

コメント

ocSessionInit() コールにより作成された構造体とメモリーをすべて消去します。ocSessionInit() と ocSessionTerm() は常に一対でコールする必要があります。

3.7.3.3 ocSaveUserInfo

ユーザー設定を conscli.odb データベース・ファイルに保存します。

構文

```
int ocSaveUserInfo( ocEnv *env );
```

ocSaveUserInfo 関数のパラメータを表 3-23 に示します。

表 3-23 ocSaveUserInfo のパラメータ

名前	説明
env	同期環境へのポインタ。

コメント

この関数は、ユーザー設定をクライアント側のファイルまたはデータベースに保存するか上書きします。環境構造体で指定された次の情報が保存されます。

- username
- password
- savePassword
- newPassword
- priority
- secure
- pushOnly
- syncApps
- syncNewPublication

これらのフィールドの使用方法は、[3.7.3.7 項「C/C++ データ構造」](#)を参照してください。

3.7.3.4 ocDoSynchronize

同期プロセスを開始します。

構文

```
int ocDoSynchronize( ocEnv *env );
```

ocDoSynchronize 関数のパラメータを[表 3-24](#)に示します。

表 3-24 ocDoSynchronize のパラメータ

名前	説明
env	同期環境へのポインタ。

コメント

この関数は、同期サイクルを開始します。syncDirection が 0（デフォルト）の場合、ラウンドトリップ同期がアクティブになります。syncDirection が 1 の場合は、アップロード（送信操作）のみ実行されます。syncDirection が 2 の場合は、ダウンロード（受信操作）のみ実行されます。クライアントがサーバーからのデータのダウンロードを必要としない場合、アップロードのみの同期を実行すると有効です。

3.7.3.5 ocSetTableSyncFlag

選択同期用の表フラグを更新します。それぞれの表に対してこの関数をコールし、次のセッションで表を同期する (1) か、またはしない (0) かを指定します。

このオプションは、ocDoSynchronize の前に使用する必要があります。

ocSetTableSyncFlag の sync_flag のデフォルト設定は、すべての表で TRUE (1) です。デフォルトでは、すべての表にフラグが設定されて、同期されます。特定の表を選択して同期する場合は、最初にすべての表を同期するデフォルト設定を無効にする必要があります。その後、同期したい特定の表の選択同期を有効にします。

構文

```
ocSetTableSyncFlag(ocEnv *env, const char* publication_name,
const char* table_name, short sync_flag)
```

ocSetTableSyncFlag 関数のパラメータを次の表に示します。

表 3-25 に、ocSetTableSyncFlag パラメータの名前およびその説明を示します。

表 3-25 ocSetTableSyncFlag のパラメータ

名前	説明
env	同期環境へのポインタ。
publication_name	同期がとられるパブリケーションの名前です。publication_name の値が NULL の場合は、すべてのパブリケーションがデータベースにあることを表します。文字列は Consolidator の CreatePublication メソッドの client_name_template パラメータと同じです。多くの場合、このパラメータには NULL を使用します。詳細は、第 6 章「Consolidator」の 6.2.4 項「パブリケーションの作成」を参照してください。
table_name	スナップショットの名前です。これは、CreatePublicationItem() である 3 番目のパラメータである store の名前と同じです。詳細は、第 6 章「Consolidator」の 6.2.5 項「パブリケーション・アイテムの作成」を参照してください。
sync_flag	sync_flag が 1 に設定されている場合は、パブリケーションを同期する必要があります。sync_flag が 0 に設定されている場合は、同期しません。sync_flag の値は、永続的には格納されません。ocDoSynchronize() の前に毎回、ocSetTableSyncFlag() をコールする必要があります。

コメント

この関数により、クライアント・アプリケーションは、特定の表の同期方法を選択できます。

個々の表または個々のパブリケーションに対して `sync_flag` を設定します。
`sync_flag=0` の場合、表は同期されません。

特定の表のみを同期する場合、次の手順を実行します。

- 1. すべての表が 1 (TRUE) に設定されているデフォルト設定を無効にします。

例：

```
ocSetTableSyncFlag(&env, <publication_name>,null,0)
```

<publication_name> は、パブリケーションの実際の名前に置き換える必要があります。また、null 値は、例外なくパブリケーションのすべての表を指定します。

- 2. 特定の表の選択同期を有効にします。

例：

```
ocSetTableSyncFlag(&env, <publication_name>,<table_name>,1)
```

3.7.3.6 ocGetPublication

この関数は、Web-to-Go のアプリケーション名からクライアントのパブリケーション名を取得します。アプリケーションをパブリッシュする前に、パッケージ・ウィザードを使用してアプリケーションをパッケージする場合、Web-to-Go のユーザーはアプリケーション名のみを認識します。

構文

```
ocError ocGetPublication(ocEnv* env, const char* application_name,
char* buf, int buf_len);
```

ocGetPublication 関数のパラメータを表 3-26 に示します。この表に、ocGetPublication のパラメータ名およびその説明を示します。

表 3-26 ocGetPublication のパラメータ

名前	説明
env	戻される同期環境を保持する ocEnv 構造バッファへのポインタ。
application_name	アプリケーションの名前です。
buf	パブリケーション名が格納されるバッファです。
buf_len	バッファ長です。32 バイト以上である必要があります。

コメント

戻り値 0 は、関数が正常に実行されたことを示します。その他の値は、エラー・コードです。

この関数を使用すると、Web-to-Go のアプリケーション名からパブリケーション名が取得され、バッファに格納されます。

例

次のコード例は、Field オブジェクトの使用方法を示しています。

```
void sync()
{
    ocEnv env;
    int rc;

    // Clean up ocnv
    memset(&env 0, sizeof(env) );

    // init OCAPI
    rc = ocSessionInit(&env);

    strcpy(env.username, "john");
    strcpy(env.password, "john");

    // We use transportEnv as HTTP paramters
    ocTrHttp* http_params = (ocTrHttp*)(env.transportEnv.ocTrInfo);
    strcpy(http_params->url, "your_host");

    // Do not sync webtogo applicaton "Sample3"
    charbuf[32];
    rc = ocGetPublication(&env, "Sample3", buf, sizeof(buf));
    rc = ocSetTableSyncFlag(&env, buf, NULL, 0);

    // call sync
    rc = ocDoSynchronize(&env);
    if (rc < 0)
        fprintf(stderr, "ocDoSynchronize failed with %d:%d\n",
            rc, env.exError);
    else
        printf("Sync compeleted\n");

    // close OCAPI session
    rc = ocSessionTerm(&env);
    return 0;
}
```

3.7.3.7 C/C++ データ構造

Mobile Sync API に含まれるデータ構造体には、ocEnv と ocTransportEnv の 2 つがあります。

3.7.3.7.1 ocEnv

ocEnv は、内部メモリー・バッファと状態情報を保持するため、すべての Mobile Sync モジュール関数によって使用されるデータ構造です。この構造体を使用する前に、アプリケーションはこれを ocSessionInit に渡して環境を初期化する必要があります。構造体のパラメータを表 3-27 に示します。

表 3-27 ocEnv 構造体のフィールドのパラメータ

フィールド	型	使用方法	説明
username	char [MAX_USERNAME]	コール側は、ocSessionInit をコールする前に、このフィールドを設定する必要があります。	認証するユーザーの名前。
password	char [MAX_USERNAME]	コール側は、ocSessionInit をコールする前に、このフィールドを設定する必要があります。	ユーザーのパスワード（クリア・テキスト）。
newPassword	char [MAX_USERNAME]	コール側は、ocSessionInit をコールした後に、これらのフィールドをオプションで設定できます。	文字列の最初の文字が NULL ではなく、(char) 0 の場合、次の同期セッションについて有効なユーザーのパスワードに変更するようにサーバーに要求するために、この文字列はサーバーに送信されます。
savePassword	short	コール側は、ocSessionInit をコールした後に、これらのフィールドをオプションで設定できます。	1 に設定すると、「パスワード」フィールド内のパスワードがローカルに保存され、次回 ocSessionInit がコールされる際にロードされます。
appRoot	char [MAX_USERNAME]	コール側は、ocSessionInit をコールした後に、これらのフィールドをオプションで設定できます。	アプリケーションのコピー先であるディレクトリ。最初の文字が NULL である場合、デフォルトのディレクトリが使用されます。
priority	short	コール側は、ocSessionInit をコールした後に、これらのフィールドをオプションで設定できます。	予約済。

表 3-27 ocEnv 構造体のフィールドのパラメータ (続き)

フィールド	型	使用方法	説明
secure	short	コール側は、ocSessionInit をコールした後に、これらのフィールドをオプションで設定できます。	0 に設定すると、トランスポートでの安全が確保されません。 OC_DATA_ENCRYPTION に設定すると、CAST5 同期を使用します。 OC_SSL_ENCRYPTION に設定すると、SSL 同期を使用します (Win32 のみ)。
syncDirection	enum	コール側は、ocSessionInit をコールした後に、これらのフィールドをオプションで設定できます。	0 (OC_SENDRECEIVE) に設定すると、同期は双方向になります (デフォルト)。 OC_SENDONLY に設定すると、変更はサーバーのみにプッシュされます。これは、ローカルでの変更が収集されて送信された後に、同期を停止するために行われます。 エンジンによって (フロッピー・ベースなどの) 異なる段階を分離する必要がある同期の場合に便利です。OC_RECEIVEONLY に設定すると、変更は送信されず、サーバーからの更新のみ受信します。この設定では、受信操作のみ実行され、ローカル・データベースの段階への変更を可能にします。
trType	enum	ocSessionInit をコールする前に設定する必要があります。	0 (OC_BUILDIN_HTTP) に設定すると、HTTP 内蔵トランスポート・サーバーを使用します。OC_USER_METHOD に設定すると、ユーザーの提供したトランスポート・ファンクションを使用します。
exError	ocError	OCAPI によって更新される読み込み専用情報です。	拡張エラー・コード - OS または OKAPI エラー・コード。
transportEnv	ocTransportEnv		トランスポート・バッファ。3.7.3.7.2 項を参照してください。
fnProgress	progressProc	コール側は、ocSessionInit をコールした後に、これらのフィールドをオプションで設定できます。	NULL ではない場合、プログレス・リスナーのコールバックにポイントします。
totalSendDataLen	long	OCAPI によって更新される読み込み専用情報です。	合計送信バイト数をトランスポートへ通知する OCAPI によって設定されます。最初の fnSend() がコールされる前に設定します。
totalReceiveDataLen	long	OCAPI によって更新される読み込み専用情報です。	合計受信バイト数をトランスポートへ通知する OCAPI によって設定されます。最初の fnReceive() コール時に設定します。

表 3-27 ocEnv 構造体のフィールドのパラメータ (続き)

フィールド	型	使用方法	説明
userContext	void*	コール側は、ocSessionInit をコールした後に、これらのフィールドをオプションで設定できます。	コール側によって任意のコンテキスト情報 (処理する進行状況ダイアログ、レンダラのオブジェクト・ポインタなど) を設定できます。
ocContext	void*		予約済。
logged	short		予約済。
bufferSize	long		予約済 (ワイヤレス /Nettech のみ)。
pushOnly	short	コール側は、ocSessionInit をコールした後に、これらのフィールドをオプションで設定できます。	1 に設定すると、変更はサーバーのみにプッシュされます。
syncApps	short	コール側は、ocSessionInit をコールした後に、これらのフィールドをオプションで設定できます。	1 (デフォルト) に設定すると、アプリケーションの配布を行います。 0 に設定すると、サーバーからはいずれのアプリケーションも受信されません。
syncNewPublications	short	コール側は、ocSessionInit をコールした後に、これらのフィールドをオプションで設定できます。	1 (デフォルト) に設定すると、最後の同期以降に作成された新しいパブリケーションが受信されます。 0 に設定すると、既存のパブリケーションのみ同期します (ワイヤレスなどの低速な転送に便利です)。
clientDBMode	enum	コール側は、ocSessionInit をコールした後に、これらのフィールドをオプションで設定できます。	OC_DBMODE_EMBEDDED (デフォルト) に設定すると、ローカルの Oracle Lite ODBC ドライバが使用されます。 OC_DBMODE_CLIENT に設定すると、Branch Office ドライバが使用されます。
syncTimeLog	short	コール側は、ocSessionInit をコールした後に、これらのフィールドをオプションで設定できます。	1 に設定すると、同期開始時刻が conscli.odt に記録されます。
updateLog	short	コール側は、ocSessionInit をコールした後に、これらのフィールドをオプションで設定できます。	デバッグのみ。 1 に設定すると、サーバー側の行挿入情報および行更新情報がパブリケーション odt に記録されます。

表 3-27 ocEnv 構造体のフィールドのパラメータ (続き)

フィールド	型	使用方法	説明
options	short	コール側は、ocSessionInit をコールした後に、これらのフィールドをオプションで設定できます。	<div>デバッグのみ。次のフラグのビット・セット。</div> <ul style="list-style-type: none">■ OCAPI_OPT_SENDMETADATA メタ情報をサーバーへ送信します。■ または OCAPI_OPT_DEBUG デバッグ・メッセージを使用可能にします。■ OCAPI_OPT_DEBUG_F デバッグのために送信および受信されたすべてのバイトを保存します。■ OCAPI_OPT_NOCOMP 圧縮を使用禁止にします。■ OCAPI_OPT_ABORT 設定すると、OCAPI によって現在の同期セッションの強制終了が試みられます。■ OCAPI_OPT_FULLREFRESH OCAPI に対して、既存のすべてのデータをパージし、完全リフレッシュを実行するように強制します。

環境構造には、Mobile Sync モジュール関数の動作方法を変更するためにコール側が更新できるフィールドも含まれています。

```
typedef struct ocEnv_s {
    // User infos
    char    username[MAX_USERNAME]; // consolidator client id
    char    password[MAX_USERNAME]; // consolidator client password for
                                     // authentication during synchronization
    char    newPassword[MAX_USERNAME]; // resetting consolidator client password
                                     // on server side if this field is not blank
    short   savePassword;             // if set to 1, save 'password' as preference data on client device
    char    appRoot[MAX_PATHNAME];   // directory path on client device for deploying files
    short   priority;                 // reserved
    short   secure;                   // if set to 1, data encrypted over the wire
    enum {
        OC_SENDRECEIVE = 0,          // full step of synchronize
        OC_SENDOONLY,                // send phase only
    };
};
```

```

    OC_RECEIVEONLY,           // receive phase only

    // For Palm Only
    OC_SENDFILE,              // send into local file | pdb
    OC_RECEIVEFROMFILE        // receive from local file | pdb
}syncDirection;             // synchronize direction

enum {
    OC_BUILDIN_HTTP = 0,      // Use build-in Http transport method
    OC_USER_METHOD            // Use user defined transport method
}trType;                     // type of transport

ocError exError;             // extra error code

ocTransportEnv transportEnv;  // transport control information

// GUI related function entry
progressProc fnProgress;     // callback to track progress; this is optional

// Values used for Progress Bar. If 0, progress bar won't show.
long totalSendDataLen;        // set by OCAPI informing transport total number
                                // of bytes to send; set before the first fnSend() is called
long totalReceiveDataLen;     // to be set by transport informing OCAPI
                                // total number of bytes to receive;
                                //should be set at first fnReceive() call.

void* userContext;           // user defined context

void* ocContext;             // internal use only
short logged;                // internal use only

long bufferSize;             // send/receive buffer size, default is 0
short pushOnly;              // Push only flag, default is 0
short syncApps;              // Application deployment flag, default is 1
short syncNewPublication;    // Sync New publication flag, default is 1

enum {
    OC_DBMODE_EMBEDDED = 0,   // WIN32 only,
                                // Client database is accessed by conventional ODBC
    OC_DBMODE_CLIENT          // Client database is accessed by multi client ODBC
} clientDBMode;              // Default is OC_DBMODE_EMBEDDED.
short syncTimeLog;           // If 1, Log sync start time and end time to conscli.odbc.
short updateLog;             // If 1, Log serverside insert/update row info to user odb. Default is 0.
//short sendMetaData;        // If 1, Client send Table Meta data. Default is 0.
unsigned short options;      // refer OCPI_OPT
short cancel;                // Cancel button is on or off
} ocEnv;

```

3.7.3.7.2 ocTransportEnv

この構造体は、組み込み転送関数を上書きするために使用します。構造体に関数のリストを指定することにより、アプリケーションは同期エンジンによって使用されるトランスポート層に対して独自の実装を定義できます。

```
typedef struct ocTransportEnv_s {
void* ocTrInfo;           // transport internal context
                           // for built-in Http, mapped to ocTrHttp
connectProc fnConnect;    // plug-in callback to establish a connection from
                           // device to server
disconnectProc fnDisconnect; // plug-in callback to dismantle connection
                           // from device to server
sendProc fnSend;         // plug-in callback to send data
receiveProc fnReceive;   // plug-in callback to receive data
}ocTransportEnv;
```

3.7.4 選択同期

この機能により、モバイル・アプリケーションが、特定の表の同期方法を選択できます。モバイル・アプリケーションは、C/C++ インタフェース API の `ocSetTableSyncFlag` 関数を使用して、どのパブリケーションおよびパブリケーション・アイテムの同期が必要かを判断します。アプリケーションはこの関数をコールし、表ごとに `sync_flag` パラメータを 1（同期する）または 0（同期しない）に設定します。したがって、表のリストは実行中に動的に変更でき、アプリケーション開発者は選択同期をプログラムで制御できます。

ActiveX Data Objects for Windows CE

この章では、Oracle Lite データベースの ActiveX Data Objects for Windows CE について説明します。各項で、それぞれ異なる項目を記載しています。内容は次のとおりです。

- 4.1 項「概要」
- 4.2 項「機能」
- 4.3 項「ActiveX Data Objects for Windows CE のオブジェクト」
- 4.4 項「Connection オブジェクトのメソッド」
- 4.5 項「Connection オブジェクトのプロパティ」
- 4.6 項「Recordset オブジェクトのメソッド」
- 4.7 項「Recordset オブジェクトのプロパティ」
- 4.8 項「Field オブジェクトのプロパティ」
- 4.9 項「Fields コレクション」
- 4.10 項「SQL およびデータベースの参照」
- 4.11 項「エラー・メッセージ」

4.1 概要

ActiveX Data Objects はプログラミング・インタフェースです。これを使用すると、Visual Basic アプリケーションで Oracle Lite データベースの機能にアクセスできます。これは COM インタフェース標準を基にしたオブジェクト指向プログラミング・インタフェースで、基礎となるデータベース・エンジンにアクセスする機能を実現します。ActiveX Data Objects for Windows CE には、Windows CE 環境で実行できるように設計された ActiveX Data Objects 機能のサブセットが含まれます。

ActiveX Data Objects for Windows CE インタフェースをサポートするために、Oracle Lite データベースはまったく同じインタフェースを実現するモジュールを実装しました。これによって、Visual Basic プログラマは、Oracle Lite データベースに対して同じ Microsoft 社の ActiveX Data Objects for Windows CE インタフェースを使用してアプリケーションを開発できます。

Oracle Lite データベース の ActiveX Data Objects for Windows CE (OLADOCE) は、Microsoft Windows CE ActiveX Data Objects v2.0 SDK のガイドラインに準拠しています。このガイドラインは、ActiveX Data Objects 1.0 のデスクトップ・バージョンに多少の修正を加えたものです。

ほとんどすべてのプログラミング言語で、Oracle Lite データベースの ActiveX Data Objects for Windows CE にアクセスできますが、Oracle Lite データベースの実装は Visual Basic 環境にあわせて調整されています。

Oracle Lite データベース の ActiveX Data Objects for Windows CE は、Oracle Lite データベースの ODBC インタフェースの上位に位置し、アプリケーション・プログラムに対する COM インタフェースを実現します。

注意: OLADOCE は、現在エミュレータでは実行できません。このため、Windows CE デバイスに DLL を登録するには、ランタイム・クライアント・バンドルをインストールする必要があります。

4.2 機能

ActiveX Data Objects for Windows CE インタフェースはオブジェクト指向インタフェースで、様々な COM インタフェース (クラス) へのアクセスを介してその機能を提供します。

ActiveX Data Objects for Windows CE のオブジェクト、メソッドおよびプロパティを表 4-1 に示します。

表 4-1 ADOCE のオブジェクト型

インタフェース	メソッド	プロパティ
Connection	Open	ConnectionString
	Close	IsolationLevel
	BeginTrans	

表 4-1 ADOCE のオブジェクト型（続き）

インタフェース	メソッド	プロパティ
Recordset オブジェクト	CommitTrans	
	RollbackTrans	
	AddNew	AbsolutePage***
	CancelUpdate	AbsolutePosition
	Clone***	ActiveConnection***
	Close	BOF
	Delete	Bookmark***
	GetRows	CacheSize***
	Move	CursorType
	MoveFirst	EditMode
	MoveLast	EOF
	MoveNext	LockType
	MovePrevious	PageCount***
	Open	PageSize***
	Supports	RecordCount
	Update	Source
Field オブジェクト	(メソッドなし)	ActualSize
		Attributes
		DefinedSize
		Name
		Type
Fields（コレクション）	(メソッドなし)	UnderlyingValue
		Value
		Count

*** は未実装を示します。

ActiveX Data Objects for Windows CE の各オブジェクトには、一連のメソッドと一連のプロパティがあります。メソッドは、アプリケーションからコールできる関数です。プロパティはデータ・メンバーで、これを設定または取り出すことで機能の特定の部分を変更できます。

4.3 ActiveX Data Objects for Windows CE のオブジェクト

ActiveX Data Objects for Windows CE (ADOCE) COM ライブラリには、Connection、Recordset、Field および Fields という COM インタフェースまたはオブジェクトがあります。Connection オブジェクトを作成し、ODBC.txt に含まれる DSN エントリに基づく接続をアクティブにするには、CreateObject を使用します。新規の Recordset オブジェクトを作成するには、CreateObject を使用します。Field オブジェクトは、直接作成できません。既存のレコードセットのコンテキストにのみ存在するためです。特定の Field オブジェクトを参照するには、Set を使用します。たとえば、Connection、Recordset および Field オブジェクトを含む典型的な OLADOCE コード・セグメントは次のとおりです。

```
Option Explicit
'CursorType
Const adOpenKeyset = 1
'LockType
Const adLockOptimistic = 3
'Options
Const adCmdText = 1
Const adCmdTable = 2

Sub Main()
Dim conn
Dim rs, fldName
Set conn = CreateObject("oladoce.connection")
conn.Open "POLITE", "SYSTEM", "MANAGER"
conn.BeginTrans
Set rs = CreateObject("oladoce.recordset")
rs.Open "drop table newtable", conn, adOpenKeyset, adLockOptimistic, adCmdText
rs.Open "create table newtable (f1 varchar(3))", conn, adOpenKeyset,
adLockOptimistic, adCmdText
rs.Open "newtable", conn, adOpenKeyset, adLockOptimistic, adCmdTable
rs.AddNew "f1", "a"
rs.Close
conn.CommitTrans
rs.Open "newtable", conn, adOpenKeyset, adLockOptimistic, adCmdTable
Set fldName = rs.Fields("f1")
MsgBox fldName.Value
rs.Close
conn.Close
Set fldName = Nothing
```

```
Set rs = Nothing
Set conn = Nothing
End Sub
```

4.4 Connection オブジェクトのメソッド

Connection オブジェクトは、データベースへの接続をオープンするために使用します。オブジェクトには、[表 4-2](#) に示すメソッドがあります。

表 4-2 Connection オブジェクトのメソッド

メソッド	説明
Open	データベース・ソースへの接続をオープンします。
BeginTrans	自動コミット・モードを停止し、新しいトランザクションを開始します。
CommitTrans	現在のトランザクションをコミットし、自動コミット・モードへ戻ります。
RollbackTrans	現在のトランザクションをロールバックし、自動コミット・モードへ戻ります。
Close	現在の接続をクローズします。

4.4.1 Open

データベースへの接続をオープンします。

構文

```
Connection.Open ConnectionString, UserID, Password, Options
```

Connection.Open のパラメータを[表 4-3](#) に示します。

表 4-3 Connection.Open のパラメータ

パラメータ	説明
ConnectionString	接続しているデータベースの DSN です。
UserID	オプションです。デフォルト値は SYSTEM です。指定した場合、データベースへの接続を認証するためにこの UserID（ユーザー名）が使用されます。
Password	オプションです。デフォルト値は MANAGER です。指定されたユーザーのパスワードです。
Options	これは予約済フィールドです。空白にするか「0」に設定します。

コメント

メソッド `Open` は、データベースへの接続をオープンします。データベースは、`ConnectionString` パラメータのデータソース名によって識別されます。データベースに対して認証が必要な場合は、オプションで `UserId` および `Password` を提供できます。引数 `Options` は、現在は使用されておらず、互換性のためのみに定義されています。

すでに接続された接続オブジェクトで `Connection.Open` をコールした場合、まず前回の接続がクローズになり、新しい接続がオープンになります。前回の接続に「コミットされるまでの」変更がある場合、その変更はロールバックされます。

例

```
Dim conn
Set conn = CreateObject("oladoce.connection")
conn.open "MyDSN", "sam", "123"
```

4.4.2 BeginTrans

このメソッドは、自動コミット・モードを使用禁止にして、新しいトランザクションを開始します。

構文

```
connection.BeginTrans
```

コメント

デフォルトでは、すべての変更は、レコードごとにのみ保持されます。

`Recordset.Update` (例では `rs.update`) がコールされると、変更はデータベースにコミットされます。ただし、いくつかの行に変更を加え、1つのトランザクションでコミットする場合は、まず `BeginTrans` をコールし、次にトランザクションを終了するために `CommitTrans` または `RollbackTrans` をコールする必要があります。正常に行われた `BeginTrans` へのコールの後のすべての操作は、`CommitTrans` または `RollbackTrans` がコールされるまでコミットされません。`CommitTrans` または `RollbackTrans` がコールされる前に `Recordset` オブジェクトがクローズにされると、トランザクションは自動的にロールバックされます。

例

```
Option Explicit
'CursorType
Const adOpenKeyset = 1
'LockType
Const adLockOptimistic = 3
'Options
Const adCmdText = 1
```

```
Const adCmdTable = 2

Sub Main()
Dim conn, rs

Set conn = CreateObject("oladoce.connection")
conn.Open "polite", "system", "manager"
conn.BeginTrans

Set rs = CreateObject("oladoce.recordset")
rs.Open "drop table newtable", conn, adOpenKeyset, adLockOptimistic, adCmdText
Err.Clear

rs.Open "create table newtable(c1 number, c2 varchar(10))", conn, adOpenKeyset,
adLockOptimistic, adCmdText
rs.Close

rs.Open "newtable", conn, adOpenKeyset, adLockOptimistic, adCmdTable
rs.AddNew
rs.Fields("c1") = 777
rs.Fields("c2") = "AAA"
rs.Update

conn.CommitTrans
rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
End Sub
```

4.4.3 CommitTrans

このメソッドは、現在のトランザクションをコミットします。

構文

Connection.CommitTrans

コメント

関数は、BeginTrans コールによってオープンにされた現在のトランザクションをコミットします。トランザクションのコミットが失敗した場合、エラー・オブジェクトが発生します。

4.4.4 RollbackTrans

このメソッドは、現在のトランザクションをロールバックします。

構文

Connection.RollbackTrans

コメント

RollbackTrans メソッドは、BeginTrans コールによって開始された現在のトランザクションをロールバックします。

4.4.5 Close

現在の接続をクローズします。

構文

Connection.Close

コメント

このメソッドは、Connection オブジェクトと関連するデータベースへの現在の接続をクローズするために使用します。この接続で作成された Recordset オブジェクトは、Connection.Close をコールする前にクローズする必要があります。

4.5 Connection オブジェクトのプロパティ

Connection オブジェクトによってサポートされるプロパティを[表 4-4](#)に示します。

表 4-4 Connection オブジェクトのプロパティ

プロパティ	説明
ConnectionString	現在の接続文字列です。
IsolationLevel	現在の分離レベルです。

4.5.1 ConnectionString

接続の現在の DSN を含む文字列です。

コメント

ConnectionString は、現在の接続の DSN です。ConnectionString を変更した場合、Connection.Open がコールされるまで現在の接続は変更されません。また、Connection.Open とともに ConnectionString パラメータがコールされた場合、ConnectionString プロパティの中身が上書きされます。

アプリケーションは、このプロパティから読み込むことによって、接続オブジェクトによって使用されている DSN を特定できます。

4.5.2 IsolationLevel

現在の分離レベルを設定または取得します。

コメント

デフォルトでは、IsolationLevel はコミット読み込みです。また、ユーザーは、特定の DSN のために **odbc.txt** ファイル内の IsolationLevel フィールドを変更できます。このフィールドが設定されている場合、その DSN への接続が確立されたときに、デフォルトとして特定の分離レベルが使用されます。

実行時には、ユーザーは、アクティブな Connection オブジェクトの IsolationLevel プロパティを設定することにより分離レベルを変更できます。これは、特定の Connection オブジェクトを使用して作成されたすべてのレコードセットに影響します。

プロパティは、次の値を持つ IsolationLevelEnum を返します。

- adXactUnspecified - レベルが判断できないことを表します。
- adXactReadUncommitted - 分離レベルが非コミット読み込み（内容を保証しない読み込み）であることを表します。
- adXactReadCommitted - 分離レベルがコミット読み込みであることを表します。
- adXactRepeatableRead - 分離レベルがリピータブル・リードであることを表します。
- adXactSerializable - 分離レベルがシリアル可能であることを表します。

4.6 Recordset オブジェクトのメソッド

Recordset オブジェクトによってサポートされるメソッドを表 4-5 に示します。

表 4-5 Recordset オブジェクトのメソッド

メソッド	説明
AddNew	レコードセットに新規の行を 1 行挿入します。
CancelUpdate	メモリーに保持されている変更を取り消します。
Clone***	レコードセットを複製します。
Close	レコードセットをクローズします。
Delete	レコードセットから 1 行を削除します。
GetRows	レコードセットに格納されているデータを返します。
Move	レコードセットのアクティブ行にポインタを変更します。
MoveFirst	最初の行をアクティブにします。
MoveLast	最後の行をアクティブにします。
MoveNext	レコードセットのアクティブ行ポインタを次の行に移動します。
MovePrevious	レコードセットのアクティブ行ポインタを前の行に移動します。
Open	レコードセットを定義してオープンします。SQL コマンドを実行します。
Update	メモリーに保持されている変更をコミットし、実際の表を更新します。
Supports	レコードセットが特定の機能をサポートするかどうかを判断します。

*** は未実装を示します。

ActiveX Data Objects for Windows CE データベースは単一ユーザーによるアクセスを想定しているので、バッチ更新モードはありません。ActiveX Data Objects for Windows CE では、次のメソッドをサポートしません。

- CancelBatch
- NextRecordset
- Requery
- Resync
- UpdateBatch

4.6.1 AddNew

更新可能な Recordset オブジェクトに新規レコードを作成します。

構文

`recordset.AddNew Fields, Values`

AddNew のパラメータを表 4-6 に示します。

表 4-6 AddNew のパラメータ

パラメータ	説明
Fields	オプションです。新規レコードの単一のフィールド名、または複数のフィールド名やフィールドの位置を示す順序数を表す配列です。
Values	オプションです。新規レコードのフィールドの単一の値、または複数の値を表す配列です。

コメント

新規レコードを作成および初期化するには、AddNew メソッドを使用します。現在の Recordset オブジェクトにレコードを追加できるかどうかを検証するには、Supports メソッド (Const AdAddNew = &H01000400) を使用します。デフォルトの LockType (Const AdLockReadOnly = 1) でレコードセットをオープンしている場合は、レコードを追加できません。これらの関数の詳細は、4.6.8 項「Open」および 4.6.9 項「Supports」を参照してください。

AddNew メソッドをコールすると、新規レコードがカレント・レコードになります。Update メソッドをコールした後もカレント・レコードのままになります。新規レコードは Recordset の最後に追加されます。

カレント・レコードの編集時または新規レコードの追加時に AddNew をコールすると、ActiveX Data Objects for Windows CE は Update メソッドをコールしてすべての変更を保存した後で新規レコードを作成します。

Fields が配列の場合、Values も同じメンバー数の配列である必要があります。それ以外の場合はエラーになります。両方の配列で、フィールド名の順序はフィールド値の順序と一致する必要があります。

引数なしで AddNew メソッドをコールすると EditMode プロパティが 2 (AdEditAdd) に設定され、ActiveX Data Objects for Windows CE はすべてのフィールド値の変更をローカルにキャッシュします。Update メソッドをコールすると新規レコードがデータベースに転送され、EditMode プロパティが 0 (AdEditNone) に設定されます。Fields 引数および Values 引数を渡すと、ActiveX Data Objects for Windows CE は即時に新規レコードをデータベースに転送します。Update のコールは必要ありません。EditMode プロパティ値は 0 (AdEditNone) から変更されません。

ActiveX Data Objects for Windows CE の更新は、常に即時モードです。バッチ・モードはサポートされていません。

例

```
Option Explicit
'CursorType
Const adOpenKeyset = 1
'LockType
Const adLockOptimistic = 3
'Options
Const adCmdText = 1
Const adCmdTable = 2

Sub Main()
Dim conn
Dim rs, fldName
Set conn = CreateObject("oladoce.connection")
conn.Open "POLITE", "SYSTEM", "MANAGER"
conn.BeginTrans
Set rs = CreateObject("oladoce.recordset")
rs.Open "drop table newtable", conn, adOpenKeyset, adLockOptimistic, adCmdText
rs.Open "create table newtable (f1 varchar(3))", conn, adOpenKeyset,
    adLockOptimistic, adCmdText
rs.Open "newtable", conn, adOpenKeyset, adLockOptimistic, adCmdTable
rs.AddNew "f1", "a"
rs.Close
conn.CommitTrans
rs.Open "newtable", conn, adOpenKeyset, adLockOptimistic, adCmdTable
Set fldName = rs.Fields("f1")
MsgBox fldName.Value
rs.Close
conn.Close
Set fldName = Nothing
Set rs = Nothing
Set conn = Nothing
End Sub
```

4.6.2 CancelUpdate

Update メソッドをコールする前に、カレント・レコードまたは新規レコードに対して加えられたすべての変更を取り消します。

構文

```
recordset.CancelUpdate
```

コメント

カレント・レコードに対するすべての変更を取り消すか、新規に追加したレコードを廃棄するには、CancelUpdate メソッドを使用します。オープンされたトランザクション内でないかぎり、Update メソッドをコールした後に、カレント・レコードまたは新しいレコードに対しての変更を元に戻せません (BeginTrans がコールされます)。この場合、BeginTrans へのコール後に、RollbackTrans をコールして変更を元に戻せます。

新規レコードの追加中に CancelUpdate メソッドをコールした場合は、AddNew をコールする前にカレントだったレコードが再びカレント・レコードになります。EditMode プロパティは 0 にリセットされます。

カレント・レコードの変更または新規レコードの追加をしていない場合、CancelUpdate メソッドはエラーになります。変更があった場合、レコードセットの EditMode プロパティは 0 以外の値になります。

例

```
Option Explicit
'CursorType
Const adOpenKeyset = 1
'LockType
Const adLockOptimistic = 3
'Options
Const adCmdTable = 2
'EditMode
Const adEditNone = 0
Const adEditAdd = 2

Sub Main()
Dim conn
Dim rs
Set conn = CreateObject("oladoce.connection")
conn.Open "POLITE", "SYSTEM", "MANAGER"
Set rs = CreateObject("oladoce.recordset")
rs.Open "emp", conn, adOpenKeyset, adLockOptimistic, adCmdTable
rs.AddNew 'Changes the EditMode from 0 to 2
If rs.EditMode = adEditAdd Then rs.CancelUpdate 'Change the EditMode from 2 to 0
MsgBox rs.EditMode
rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
End Sub
```

4.6.3 Close

オープンしているオブジェクトと依存オブジェクト（ある場合はすべて）をクローズします。

構文

```
recordset.Close
```

コメント

Recordset オブジェクトをクローズしてそのオブジェクトに関連するすべてのシステム・リソースを解放するには、Close メソッドを使用します。ただし、オブジェクトをクローズしてもメモリーからは削除されません。プロパティの設定を変更して、後で再びオープンできます。メモリーからオブジェクトを完全に除去するには、Recordset オブジェクト変数を Nothing に設定します。

Recordset オブジェクトをクローズすると、この特定の Recordset オブジェクトを介して取得した関連データは解放されますが、その元になるデータベースには影響しません。後で Open メソッドをコールして、そのレコードセットを同じ属性または変更された属性で再オープンできます。基礎となるデータベースの構造を変更する前に、レコードセットをクローズする必要があります。

Recordset オブジェクトがクローズされている間は、現在行を要求するメソッドはすべてエラーになります。また、クローズされているレコードセットを再度クローズしようとすると、エラーになります。

編集中に Close メソッドをコールすると、エラーになります。最初に Update メソッドまたは CancelUpdate メソッドをコールします。

4.6.4 Delete

オープンしている Recordset オブジェクトのカレント・レコードを削除します。

構文

```
recordset.Delete AffectRecords
```

Delete メソッドのパラメータを [表 4-7](#) に示します。

表 4-7 Delete のパラメータ

パラメータ	説明
AffectRecords	オプションです。AffectRecords の有効な値は、1 (AdAffectCurrent) のみです。他の値では、ランタイム・エラーになります。この機能は、デスクトップ・コンピュータの ActiveX Data Objects との互換性のために提供されています。

コメント

Delete メソッドを使用すると、データベースからカレント・レコードが削除されます。Recordset オブジェクトがレコードの削除を許可していない場合はエラーになります。

削除されたレコードからフィールドの値を取り出そうとすると、エラーになります。

削除されたレコードがアクセス不可になった後でも、別のレコードに移動するまでは削除されたレコードがカレント・レコードです。レコードの削除に失敗すると、ランタイム・エラーになります。

4.6.5 GetRows

Recordset の複数のレコードを配列に取り出します。

構文

recordset.GetRows (Rows, Start, Fields

GetRows メソッドのパラメータを表 4-8 に示します。

表 4-8 GetRows のパラメータ

パラメータ	説明
Array	返されたデータが格納される VARIANT 型変数。
Rows	オプションです。LONG 型の式で、取り出すレコードの数を示します。デフォルト値は AdGetRowsRest または -1 です。
Start	オプションです。STRING 型または VARIANT 型で、GetRows 操作の開始レコードに対するブックマークに評価されます。
Fields	オプションです。VARIANT 型で、単一のフィールド名または位置を示す順序数、あるいは複数のフィールド名やフィールドの位置を示す順序数の配列を表します。ADOCE はここで指定したフィールドのデータのみを返します。

コメント

GetRows メソッドは、Recordset から 2 次元配列にレコードをコピーするために使用します。最初の添字はフィールドを示し、2 番目の添字はレコード番号を示します。GetRows メソッドがデータを返すときに、Array 変数のディメンションが自動的に設定されます。

Rows 引数の値を指定しないと、GetRows メソッドは自動的に Recordset オブジェクトからすべてのレコードを取り出します。使用可能なレコード数を超過して要求すると、GetRows は使用可能な数のレコードのみを返します。

Recordset オブジェクトがブックマークをサポートしている場合、どのレコードからデータの取出しを開始するかを、レコードの **Bookmark** プロパティの値を **GetRows** メソッドに渡して指定できます。

GetRows コールから返されるフィールドを制限する必要がある場合、単一のフィールド名またはフィールド数、あるいはフィールド名またはフィールド数の配列を **Fields** 引数として渡せます。

GetRows をコールした後は、まだ読み込まれていない次のレコードがカレント・レコードになりますが、それ以上レコードがない場合は **EOF** プロパティが **TRUE** に設定されます。

例

```
Option Explicit
'CursorType
Const adOpenKeyset = 1
'LockType
Const adLockOptimistic = 3
'Options
Const adCmdTable = 2

Sub Main()
Dim conn
Dim rstEmployees
Dim avarRecords As Variant
Dim intRecCount

Set conn = CreateObject("oladoce.Connection")
conn.Open "POLITE", "SYSTEM", "MANAGER"
Set rstEmployees = CreateObject("oladoce.recordset")
rstEmployees.Open "emp", conn, adOpenKeyset, adLockOptimistic, adCmdTable

'Store results of GetRows method in array.
avarRecords = rstEmployees.GetRows(3)
intRecCount = UBound(avarRecords, 2) + 1

MsgBox intRecCount

rstEmployees.Close
conn.Close
Set rstEmployees = Nothing
Set conn = Nothing
End Sub
```

4.6.6 Move

Recordset オブジェクトのカレント・レコードの位置を移動します。

構文

`recordset.Move NumRecords, Start`

Move メソッドのパラメータを表 4-9 に示します。

表 4-9 Move のパラメータ

パラメータ	説明
NumRecords	移動するカレント・レコードの位置をレコード数で指定する符号付 LONG 型の式。
Start	オプションです。STRING 型または VARIANT 型で、ブックマークに評価されます。次に説明する値のいずれかです。 <ul style="list-style-type: none"> AdBookmarkCurrent — 値は 0。これがデフォルトです。カレント・レコードから開始します。Start に値を渡さないことと、論理的に等価です。 AdBookmarkFirst — 値は 1。最初のレコードから開始します。 AdBookmarkLast — 値は 2。最後のレコードから開始します。

コメント

NumRecords 引数が正の値の場合、カレント・レコードの位置は順方向にレコードセットの最後に向かって移動します。NumRecords が負の値の場合、カレント・レコードの位置は逆方向にレコードセットの最初に向かって移動します。

Move コールでカレント・レコードの位置を最初のレコードより前に移動しようとする、ActiveX Data Objects for Windows CE はカレント・レコードを最初のレコードより前に移動し、BOF が TRUE になります。BOF プロパティがすでに TRUE のときにさらに前に移動しようすると、エラーになります。

Move コールでカレント・レコードの位置を最後のレコードより後に移動しようとする、ActiveX Data Objects for Windows CE はカレント・レコードを最後のレコードより後に移動し、EOF が TRUE になります。EOF プロパティがすでに TRUE のときにさらに後に移動しようすると、エラーになります。

空の Recordset オブジェクトで Move メソッドをコールすると、エラーになります。

Start 引数を渡すと、(Recordset オブジェクトがブックマークをサポートしている場合) このブックマークがあるレコードに対して相対的に移動します。指定されない場合は、カレント・レコードに対して相対的に移動します。

次のコード例は、Move メソッド用の定数を設定しています。

```
Option Explicit
'CursorType
Const adOpenKeyset = 1
'LockType
Const adLockOptimistic = 3
'Options
Const adCmdTable = 2
'Start
Const adBookmarkLast = 2

Sub Main()
Dim conn
Dim rstEmployees
Dim fldName
Dim avarRecords As Variant

Set conn = CreateObject("oladoce.Connection")
conn.Open "POLITE", "SYSTEM", "MANAGER"
Set rstEmployees = CreateObject("oladoce.recordset")
rstEmployees.Open "emp", conn, adOpenKeyset, adLockOptimistic, adCmdTable

rstEmployees.Move -3, adBookmarkLast

Set fldName = rstEmployees.Fields("ename")
MsgBox fldName.Value

rstEmployees.Close
conn.Close
Set fldName = Nothing
Set rstEmployees = Nothing
Set conn = Nothing
End Sub
```


4.6.7 MoveFirst、MoveLast、MoveNext、MovePrevious

これらのメソッドは、それぞれ、指定された **Recordset** オブジェクトの中で最初のレコード、最後のレコード、次のレコードまたは前のレコードに移動し、移動先のレコードをカレント・レコードにします。

構文

```
recordset.MoveFirst  
recordset.MoveLast  
recordset.MoveNext  
recordset.MovePrevious
```

コメント

MoveFirst メソッドは、カレント・レコードの位置をレコードセットの最初のレコードに移動するために使用します。

MoveLast メソッドは、カレント・レコードの位置をレコードセットの最後のレコードに移動するために使用します。

MoveNext メソッドは、カレント・レコードの位置をレコードセットの次のレコードに（最後に向かって順方向に 1 つ）移動するために使用します。最後のレコードがカレント・レコードで **MoveNext** メソッドをコールした場合、**ActiveX Data Objects for Windows CE** はカレント・レコードをレコードセットの最後のレコードより後に移動し、**EOF** が **TRUE** になります。**EOF** プロパティがすでに **TRUE** のときにさらに後に移動しようとすると、エラーになります。

MovePrevious メソッドは、カレント・レコードの位置をレコードセットの前のレコードに（最初に向かって逆方向に 1 つ）移動するために使用します。最初のレコードがカレント・レコードで **MovePrevious** メソッドをコールした場合、**ActiveX Data Objects for Windows CE** はカレント・レコードをレコードセットの最初のレコードより前に移動し、**BOF** が **TRUE** になります。**BOF** プロパティがすでに **TRUE** のときにさらに前に移動しようとすると、エラーになります。

4.6.8 Open

このメソッドはカーソルをオープンします。デフォルトは読み込み専用です。

構文

```
recordset.Open Source, ActiveConnection, CursorType, LockType, Options
```

Open メソッドのパラメータを表 4-10 に示します。

表 4-10 Open のパラメータ

パラメータ	説明
Source	必須です。VARIANT 型で、表の名前または SQL 文に評価されます。
ActiveConnection	有効な Connection オブジェクトに設定する必要があります。
CursorType	<p>オプションです。レコードセットで許可されている移動、および基礎となるデータベースに対する更新でレコードセットに反映されるものを判断します。内容は次のとおりです。</p> <ul style="list-style-type: none">■ AdOpenForwardOnly — 値は 0。デフォルトです。Forward-only カーソルです。静的カーソルと同じですが、レコードを順方向でしかスクロールできない点が異なります。デスクトップ・コンピュータの ActiveX Data Objects との互換性のためのものです。■ AdOpenKeyset — 値は 1。Keyset カーソルです。他のユーザーによる追加、変更および削除は参照できません。レコードセット内のすべての種類の移動が許可されます。 <p>他の値を使用すると、CursorType は 1（AdOpenKeyset）になります。動的カーソルおよび静的カーソルは、ActiveX Data Objects for Windows CE では使用できません。</p>
LockType	<p>オプションです。Recordset をオープンするときにプロバイダが使用する必要がある、ロック（並行性）のタイプを判断します。値は、次のとおりです。</p> <ul style="list-style-type: none">■ AdLockReadOnly — 値は 1。これがデフォルトです。レコードを追加、削除または変更することはできません。■ AdLockOptimistic — 値は 3。レコードを追加、削除および変更できます。 <p>他の値を使用すると、LockType は 3（AdLockOptimistic）になりますが、表自体が読み専用の場合は別です。その場合、LockType は 1（AdLockReadOnly）になります。デスクトップ・コンピュータの ActiveX Data Objects との互換性のために、デフォルトの LockType は 1（AdLockReadOnly）になっています。即時バッチ・ロックおよび最適バッチ・ロックは ActiveX Data Objects for Windows CE では使用できません。</p>

表 4-10 Open のパラメータ (続き)

パラメータ	説明
Options	Options の内容は次のとおりです。 <ul style="list-style-type: none"> ■ AdCmdText — 値は 1。Source を SQL 文として評価します。 ■ AdCmdTable — 値は 2。Source を MSysTables の表の名前として評価します。 ■ AdCmdStoredProc — 値は 4。Source を MSysProcs のストアード・プロシージャとして評価します。 ■ AdCmdUnknown — 値は 8。これがデフォルトです。Source プロパティ内のコマンドのタイプは不明です。

コメント

ActiveX Data Objects for Windows CE には Command オブジェクトはありません。したがって、Source 引数はオブジェクトである必要があります。

使用しているコマンドのタイプがわかっている場合、Options 引数を設定すると ActiveX Data Objects for Windows CE が関連コードに直接移動します。Options 引数が Source 引数のコマンドのタイプと一致しない場合、Open メソッドをコールしたときにエラーになります。

CREATE TABLE のような行を返さない SQL コマンドを設定して Open メソッドがコールされると、レコードセットは返されません。また、レコードセットの状態はクローズのままです。

ActiveX Data Objects for Windows CE は、オープンする各レコードセットから参照されるすべてのデータベース行へのポインタを保持するキーセットを生成します。マルチユーザーの場合は、基礎となるデータベースの変更によってこのキーセットが動的に更新されないの、レコードセットを最初にオープンした後に他のアプリケーションによって削除されたレコードへのポインタを保持し続ける可能性があります。これらの削除済レコードにアクセスしようすると、エラーになります。現在オープンしているレコードセットを使用してデータベースに行が追加されると、そのレコードセットに対するキーセットも更新されます。

マルチユーザーの場合、別のプログラムによってデータベースに行が追加されたときに、そのデータベースを参照している他のレコードセットに対するキーセットは動的に更新されないため、別のプログラムによって追加された行は参照できません。キーセットが認識している行を別のアプリケーションが変更した場合は、その行に対するポインタは有効であるため参照できます。新規のキーセットを生成して別のプログラムによる追加と削除を参照するには、レコードセットをクローズして再オープンします。複製されたレコードセットは共通のキーセットを共有します。

例

次のコード例は、Open メソッド用の `CursorType` 定数と `LockType` 定数を設定しています。

```
Option Explicit
'CursorType
Const adOpenForwardOnly = 0
Const adOpenKeyset = 1
'LockType
Const adLockReadOnly = 1
Const adLockOptimistic = 3
'Options
Const adCmdTable = 2

Sub Main()
Dim conn
Dim rstLocked, rstUpdateable

Set conn = CreateObject("oladoce.Connection")
conn.Open "POLITE", "SYSTEM", "MANAGER"
Set rstLocked = CreateObject("oladoce.recordset")
Set rstUpdateable = CreateObject("oladoce.recordset")
'The default is to open a read-only, forward-only recordset
rstLocked.Open "emp", conn
'You must specify other parameters to make a recordset updateable
rstUpdateable.Open "customer", conn, adOpenKeyset, adLockOptimistic

rstLocked.Close
rstUpdateable.Close
conn.Close
Set rstLocked = Nothing
Set rstUpdateable = Nothing
Set conn = Nothing
End Sub
```

4.6.9 Supports

指定された Recordset オブジェクトが特定のタイプの機能をサポートするかどうかを判断します。

構文

```
Boolean = recordset.Supports(CursorOptions)
```

`Supports` メソッドのパラメータを[表 4-11](#)に示します。

表 4-11 Supports のパラメータ

パラメータ	説明
CursorOptions	<p>LONG 型の式。次の値のいずれか 1 つまたは値の組合せです。</p> <ul style="list-style-type: none"> ■ adAddNew — 値は 16778240。AddNew メソッドをサポートします。 ■ AdApproxPosition — 値は 16384。AbsolutePosition プロパティおよび AbsolutePage プロパティをサポートします。 ■ AdBookmark — 値は 8192。Bookmark プロパティをサポートします。 ■ AdDelete — 値は 16779264。Delete メソッドをサポートします。 ■ AdMovePrevious — 値は 512。MovePrevious メソッドと、Move メソッドによるカレント・レコードの位置の逆方向移動をサポートします。 ■ AdUpdate — 値は 16809984。Update メソッドをサポートします。 ■ AdHoldRecords — 値は 256。サポートされません。 ■ AdResync — 値は 131072。サポートされません。 ■ AdUpdateBatch — 値は 65536。サポートされません。

コメント

Supports メソッドは、Recordset オブジェクトがサポートする機能のタイプを判断するために使用します。*CursorOptions* に含まれている定数に対応する機能を Recordset オブジェクトがサポートする場合、Supports メソッドは TRUE を返します。それ以外の場合は、FALSE を返します。

定数 AdHoldRecords、AdResync および AdUpdateBatch の場合、ActiveX Data Objects for Windows CE は常に FALSE を返します。他の定数に対しては、TRUE を返す可能性があります。

例

次のコード例は、Supports メソッド用の定数を設定しています。

```
Option Explicit
'CursorType
Const adOpenForwardOnly = 0
Const adOpenKeyset = 1
'LockType
Const adLockReadOnly = 1
Const adLockOptimistic = 3
```

```
'Options
Const adCmdTable = 2
'CursorOptions
Const adAddNew = 16778240
Const adApproxPosition = 16384
Const adBookMark = 8192
Const adDelete = 16779264
Const adHoldRecords = 256
Const adMovePrevious = 512
Const adResync = 131072
Const adUpdate = 16809984
Const adUpdateBatch = 65536

Sub Main()
Dim conn
Dim rs
Dim supported As Boolean

Set conn = CreateObject("oladoce.Connection")
conn.Open "POLITE", "SYSTEM", "MANAGER"
Set rs = CreateObject("oladoce.recordset")
rs.Open "customer", conn, adOpenKeyset, adLockOptimistic

If rs.Supports(adAddNew) = False Then 'Supported
    MsgBox "adAddNew Failed"
ElseIf rs.Supports(adApproxPosition) = True Then 'Not supported
    MsgBox "adApproxPosition Failed"
ElseIf rs.Supports(adBookMark) = True Then 'Not supported
    MsgBox "adBookMark Failed"
ElseIf rs.Supports(adDelete) = False Then 'supported
    MsgBox "adDelete Failed"
ElseIf rs.Supports(adHoldRecords) = True Then 'Not supported
    MsgBox "adHoldRecords Failed"
ElseIf rs.Supports(adMovePrevious) = False Then 'supported
    MsgBox "adMovePrevious Failed"
ElseIf rs.Supports(adResync) = True Then 'Not supported
    MsgBox "adResync Failed"
ElseIf rs.Supports(adUpdate) = False Then 'supported
    MsgBox "adUpdate Failed"
ElseIf rs.Supports(adUpdateBatch) = True Then 'Not supported
    MsgBox "adUpdateBatch Failed"
Else
    MsgBox "Passed"
End If

rs.Close
conn.Close
```

```
Set rs = Nothing
Set conn = Nothing
End Sub
```

4.6.10 Update

Recordset オブジェクトのカレント・レコードに対するすべての変更を保存します。
Recordset.Open メソッドから直接 SQL 文を実行する場合は、このメソッドは不要です。

構文

recordset.Update Fields, Values

Update メソッドのパラメータを表 4-12 に示します。

表 4-12 Update のパラメータ

パラメータ	説明
Fields	オプションです。変更するフィールドの単一フィールド名、または複数のフィールド名やフィールドの位置を示す順序数を表す配列です。
Values	オプションです。新規レコードのフィールドの単一の値、または複数の値を表す配列です。

コメント

Update メソッドは、AddNew メソッドのコール後、または既存レコードのフィールド値の変更後の、Recordset オブジェクトのカレント・レコードに対するすべての変更を保存するために使用します。Recordset オブジェクトは更新をサポートする必要があります。フィールドの値を設定するには、次のいずれかの方法を実行します。

- Field オブジェクトの Value プロパティに値を割り当てて Update メソッドをコールします。
- フィールド名と値を Update コールの引数として渡します。
- フィールド名の配列と値の配列を Update コールに渡します。

フィールド名の配列と値の配列を使用する場合は、両方の要素数が同一である必要があります。また、フィールド名の順序はフィールド値の順序と一致する必要があります。フィールド名と値の数および順序が一致しない場合、エラーになります。

Update メソッドをコールする前に追加または編集中のレコードから移動すると、ActiveX Data Objects for Windows CE は自動的に Update をコールして変更を保存します。カレント・レコードに対するすべての変更を取り消す、または新規に追加したレコードを廃棄する場合は、CancelUpdate メソッドをコールする必要があります。

カレント・レコードは、Update メソッドのコール後もカレントのままです。

例

```
Option Explicit
'CursorType
Const adOpenKeyset = 1
'LockType
Const adLockOptimistic = 3
'Options
Const adCmdText = 1
Const adCmdTable = 2

Sub Main()
Dim conn
Dim rs, fldName
Set conn = CreateObject("oladoce.connection")
conn.Open "POLITE", "SYSTEM", "MANAGER"
conn.BeginTrans
Set rs = CreateObject("oladoce.recordset")
rs.Open "drop table newtable", conn, adOpenKeyset, adLockOptimistic, adCmdText
rs.Open "create table newtable (f1 varchar(3))", conn, adOpenKeyset,
    adLockOptimistic, adCmdText
rs.Open "newtable", conn, adOpenKeyset, adLockOptimistic, adCmdTable
rs.AddNew "f1", "a"
rs.Close
conn.CommitTrans
rs.Open "newtable", conn, adOpenKeyset, adLockOptimistic, adCmdTable
rs.Update "f1", "b"
Set fldName = rs.Fields("f1")
MsgBox fldName.Value
rs.Close
conn.Close
Set fldName = Nothing
Set rs = Nothing
Set conn = Nothing
End Sub
```


4.7 Recordset オブジェクトのプロパティ

Recordset オブジェクトがサポートするプロパティを表 4-13 に示します。

表 4-13 Recordset オブジェクトのプロパティ

プロパティ	説明
AbsolutePage***	新規のカレント・レコードに対して移動するページを指定します。
AbsolutePosition	Recordset オブジェクト内でのカレント・レコードの位置を示す順序数を指定します。
ActiveConnection***	現在のデータベース接続を設定します。有効およびアクティブな Connection オブジェクトに設定する必要があります。
BOF	カレント・レコードの位置が Recordset オブジェクトの最初のレコードより前かどうかを示します。
EOF	カレント・レコードの位置が Recordset オブジェクトの最後のレコードより後であることを示します。
Bookmark***	Recordset オブジェクト内でレコードを一意に識別するブックマークを指定します。
CacheSize***	Recordset オブジェクトからローカルのメモリーにキャッシュされるレコード数を指定します。
CursorType	Recordset オブジェクト内で使用するカーソルのタイプを示します。
EditMode	カレント・レコードの編集ステータスを示します。
LockType	編集中のレコードに対するロックのタイプを示します。
PageCount***	Recordset オブジェクトに含まれるデータのページ数を示します。
PageSize***	Recordset の 1 ページを構成するレコード数を示します。
RecordCount	Recordset オブジェクトのカレント・レコード数を示す LONG 型の値を返します。
Source	Recordset オブジェクト、SQL 文または表の名前に含まれるデータのソースを示します。

*** は未実装を示します。

レコードセットは仮想データベース表で、そのフィールドおよび行は H/PC の実際のデータベース表のフィールドおよび行のサブセットと対応しています。レコードセットの行に含まれる情報を追加、削除または変更すると、基礎となる表の対応している部分にその変更内容を渡せます。レコードセットのデータを変更するとその変更内容はメモリーに格納されるので、基礎となるデータベースの更新前に変更を取り消すことができます。ActiveX Data Objects for Windows CE では、バッチ更新をサポートしていません。データが変更されても基礎となるデータベースがコミットされていない行は、同時に 1 つしか存在できません。

レコードセット表の構造を変更すると、基礎となるデータベース表に即時に反映されます。

4.7.1 AbsolutePosition

Recordset オブジェクト内でのカレント・レコードの位置を示す順序数を指定します。

構文

```
var = recordset.AbsolutePosition
```

戻り値

1 から Recordset オブジェクトに含まれるレコード数 (RecordCount) までの LONG 型の値を返します。

コメント

AbsolutePosition プロパティは、Recordset オブジェクト内でのレコードの位置を示す順序数を基にして移動するために使用します。

AbsolutePage プロパティと同様に、AbsolutePosition は 1 から数えられ、カレント・レコードが Recordset の最初のレコードの場合に 1 になります。RecordCount プロパティから、Recordset オブジェクトに含まれるレコード数の合計を取得できます。

AbsolutePosition プロパティを、レコード番号の代用として使用しないでください。指定されたレコードの位置は、先行するレコードを削除したときに変更されます。また、Recordset オブジェクトが再オープンされた場合に、指定されたレコードが同じ AbsolutePosition を持つという保証はありません。指定した位置を保持してそこに戻する方法としては、ブックマークをお勧めします。これは、すべてのタイプの Recordset オブジェクトで通用する唯一の方法です。

AbsolutePosition は 3 つの特殊な値、-1 (AdPosUnknown)、-2 (AdPosBOF) および -3 (AdPosEOF) をサポートします。

AbsolutePosition に 0 または負の値を設定すると、エラーになります。

4.7.2 BOF、EOF

Recordset オブジェクトで、BOF はカレント・レコードの位置が最初のレコードより前であることを示します。EOF はカレント・レコードの位置が最後のレコードより後であることを示します。

構文

```
recordset.BOF  
recordset.EOF
```

戻り値

BOF プロパティおよび EOF プロパティの戻り値はブール値です。

BOF プロパティは、カレント・レコードの位置が最初のレコードより前にあるときに TRUE を返し、カレント・レコードの位置が最初のレコードまたはそれより後にあるときは FALSE を返します。

EOF プロパティは、カレント・レコードの位置が最後のレコードより後にあるときに TRUE を返し、カレント・レコードの位置が最後のレコードまたはその前にあるときは FALSE を返します。

BOF プロパティまたは EOF プロパティが TRUE であるとき、カレント・レコードはありません。

コメント

BOF プロパティおよび EOF プロパティは、Recordset オブジェクトにレコードが含まれているか、あるいはレコードからレコードへの移動時に Recordset オブジェクトの外側に出たかどうかを判断するために使用します。

レコードがない Recordset オブジェクトをオープンした場合、BOF および EOF の両プロパティが TRUE に設定され、Recordset オブジェクトの RecordCount プロパティは 0 に設定されます。少なくともレコードを 1 つ含んでいる Recordset オブジェクトをオープンすると、最初のレコードがカレント・レコードになり、BOF プロパティおよび EOF プロパティは FALSE になります。

Delete メソッドをコールすると、Recordset の最後のレコードを削除した場合でも BOF プロパティまたは EOF プロパティの設定は変更されません。

表 4-14 に、様々な Move メソッドをコールして正しくレコードを再配置できなかった場合、BOF プロパティおよび EOF プロパティがどのように設定されるかを示します。

表 4-14 BOF および EOF のパラメータ

メソッド	BOF	EOF
MoveFirst、MoveLast	TRUE	TRUE
Move=0	変更なし	変更なし
MovePrevious、Move<0	TRUE	変更なし
MoveNext、Move>0	変更なし	TRUE

4.7.3 CursorType

Recordset オブジェクト内で使用するカーソルのタイプを示します。

構文

`recordset.CursorType`

戻り値

レコードセットで許可されている移動、および基礎となるデータベースに対する更新でレコードセットに反映されるものを示します。0 (AdOpenForwardOnly) または 1 (AdOpenKeyset) のどちらかです。

コメント

CursorType プロパティは、Recordset オブジェクトをオープンするときに使用するカーソルのタイプを指定するために使用します。CursorType プロパティは、レコードセットがクローズしている場合は読み込み / 書き込み可能、オープンしている場合は読み込み専用です。

Recordset オブジェクトがオープンしているときは、使用している実際のカーソルのタイプに一致するように CursorType プロパティが変更される可能性があります。返されたカーソルの特定の機能を検証するには、Supports メソッドを使用します。

ActiveX Data Objects for Windows CE が実装しているのは AdOpenForwardOnly および AdOpenKeyset のみなので、その他の値はすべて AdOpenKeyset にマップされます。

4.7.4 EditMode

カレント・レコードの編集ステータスを示します。

構文

`object.EditMode`

戻り値

表 4-15 に示す値の 1 つが返されます。

表 4-15 編集モードの戻り値

EditMode	値	説明
AdEditNone	0	進行中の編集作業はありません。
AdEditInProgress	1	カレント・レコードのデータが変更され、まだ保存されていません。
AdEditAdd	2	AddNew メソッドが起動されていて、コピー・バッファにあるカレント・レコードはまだデータベースに保存されていない新規レコードです。

コメント

EditMode プロパティは、カレント・レコードの編集ステータスを判断するために使用します。編集プロセスが中断されて Update メソッドまたは CancelUpdate メソッドを実行する必要があるかどうかを判断する場合に、保留中の変更があるかどうかをテストできます。

異なる編集条件での EditMode プロパティの詳細は、AddNew メソッドを参照してください。

4.7.5 LockType

編集中のレコードに対するロックのタイプを示します。

構文

```
recordset.LockType
```

戻り値

表 4-16 に示す値の 1 つが返されます。

表 4-16 ロック・モードの戻り値

LockType	値	説明
AdLockReadOnly	1	デフォルトです。レコードを追加、削除または変更することはできません。
AdLockOptimistic	3	レコードを追加、削除および変更できます。

コメント

LockType プロパティは、Recordset オブジェクトをオープンするときにプロバイダが使用する必要があるロックのタイプを判断するため、または Recordset オブジェクトをオープンしたときに使用したロックのタイプを返すために使用します。LockType プロパティは、Recordset がクローズしている場合は読み込み / 書き込み可能、オープンしている場合は読み込み専用です。

ActiveX Data Objects for Windows CE は AdLockPessimistic または AdLockBatchOptimistic をサポートしません。通告なしで AdLockOptimistic に置き換えます。

4.7.6 RecordCount

Recordset オブジェクト内の現在のレコード数を示します。

構文

```
var = recordset.RecordCount
```

戻り値

LONG 型の値を返します。

コメント

RecordCount プロパティは、Recordset オブジェクトに含まれるレコード数を判断するために使用します。ActiveX Data Objects for Windows CE がレコード数を判断できないときは、-1 (AdUnknown) を返します。

クローズしている Recordset オブジェクトの RecordCount プロパティを読み込もうとすると、エラーになります。

4.7.7 Source

Recordset オブジェクト、SQL 文または表の名前に含まれるデータのソースを示します。

構文

```
var = recordset.Source
```

戻り値

STRING 型の値を返します。

コメント

Source プロパティは、Recordset オブジェクトのデータ・ソースを指定するために使用します。

Source 引数は Open 文で必須であり、ActiveX Data Objects for Windows CE には Execute メソッドと Command オブジェクトがないので、このプロパティを設定しても影響がありません。

4.8 Field オブジェクトのプロパティ

Field オブジェクトでサポートされるプロパティを表 4-17 に示します。

表 4-17 Field オブジェクトのプロパティ

メソッド	説明
ActualSize	フィールド値の実際の長さを、バイト数で示します。
Attributes	Field オブジェクトの 1 つまたは複数の特性を示す値を返します。このプロパティは、読み込み専用です。
DefinedSize	Field オブジェクトのデータ容量を判断するために使用します。定義されたフィールドのサイズを、文字数で返します。サイズをバイト数で返す ActualSize と比較してください。
Name	フィールドの名前を返します。このプロパティは、読み込み専用です。
Type	Field オブジェクトのデータ型を示します。 Type プロパティは、読み込み専用です。
UnderlyingValue	データベース内の、Field オブジェクトの現在の値を示します。
Value (デフォルト)	レコードセット内の、Field オブジェクトの現在の値を示します。

備考

Field オブジェクトにはメソッドもイベントもありません。**Value** を除いてすべてのプロパティが読み込み専用です。次のコード例は、Field オブジェクトの使用方法を示しています。

```
Option Explicit
'CursorType
Const adOpenKeyset = 1
'LockType
Const adLockOptimistic = 3
'Options
Const adCmdText = 1
Const adCmdTable = 2

Sub Main()
On Error Resume Next
Dim conn
Dim rs, fld

Set conn = CreateObject("oladoce.connection")
conn.Open "POLITE", "SYSTEM", "MANAGER"
Set rs = CreateObject("oladoce.recordset")
rs.Open "emp", conn, adOpenKeyset, adLockOptimistic, adCmdTable
```

```
Set fld = rs.Fields(2)
MsgBox fld.Value, , fld.Name

rs.Close
conn.Close
Set fld = Nothing
Set rs = Nothing
Set conn = Nothing
End Sub
```

4.8.1 ActualSize

フィールド値の実際の長さを、バイト数で示します。

構文

```
var = field.ActualSize
```

戻り値

LONG 型の値を返します。

コメント

`ActualSize` は、Field オブジェクトの値の実際の長さをバイト数で返すために使用します。すべてのフィールドに対して、`ActualSize` プロパティは読み込み専用です。ActiveX Data Objects for Windows CE がフィールド値の長さを判断できない場合、`ActualSize` プロパティは -1 (`AdUnknown`) を返します。

文字データ型では、最大許容サイズを判断するには `DefinedSize` を使用の方が便利な場合があります。

4.8.2 Attributes

Field オブジェクトの 1 つまたは複数の特性を示します。このプロパティは、読み込み専用です。

構文

```
var = field.Attribute
```

戻り値

Field オブジェクトでは、この値がフィールドの特性を示します。[表 4-18](#) に示す 1 つまたは複数の値の合計をとることができます。

表 4-18 Attributes の戻り値

戻り値	値	説明
adFldMayDefer	2	フィールドが遅延処理され、明示的にアクセスしたときを除いて、フィールドの値がレコード全体とともにデータ・ソースから取り出されるわけではないことを示します。
AdFldUpdatable	4	フィールドに書き込めることを示します。
adFldUnknownUpdatable	8	フィールドに書き込めるかどうか、プロバイダが判断できないことを示します。
AdFldFixed	16	フィールドに固定長データが含まれていることを示します。すべてのデータ型で使用できますが、AdVarWChar、AdLongVarWChar、AdVarBinary および AdLongVarbinary では使用できません。
AdFldIsNullable	32	フィールドが NULL 値を受け入れることを示します。
AdFldMayBeNull	64	フィールドから NULL 値を読み込めることを示します。
adFldLong	128	フィールドがロング・バイナリ型のフィールドであることを示します。または、AppendChunk メソッドおよび GetChunk メソッドを使用できることを示します。
adFldRowID	256	フィールドにレコード番号、一意の識別子などのレコード識別子が含まれていることを示します。

ActiveX Data Objects for Windows CE では、AdFldRowVersion または AdFldCacheDeferred の値を返しません。

例

次のコード例は、Attributes プロパティ用の定数を設定しています。

```
Option Explicit
'CursorType
Const adOpenKeyset = 1
'LockType
Const adLockOptimistic = 3
'Options
Const adCmdText = 1
Const adCmdTable = 2
'Attributes return values
Const adFldMayDefer = 2
Const adFldUpdatable = 4
Const adFldUnknownUpdatable = 8
```

```
Const adFldFixed = 16
Const adFldIsNullable = 32
Const adFldMayBeNull = 64
Const adFldLong = 128
Const adFldRowID = 256

Sub Main()
On Error Resume Next
Dim conn
Dim rs, n

Set conn = CreateObject("oladoce.connection")
conn.Open "POLITE", "SYSTEM", "MANAGER"
Set rs = CreateObject("oladoce.recordset")
rs.Open "emp", conn, adOpenKeyset, adLockOptimistic, adCmdTable
For n = 0 To rs.Fields.Count - 1
    MsgBox rs.Fields(n).Attributes
Next

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
End Sub
```

4.8.3 DefinedSize

Field オブジェクトのデータ容量を判断するために使用します。定義されたフィールドのサイズを、文字数で返します。サイズをバイト数で返す **ActualSize** と比較できます。

構文

```
var = field.DefinedSize
```

戻り値

フィールドの最大長を指定する数を返します。

コメント

ActiveX Data Objects for Windows CE は、256 文字より少ない文字列と最大 32,733 の Unicode 文字をサポートできるメモ文字列の、2 種類の文字列型をサポートします。長さは表が作成されたときに設定され、データが設定されたときに施行されます。256 文字より少ない文字列の場合、DefinedSize は表が作成されたときに指定された長さを返します。長さが定義されていないテキスト・フィールドの場合、DefinedSize はフィールドに保持できる最大 Unicode 文字数である 32,733 を返します。

DefinedSize プロパティと ActualSize プロパティは異なります。たとえば、データ型 202 (AdVarChar) で最大長 50 文字として宣言された Field オブジェクトの場合に返される DefinedSize プロパティの値は 50 ですが、ActualSize プロパティの値にはカレント・レコードのそのフィールドに格納されているデータのバイト数が返されます。AdVarChar は 1 文字当たり 2 バイトを使用するため、定義されたサイズ (文字数) より格納データのバイト数の方が大きい場合もあります。

4.8.4 Name

フィールドの名前を返します。このプロパティは、読み専用です。

構文

```
var = field.Name
```

戻り値

名前を指定する STRING 型の値を返します。

コメント

フィールド名は 64 文字以下です。

例

```
Option Explicit
'CursorType
Const adOpenKeyset = 1
'LockType
Const adLockOptimistic = 3
'Options
Const adCmdText = 1
Const adCmdTable = 2

Sub Main()
On Error Resume Next
Dim conn
Dim rs, n

Set conn = CreateObject("oladoce.connection")
conn.Open "POLITE", "SYSTEM", "MANAGER"
Set rs = CreateObject("oladoce.recordset")
rs.Open "emp", conn, adOpenKeyset, adLockOptimistic, adCmdTable
For n = 0 To rs.Fields.Count - 1
    MsgBox rs.Fields(n).Name
Next

rs.Close
```

```
conn.Close
Set rs = Nothing
Set conn = Nothing
End Sub
```

4.8.5 Type

Field オブジェクトのデータ型を示します。このプロパティは、読み込み専用です。

構文

```
var = field.Type
```

戻り値

表 4-19 に示す値の 1 つです。

表 4-19 Type の戻り値

定数	値	説明
AdVarChar	202	256 文字より少ない、NULL で終了する Unicode 文字の文字列。
AdLongVarChar	203	NULL で終了する Unicode 文字の文字列。
AdVarBinary	204	256 文字より少ないバイナリ値。
AdLongVarBinary	205	65533 (4096*16 - 3) バイト以下のバイナリ値。
AdInteger	3	4 バイトの符号付き整数。
AdSmallInt	2	2 バイトの符号付き整数。
AdDouble	5	倍精度浮動小数点値。
AdDate	7	DATE 型の値。
AdUnsignedSmallInt	18	2 バイトの符号なし整数。
AdUnsignedInt	19	4 バイトの符号なし整数。
AdBoolean	11	ブール型の TRUE/FALSE 値。
AdDouble	5	倍精度浮動小数点数。

デスクトップ・コンピュータの ActiveX Data Objects では Type に他の値も使用できますが、ActiveX Data Objects for Windows CE ではサポートされず戻り値としても使用されません。

例

次のコード例は、Type プロパティ用の定数を設定しています。

```
Option Explicit
'CursorType
Const adOpenKeyset = 1
'LockType
Const adLockOptimistic = 3
'Options
Const adCmdText = 1
Const adCmdTable = 2
'Type return values
Const adVarChar = 202
Const adLongVarChar = 203
Const adVarBinary = 204
Const adLongVarBinary = 205
Const adInteger = 3
Const adSmallInt = 2
Const adDouble = 5
Const adDate = 7
Const adUnsignedSmallInt = 18
Const adUnsignedInt = 19
Const adBoolean = 11

Sub Main()
On Error Resume Next
Dim conn
Dim rs, n

Set conn = CreateObject("oladoce.connection")
conn.Open "POLITE", "SYSTEM", "MANAGER"
Set rs = CreateObject("oladoce.recordset")
rs.Open "emp", conn, adOpenKeyset, adLockOptimistic, adCmdTable
For n = 0 To rs.Fields.Count - 1
    MsgBox rs.Fields(n).Type
Next

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
End Sub
```

4.8.6 UnderlyingValue

データベース内の、Field オブジェクトの現在の値を示します。

構文

```
var = field.UnderlyingValue
```

戻り値

VARIANT 型の値を返します。

コメント

UnderlyingValue プロパティは、データベースからフィールドの現在の値を返すために使用します。

4.8.7 Value

Field オブジェクトに割り当てられた値を示します。

構文

```
var = field.Value
```

戻り値

VARIANT 型の値を返します。デフォルト値は、Type プロパティに依存します。

コメント

Value プロパティは、Field オブジェクトにデータを設定するかデータを返すために使用します。

ActiveX Data Objects for Windows CE では、Value プロパティでロング・バイナリ値を設定または返せます。

4.9 Fields コレクション

Fields コレクションには、Recordset の各列の Field オブジェクトが 1 つずつ含まれます。

備考

名前または索引によって、特定のフィールドを参照できます。Fields コレクションは Count プロパティをサポートします。

例

次のコード例に、Fields コレクションを使用して、newtable 表からすべてのフィールド名を取得する方法を示します。

```
Option Explicit
'CursorType
Const adOpenKeyset = 1
'LockType
Const adLockOptimistic = 3
'Options
Const adCmdText = 1
Const adCmdTable = 2

Sub Main()
    Dim conn, rs, n
    Dim myStr As String

    Set conn = CreateObject("oladoce.connection")
    conn.Open "polite", "system", "manager"

    Set rs = CreateObject("oladoce.recordset")
    rs.Open "create table newtable(column1 int, column2 varchar(10))", conn,
        adOpenKeyset, adLockOptimistic, adCmdText
    Err.Clear
    rs.Close

    rs.Open "newtable", conn, adOpenKeyset, adLockOptimistic, adCmdTable
    For n = 0 To rs.Fields.Count - 1
        myStr = myStr + rs.Fields(n).Name
        myStr = myStr + " "
    Next
    MsgBox myStr

    rs.Close
    conn.Close
    Set rs = Nothing
    Set conn = Nothing
End Sub
```

4.10 SQL およびデータベースの参照

ActiveX Data Objects for Windows CE は Oracle Lite database for Windows CE の SQL 構文をすべてサポートしますが、次の 2 つはサポートしません。

- 文の最後のセミコロン。
- COMMIT や ROLLBACK などの単一語で構成された文。

Oracle Lite データベースの ActiveX Data Objects for Windows CE ではサポートされていない ActiveX Data Objects for Windows CE システム表を表 4-20 に示します。

表 4-20 OLADOCE ではサポートされていない Microsoft ADOCE システム

表	説明
MsysTables	ActiveX Data Objects for Windows CE が認識するすべての表とその特性。
MsysIndexes	ActiveX Data Objects for Windows CE が認識するすべての表に対する索引。
MsysFields	ActiveX Data Objects for Windows CE が認識するすべての表のフィールド。
MsysProcs	名前で行えるストアード SQL 文。

4.11 エラー・メッセージ

ActiveX Data Objects for Windows CE は、ActiveX Data Objects エラーと SQL エラーを生成します。ActiveX Data Objects エラーは、コントロールのメソッドおよびプロパティが適切に使用されなかったときに発生します。SQL エラーは、Open メソッドの中で使用された SQL 文に問題があるときに生成されます。

エラーには、番号と関連するエラー・テキストがあります。テキストは、Err オブジェクトの Description プロパティを使用して取り出されます。Helpfile 情報または HelpContext 情報はありません。

エラーは、インライン・エラー・トラップで On Error Resume Next 文を使用して獲得されます。インラインでエラーをトラップするには、操作の完了後に Err オブジェクトをチェックして、Err.Number が 0 以外の値かどうかを確認します。Err.Number に返されるエラー・コードを使用すると、それが SQL エラーであっても ActiveX Data Objects エラーであっても適切に処理できます。

4.11.1 ActiveX Data Objects のエラー

表 4-21 に ActiveX Data Objects のエラー値を示します。

表 4-21 ADOCE のエラー

エラー	値	説明
AdErrInvalidArgument	3001	型が誤っている、許容域を超えている、または互いに競合している引数をアプリケーションが使用しています。
adErrNoCurrentRecord	3021	BOF または EOF が TRUE であるか、カレント・レコードが削除されています。アプリケーションが要求している操作には、カレント・レコードが必要です。
adErrIllegalOperation	3219	アプリケーションが要求している操作は、このコンテキストではできません。
adErrFeatureNotAvailable	3251	アプリケーションが要求している操作は、プロバイダによってサポートされていません。
adErrItemNotFound	3265	ActiveX Data Objects は、アプリケーションが要求した名前または序数による参照に対応するオブジェクトを、コレクションの中で検索できませんでした。
adErrObjectNotSet	3420	アプリケーションによるオブジェクト参照が、有効なオブジェクトを指していません。
adErrDataConversion	3421	アプリケーションが現在の操作に使用している値の型が誤っています。
AdErrObjectClosed	3704	アプリケーションが要求している操作は、オブジェクトがクローズしているときは実行できません。
AdErrObjectOpen	3705	アプリケーションが要求している操作は、オブジェクトがオープンしているときは実行できません。
adErrProviderNotFound	3706	ActiveX Data Objects は、指定されたプロバイダを見つけられませんでした。
adErrInvalidConnection	3709	指定した接続文字列が無効です。

4.11.2 SQL エラー

表示される SQL エラーの種類を表 4-22 に示します。表に示したエラー値は 16 進表現で、コード中では Hex(Err.Number) を使用して表示できます。Oracle Lite のネイティブ・エラー・コードは、Err オブジェクトの Description プロパティから表示できます。

Oracle Lite データベースの ActiveX Data Objects for Windows CE は、ActiveX Data Objects for Windows CE で発生する可能性があるすべての SQL エラーをネイティブ・エラーにマップします。

表 4-23 に、SQL エラー（表 4-22 を参照）と Oracle Lite 内部のネイティブ・エラーの直接マッピングを示します。

表 4-22 SQL エラー

エラー	値	説明	説明
DB_E_CANTCONVERTVALUE	80040E07	型 "constant" を変換できません。	コマンド・テキスト中のリテラル値が、(データ・オーバーフロー以外の理由で) 関連する列の型に変換できませんでした。エラー文字列には問題の定数が含まれています。
DB_E_DATAOVERFLOW	80040E57	定数値 "constant" がオーバーフローしました。	コマンド・テキスト中のリテラル値が、関連する列によって指定されている型をオーバーフローしました。エラー文字列には問題の定数が含まれています。
DB_E_ERRORSINCOMMAND	80040E14	"token" の近くに不正な構文があります。	コマンド・テキストに 1 つまたは複数のエラーがありました。一般には、構文エラーまたは予想外のキーワードです。エラー文字列には予想外のトークンが含まれています。
E_OUTOFMEMORY	8007000E	メモリー不足です。	メモリー不足です。
DB_E_NOTABLE	80040E37	表 "table" が存在しません。	指定された表が存在しません。エラー文字列にはエラーになった表の名前が含まれています。
DB_E_BADCOLUMNID	80040E11	"field" フィールドが存在しません。	指定された列が存在しませんでした。エラー文字列にはエラーになったフィールドの名前が含まれています。

表 4-22 SQL エラー（続き）

エラー	値	説明	説明
DB_E_DUPLICATETABLEID	80040E3F	表 "table" はすでに存在します。	指定された表は、すでに現在のデータ・ソースに存在しています。
DB_E_DUPLICATEINDEXID	80040E34	索引 "index" はすでに存在します。	指定された索引は、すでに現在のデータ・ソース・オブジェクトに存在しています。
DB_E_NOINDEX	80040E35	索引 "index" が存在しません。	指定された索引が現在のデータ・ソースに存在しないか、指定された表に適用されませんでした。
DB_E_DUPLICATECOLUMNID	80040E3E	"field" フィールドはすでに存在します。	複数の要素のフィールド名が同じでした。
DB_E_NOCOMMAND	80040E0C	未対応	コマンドが設定されていません。
DB_E_BADBOOKMARK	80040E0E	未対応	ブックマークが無効です。
DB_E_DELETEDROW	80040E23	未対応	行が削除されています。
DB_E_CANTFETCHBACKWARDS	80040E24	未対応	forward-only カーソルは逆方向には読み込めません。
DB_E_FIELDDIFFERENT	80040E41	"%1s" の近くに無効なフィールド比較があります。	型が異なる 2 つのフィールドを比較しようとしてしました。
DB_E_FIELDMAXEXCEED	80040E42	表の "%1s" の近くで最大列数を超過しました。	表当たりの最大数を超える列を使用しようとしてしました。

表 4-23 Oracle Lite 内部のネイティブ・エラーへの SQL エラーのマッピング

SQL エラー	Oracle Lite のネイティブ・エラー・コード
DB_E_CANTCONVERTVALUE	POL-2404: キャスト変換に無効な文字が指定されています
DB_E_DATAOVERFLOW	POL-2401: 数値が有効範囲ではありません POL-2403: 列の値が大きすぎます
DB_E_ERRORSINCOMMAND	POL-5228: 構文エラーです

表 4-23 Oracle Lite 内部のネイティブ・エラーへの SQL エラーのマッピング (続き)

SQL エラー	Oracle Lite のネイティブ・エラー・コード
E_OUTOFMEMORY	POL-2000: これ以上メモリーを割り当てられません POL-2001: 要求した配列サイズが大きすぎます POL-2202: 共有メモリー領域が不足しています POL-2408: メモリー不足です POL-3005: ヒープ・メモリーが不足しています POL-5105: メモリー割当てに失敗しました POL-6057: メモリー割当てに失敗しました
DB_E_NOTABLE	POL-5130: 表またはビューが見つかりません
DB_E_BADCOLUMNID	POL-5205: 列が見つかりません
DB_E_DUPLICATETABLEID	POL-5128: 表名が重複しています
DB_E_DUPLICATEINDEXID	POL-5153: 索引はすでに存在します
DB_E_NOINDEX	POL-5167: 索引が存在しません
DB_E_DUPLICATECOLUMNID	POL-5202: 列名が重複しています
DB_E_NOCOMMAND	未対応
DB_E_BADBOOKMARK	未対応
DB_E_DELETEDROW	未対応
DB_E_CANTFETCHBACKWARDS	未対応
DB_E_FIELDDIFFERENT	POL-5213: 指定されたデータ型が比較可能ではありません
DB_E_FIELDMAXEXCEED	POL-5237: 列が多すぎます (最大で 1000 までです)

パッケージ・ウィザードの使用

この章では、Mobile Development Kit のパッケージ・ウィザード・ユーティリティについて説明します。内容は次のとおりです。

- [5.1 項「パッケージ・ウィザードの概要」](#)
- [5.2 項「パッケージ・ウィザードの起動」](#)
- [5.3 項「プラットフォームの選択」](#)
- [5.4 項「新規アプリケーションの命名」](#)
- [5.5 項「アプリケーション・ファイルのリスト表示」](#)
- [5.6 項「データベース情報の入力」](#)
- [5.7 項「レプリケーション用スナップショットの定義」](#)
- [5.8 項「アプリケーションの完了」](#)

5.1 パッケージ・ウィザードの概要

パッケージ・ウィザードは、次の目的に使用できるグラフィカル・ツールです。

- Windows CE プラットフォーム用の新規 Mobile サーバー・アプリケーションを作成する。
- 既存の Mobile サーバー・アプリケーションを編集する。
- アプリケーションを Mobile サーバー・リポジトリにパブリッシュする。

新規 Mobile サーバー・アプリケーションを作成するときは、コンポーネントを定義し Mobile サーバー・リポジトリにパブリッシュします。場合によっては、既存の Mobile サーバー・アプリケーションのコンポーネントの定義を編集することがあります。たとえば、アプリケーションの新規バージョンを開発する場合は、パッケージ・ウィザードを使用してアプリケーション定義を更新します。パッケージ・ウィザードを使用すると、アプリケーション・コンポーネントを **.jar** ファイルにパッケージ化することもできます。

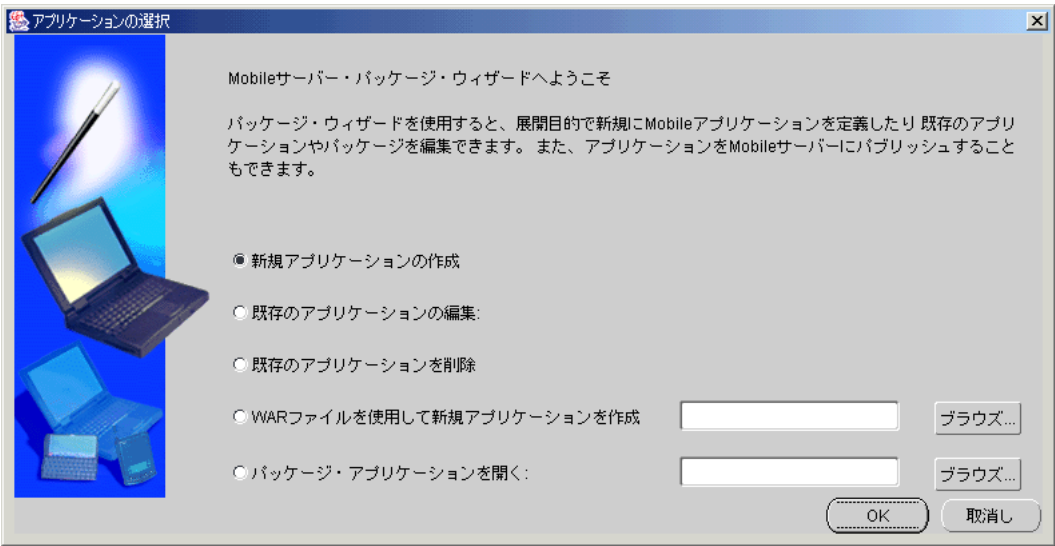
5.2 パッケージ・ウィザードの起動

パッケージ・ウィザードを起動するには、DOS プロンプトで次のように入力します。

wtgpack

パッケージ・ウィザードが表示され、デフォルトで「ようこそ」パネルが表示されます。「ようこそ」パネルでは、[図 5-1](#) に示す機能を使用して、パッケージ化されたアプリケーションの作成、編集またはオープンができます。

図 5-1 「ようこそ」 パネル



「ようこそ」 パネルのオプションを表 5-1 に示します。

表 5-1 「ようこそ」 パネルのオプション

機能	説明
新規アプリケーションの作成	このオプションを選択すると、新規アプリケーションを定義できます。
既存のアプリケーションの編集	このオプションを選択すると、既存のアプリケーションを編集できます。隣にあるドロップダウン・リストから既存のアプリケーションを選択できます。
既存のアプリケーションを削除	このオプションは、指定されたアプリケーションへの参照をファイルからすべて削除します。以前にパブリッシュされたアプリケーションの場合は、Mobile サーバー・リポジトリから削除しません。これは、コントロール・センターを使用して行う必要があります。
WAR ファイルを使用して新規アプリケーションを作成	WAR ファイルを使用する Web ベースのアプリケーションは、Web-to-Go に対してのみパッケージ化されます。WAR ファイルのサポートの詳細は、『Oracle9i Lite for Web-to-Go 開発者ガイド』を参照してください。
パッケージ・アプリケーションを開く	このオプションを選択すると、.jar ファイルとしてパッケージ化されているアプリケーションを選択できます。隣にあるフィールドにパッケージ・アプリケーションの名前を入力するか、「ブラウズ」ボタンを使用して、編集するアプリケーションを検索します。

5.3 プラットフォームの選択

この画面を使用して、アプリケーションをパッケージ化する対象プラットフォームを選択できます。プラットフォームは最低 1 つ選択する必要があります。アプリケーションが 2 種類以上のクライアントで実行される場合は、複数を選択できます。上の「使用可能プラットフォーム」のリストでプラットフォームを選択し、左にある下向き矢印ボタンを使用して、そのプラットフォームを「選択済プラットフォーム」に移動します。「使用可能プラットフォーム」のリストを [図 5-2](#) に示します。

図 5-2 プラットフォームの選択画面



5.4 新規アプリケーションの命名

「アプリケーション」パネルは、Mobile サーバー・アプリケーションに名前を付けて、このアプリケーションを Mobile サーバー上のどこに格納するかを指定するために使用します。このパネルには、[図 5-3](#) に示すフィールドが含まれます。

図 5-3 「アプリケーション」 パネル



「アプリケーション」 パネルのオプションを表 5-2 に示します。

表 5-2 「アプリケーション」 パネルのオプション

フィールド	説明	必須
仮想パス	これは、アプリケーションに対して一意の ID を提供します。 Mobile サーバー・リポジトリのルート・ディレクトリからアプリケーション自体の場所にマップされるパスです。仮想パスにより、アプリケーションのディレクトリ構造全体を参照する必要がなくなります。	<input type="radio"/>
アプリケーション名	Mobile サーバーにログインするときに表示されるアプリケーションの名前。	<input type="radio"/>
説明	Windows アプリケーションの概要。	<input type="radio"/>
ローカル・アプリケーションのディレクトリ	ローカル・マシン上でこのアプリケーションの全コンポーネントが含まれているディレクトリです。この場所は、入力するかまたは「ブラウズ」ボタンをクリックして選択します。このディレクトリに必要な形式は、 5.4.1 項「プラットフォーム・ファイルの検索」 を参照してください。	<input type="radio"/>

5.4.1 プラットフォーム・ファイルの検索

ローカル・アプリケーションのディレクトリが必要です。アプリケーションに、Windows CE のファイルが含まれている場合は、ローカル・アプリケーション・ディレクトリの「wce」サブディレクトリにそのファイルを入れます。

単純化された新規カテゴリをサポートするために、バイナリ・ファイルの位置も変更されています。ファイルは、次のディレクトリにある Mobile サーバー・リポジトリに格納されます。

<AppRoot> / wce / <フォーム・ファクタ> / <言語> / <チップセット>

<AppRoot> は、アプリケーション・リポジトリのルートです。

特定のデバイス用のディレクトリに入れられないファイルは、Web-to-Go アプリケーションに使用されるものと想定されます。Web-to-Go ファイルには特定のディレクトリは不要です。このファイルはローカル・アプリケーション・ディレクトリのルート・レベルにも入れられます。

Mobile サーバーでは、同一アプリケーションの複数のバージョンを Mobile サーバー・ディレクトリにパブリッシュおよび管理できます。これは、同じアプリケーションに複数の実装が存在し、それぞれが Oracle データベース・サーバー内の同一のアプリケーション表にアクセスすることを意味します。たとえば、Palm と Compaq iPAQ 両用の C/C++ アプリケーションを持つこともできます。C++ ソース・コードはその一部またはすべてを再利用できる場合がありますが、Palm 用と iPAQ 用にそれぞれファイルを再コンパイルし、異なる実行可能バージョンを作成する必要があります。同一アプリケーションに 2 つのバージョンが存在し、それぞれが同一のデータベース表を使用します。アプリケーション・ファイルは、個別の名前を持つ専用サブディレクトリに格納することが重要です。ローカル・アプリケーション・ディレクトリは Windows 開発システム上のディレクトリで、アプリケーションの複数のバージョンが格納される場所です。パッケージ・ウィザードは、このアプリケーション（ルート）・ディレクトリの下にあるアプリケーション・ファイルを再帰的に読み込みます。

例 1 — 米語

'Applications' というローカル・アプリケーション・ディレクトリに、アプリケーションの複数のバージョンが格納されます。

C:\Applications

iPAQ 用の実行可能ファイルは、**¥wce¥Pocket_PC¥us¥arm** サブディレクトリの下に格納する必要があります。たとえば、次のようになります。

C:\Applications¥wce¥Pocket_PC¥us¥arm

HP Jornada 720 用の実行可能ファイルは、¥wce¥HPC_Pro¥us¥sh3 サブディレクトリの下に格納する必要があります。たとえば、次のようになります。

C:¥Applications¥wce¥HPC_Pro¥us¥sh3

ローカル・アプリケーション・ディレクトリは **C:¥Applications** ですが、これにはサブディレクトリが 2 つあります。

C:¥Applications — これがローカル・アプリケーション・ディレクトリです。

C:¥Applications¥wce — Windows CE 用のアプリケーション・サブディレクトリです。

C:¥Applications¥wce¥Pocket_PC¥us¥arm — Windows CE の StrongArm パージョン用のアプリケーション・サブディレクトリです。

C:¥Applications¥wce¥HPC_Pro¥us¥sh3 — Windows CE の SH3 パージョン用のアプリケーション・サブディレクトリです。

注意： ローカル・アプリケーション・ディレクトリ・フィールドにディレクトリ文字列を入力するときに必要なのは **C:¥Applications** のみです。この時点で他のサブディレクトリを入力すると、パッケージ作成プロセスが失敗する原因になります。

例 2 — 日本語

iPAQ のような日本語対応デバイスに配布されるアプリケーションの実行可能ファイルは、¥wce¥Pocket_PC¥ja¥arm サブディレクトリに格納します。たとえば、次のようになります。

C:¥Applications¥wce¥Pocket_PC¥ja¥arm

他のプラットフォーム用のディレクトリのリストは、該当するプラットフォーム用の開発者ガイドを参照してください。

5.5 アプリケーション・ファイルのリスト表示

「ファイル」パネルは、アプリケーション・ファイルをリスト表示し、各ファイルがローカル・マシン上のどこにあるかを示します。パッケージ・ウィザードはローカル・アプリケーションのディレクトリの内容を分析し、各ファイルのローカル・パスを表示します。

図 5-4 「ファイル」 パネル

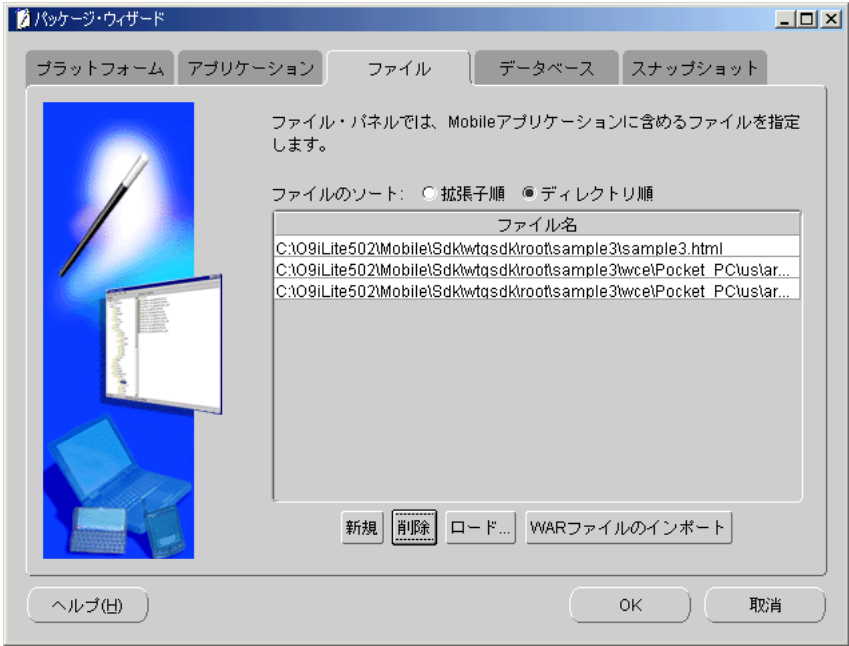


図 5-4 に示すとおり、「ファイル」パネルにリストされているファイルは、すべて追加、削除またはロードできます。新規アプリケーションを作成する場合、「ファイル」パネルに進むと、ローカル・ディレクトリにリストされているすべてのファイルが自動的に分析されロードされます。既存アプリケーションを編集する場合は、「ロード」ボタンを使用して個々のファイルをロードできます。

「ファイル」パネルのオプションを表 5-3 に示します。

表 5-3 「ファイル」パネルのオプション

フィールド	説明	必須
ローカルのパス	各 Mobile サーバー・アプリケーション・ファイルの絶対パスです。リスト内の各エントリには、各ファイルまたはディレクトリの完全なパスが含まれています。	<input type="radio"/>

表 5-3 「ファイル」 パネルのオプション（続き）

フィールド	説明	必須
ファイルのソート	<ul style="list-style-type: none"> ■ 拡張子順 — ファイルを拡張子ごとにアルファベット順に表示します。 ■ ディレクトリ順 — ファイルをディレクトリごとにアルファベット順に表示します。 	

5.5.1 ソート

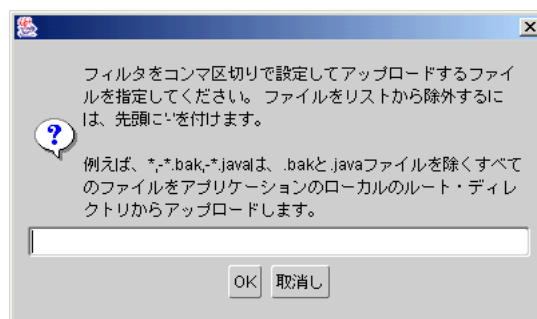
ファイルは、拡張子または含まれているディレクトリごとにソートできます。ファイルをソートするには、「拡張子順」または「ディレクトリ順」ラジオ・ボタンをクリックします。

5.5.2 フィルタ

「ロード」ボタンをクリックすると「入力」ダイアログ・ボックスが表示されます。図 5-5 に示すとおり、「入力」ダイアログ・ボックスは、アップロード・プロセスからのアプリケーション・ファイルを含めるかまたは除外するかを指定する（カンマで区切られた）フィルタのリストを作成するために使用します。ファイルを除外するには、ファイル名の前に負符号 (-) を付けます。たとえば、**.dll** および **.exe** 拡張子の付いたファイルを除くすべてのファイルをロードするには、次のように入力します。

***,-*.dll,-*.exe**

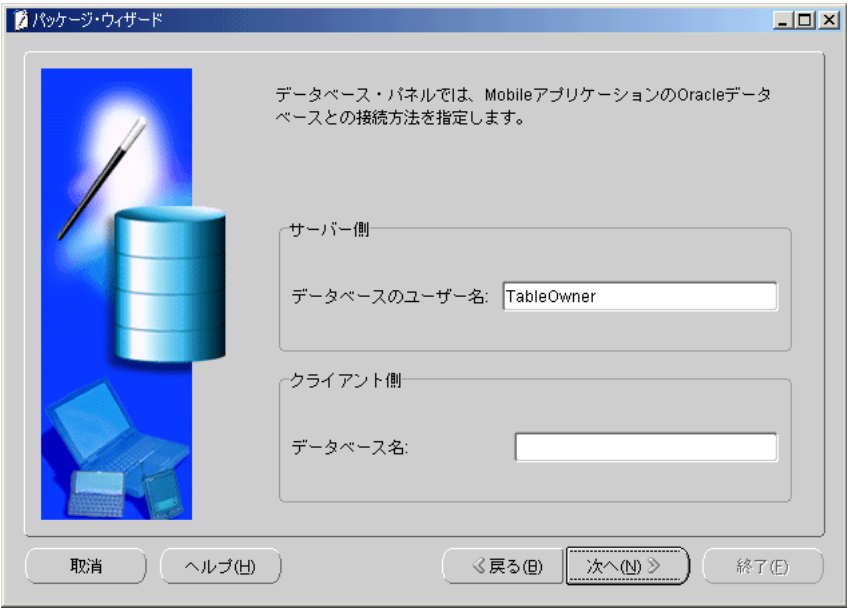
図 5-5 「フィルタ」 パネル



5.6 データベース情報の入力

図 5-6 に示すとおり、「データベース」パネルは、データを同期化する、Oracle サーバー上のデータベースを指定するために使用します。

図 5-6 「データベース」パネル



「データベース」パネルのフィールドを表 5-4 に示します。

表 5-4 「データベース」パネルのオプション

フィールド	説明	必須
データベースのユーザー名	データの同期をとるためにアプリケーションによって使用されるデータベースのユーザー名。	<input type="radio"/>
データベース名	Mobile クライアント・デバイス上で接続中のデータベースの名前。アプリケーションで使われる名前である必要があります。空白のまま何も指定しないと、自動的に名前が生成されます。	<input type="radio"/>

5.7 レプリケーション用スナップショットの定義

「スナップショット」パネルは、アプリケーションのレプリケーション・スナップショットを作成するために使用します。スナップショットは、全アプリケーション間で一意にする必要があります。これを実現する方法の1つは、スナップショット名の前にアプリケーション名を付けて変更することです。パッケージ・ウィザードでは、次のプラットフォームに対してスナップショットを作成できます。

- Windows CE
- Win32
- Palm
- EPOC

この画面には、[図 5-7](#) に示すフィールドが含まれます。

図 5-7 「スナップショット」パネル



「スナップショット」パネルのフィールドを表 5-5 に示します。

表 5-5 スナップショット・パラメータ

フィールド	説明	必須
全プラットフォーム	<p>現在のスナップショットのプラットフォームのド ロップダウン・リストです。ドロップダウン・リス トには、次のプラットフォームをすべて含めること ができます。</p> <ul style="list-style-type: none">Win32PalmEPOCWindows CE全プラットフォーム <p>ドロップダウン・リストからプラットフォームを選 択すると、そのプラットフォーム用のスナップ ショットのみが「スナップショット」パネルに表示 されます。たとえば、「全プラットフォーム」ド ロップダウン・リストから「Win32」を選択する と、Win32 ベースのスナップショットのみが表示さ れます。ドロップダウンから「全プラットフォーム」オプ ションを選択すると、現在使用中のプラッ トフォームごとにすべてのスナップショットが表示 されます。ユーザーが新しいスナップショットを追 加した場合、ドロップダウン・リストには追加のプ ラットフォームがリスト表示されます。</p>	×
名前	<p>Mobile サーバー・アプリケーションに関連付けら れているスナップショット定義の名前です。この名 前は、Oracle Lite データベースに既存のデータベ ース表と同じである必要があります、存在しない場合は作 成する必要があります。</p>	○

表 5-5 スナップショット・パラメータ

フィールド	説明	必須
テンプレート	テンプレートとは、スナップショットの作成に使用される SQL 文です。テンプレートには変数を含められます。テンプレートを Mobile サーバーにパブリッシュした後は、Mobile サーバー・コントロール・センターを使用して、ユーザー固有のテンプレート変数を指定できます。ただし、Mobile サーバー・コントロール・センターではスナップショット定義テンプレートを変更できません。	○
プラットフォーム	スナップショット定義のプラットフォームです。ユーザーは異なるプラットフォームに対してスナップショット定義を作成できます。クライアントのデータと同期をとった場合、クライアント・アプリケーションを実行しているプラットフォームに適したスナップショット定義のみが取得されます。	○
比率	これは、データベース表の同期順序を決定する正の整数です。マスター・ディテール・リレーションを持つ表の場合、マスター表はディテール表より先にレプリケートされるように、低い比率を持つ必要があります。	○

「スナップショット」パネルで「新規」ボタンまたは「削除」ボタンをクリックすると、「スナップショット」パネルにスナップショットを追加または削除できます。スナップショットはインポートまたは編集することもできます。

注意：「スナップショット」パネルから複数のスナップショットをインポートできますが、「新規表」ダイアログ・ボックスから新規表を作成するときにインポートできるスナップショットは1つのみです。

5.7.1 新規スナップショットの作成

新規スナップショットを作成するには、「新規」ボタンをクリックします。図 5-8 に示すような「新規スナップショット」ウィンドウが表示されます。

図 5-8 「新規スナップショット」ウィンドウ

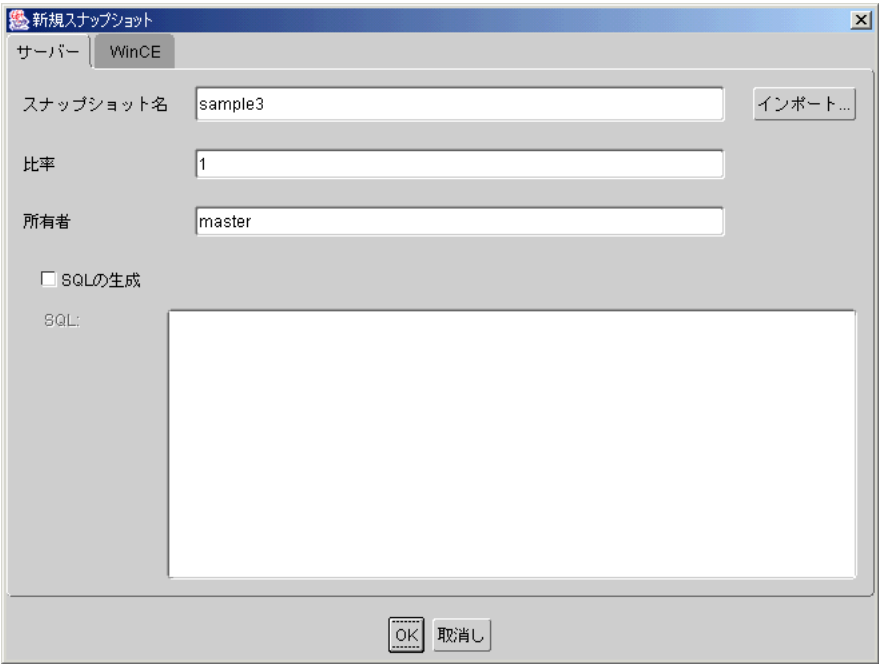


表 5-6 に示す「新規スナップショット」ウィンドウの機能を変更して、新規スナップショットを作成します。

表 5-6 「新規スナップショット」ウィンドウのオプション

機能	説明
プラットフォーム	タブには、「プラットフォーム」画面での選択に基づくプラットフォームが表示されます。
スナップショット名	スナップショット定義の基になるデータベース・サーバー表の名前です。
比率	この表に対する表の比率を設定できます。表の比率は、同期時の競合を解決するために使用されます。詳細は、 6.2.8.3 項 を参照してください。

表 5-6 「新規スナップショット」ウィンドウのオプション（続き）

機能	説明
所有者	実表が属するスキーマ名です。
SQL の生成	この機能を選択すると、パッケージ・ウィザードにより SQL スクリプトへの出力情報が収集されます。この SQL スクリプトは、Mobile サーバーに関連付けられているデータベース上にデータベース表を作成するために使用できます。データベースに実表が存在し、SQL スクリプトを使用して実表を作成する必要がない場合は、このチェックボックスの選択を解除します。
SQL	名前付きの表を定義する SQL の CREATE TABLE 文を表示します。この文は変更できます。「SQL の生成」ボックスが選択されている場合は、作成される SQL スクリプトにこの SQL 文が含まれます。

5.7.1.1 スナップショットの索引の作成

パッケージ・ウィザードを使用してスナップショットの索引を作成するには、次の手順を実行します。


1.  図 5-7 に示すとおり、「スナップショット」パネルから「編集」ボタンを選択して、既存のスナップショットから索引を作成します。スナップショットおよび索引を新規作成する場合は、「新規」ボタンを選択します。

図 5-9 索引の作成

スナップショットの編集

サーバー WinCE

☒ クライアントで作成

☒ 更新可能

競合解決: ☒ サーバー優先 ☐ クライアント優先 DMLプロシージャ

リフレッシュ・タイプ: ☒ 高速リフレッシュ ☐ 完全リフレッシュ

テンプレート: `SELECT * FROM MASTER.RECORDINGS WHERE USERCODE=USER`

索引

名前	タイプ	列
ID INDEX	primary	ID

新規 削除

OK 取消し

2. 図 5-9 に示すとおり、パネルからプラットフォームのタブを選択します。「テンプレート」フィールドにスナップショットを定義する SQL 文が表示されます。その下にあるのが「索引」表です。新規索引を作成するには、この表の下にある「新規」ボタンを選択します。
3. 「索引」表には、次の 3 列があります。
 - 名前 — 索引の名前です。
 - タイプ — 索引は、**regular**（標準）、**primary**（1 次）または **unique**（一意）です。これらを選択するにはドロップダウン・メニューを使用します。
 - 列 — 索引で使用する列名を入力します。

5.7.2 スナップショットのインポート

Oracle データベースまたは Oracle Lite データベースからスナップショットをインポートするには、「インポート」ボタンをクリックします。接続を指定していない場合は、「データベースへの接続」ウィンドウが表示されます。

注意： 一度指定したデータベース接続は、パッケージ・ウィザードの残りの部分でも使用されます。すでに接続が確立されているが、Oracle と Oracle Lite データベースを切り替える必要がある場合は、パッケージ・ウィザード・アプリケーションを完全に終了して、再度 **wtgpack.exe** を実行します。

図 5-10 「データベースへの接続」ウィンドウ



図 5-10 に示すとおり、スナップショットのインポート元の Oracle データベースのユーザー名、パスワードおよびデータベース URL を入力します。「OK」ボタンをクリックして続けます。「表」ウィンドウが表示されます。

注意： Oracle データベースのデータベース URL を入力する場合は、次の書式を使用します。

`jdbc:oracle:thin:@<databasehostname>:<port>:<SID>`

たとえば、`jdbc:oracle:thin:@<HOSTNAME>:1521:orcl` と指定します。

Oracle Lite の場合は、`jdbc:polite:webtogo` を使用します。

図 5-11 「表」ウィンドウ



図 5-11 に示すとおり、スナップショット定義の作成元のスキーマを選択した後、表を選択します。「追加」をクリックしてから「閉じる」をクリックします。パッケージ・ウィザードの「スナップショット」パネルに表が表示されます。

5.7.3 スナップショットの編集

図 5-12 に示すとおり、スナップショット定義を編集するには、「スナップショット」パネルからスナップショットを選択し、「編集」をクリックします。「スナップショットの編集」ウィンドウが表示されます。最初に選択したプラットフォームがタブに表示されます。スナップショットがまったく同じ場合でも、プラットフォームごとにタブを使用してスナップショットを定義する必要があります。

図 5-12 「スナップショットの編集」ウィンドウ



スナップショット定義を編集するには、表 5-7 に示す「スナップショットの編集」ウィンドウのパラメータを変更します。

表 5-7 「スナップショットの編集」ウィンドウのオプション

機能	説明
クライアントで作成	このチェックボックスが選択されていると、次のことを実行できます。 <ul style="list-style-type: none">■ 更新可能スナップショットの作成。■ スナップショット・テンプレートの作成。管理者は、Mobile サーバー・コントロール・センターを使用して、複数の異なるユーザー用変数をこのテンプレートに対してインスタンス化できます。
更新可能	このチェックボックスは、更新可能として作成されるスナップショットを定義します。

表 5-7 「スナップショットの編集」ウィンドウのオプション

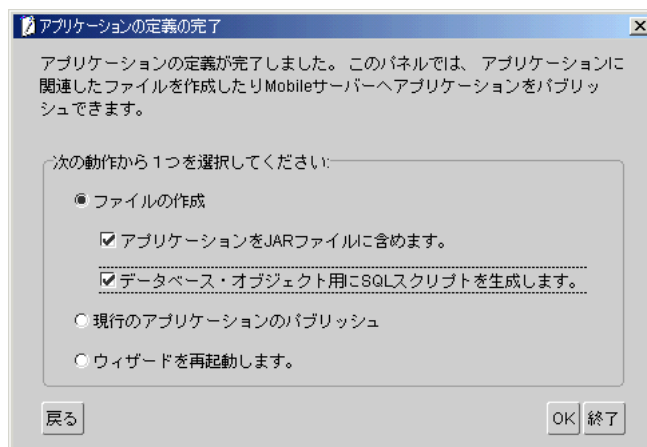
機能	説明
競合解決	このオプションは、すべての競合解決でサーバーが優先するかクライアントが優先するかを定義します。デフォルト設定は「サーバー優先」です。競合解決の詳細は、 6.5.2 項 を参照してください。
DML プロシージャ	このフィールドは、次の形式で DML プロシージャを指定するために使用できます。 AnySchema.AnyPackage.AnyName DML プロシージャを追加すると、「競合解決」オプションの選択が無効になります。 DML プロシージャの作成方法の詳細は、 6.4.13 項 を参照してください。
リフレッシュ・タイプ	このオプションには次の 2 つの選択肢があります。 <ul style="list-style-type: none">■ 高速リフレッシュデフォルトです。変更されたデータのみが転送されます。■ 完全リフレッシュデータはすべてリフレッシュされます。
テンプレート	名前付きの表のスナップショット・テンプレートを表示します。スナップショット・テンプレートは変更できます。テンプレート変数は、指定された表からデータのサブセット化を行い、クライアント用に構成されたデータを詳細化するために使用します。管理者は、Mobile サーバー・コントロール・センターを使用して、複数の異なるユーザー用変数をこのテンプレートに対してインスタンス化できます。テンプレート変数の詳細は、 5.7 項 を参照してください。

5.8 アプリケーションの完了

パッケージ・ウィザードの全パネルを完了すると、[図 5-13](#) に示す次のオプションを含む「アプリケーションの定義の完了」ウィンドウが表示されます。

- ファイルの作成
- 現行のアプリケーションのパブリッシュ
- ウィザードを再起動します。

図 5-13 「アプリケーションの定義の完了」ウィンドウ



5.8.1 アプリケーション・ファイル

パッケージ・ウィザードは、アプリケーション情報のすべてをファイルに自動的に保存します。パッケージ・ウィザードはローカル・マシン上にこのファイルを保持します。さらに、Mobile サーバーにこのファイルをパブリッシュするオプションも提供しています。アプリケーション・ファイルを Mobile サーバーにパブリッシュできるのは、サーバーが実行されているときのみです。

5.8.2 JAR ファイルの作成

「ファイルの作成」オプションを使用すると、アプリケーション・コンポーネントを **.jar** ファイルにパッケージ化できます。アプリケーション・コンポーネントを **.jar** ファイルにパッケージ化するには、「ファイルの作成」をクリックしてから「アプリケーションを JAR ファイルに含めます。」をクリックします。**.jar** ファイルの位置を指定するよう要求されます。

アプリケーションをパッケージ化した後は、パッケージ・ウィザードにより **.jar** ファイルが作成されます。Mobile サーバー・インスタンスに対する管理権限を持つユーザーなら、だれでも、コントロール・センターを使用して Mobile サーバー・リポジトリにパブリッシュできます。

5.8.3 SQL ファイルの作成

SQL スクリプトを生成するには、「ファイルの作成」をクリックしてから「データベース・オブジェクト用に SQL スクリプトを生成します。」をクリックします。生成されたスクリプトは、アプリケーションのローカル・ルート・ディレクトリの下の SQL サブディレクトリ内に入れられます。SQL スクリプトは、スナップショットおよび順序に関して指定した情報を使用します。この SQL スクリプトをデータベースに対して実行して、これらのデータベース・オブジェクトを作成できます。

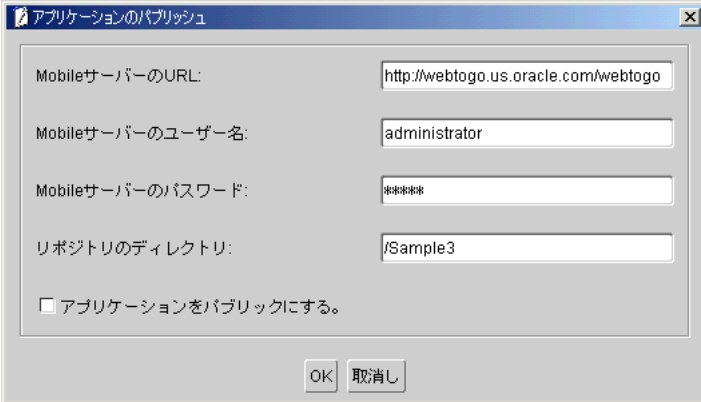
5.8.4 パッケージ・ウィザードの再起動

「ウィザードを再起動します。」オプションを使用すると、パッケージ・ウィザードを再起動できます。このオプションを使用すると、パッケージ・ウィザードの「ようこそ」パネルに戻ります。パッケージ・ウィザードを再起動するには、「ウィザードを再起動します。」をクリックしてから「OK」をクリックします。

5.8.5 アプリケーションのパブリッシュ

「現行のアプリケーションのパブリッシュ」オプションを使用すると、パッケージ・ウィザードで作成し定義したアプリケーションをパブリッシュできます。Mobile サーバー・アプリケーションをパブリッシュするには、「現行のアプリケーションのパブリッシュ」チェックボックスを選択してから「OK」をクリックします。図 5-14 に示す「アプリケーションのパブリッシュ」ウィンドウが表示されます。

図 5-14 「アプリケーションのパブリッシュ」ウィンドウ



MobileサーバーのURL:	http://webtogo.us.oracle.com/webtogo
Mobileサーバーのユーザー名:	administrator
Mobileサーバーのパスワード:	*****
リポジトリのディレクトリ:	/Sample3
<input type="checkbox"/> アプリケーションをパブリックにする。	
OK 取消し	

表 5-8 に示す「アプリケーションのパブリッシュ」ウィンドウの各フィールドに必要な情報を入力します。

表 5-8 「アプリケーションのパブリッシュ」ウィンドウのオプション

フィールド	説明	必須
Mobile サーバーの URL	サーバー名とポート番号を含む、Mobile サーバーの URL です。サーバー名とポート番号は、次の書式で指定します。 http://mobileserver:port/webtogo mobileserver は、Mobile サーバーのホスト名で、port は TCP/IP ポートです。デフォルト・ポートはポート 80 です。	<input type="radio"/>
Mobile サーバーの ユーザー名	Mobile サーバー・ユーザーの名前です。	<input type="radio"/>
Mobile サーバーの パスワード	Mobile サーバー・ユーザーのパスワードです。	<input type="radio"/>
リポジトリのディレクトリ	Mobile サーバー・リポジトリの宛先ディレクトリです。パッケージ・ウィザードはアプリケーション・ファイルをこのディレクトリにパブリッシュし、ローカル・アプリケーションのディレクトリ上でディレクトリ構造をメンテナン스します。	<input type="radio"/>
アプリケーションをパブリックにする。	このアプリケーションをパブリック・アプリケーションとしてパブリッシュするには、これを選択します。パブリック・アプリケーションに対しては、すべてのユーザーがアクセス権を持ちます。	<input checked="" type="radio"/>

注意： アプリケーションを Mobile サーバーにパブリッシュするには、publish 権限が必要です。Mobile サーバー管理者は Mobile サーバー・コントロール・センターを使用して権限を割り当てます。

5.8.6 アプリケーションの編集

パッケージ・ウィザードを起動して「既存のアプリケーションの編集」を選択すると、アプリケーションを編集できます。

Consolidator

この章では、Consolidator、パブリッシュ・サブスクライブ・モデル、アプリケーションをカスタマイズするための Consolidator および Resource Manager API の使用方法、および Mobile Sync Client モジュール・プログラミング・インタフェースについて説明します。内容は次のとおりです。

- 6.1 項「Consolidator API を使用したパブリッシュ・サブスクライブ・モデル」
- 6.2 項「Consolidator を使用したサンプル・プログラム Sample11.java の定義」
- 6.3 項「Consolidator のその他の標準機能」
- 6.4 項「Consolidator をカスタマイズするための拡張機能」
- 6.5 項「同期エラーと競合」
- 6.6 項「Oracle サーバーとクライアント間でのデータ型のマッピング」

これらの項目では、デフォルトの同期プロセスを変更する場合の説明を行います。この章の内容を理解するには、Java プログラミング言語の十分な知識が必要です。

6.1 Consolidator API を使用したパブリッシュ・サブスクライブ・モデル

Mobile サーバーではパブリッシュ・サブスクライブ・モデルを使用して、Oracle データベースのサーバーとクライアント間のデータ配分を集中管理します。パブリケーション・アイテムやパブリケーションの作成などの基本機能は、パッケージ・ウィザードを使用すると簡単に実装できます。このような機能は、Consolidator API や Resource Manager API でも実行できます。その場合は、Java プログラムを作成して、必要に応じて機能をカスタマイズします。拡張機能をプログラムにより使用可能にするには、Consolidator API および Resource Manager API を使用する必要があります。

パブリッシュ・サブスクライブ・モデルでは、表 6-1 に示すデータベース・オブジェクトが使用されます。

表 6-1 パブリッシュ・サブスクライブ・モデルの要素

項目	説明
パブリケーション・アイテム	パブリケーション・アイテムは、ユーザーがアクセスできるデータ・サブセットを指定する SQL の SELECT 文です。パブリケーション・アイテムは、クライアント上のレプリカ表に対応し、サーバー上の表のスナップショット定義となります。
パブリケーション	パブリケーションは、パブリケーション・アイテムのグループです。
スナップショット	スナップショットは、パブリケーション・アイテムのスナップショット定義によって定義された、Oracle データベース実表内のデータのサブセットです。
サブスクリプション	サブスクリプションは、ユーザーをパブリケーションに対応付けます。サブスクリプション・パラメータを含む場合もあります。サブスクリプション・パラメータは、クライアントがサブスクライブされるパブリケーション内のすべてのパブリケーション・アイテムのセットです。
サブスクリプション・パラメータ	サブスクリプション・パラメータは、名前と文字列値を使用して、個々のパブリケーションに対する個々のクライアントのサブスクリプションを定義します。サブスクリプション・パラメータを使用すると、クライアントはデータのサブセット化を実行し、各クライアントに割り当てられる行数を制限できます。一般的なサブスクリプション・パラメータには、ユーザー名とアプリケーション固有の値（社員番号や市外局番など）を含めることができます。

表 6-1 パブリッシュ・サブスクライブ・モデルの要素（続き）

項目	説明
ユーザー	<p>ユーザーは、ユーザー名とパスワードで定義されます。Mobile サーバーは、クライアントのサブスクリプションに従ってデータを同期します。</p> <ul style="list-style-type: none"> ■ ユーザーは、単一のユーザー名を使用して、単一クライアントのデータを同期させることができます。この使用モードをお勧めします。 ■ ユーザーは単一のユーザー名を使用して、複数のデバイスに格納されたデータを同期できます。ユーザーがデバイスを変更すると、Mobile サーバーにより新規デバイス上でそのユーザーの全サブスクリプションの完全リフレッシュが実行されます。この方法は、一般的な使用ではお勧めしません。

パブリッシュ・サブスクライブ・モデルは、次のいずれかを使用して実装できます。

- パッケージ・ウィザードを使用して、アプリケーションのパッケージ化およびパブリッシュを行います。この方法をお勧めします。この方法の詳細は、[第 5 章](#)を参照してください。
- Resource Manager API および Consolidator API を使用して、特定の拡張機能を起動したり、または実装をカスタマイズします。この方法は、特化した機能を必要とする上級者にお勧めします。

6.1.1 パブリッシュ・サブスクライブ・モデルの使用方法

パブリッシュ・サブスクライブ・モデルは、Consolidator API および Resource Manager API を使用してプログラムによりカスタマイズできます。Consolidator を起動して、パブリッシュ・サブスクライブ・モデルを実装するための基本手順は、次のとおりです。

1. データベース表を作成します。
2. Mobile サーバーに接続します。
3. パブリケーションを作成します。
4. パブリケーション・アイテムを作成します。
5. 必要に応じて、パブリケーション・アイテム索引を作成します。
6. ユーザーを作成します。
7. パブリケーションにパブリケーション・アイテムを追加します。
8. ユーザーをパブリケーションにサブスクライブします。

- 9. パブリケーションに対してユーザーのサブスクリプション・パラメータを定義します。
- 10. サブスクリプションをインスタンス化します。

6.1.2 Consolidator の機能の比較

パッケージ・ウィザードおよびコントロール・センターでは、アプリケーションをパッケージ化してパブリッシュする、ユーザーを作成または削除する、サブスクリプションを作成または削除するなど、最も一般的に使用するパブリッシュ・サブスクリブ・モデルの機能が用意されています。さらに高度な機能は、Consolidator API および Resource Manager API により提供されます。基本機能の比較を表 6-2 に示します。

表 6-2 Consolidator の基本機能の比較

機能	パッケージ・ウィザード	コントロール・センター	API
接続のオープン	×	×	○
ユーザーの作成	×	○	○
ユーザーの削除	×	○	○
パブリケーションの作成	○ (Web アプリケーションのみ)	×	○
パブリケーション・アイテムの作成	○ (Web アプリケーションのみ)	×	○
パブリケーション・アイテム索引の作成	○ (Web アプリケーションのみ)	×	○
パブリケーションの削除	×	×	○
パブリケーション・アイテムの削除	特殊。詳細は、パッケージ・ウィザードについてのドキュメントを参照してください。	×	○
パブリケーション・アイテム索引の削除	○ (Web アプリケーションのみ)	×	○
シーケンスの作成	○ (Web アプリケーションのみ)	×	○
シーケンス・パーティションの作成	○ (Web アプリケーションのみ)	×	○
シーケンスの削除	○ (Web アプリケーションのみ)	×	○
シーケンス・パーティションの削除	○ (Web アプリケーションのみ)	×	○

表 6-2 Consolidator の基本機能の比較（続き）

機能	パッケージ・ウィザード	コントロール・センター	API
パブリケーション・アイテムの追加	○	×	○
パブリケーション・アイテムの取消し	×	×	○
サブスクリプションの作成	×	○	○
サブスクリプションのインスタンス化の解除	×	×	○
サブスクリプション・パラメータの設定	×	○	○
サブスクリプションの削除	×	○	○
トランザクションのコミット	×	×	○
トランザクションのロールバック	×	×	○
接続のクローズ	×	×	○

Consolidator の拡張機能は、Consolidator および Resource Manager API を使用することによってのみ、一般的に使用可能となります。拡張機能の比較を表 6-3 に示します。

表 6-3 Consolidator の拡張機能の比較

機能	パッケージ・ウィザード	コントロール・センター	API
仮想主キー列の作成	×	×	○
仮想主キー列の削除	×	×	○
モバイル DML プロシージャの追加	○	×	○
モバイル DML プロシージャの削除	○	×	○
パブリケーション・アイテムの再インスタンス化	×	×	○
親表のヒント	×	×	○
依存関係のヒント	×	×	○
依存関係のヒントの削除	×	×	○
パブリケーション・アイテムの問合せキャッシュの有効化	×	×	○

表 6-3 Consolidator の拡張機能の比較（続き）

機能	パッケージ・ウィザード	コントロール・センター	API
パブリケーション・アイテムの問合せ キャッシュの無効化	×	×	○
主キーのヒント	×	×	○
トランザクションのパージ	×	×	○
トランザクションの実行	×	×	○
完全リフレッシュ	×	○	○
文の実行	×	×	○
メタデータの生成	×	×	○
キャッシュのリセット	×	×	○
キャッシュの依存関係	×	×	○
キャッシュの依存関係の削除	×	×	○
現在の時刻の取得	×	×	○
認証	×	○	○
制限選択条件の設定	×	×	○
パブリケーションの変更	×	×	○

6.2 Consolidator を使用したサンプル・プログラム Sample11.java の定義

これらの API を使用して Consolidator を定義する方法を示すために、この項では、Oracle9i Lite に付属している **sample11.java** という Java プログラムのサンプルを使用します。Resource Manager パッケージを参照するエントリは、Web-to-Go API にある Mobile Admin クラスの子です。Consolidator クラスを参照するエントリは、Consolidator API に含まれています。

このサンプルは次の場所にあります。

Solaris の場合：

<ORACLE_HOME>/mobile/server/samples/sample11

Windows NT の場合：

<ORACLE_HOME>%mobile%server%samples%sample11

6.2.1 Sample11.java

プログラムのソース・コードは次のとおりです。

```
import java.sql.SQLException;
import java.sql.*;

import oracle.lite.sync.Consolidator;

public class sample11
{
    static String CONS_SCHEMA;
    static String DEFAULT_PASSWORD;

    public static void main(String argv[]) throws Throwable
    {
        //////////////////////////////////////
        //SAMPLE11
        //////////////////////////////////////
        if(argv.length != 2)
        {
            System.out.println("Syntax: java sample11 <Schema> <Password>");
            return;
        }
        CONS_SCHEMA = argv[0] ;
        DEFAULT_PASSWORD = argv[1] ;

        // 標準 JDBC を使用した必須表の作成
        DriverManager.registerDriver ((Driver)Class.forName
        ("oracle.jdbc.driver.OracleDriver").newInstance ());
        Connection c = null;
        Statement s = null;
        try
        {
            c = DriverManager.getConnection ("jdbc:oracle:oci8:@WEBTOGO.WORLD",
            "MASTER", "MASTER" );
            s = c.createStatement ();
            s.executeUpdate("create table MASTER.ORD_MASTER("
            + "ID number(9),"
            + "DDATE DATE default TO_DATE('1990-01-01 15:35:40', 'YYYY-MM-DD
            HH24:MI:SS'),"
            + "STATUS number(9),"
```

```
+ "NAME varchar2(20),"
+ "DESCRIPTION varchar2(20)"
+ ")";

s.executeUpdate("alter table MASTER.ORD_MASTER add constraint"
+ " orders_pk primary key(ID)");

s.execute("GRANT ALL ON MASTER.ORD_MASTER to " + CONS_SCHEMA + " WITH GRANT
OPTION");

s.executeUpdate("create table MASTER.ORD_DETAIL("
+ "ID number(9),"
+ "KEY number(9),"
+ "DATE DATE default TO_DATE('1995-01-01 15:35:40', 'YYYY-MM-DD
HH24:MI:SS'),"
+ "DESCRIPTION varchar2(20),"
+ "QTYORDERED number(9),"
+ "QTYSHIPPED number(9),"
+ "QTYRECEIVED number(9),"
+ "COST number(9)"
+ ")");

s.executeUpdate("alter table MASTER.ORD_DETAIL add constraint"
+ " items_pk primary key(ID, KEY)");

s.execute("GRANT ALL ON MASTER.ORD_DETAIL to " + CONS_SCHEMA + " WITH
GRANT OPTION");
c.commit ();
}
catch (SQLException ee)
{
    ee.printStackTrace ();
}
finally
{
    if (s!= null) try {s.close ();}catch (SQLException e1){}
    if (c!= null) try {c.close ();}catch (SQLException e2){}
}

//Mobile サーバーへの接続
oracle.mobile.admin.ResourceManager.openConnection(CONS_SCHEMA, DEFAULT_
PASSWORD);
// パブリケーションの作成
try {
```

```
Consolidator.DropPublication("T_SAMPLE11");
} catch (Throwable e) {
//e.printStackTrace(); ignore error
}
Consolidator.CreatePublication("T_SAMPLE11", Consolidator.OKPI_CREATOR_ID,
"OrdersODB.%s", null);

    // パブリケーション・アイテムの作成
try {
Consolidator.DropPublicationItem("P_SAMPLE11-M");
} catch (Throwable e) {
//e.printStackTrace(); ignore error
}

    try
    {
        Consolidator.CreatePublicationItem("P_SAMPLE11-M","MASTER","ORD_MASTER", "F",
"SELECT * FROM MASTER.ORD_MASTER", null, null);
    } catch (Throwable e) {
e.printStackTrace();
}

try {
Consolidator.DropPublicationItem("P_SAMPLE11-D");
} catch (Throwable e) {
//e.printStackTrace();
}

    try
    {
Consolidator.CreatePublicationItem("P_SAMPLE11-D","MASTER","ORD_DETAIL", "F",
"SELECT * FROM MASTER.ORD_DETAIL", null, null);

    // パブリケーション・アイテム索引の作成

Consolidator.CreatePublicationItemIndex("P_SAMPLE11M-I1", "P_SAMPLE11-M", "I",
"DDATE");
Consolidator.CreatePublicationItemIndex("P_SAMPLE11M-I2", "P_SAMPLE11-M", "I",
"STATUS");
Consolidator.CreatePublicationItemIndex("P_SAMPLE11M-I3", "P_SAMPLE11-M", "I",
"NAME");
Consolidator.CreatePublicationItemIndex("P_SAMPLE11D-I2", "P_SAMPLE11-D", "I",
"KEY");
Consolidator.CreatePublicationItemIndex("P_SAMPLE11D-I3", "P_SAMPLE11-D", "I",
"DESCRIPTION");
```

```
// パブリケーションへのパブリケーション・アイテムの追加

Consolidator.AddPublicationItem("T_SAMPLE11", "P_SAMPLE11-M", null, null, "S", null,
null);
Consolidator.AddPublicationItem("T_SAMPLE11", "P_SAMPLE11-D", null, null, "S", null,
null);
    }
    catch (Throwable e)
    {
        e.printStackTrace ();
    }

// ユーザーの作成
try {
oracle.mobile.admin.ResourceManager.Example("S11U1");
} catch (Throwable e) {
//e.printStackTrace(); ignore error
}

oracle.mobile.admin.ResourceManager.Example("S11U1","manager","S11U1","C");

// サブスクリプションのインスタンス化
Consolidator.Example("T_SAMPLE11", "S11U1");
Consolidator.InstantiateSubscription("T_SAMPLE11", "S11U1");

oracle.mobile.admin.ResourceManager.commitTransaction();
oracle.mobile.admin.ResourceManager.closeConnection();

    }
}
```

6.2.2 標準 JDBC を使用した必須表の作成

このプログラムの最初のセクションは、データベース MASTER への JDBC 接続を取得し、データベース内に実表 ORD_MASTER および ORD_DETAIL を作成します。この部分の処理は、SQL を使用して行うこともできます。3.6 項「同期のテスト」に説明されている手順まで進むと、Mobile サーバー・リポジトリおよびクライアントにこれらの表が作成されます。

6.2.3 Mobile サーバーへの接続

次の構文は、Mobile サーバーへの接続を行います。

6.2.3.1 openConnection

例

```
ResourceManager.openConnection(<USERNAME>, <PASSWORD>);  
oracle.mobile.admin.ResourceManager.openConnection  
    (CONS_SCHEMA,  
     DEFAULT_PASSWORD);
```

この例では、<USERNAME> は MOBILEADMIN で、<PASSWORD> は MANAGER です。

6.2.4 パブリケーションの作成

次の手順は、Consolidator クラスを使用してパブリケーションを作成することです。パブリケーションは、基本的にパブリケーション・アイテムのセットです。Sample11.java により、2つのパブリケーションが作成されます。最初に、DropPublication コマンドを使用して、作成するパブリケーションがまだ存在していないことを確認します。

パブリケーション名には、空白などの特殊文字を使用できます。

6.2.4.1 CreatePublication

CreatePublication の構文は、次のとおりです。

```
public static void CreatePublication  
    (String name,  
     int client_storage_type,  
     String client_name_template,  
     String enforce_ri) throws Throwable
```

例

Sample11.java では、T_SAMPLE11 というパブリケーションを作成します。

```
Consolidator.CreatePublication("T_SAMPLE11", Consolidator.OKPI_CREATOR_ID,  
"OrdersODB.%s", null);
```

CreatePublication のパラメータを[表 6-4](#) に示します。

表 6-4 CreatePublication のパラメータ

パラメータ	定義
name	作成するパブリケーションの名前です。
client_storage_type	プラットフォーム・タイプを定義する定数。詳細は、 表 6-5「クライアントの格納タイプ」 を参照してください。
client_name_template	クライアント・デバイスを対象としたパブリケーション・アイテム名のテンプレートです。次のうちの 1 つを選択します。 <ul style="list-style-type: none">■ %s — これは、デフォルト設定で、パブリケーション・アイテムをデフォルト・データベースである conscli.odb に格納します。■ <DATABASE>.%s — このオプションは、パブリケーション・アイテムを <DATABASE> という名前のデータベースに格納します。このオプションは、ファイル名拡張子はサポートしません。■ テンプレートのかわりに、1 つのパブリケーション・アイテムを含むパブリケーションに特定の値を使用することができます。たとえば、「AddressBook」を Palm OS Address Book アプリケーションに使用できます。
enforce_ri	これは、将来の拡張用に確保されているパラメータで、常に NULL です。

注意： クライアントの格納タイプとして Oracle Lite データベースを使用する場合、データベースには拡張子は付きません。

表 6-5 クライアントの格納タイプ

パラメータ値	定義
Consolidator.DFLT_CREATOR_ID	Oracle Kernal API（OKAPI）は、すべてのデバイスでのデフォルトです。OKAPI は、クライアント上の Oracle Lite データベースのフォーマットを定義します。この格納タイプを選択すると、データベースは、OKAPI と互換性のあるすべてのデバイスで稼働します。
Consolidator.OKPI_CREATOR_ID	OKAPI の使用方法を指定します。この格納タイプを選択すると、データベースは、OKAPI と互換性のあるすべてのデバイスで稼働します。

表 6-5 クライアントの格納タイプ（続き）

パラメータ値	定義
Consolidator.OKAPI_PALM	格納タイプを Palm OS 専用として定義します。
Consolidator.OKAPI_EPOC	格納タイプを EPOC 専用として定義します。
Consolidator.OKAPI_WIN32	格納タイプを Windows 専用として定義します。
Consolidator.OKAPI_WINCE	格納タイプを Windows CE 専用として定義します。
Consolidator.PALMDB_EXPENSE	格納タイプを Palm OS Expense アプリケーション専用として定義します。
Consolidator.PALMDB_DATEBOOK	格納タイプを Palm OS の DateBook (Calendar) アプリケーション専用として定義します。
Consolidator.PALMDB_EMAIL	格納タイプを Palm OS 電子メール・アプリケーション専用として定義します。
Consolidator.PALMDB_ADDRESSBOOK	格納タイプを Palm OS Address Book アプリケーション専用として定義します。
Consolidator.PALMDB_TODO	格納タイプを Palm OS Todo アプリケーション専用として定義します。
Consolidator.PALMDB_MEMO	格納タイプを Palm OS Memo アプリケーション専用として定義します。

6.2.5 パブリケーション・アイテムの作成

パブリケーションを作成した後は、パブリケーション・アイテムを作成する必要があります。パブリケーション・アイテムは、Oracle Lite データベースにダウンロードされる実表のスナップショットを定義します。パブリケーション・アイテムのリフレッシュ・モードは作成時に指定するため、高速リフレッシュまたは完全リフレッシュに対応するように事前構成します。データ要件について細部まで制御できるように、パブリケーション・アイテムの作成時に、特定のクライアントに対してデータのサブセット化パラメータを設定することもできます。

パブリケーション・アイテム名は、26 文字に制限され、すべてのパブリケーションで一意である必要があります。次の例では、P_SAMPLE11-M というパブリケーション・アイテムを作成します。このパブリケーション・アイテムを作成する前に、サンプルでは、DropPublicationItem を使用して、名前が重複している既存のすべてのパブリケーション・アイテムをクリーンアップします。

6.2.5.1 CreatePublicationItem

CreatePublicationItem の構文は、次のとおりです。

```
public static void CreatePublicationItem
    (String name,
     String owner,
     String store,
     String refresh_mode,
     String select_stmt,
     String cbk_owner,
     String cbk_name) throws Throwable
```

CreatePublicationItem のパラメータを表 6-6 に示します。

表 6-6 CreatePublicationItem のパラメータ

パラメータ	定義
name	パブリケーション・アイテム名を指定します。
owner	ベース・オブジェクト・スキーマの所有者を指定します。たとえば、MASTER がベース・オブジェクト ORD_MASTER の所有者です。
store	Oracle データベース内の実表名またはビュー名を指定します。定義済スナップショットにもこの名前が割り当てられます。
refresh_mode	リフレッシュ・モードを高速または完全として定義します。詳細は、6.4.5 項「複数表パブリケーション（ビュー）の高速リフレッシュおよび更新操作」を参照してください。
select_stmt	データベース表内の、指定された列のデータを識別する SQL SELECT 文です。
cbk_owner	コールバック・パッケージの所有者を指定します。詳細は、6.4.12 項「構成と適用を使用する前後に行うコールバックのカスタマイズ」を参照してください。
cbk_name	コールバック・パッケージ名を指定します。詳細は、6.4.12 項「構成と適用を使用する前後に行うコールバックのカスタマイズ」を参照してください。

例

Sample11.java プログラムでは、次のコマンドにより、前述の手順でリポジトリ内に作成されたデータベース表 ORD_MASTER および ORD_DETAIL の P_SAMPLE-M および P_SAMPLE-D というスナップショット定義、つまりパブリケーション・アイテムが作成されます。

```
Consolidator.CreatePublicationItem("P_SAMPLE11-M", "MASTER", "ORD_MASTER", "F",  
"SELECT * FROM MASTER.ORD_MASTER", null, null);  
Consolidator.CreatePublicationItem("P_SAMPLE11-D", "MASTER", "ORD_DETAIL", "F",  
"SELECT * FROM MASTER.ORD_DETAIL", null, null);
```

6.2.5.2 更新可能な複数表ビューに対するパブリケーション・アイテムの定義

パブリケーション・アイテムは、表とビューの両方に対して定義できます。

更新可能な複数表ビューをパブリッシュする場合、特定の制限が適用されます。

- ビューには、主キーの定義された親表が含まれている必要があります。
- ビューのデータ操作言語（DML）操作に対して INSTEAD OF トリガーを定義する必要があります。詳細は、[6.4.5 項「複数表パブリケーション（ビュー）の高速リフレッシュおよび更新操作」](#)を参照してください。
- ビューの実表は、すべてパブリッシュする必要があります。

6.2.5.3 データのサブセット化

データのサブセット化は、特定のデータ・サブセットを作成してパラメータ名に割り当てる機能です。このパラメータ名は、サブスクライブするユーザーに割り当てられます。パブリケーション・アイテムを作成する場合、最大 8KB の文字制限を持つ、パラメータ化された SELECT 文を定義できます。サブスクリプション・パラメータは、パブリケーション・アイテムが作成された時点で指定される必要があり、特定のクライアントにパブリッシュされたデータを制御するために、同期時に使用されます。

データ・サブセットの作成例

```
Consolidator.CreatePublicationItem("CORP_DIR1", "DIRECTORY1", "ADDR14P", "F",  
"SELECT LastName, FirstName, company, phone1, phone2, phone3, phone4,  
phone5, phone1id, phone2id, phone3id, displayphone, address, city, state,  
zipcode, country, title, custom1, custom2, custom3, note  
FROM directory1.addr14p WHERE company > :COMPANY", null, null);
```

このサンプル文では、CORP_DIR1 というパブリケーションからデータが取り出され、企業 (company) によってサブセット化されます。

注意： SELECT 文内では、データ・サブセットのパラメータ名の前に、コロンを接頭辞として指定する必要があります（たとえば、:COMPANY）。

6.2.6 パブリケーションに対するクライアント・サブスクリプション・パラメータの定義

パブリケーションでデータのサブセット化パラメータを使用する場合は、これらのパラメータをパブリケーションのサブスクリプション用に設定する必要があります。[6.2.5.3 項「データのサブセット化」](#)に説明されている「COMPANY」はパラメータの一例です。

6.2.6.1 SetSubscriptionParameter

```
public static void SetSubscriptionParameter
    (String publication,
     String clientid,
     String param_name,
     String param_value) throws Throwable
```

SetSubscriptionParameter のパラメータを[表 6-7](#)に示します。

表 6-7 SetSubscriptionParameter のパラメータ

パラメータ	定義
publication	サブセットの導出元となるパブリケーションを定義します。
clientid	サブセット化されたデータのクライアント ID を定義します。
param_name	パラメータ名を定義します。
param_value	渡されるパラメータ値を定義します。この値により、このパラメータを使用してパブリケーション・アイテムの問合せから返されるデータが決まります。

例

この例では、パラメータを使用して、クライアント DAVIDL をパブリケーション CORP_DIR1 にサブスクライブします。

```
Consolidator.SetSubscriptionParameter("CORP_DIR1", "DAVIDL", "COMPANY", "'DAVECO'");
```

注意： この方法は、Consolidator API を使用して作成されたパブリケーションに対してのみ使用してください。同様の処理を行うには、パッケージ・ウィザードを使用して、データ・サブセットのもう 1 つのフォームであるテンプレート変数を作成する方法もあります。

6.2.7 パブリケーション・アイテム索引の作成

Mobile サーバーでは、クライアント上の Oracle Lite データベースへの索引の自動配布をサポートしています。Mobile サーバーは、主キー索引をサーバー・データベースから自動的にレプリケートします。Consolidator API では、クライアントに一意、標準および主キーの各索引を明示的に配布するコールも同様に提供しています。

6.2.7.1 CreatePublicationItemIndex

CreatePublicationItemIndex の構文は、次のとおりです。

```
public static void CreatePublicationItemIndex
    (String name,
     String publication_item,
     String pmode,
     String columns) throws Throwable
```

CreatePublicationItemIndex のパラメータを[表 6-8](#)に示します。

表 6-8 CreatePublicationItemIndex のパラメータ

パラメータ	定義
name	作成する索引名を定義します。
publication_item	索引のパブリケーション・アイテムを定義します。
pmode	索引モード（I—標準、U—一意、P—主キー・モード）を定義します。詳細は、 6.2.7.2 項「クライアント索引の定義」 を参照してください。
columns	索引に含める列名を定義します。それぞれの文に 2 つ以上の列をリストする場合、列のリストをカンマで区切る必要があります（空白は入れないでください）。

例 1

Sample11.java サンプル・コードでは、次の書式をとります。

```
Consolidator.CreatePublicationItemIndex("P_SAMPLE11M-I1", "P_SAMPLE11-M", "I",
"DDATE");
Consolidator.CreatePublicationItemIndex("P_SAMPLE11M-I2", "P_SAMPLE11-M", "I",
"STATUS");
Consolidator.CreatePublicationItemIndex("P_SAMPLE11M-I3", "P_SAMPLE11-M", "I",
"NAME");
Consolidator.CreatePublicationItemIndex("P_SAMPLE11D-I2", "P_SAMPLE11-D", "I",
"KEY");
Consolidator.CreatePublicationItemIndex("P_SAMPLE11D-I3", "P_SAMPLE11-D", "I",
"DESCRIPTION");
```

Sample11.java では、5 つの索引が作成されます。P_SAMPLE-M パブリケーション・アイテムの「DDATE」、「STATUS」および「NAME」列、および P_SAMPLE-D パブリケーション・アイテムの「KEY」および「DESCRIPTION」列について標準索引が設定されます。1 つの索引に、2 つ以上の列を含めることができます。次のように、複数列を持つ索引を定義することもできます。

例 2

```
Consolidator.CreatePublicationItemIndex("P_SAMPLE11D-I1", "P_SAMPLE11-D", "I",  
"KEY,DESCRIPTION");
```

6.2.7.2 クライアント索引の定義

クライアント側の索引は、既存のパブリケーション・アイテムに対して定義できます。3 種類の索引を指定できます。

- P – 主キー
- U – 一意
- I – 標準

注意：パブリケーション・アイテムに「U」または「P」タイプの索引を定義した場合、サーバー上では重複キーの検査は行われません。パブリケーション・アイテムのベース・オブジェクトに同一の制約がない場合、Mobile Sync は重複キー違反で失敗する可能性があります。詳細は、『Consolidator Admin API Specification』を参照してください。

6.2.8 パブリケーションへのパブリケーション・アイテムの追加

パブリケーション・アイテムを作成した後、これをパブリケーションに対応付ける必要があります。定義を変更するには、パブリケーション・アイテムを削除して新しい定義で作成しなおすか、要件に応じてスキーマの発展を使用します。詳細は、『Consolidator Admin API Specification』の「DropPublicationItem」および「AlterPublicationItem」を参照してください。

6.2.8.1 AddPublicationItem

AddPublicationItem の構文は、次のとおりです。

```
public static void AddPublicationItem  
(String publication,  
String item,  
String columns,  
String disabled_dml,  
String conflict_rule,  
String restricting-predicate,  
String weight) throws Throwable
```

次の例では、P_SAMPLE1 という名前のパブリケーション・アイテムをパブリケーション T_SAMPLE1 に追加します。AddPublicationItem のパラメータを表 6-9 に示します。

表 6-9 AddPublicationItem のパラメータ

パラメータ	定義
publication	新規アイテムを受け取るパブリケーションを定義します。
item	追加するパブリケーション・アイテムを定義します。
columns	<p>パブリケーション・アイテム列の新しい名前を指定します。NULL は、列の名前を変更しないことを指定します。パブリケーション・アイテムの問合せ内のすべての列は、次のいずれかの適切な順序で指定する必要があります。</p> <ul style="list-style-type: none"> ■ パブリケーション・アイテムの SELECT 文に指定される順序です。 ■ 「SELECT * FROM ...」を指定した文を使用する場合、列名の順序は実表またはビューの順序と同一である必要があります。
disabled_dml	<p>DML を使用禁止にするためのオプションを指定します。可能な値は、次のとおりです。</p> <ul style="list-style-type: none"> ■ Y — 完全に更新可能なパブリケーション・アイテムを定義します。 ■ N — 読み込み専用のパブリケーション・アイテムを定義します。読み込み専用のパブリケーション・アイテムは、「IUD」オプションを使用しても定義できます。 ■ I — 個々の挿入操作の伝播を使用禁止にします。 ■ U — 個々の更新操作の伝播を使用禁止にします。 ■ D — 個々の削除操作の伝播を使用禁止にします。 ■ NULL — DML を使用禁止するためのオプションを選択しないことを指定します。
conflict_rule	競合解決で優先する側を定義します。クライアントを優先する場合は「C」、サーバーを優先する場合は「S」を指定します。詳細は、 6.2.8.2 項「競合ルールの定義」 を参照してください。
restricting_predicate	高優先順位モードを示します。制限選択条件は、パブリケーションにパブリケーション・アイテムを追加するときに、パブリケーション・アイテムに割り当てることができます。クライアントが高優先順位モードで同期される場合、クライアントにプッシュされるデータを制限するために選択条件が使用されます。このパラメータは、NULL にできます。このパラメータは、上級者向けです。

表 6-9 AddPublicationItem のパラメータ（続き）

パラメータ	定義
weight	NULL、またはマスター表に対してクライアント操作を実行する優先順位を指定する整数を指定します。詳細は、 6.2.8.3 項「表の比率の使用」 を参照してください。この値には、1 ～ 1023 の整数を使用する必要があります。

例

```
Consolidator.AddPublicationItem("T_SAMPLE1", "P_SAMPLE1", null, null, "S", null, null);
```

6.2.8.2 競合ルールの定義

パブリケーションにパブリケーション・アイテムを追加する場合、クライアント「C」とサーバー「S」のいずれかを優先して同期の競合を解決するウィニング・ルールを指定できます。Mobile サーバーの同期の競合は、次のいずれかの状況で検出されます。

- クライアントとサーバーで同一行が更新された場合。
- クライアントとサーバーの両方で同じ主キーを持つ行を作成した場合。
- クライアントが削除した行をサーバーが更新した場合。
- クライアントが更新した行をサーバーが削除した場合。これは、Oracle データベースのアドバンスド・レプリケーションとの互換性が理由の同期エラーとみなされます。
- クライアントのデータが実表に直接適用されない遅延データ処理があるシステム（たとえば、3 層アーキテクチャ）の場合、クライアントが最初に行を挿入し、次に同じ行を更新する場合に、実表にまだ行が挿入されていない状況が発生する可能性があります。この場合、C\$ALL_CONFIG の DEF_APPLY パラメータが TRUE に設定されていると、UPDATE でなく INSERT 操作が実行されます。結果として発生する主キー競合を解決するのは、アプリケーション開発者の責任です。ただし、DEF_APPLY が設定されていない場合、「NO DATA FOUND」例外が発生します（同期エラーの処理は後述を参照してください）。
- NULL の使用違反や外部キー制約違反などのその他のエラーはすべて同期エラーです。
- 同期エラーが自動的に解決されない場合、対応するトランザクションがロールバックされ、トランザクション操作は、C\$EQ 内の Mobile サーバー・エラー・キューに移動されます。データは CEQ\$ に格納されます。Mobile サーバーのデータベース管理者は、これらのトランザクション操作を変更して再実行したり、エラー・キューからバージできます。

6.2.8.3 表の比率の使用

表の比率は、パブリケーションとパブリケーション・アイテムを関連付ける整数プロパティです。Mobile サーバーは、表の比率を使用して、各パブリケーション内のマスター表にクライアント操作を適用する順序を次のように決定します。

1. 最初に、クライアントの INSERT 操作が、表の比率の低いものから高いものへと順に実行されます。
2. 次に、クライアントの DELETE 操作が、表の比率の高いものから低いものへと順に実行されます。
3. 最後に、クライアントの UPDATE 操作が、表の比率の低いものから高いものへと順に実行されます。
4. この値には、1 ～ 1023 の整数を割り当てる必要があります。

表の比率は、特定のパブリケーション内のパブリケーション・アイテムに対して適用されます。たとえば、1つのパブリケーションに、比率2のパブリケーション・アイテムが2つ以上存在することができます。この場合、同一パブリケーション内の、比率の低いすべてのパブリケーション・アイテムに対して INSERT 操作が実行されます。

6.2.9 ユーザーの作成

Sample11 では、新規ユーザーを作成する前に、`dropUser()` を使用してユーザーを削除します。これは、新しいユーザー ID を作成する前に、疑似ユーザー ID を一掃する役目を果たします。詳細は、[6.2.10 項「ユーザーの削除」](#)を参照してください。この関数のパラメータでは、大 / 小文字は区別されません。

6.2.9.1 createUser

`createUser` の構文は、次のとおりです。

```
public static boolean createUser
    (String userName,
     String password,
     String fullName,
     String privilege) throws Throwable;
```

`createUser` のパラメータを[表 6-10](#)に示します。

表 6-10 createUser のパラメータ

パラメータ	定義
userName	Mobile クライアントのユーザー名を定義します。
password	ユーザー名に対応するパスワードを定義します。

表 6-10 createUser のパラメータ（続き）

パラメータ	定義
fullName	オプションです。たとえば John Smith のように、ユーザーのフルネームを指定します。
privilege	このパラメータは、Mobile サーバーのユーザー権限を定義します。次のいずれかの値となります。 <ul style="list-style-type: none">■ O - アプリケーションをパブリッシュします。■ U - Mobile サーバーに接続します。■ A - Mobile サーバーを管理します。■ NULL - 権限がないことを表します。

次の例では、表 6-10 に示されたパラメータを使用してユーザー「S11U1」を作成します。

例

```
oracle.mobile.admin.ResourceManager.createUser("S11U1","manager","John Smith","C")
```

6.2.10 ユーザーの削除

dropUser 関数を使用して、既存の Mobile サーバー・ユーザーを削除できます。この関数のパラメータでは、大 / 小文字は区別されません。

6.2.10.1 dropUser

dropUser の構文は、次のとおりです。

```
public static void dropUser(String userName);
```

dropUser のパラメータを[表 6-11](#)に示します。

表 6-11 dropUser のパラメータ

パラメータ	定義
userName	Mobile クライアントのユーザー名を指定します。

次の例では、ユーザー「S11U1」を削除します。

例

```
oracle.mobile.admin.ResourceManager.dropUser("S11U1");
```

6.2.11 パブリケーションへのユーザーのサブスクライブ

CreateSubscription 関数を使用すると、ユーザーをパブリケーションにサブスクライブできます。

6.2.11.1 CreateSubscription

CreateSubscription の構文は、次のとおりです。

```
public static void CreateSubscription
    (String publication,
     String clientid) throws Throwable
```

表 6-12 CreateSubscription のパラメータ

パラメータ	定義
publication	サブスクライブ先となるパブリケーションを指定します。
clientid	パブリケーションにサブスクライブするユーザーを指定します。

次の例では、表 6-12 に示されたパラメータを使用して、クライアント S11U1 をパブリケーション T_SAMPLE11 にサブスクライブします。

例

```
Consolidator.CreateSubscription("T_SAMPLE11", "S11U1");
```

6.2.12 サブスクリプションのインスタンス化

ユーザーをパブリケーションにサブスクライブしてから、サブスクリプションをインスタンス化してサブスクリプション処理を完了します。Mobile サーバーでサブスクリプションをインスタンス化する際、サブスクリプションの完全な内部表現が作成されます。

注意： データのサブセット化に必要なサブスクリプション・パラメータを設定する必要がある場合は、サブスクリプションをインスタンス化する前に、このパラメータの設定処理を完了する必要があります。詳細は、[6.2.5.3 項「データのサブセット化」](#)を参照してください。

6.2.12.1 InstantiateSubscription

InstantiateSubscription の構文は、次のとおりです。

```
public static void InstantiateSubscription
    (String publication,
     String clientid) throws Throwable
```

InstantiateSubscription のパラメータを表 6-13 に示します。

表 6-13 InstantiateSubscription のパラメータ

パラメータ	定義
publication	サブスクライブ先となるパブリケーションを指定します。
clientid	パブリケーションにサブスクライブするユーザーを指定します。

次の例では、表に指定した値を使用して、クライアントのサブスクリプションをパブリケーションに対してインスタンス化します。

例

```
Consolidator.InstantiateSubscription("T_SAMPLE1", "DAVIDL");
```

6.3 Consolidator のその他の標準機能

6.1 項「Consolidator API を使用したパブリッシュ・サブスクライブ・モデル」で使用している API コールは、パブリケーション、パブリケーション・アイテムおよびサブスクリプションをプログラムにより作成する場合に必要な API コールです。この項の内容は、使用頻度は低いですが、重要な内容です。

- 6.3.1 項「クライアント・デバイス・データベースの DDL 操作」
- 6.3.2 項「パスワードの変更」
- 6.3.3 項「シーケンスの作成」
- 6.3.4 項「クライアントのシーケンスのパーティション化」
- 6.3.5 項「リモート・データベース・リンクのサポート」

6.3.1 クライアント・デバイス・データベースの DDL 操作

クライアントの最初の同期時に、Mobile サーバーは、クライアントに自動的にスナップショット形式のデータベース・オブジェクトを作成します。デフォルトでは、表の主キー索引がサーバーから自動的にレプリケートされます。パブリケーション・アイテムを使用して、2 次索引を作成できます。1 次索引が必要ない場合は、パブリケーション・アイテムから明示的に削除する必要があります。特定の API の情報の詳細は、『Consolidator Admin API Specification』を参照してください。

6.3.2 パスワードの変更

Mobile サーバー・ユーザーのパスワードは、`setPassword()` 関数を使用して変更できます。

6.3.2.1 setPassword

`setPassword` の構文は、次のとおりです。

```
public static void setPassword
    (String userName,
     String newpwd) throws Throwable
```

`setPassword` のパラメータを表 6-14 に示します。

表 6-14 setPassword のパラメータ

パラメータ	定義
userName	Mobile クライアントのユーザー名を指定します。
newpwd	Mobile クライアントの新しいパスワードを指定します。

次の例では、ユーザー「MOBILE」のパスワードを変更します。

例

```
ResourceManager.setPassword("MOBILE", "MOBILENEW");
```

6.3.3 シーケンスの作成

Mobile サーバーでは、シーケンスのパーティション化およびクライアントとの同期をサポートしています。`CreateSequence` 関数を使用してシーケンスを作成できます。

6.3.3.1 CreateSequence

`CreateSequence` の構文は、次のとおりです。

```
public static void CreateSequence(String name) throws Throwable
```

ここで、name は、シーケンス名を指定する値です。次の例では、CUSTOM1 という名前のシーケンスを作成します。

例

```
Consolidator.CreateSequence("CUSTOM1");
```

コメント

C/C++ または Java で作成されたネイティブ・アプリケーションでのシーケンスのサポート方法は、Web-to-Go を使用して作成された Web ベース・アプリケーションと多少異なります。異なる点は、CreateSequence または CreateSequencePartition 関数によりシーケンス・オブジェクトは作成されませんが、シーケンス定義が C\$ALL_SEQUENCE_PARTITIONS 表に作成されることです。このシーケンス定義の作成は、パッケージ・ウィザードと Consolidator API の両方を使用して、宣言とプログラムによる方法を組み合わせて行う必要があります。次の手順を実行します。

1. パッケージ・ウィザードを使用してネイティブ・アプリケーションをパッケージ化します。
2. このネイティブ・アプリケーションを Mobile サーバー・リポジトリにパブリッシュします。
3. Consolidator API 関数の CreateSequence および CreateSequencePartition を使用します。
4. 同期をとります。これにより、**conscli.odt** データベースに C\$ALL_SEQUENCE_PARTITIONS 表が作成されます。
5. ネイティブ・アプリケーションが conscli.odt データベースの C\$ALL_SEQUENCE_PARTITIONS 表にアクセスし、CURR_VAL と INCR を取り出して、新しい数値を計算します。この数値は、アプリケーションにより C\$ALL_SEQUENCE_PARTITIONS 表に格納されます。
6. 同期をとります。これで、C\$ALL_SEQUENCE_PARTITIONS 表が同期されます。

6.3.4 クライアントのシーケンスのパーティション化

Mobile サーバー・シーケンスを使用して、クライアント上で一意の主キーを生成できます。シーケンスを使用するには、まずシーケンスを作成し、そのシーケンスを Mobile Sync ユーザーごとにパーティション化する必要があります。このようなシーケンスは、本当のシーケンス・オブジェクトではなく、実際には表内のレコードです。そのため、開発者は責任を持って、curr_val で示される現在の値が最新値に更新されていることを確認する必要があります。

6.3.4.1 CreateSequencePartition

CreateSequencePartition の構文は、次のとおりです。

```
public static void CreateSequencePartition
    (String name,
     String clientid,
     long curr_val,
     long incr) throws Throwable
```

CreateSequencePartition のパラメータを表 6-15 に示します。

表 6-15 CreateSequencePartition のパラメータ

パラメータ	定義
name	パーティション化するシーケンス名を定義します。
clientid	シーケンスの割当て先となるユーザーを定義します。
curr_val	シーケンスの初期値を定義します。
incr	シーケンスの増分値を定義します。

次の例では、CUSTOM1 という名前のシーケンスをパーティション化します。

例

```
Consolidator.CreateSequencePartition("CUSTOM1", "DAVIDL", 1000, 1);
```

注意： シーケンスが一意であることを保証するには、一意の開始位置を使用し、サーバー上のクライアント数に等しい増分値を設定します。

6.3.4.1.1 クライアント上での手動による現在の値の更新

クライアント上にパーティションを作成する場合、開発者は、SQL を使用してパーティションの curr_val を手動で更新することによって、確実に一意の値が使用されるようにする必要があります。たとえば、次のようになります。

```
UPDATE C$ALL_SEQUENCE_PARTITIONS
SET curr_val=curr_val+incr
WHERE name = 'CUSTOM1';
```

この処理は、現在の新しい値 (curr_val) が必要になるたびに、またはこのシーケンスを使用して新しい主キーが生成されるたびにクライアント上で実行する必要があります。

6.3.5 リモート・データベース・リンクのサポート

Mobile サーバー・リポジトリの外部にあるリモート・データベース・インスタンスに存在するデータベース・オブジェクトについて、パブリケーション・アイテムを定義することができます。リモート・オブジェクトのローカル・プライベート・シノニムは、Oracle データベース内に作成する必要があります。Consolidator のログイン・オブジェクトを作成するには、`<ORACLE_HOME>%Mobile%server%admin%consolidator_rmt.sql` の SQL スクリプトをリモート・スキーマに対して実行します。

次に、`CreatePublicationItem` API を使用して、シノニムをパブリッシュします。リモート・オブジェクトが、更新モードまたは高速リフレッシュ・モード（あるいはその両方）でパブリッシュする必要があるビューの場合は、リモート親表もローカルでパブリッシュする必要があります。ローカルの更新可能なビューまたは高速リフレッシュ可能なビュー（あるいはその両方）に使用されるものと同様のリモート・ビューのシノニムには、親表のヒントを提供する必要があります。

リモート・オブジェクトをパブリッシュすることにより発生する不明確な依存関係を処理するために、`DependencyHint` と `RemoveDependencyHint` という 2 つの API が追加されています。

Oracle データベースへのリモート・リンクは、リモート・リンク・プロシージャを試みる前に確立されている必要があります。詳細は、『Oracle9i SQL リファレンス』を参照してください。

注意： リモート・データベースから行う同期のパフォーマンスは、ネットワーク・スループットとリモート問合せ処理のパフォーマンスの制約を受けます。そのため、データが限定される単純なビューまたは表でリモート・データの同期を使用することが最適です。

6.3.5.1 `CreatePublicationItem` を使用したリモート・オブジェクトのシノニムのパブリッシュ

次のパラメータとともに使用する `CreatePublicationItem` API は、新しいスタンドアロンのパブリケーション・アイテムを、リモート・データベース・オブジェクトとして作成します。

構文

```
public static void CreatePublicationItem
    ((String rmt_jdbc_url),
     String name,
     String owner,
     String store,
     String refresh_mode,
     String select_stmt,
     String cbk_owner,
```



```
String cbk_name) throws Throwable
```

または、

```
public static void CreatePublicationItem
    ((Connection rmt_jdbc_conn),
    String name,
    String owner,
    String store,
    String refresh_mode,
    String select_stmt,
    String cbk_owner,
    String cbk_name) throws Throwable
```

CreatePublicationItem を使用するシノニム作成のパラメータを[表 6-16](#) に示します。

表 6-16 リモート・データベース・リンク用の CreatePublicationItem のパラメータ

パラメータ	説明
rmt_jdbc_url	リモート・データベース・インスタンスの jdbc URL を指定する文字列です。
rmt_jdbc_conn	リモート・インスタンスが存在する Oracle データベースへの接続です。
name	新しいパブリケーション・アイテム名を定義する文字列です。
owner	シノニムの所有者を指定する文字列です。
store	シノニム名を指定する文字列です。
refresh_mode	注意: リモート・オブジェクトをパブリッシュするには、そのオブジェクトのプライベート・シノニムを作成する必要があります。 リフレッシュ・モードを指定する文字列です。「F」は高速リフレッシュ、「C」は完全リフレッシュを表します。デフォルトは高速リフレッシュです。
select_stmt	新しいパブリケーションについての SELECT 文を指定する文字列です。この文はパラメータ化できます。次に示す例では、このパラメータは :CAP と指定され、パラメータ名の前にコロンが付いています。
cbk_owner	コールバック・パッケージの所有者を NULL に指定します。詳細は、 6.4.12 項「構成と適用を使用する前後に行うコールバックのカスタマイズ」 を参照してください。
cbk_name	コールバック・パッケージ名を NULL に指定します。詳細は、 6.4.12 項「構成と適用を使用する前後に行うコールバックのカスタマイズ」 を参照してください。

URL 文字列を使用すると、リモート接続が確立され、自動的にクローズされます。接続が NULL か、または接続が確立できない場合は、例外がスローされます。リモート接続情報は、リンクされたデータベース上にロギング・オブジェクトを作成したり、メタデータを抽出する場合に使用されます。

例

```
Consolidator.CreatePublicationItem(  
    "jdbc:oracle:oci8:@oracle.world",  
    "P_SAMPLE1",  
    "SAMPLE1",  
    "PAYROLL_SYN",  
    "F"  
    "SELECT * FROM sample1.PAYROLL_SYN"+"WHERE SALARY >:CAP", null, null);
```

注意： SELECT 文内では、データ・サブセットのパラメータ名の前に、コロンを接頭辞として指定する必要があります（たとえば、:CAP）。

6.3.5.2 依存関係のヒントの作成

不明確な依存関係についてのヒントが作成されます。

構文

```
public static void DependencyHint  
(String owner,  
    Sting store,  
    String owner_d,  
    String store_d) throws Throwable
```

CreateDependencyHint のパラメータを [表 6-17](#) に示します。

表 6-17 CreateDependencyHint のパラメータ

パラメータ	説明
owner	ビューの所有者を指定する文字列です。
store	ビューの名前を指定する文字列です。
owner_d	実表またはビューの所有者を指定する文字列です。
store_d	実表またはビューの名前を指定する文字列です。

例

```
Given remote view definition
    create payroll_view as
    select p.pid, e.name
    from payroll p, emp e
    where p.emp_id = e.emp_id;

Execute locally
    create synonym v_payroll_syn for payroll_view@<remote_link_address>;
    create synonym t_emp_syn for emp@<remote_link_address>;

<remote_link_address> は、Oracle データベースに確立されたリンクです。
DependencyHint は、ローカル・シノニム v_payroll_syn がローカル・シノニム
t_emp_syn. に依存することを示すために使用します。

Consolidator.DependencyHint ("SAMPLE1", "V_PAYROLL_SYN", "SAMPLE1", "T_EMP_SYN");
```

6.3.5.3 依存関係のヒントの削除

不明確な依存関係についてのヒントが削除されます。

構文

```
public static void RemoveDependencyHint
    (String owner,
     Sting store,
     String owner_d,
     String store_d) throws Throwable
```

RemoveDependencyHint のパラメータを[表 6-18](#)に示します。

表 6-18 RemoveDependencyHint のパラメータ

パラメータ	説明
owner	ビューの所有者を指定する文字列です。
store	ビューの名前を指定する文字列です。
owner_d	ベース・オブジェクトの所有者を指定する文字列です。
store_d	ベース・オブジェクトの名前を指定する文字列です。

6.4 Consolidator をカスタマイズするための拡張機能

次の機能には、ほとんどのアプリケーション設計で必要でない特殊関数が含まれています。このような機能を使用するには、Java および操作するデータベースの設計（問合せの構成、表の並べ方、適用される依存関係など）についての十分な知識が必要です。内容は次のとおりです。

- 6.4.1 項「MyCompose を使用した構成フェーズのカスタマイズ」
- 6.4.2 項「Sync Discovery API」
- 6.4.3 項「マップ表のパーティション API」
- 6.4.4 項「AlterPublicationItem を使用したパブリケーション・アイテムの変更」
- 6.4.5 項「複数表パブリケーション（ビュー）の高速リフレッシュおよび更新操作」
- 6.4.6 項「仮想主キー」
- 6.4.7 項「パブリケーション・アイテムの問合せのキャッシング」
- 6.4.8 項「ユーザー定義の PL/SQL プロシージャのバインド」
- 6.4.9 項「レプリケーションをカスタマイズするためのキュー・インタフェース」
- 6.4.10 項「Null Sync コールアウト」
- 6.4.11 項「更新可能パブリケーション・アイテムの外部キー制約」
- 6.4.12 項「構成と適用を使用する前後に行うコールバックのカスタマイズ」
- 6.4.13 項「DML 操作のコールバックのカスタマイズ」
- 6.4.14 項「制限選択条件」

6.4.1 MyCompose を使用した構成フェーズのカスタマイズ

構成フェーズでは、サーバー側の 1 つ以上の実表についての問合せが必要で、問合せで指定されているパブリケーション・アイテムについて生成された DML 操作は、クライアントにダウンロードするためにアウトキューに入れられます。Consolidator は、これらの DML 操作を、サーバー側の実表についての DML 物理ログを使用して「共通の」方法で管理します。これは、たとえば複合データのサブセット化の問合せが使用されているなど、DML 操作が複雑な場合、リソース集中型になる可能性があります。この処理をカスタマイズするツール製品には、上書き可能な `compose` メソッドを持つ拡張可能な MyCompose や、カスタマイズされたクラスを登録およびロードするための Consolidator API などがあります。

6.4.1.1 ユーザー定義サブクラスとしての MyCompose の拡張

MyCompose は、ユーザー定義サブクラスを作成するためのスーパークラスとなる抽象クラスです。たとえば、次のようになります。

ItemACompose

```
public class ItemACompose extends oracle.lite.sync.MyCompose
{
    ...
}
```

ユーザー定義クラスは、実表の DML ログを解析することにより、クライアント・デバイスに送信される、パブリケーション・アイテムの DML 操作を生成します。拡張 MyCompose サブクラスは、パブリケーション・アイテムに登録され、そのパブリケーション・アイテムに関する構成フェーズのすべての操作を継承します。拡張 MyCompose クラスは、十分な共通性があれば、2 つ以上のパブリケーション・アイテムに登録できますが、内部的には各パブリケーション・アイテムに対して一意の拡張クラスのインスタンスが存在します。

6.4.1.2 MyCompose のプライマリ・メソッド

MyCompose クラスでは、構成フェーズをカスタマイズするために、needCompose、doCompose、init および destroy メソッドが使用されます。カスタマイズしたサブクラスで、これらのメソッドのいずれかを上書きして、構成フェーズの操作をカスタマイズすることができます。一度に 1 クライアントずつ構成フェーズをカスタマイズする場合は、doCompose および needCompose を使用することをお勧めします。init および destroy メソッドは、個々のクライアント処理の前後に、すべてのクライアントに対してある処理を実行する必要がある場合に使用します。これ以外にも、[6.4.1.3 項「MyCompose の補助メソッド」](#)に説明されているメソッドがあります。その項には、ここで説明する 4 つのメソッドの使用方法について役立つ情報も記載されています。

6.4.1.2.1 needCompose メソッド

このメソッドは、ダウンロードする特定のパブリケーション・アイテムについての変更内容を持つクライアントの識別に使用します。このメソッドは、doCompose をトリガーする手段として有効です。

構文

```
public int needCompose(Connection conn,
    String clientid) throws Throwable
```

needCompose のパラメータを表 6-19 に示します。

表 6-19 needCompose のパラメータ

パラメータ	定義
conn	Mobile サーバー・リポジトリへのデータベース接続です。
clientid	データベースに接続するクライアントを指定します。

次の例は、クライアントの実表の変更を調べます。この例では、「使用済」レコードの有無を調べます。変更がある場合、このメソッドは doCompose メソッドをトリガーする MyCompose.YES を返します。

例

```
public int needCompose(String clientid) throws Throwable{
    boolean baseDirty = false;
    String [][] baseTables = this.getBaseTables();

    for(int i = 0; i < baseTables.length; i++){
        if(this.baseTableDirty(baseTables[i][0], baseTables[i][1])){
            baseDirty = true;
            break;
        }
    }

    if(baseDirty){
        return MyCompose.YES;
    }else{
        return MyCompose.NO;
    }
}
```

このサンプル・コードは、needCompose メソッドを上書きし、[6.4.1.3 項「MyCompose の補助メソッド」](#)に示されている補助メソッドを使用して、パブリケーション・アイテム内のいずれかの表に、クライアントに送信する必要のある変更内容が含まれているかどうかを調べます。この例では、実表を取り出し、変更（「使用済」レコード）の有無を調べます。このテストの結果が TRUE の場合、doCompose のコールをトリガーする「Yes」が返されます。

6.4.1.2.2 doCompose メソッド

このメソッドは、クライアントのサブスクライブ元となる特定のパブリケーション・アイテムについての DML ログ表を移入します。

構文

```
public int doCompose(Connection conn,
    String clientid) throws Throwable
```

doCompose のパラメータを表 6-20 に示します。

表 6-20 doCompose のパラメータ

パラメータ	定義
conn	Mobile サーバー・リポジトリへのデータベース接続です。
clientid	データベースに接続するクライアントを指定します。

次の例には、実表を 1 つのみ持つパブリケーション・アイテムが含まれています。このパブリケーション・アイテムに対しては、実表を対象とした DML（挿入、更新または削除）操作が実行されます。このメソッドは、パブリケーション・アイテムをサブスクライブしているクライアントごとにコールされます。

例

```
public int doCompose(Connection conn, String clientid) throws Throwable {
    int rowCount = 0;

    String [][] baseTables = this.getBaseTables();
    String baseTableDMLLogName =
        this.getBaseTableDMLLogName(baseTables[0][0], baseTables[0][1]);
    String baseTablePK =
        this.getBaseTablePK(baseTables[0][0], baseTables[0][1]);
    String pubItemDMLTableName = this.getPubItemDMLTableName();

    String sql = "INSERT INTO " + pubItemDMLTableName
        + " SELECT " + baseTablePK + ", DMLTYPE$$ FROM " +
        baseTableDMLLogName;

    Statement st = conn.createStatement();
    rowCount = st.executeUpdate(sql);
    st.close();
    return rowCount;
}
```

このサンプル・コードは、doCompose メソッドを上書きし、[6.4.1.3 項「MyCompose の補助メソッド」](#)に示されている補助メソッドを使用して、SQL 文を作成します。このサンプルを使用し、適切な get メソッドを使用して、MyCompose により、実表、実表の主キー、実表の DML ログ名およびパブリケーション・アイテムの DML 表名を取り出します。これらのメソッドにより返された表名およびその他の情報を使用すると、実表の DML ログから実表の主キーの内容に含まれているパブリケーション・アイテム、および DML 操作を挿入する動的 SQL 文（「sql」）を作成できます。

6.4.1.2.3 init メソッド

このメソッドは、ユーザーが作成した構成準備プロセスのフレームワークとなります。init メソッドは、個々のクライアントの構成フェーズ より前に、すべてのクライアントに対して 1 回のみコールされます。デフォルトの実装は作用しません。

構文

```
public void init(Connection conn)
```

init のパラメータを[表 6-21](#)に示します。

表 6-21 init のパラメータ

パラメータ	定義
conn	Mobile サーバー・リポジトリへのデータベース接続です。

6.4.1.2.4 destroy メソッド

このメソッドは、ユーザーが作成した構成クリーンアップ・プロセスのフレームワークとなります。destroy メソッドは、個々のクライアントの構成フェーズ の後に、すべてのクライアントに対して 1 回のみコールされます。デフォルトの実装は作用しません。

構文

```
public void destroy(Connection conn)
```

destroy のパラメータを[表 6-21](#)に示します。

表 6-22 destroy のパラメータ

パラメータ	定義
conn	Mobile サーバー・リポジトリへのデータベース接続です。

6.4.1.3 MyCompose の補助メソッド

次のメソッドは、MyCompose のプライマリ・メソッドで使用する情報を返します。

6.4.1.3.1 getPublication

このメソッドは、パブリケーションの名前を返します。

構文

```
public String getPublication()
```

6.4.1.3.2 getPublicationItem

このメソッドは、パブリケーション・アイテムの名前を返します。

構文

```
public String getPublicationItem()
```

6.4.1.3.3 getPubItemDMLTableName

このメソッドは、doCompose または init が挿入されることが想定される DML 表または DML 表ビューの名前（スキーマ名など）を返します。

構文

```
public String getPubItemDMLTableName()
```

戻り値は、動的 SQL 文に埋め込むことができます。表またはビューの構造は次のとおりです。

```
<PubItem PK> DMLTYPE$$
```

getPubItemDMLTableName のパラメータを表 6-23 に示します。

表 6-23 getPubItemDMLTableName のビュー構造のパラメータ

パラメータ	定義
PubItemPK	getPubItemPK() により返される値です。
DMLTYPE\$\$	値は、挿入を表す「I」、削除を表す「D」、または更新を表す「U」のいずれかです。

6.4.1.3.4 getPubItemPK

このメソッドは、リストされたパブリケーションの主キーを、カンマで区切られた形式 (<col1>,<col2>,<col3>) で返します。

構文

```
public String getPubItemPK() throws Throwable
```

6.4.1.3.5 getBaseTables

このメソッドは、パブリケーション・アイテムについてのすべての実表を、2つの文字列で構成された配列で返します。2つの文字列で構成される各配列には、実表のスキーマおよび名前が入っています。親表は常に最初に返される実表、つまり `baseTables[0]` です。

構文

```
public string [][] getBaseTables() throws Throwable
```

6.4.1.3.6 getBaseTablePK

このメソッドは、リストされた実表の主キーを、カンマで区切られた形式 (`<col1>,<col2>,<col3>`) で返します。

構文

```
public String getBaseTablePK  
(String owner,  
String baseTable) throws Throwable
```

`getBaseTablePK` のパラメータを表 6-24 に示します。

表 6-24 getBaseTablePK のパラメータ

パラメータ	定義
owner	実表の所有者のスキーマ名です。
baseTable	実表の名前です。

6.4.1.3.7 baseTableDirty

このメソッドは、実表に同期を必要とする変更があるかどうかを示すブール値を返します。

構文

```
public boolean baseTableDirty(String owner, String store)
```

`baseTableDirty` のパラメータを表 6-25 に示します。

表 6-25 baseTableDirty のパラメータ

パラメータ	定義
owner	実表のスキーマ名です。
store	実表の名前です。

6.4.1.3.8 getBaseTableDMLLogName

このメソッドは、DML 物理ログ表または実表の DML ログ表ビューの名前を返します。

構文

```
public string getBaseTableDMLLogName(String owner, String baseTable)
```

getBaseTableDMLLogName のパラメータを表 6-26 に示します。

表 6-26 getBaseTableDMLLogName のパラメータ

パラメータ	定義
owner	実表の所有者のスキーマ名です。
baseTable	実表の名前です。

戻り値は、動的 SQL 文に埋め込むことができます。パブリケーション・アイテムに複数の実表がある場合は、複数の物理ログが存在する可能性があります。親実表の物理主キーは、パブリケーション・アイテムの主キーに対応します。ログの構造は次のとおりです。

```
<Base Table PK> DMLTYPE$$
```

getBaseTableDMLLogName のビュー構造のパラメータを表 6-27 に示します。

表 6-27 getBaseTableDMLLogName のビュー構造のパラメータ

パラメータ	定義
Base Table PK	親実表の主キーです。
DMLTYPE\$\$	値は、挿入を表す「I」、削除を表す「D」、または更新を表す「U」のいずれかです。

6.4.1.3.9 getMapView()

このメソッドは、動的 SQL 文で使用できるマップ表のビューを返します。このマップ表には、各クライアント・デバイスの主キーのリストが入っています。ビューはインライン・ビューになる場合もあります。

構文

```
public String getMapView() throws Throwable
```

マップ表のビューの構造は次のとおりです。

```
CLID$$CS <Pub Item PK> DMLTYPE$$
```

マップ表のビューのパラメータを表 6-28 に示します。

表 6-28 getMapView のビュー構造のパラメータ

パラメータ	定義
CLID\$\$CS	クライアント ID 列です。
Pub Item PK	パブリケーション・アイテムの主キー列です。
DMLTYPE\$\$	値は、挿入を表す「I」、削除を表す「D」、または更新を表す「U」のいずれかです。

6.4.1.4 MyCompose サブクラスを登録するための Consolidator API メソッド

サブクラスを作成した後は、パブリケーションに登録する必要があります。パブリケーション・アイテムにサブクラスを追加したり、削除できるようにするために、RegisterMyCompose と DeRegisterMyCompose という 2 つのメソッドが Consolidator API に追加されています。

6.4.1.4.1 RegisterMyCompose メソッド

RegisterMyCompose メソッドは、サブクラスを登録し、クラスのバイトコードを含め、それを Mobile サーバー・リポジトリにロードします。コードをリポジトリにロードすることにより、実行時にロードすることなくサブクラスを使用できます。

構文

```
public static void RegisterMyCompose
( String publication,
  String pubItem,
  String className,
  boolean reloadBytecode) throws Throwable
```

RegisterMyCompose のパラメータを表 6-29 に示します。

表 6-29 RegisterMyCompose のパラメータ

パラメータ	定義
publication	パブリケーション・アイテムが含まれているパブリケーションの名前です。
pubItem	サブクラスの登録先となるパブリケーション・アイテムの名前です。
className	カスタマイズされた MyCompose サブクラスの名前です。
reloadBytecode	この値が TRUE の場合、Mobile サーバー・リポジトリ内にあるクラスの既存のバイトコードが上書きされます。

6.4.1.4.2 DeRegisterMyCompose

DeRegisterMyCompose メソッドは、Mobile サーバー・リポジトリからサブクラスを削除します。

構文

```
public static void DeRegisterMyCompose
( String publication,
  String pubItem,
  boolean removeBytecode) throws Throwable
```

DeRegisterMyCompose のパラメータを表 6-30 に示します。

表 6-30 DeRegisterMyCompose のパラメータ

パラメータ	定義
publication	パブリケーション・アイテムが属するパブリケーションの名前です。
pubItem	サブクラスの登録先となるパブリケーション・アイテムの名前です。
removeBytecode	この値が TRUE の場合、Mobile サーバー・リポジトリ内にあるクラスの既存のバイトコードが削除されます。バイトコードが削除されると、このクラスに登録されているすべてのパブリケーション・アイテムの登録が削除されます。

6.4.2 Sync Discovery API

Sync Discovery 機能は、特定のクライアントを対象としたダウンロードのサイズを履歴データに基づいて見積もるように要求する場合に使用します。履歴データのメンテナンスを行うために、次の統計値が収集されます。

- パブリケーション・アイテムごとに送信された行の合計数
- これらの行のデータの合計サイズ
- これらの行の圧縮データのサイズ

6.4.2.1 getDownloadInfo メソッド

API は、DownloadInfo オブジェクトを返す getDownloadInfo メソッドで構成されます。DownloadInfo オブジェクトには、一連の PublicationSize オブジェクトと access メソッドが含まれています。PublicationSize オブジェクトは、パブリケーション・アイテムのサイズ情報を扱います。Iterator iterator() メソッドは、DownloadInfo オブジェクト内の各 PublicationSize オブジェクトを表示する場合に使用できます。

構文

```
public DownloadInfo getDownloadInfo
(String clientid,
  boolean uncompressed,
  boolean completeRefresh)
```

getDownloadInfo のパラメータを表 6-31 に示します。

表 6-31 getDownloadInfo のパラメータ

パラメータ	説明
clientid	クライアントの名前です。
uncompressed	TRUE に設定すると、データ・オブジェクトのサイズが返されます。FALSE に設定すると、圧縮後のデータ・オブジェクトのサイズが返されます。
completeRefresh	TRUE に設定すると、リフレッシュ・モードとは関係なく、完全リフレッシュ時に同期がとられるすべての行のサイズが返されます。

例

```
DownloadInfo dl = Consolidator.getDownloadInfo("S11U1", true, true);
```

6.4.2.2 DownloadInfo クラスの access メソッド

DownloadInfo クラスにより提供される access メソッドを表 6-32 に示します。

表 6-32 DownloadInfo クラスの access メソッド

メソッド	定義
<code>public Iterator iterator ()</code>	このメソッドは、ユーザーが DownloadInfo オブジェクト内に含まれているすべての PublicationSize オブジェクトを横断できるように、イテレータ・オブジェクトを返します。
<code>public long getTotalSize ()</code>	このメソッドは、すべての PublicationSize オブジェクトのサイズ情報を、バイト単位で返します。また、ユーザーがサブスクライブするすべてのパブリケーション・アイテムのサイズを、拡張子順に返します。パブリケーション・アイテムに履歴情報がない場合は、「-1」を返します。
<code>public long getPubSize (String pubName)</code>	このメソッドは、文字列 pubName により参照されるパブリケーションに属するすべてのパブリケーション・アイテムのサイズを返します。パブリケーション・アイテムに履歴情報がない場合は、「-1」を返します。
<code>public long getPubRecCount (String pubName)</code>	このメソッドは、次の同期時に同期される文字列 pubName により参照されるパブリケーションに属するすべてのパブリケーション・アイテムのレコードの合計数を返します。
<code>public long getPubItemSize (String pubItemName)</code>	このメソッドは、pubItemName により参照される特定のパブリケーション・アイテムのサイズを返します。このメソッドは、次の規則に従います。 <ol style="list-style-type: none">1. パブリケーション・アイテムが空の場合は、「0」を返します。2. パブリケーション・アイテムに履歴情報がない場合は、「-1」を返します。
<code>public long getPubItemRecCount (String pubItemName)</code>	このメソッドは、pubItemName により参照される、特定のパブリケーション・アイテムのレコード数を返します。レコード数の同期は、次の同期時に行われます。

6.4.2.3 PublicationSize クラス

PublicationSize クラスにより提供される access メソッドを表 6-33 に示します。

表 6-33 PublicationSize クラスの access メソッド

パラメータ	定義
public String getPubName ()	パブリケーション・アイテムを含んでいるパブリケーションの名前を返します。
public String getPubItemName ()	PublicationSize オブジェクトにより参照されるパブリケーション・アイテムの名前を返します。
public long getSize ()	PublicationSize オブジェクトにより参照されるパブリケーション・アイテムの合計サイズを返します。
public long getNumOfRows ()	次の同期時に同期されるパブリケーション・アイテムの行数を返します。

サンプル・コード

```
import java.sql.*;
import java.util.Iterator;
import java.util.HashSet;

import oracle.lite.sync.ConsolidatorManager;
import oracle.lite.sync.DownloadInfo;
import oracle.lite.sync.PublicationSize;

public class TestGetDownloadInfo
{

    public static void main(String argv[]) throws Throwable
    {

        // Open Consolidator connection
        try
        {

            // Create a ConsolidatorManager object
            ConsolidatorManager cm = new ConsolidatorManager ();
            // Open a Consolidator connection
            cm.OpenConnection ("MOBILEADMIN", "MANAGER",
                               "jdbc:oracle:thin:@server:1521:orcl", System.out);
            // Call getDownloadInfo
            DownloadInfo dlInfo = cm.getDownloadInfo ("S11U1", true, true);
            // Call iterator for the Iterator object and then we can use that to transverse
            // through the set of PublicationSize objects.
            Iterator it = dlInfo.iterator ();
            // A temporary holder for the PublicationSize object.
```



```
        PublicationSize ps = null;
// A temporary holder for the name of all the Publications in a HashSet object.
        HashSet pubNames = new HashSet ();
// A temporary holder for the name of all the Publication Items in a HashSet
// object.
        HashSet pubItemNames = new HashSet ();
// Traverse through the set.
        while (it.hasNext ())
        {
// Obtain the next PublicationSize object by calling next ().
            ps = (PublicationSize)it.next ();

// Obtain the name of the Publication this PublicationSize object is associated
// with by calling getPubName ().
            pubName = ps.getPubName ();
            System.out.println ("Publication: " + pubName);

// We save pubName for later use.
            pubNames.add (pubName);

// Obtain the Publication name of it by calling getPubName ().
            pubItemName = ps.getPubItemName ();
            System.out.println ("Publication Item Name: " + pubItemName);

// We save pubItemName for later use.
            pubItemNames.add (pubItemName);

// Obtain the size of it by calling getSize ().
            size = ps.getSize ();
            System.out.println ("Size of the Publication: " + size);

// Obtain the number of rows by calling getNumOfRows ().
            numOfRows = ps.getNumOfRows ();
            System.out.println ("Number of rows in the Publication: "
                                + numOfRows);
        }

// Obtain the size of all the Publications contained in the
// DownloadInfo objects.
        long totalSize = dlInfo.getTotalSize ();
        System.out.println ("Total size of all Publications: " + totalSize);

// A temporary holder for the Publication size.
        long pubSize = 0;

// A temporary holder for the Publication number of rows.
        long pubRecCount = 0;
```

```
// A temporary holder for the name of the Publication.
String tmpPubName = null;

// Transverse through the Publication names that we saved earlier.
it = pubNames.iterator ();
while (it.hasNext ())
{
// Obtain the saved name.
tmpPubName = (String) it.next ();

// Obtain the size of the Publication.
pubSize = dlInfo.getPubSize (tmpPubName);
System.out.println ("Size of " + tmpPubName + ": " + pubSize);

// Obtain the number of rows of the Publication.
pubRecCount = dlInfo.getPubRecCount (tmpPubName);
System.out.println ("Number of rows in " + tmpPubName + ": "
+ pubRecCount);
}

// A temporary holder for the Publication Item size.
long pubItemSize = 0;

// A temporary holder for the Publication Item number of rows.
long pubItemRecCount = 0;

// A temporary holder for the name of the Publication Item.
String tmpPubItemName = null;

// Traverse through the Publication Item names that we saved earlier.
it = pubItemNames.iterator ();
while (it.hasNext ())
{
// Obtain the saved name.
tmpPubItemName = (String) it.next ();

// Obtain the size of the Publication Item.
pubItemSize = dlInfo.getPubItemSize (tmpPubItemName);
System.out.println ("Size of " + pubItemSize + ": " + pubItemSize);

// Obtain the number of rows of the Publication Item.
pubItemRecCount = dlInfo.getPubItemRecCount (tmpPubItemName);
System.out.println ("Number of rows in " + tmpPubItemName + ": "
+ pubItemRecCount);
}
System.out.println ();
```

```
// Close the connection
    cm.CloseConnection ();
}
catch (Exception e)
{
    e.printStackTrace();
}
}
```

6.4.3 マップ表のパーティション API

マップ表と呼ばれる Consolidator データベース・オブジェクトは、各 Mobile クライアントの状態のメンテナンスに使用します。クライアント数が多く、各クライアントが大量のデータをサブスクライブする場合、マップ表のサイズは非常に大きくなり、拡張性の問題が発生する可能性があります。次に説明する API を使用すると、マップ表をクライアント ID 別にパーティション化でき、マップ表が管理しやすくなります。

この API では、マップ表のパーティションの作成、パーティションの追加、1 つまたはすべてのパーティションの削除、マップ表のパーティションのマージが可能です。マップ表のパーティションは、ALL_PARTITIONS データベース・カタログ・ビューを使用して監視できます。

注意： このようなパーティション化は、Oracle サーバーが提供するパーティション機能とは関係なく、Oracle9i Lite でのみ使用されます。

6.4.3.1 マップ表のパーティションの作成

参照パブリケーション・アイテムのマップ表に対応するパーティションを作成します。マップ表にデータがある場合、そのデータは作成するパーティションに転送されます。パーティションが正常に作成された後、SQL コマンド TRUNCATE TABLE を使用して、マップ表を切り捨て、冗長なデータを削除できます。

構文

```
public static void PartitionMap
    (String pub_item,
     int num_parts,
     String storage,
     String ind_storage) throws Throwable
```

PartitionMap のパラメータを [表 6-34](#) に示します。

表 6-34 PartitionMap のパラメータ

パラメータ	定義
pub_item	マップ表がパーティション化されるパブリケーション・アイテムです。
num_parts	パーティションの数です。
storage	記憶域パラメータを指定する文字列です。このパラメータには、SQL コマンド CREATE TABLE と同じ構文を使用する必要があります。詳細は、『Oracle9i SQL リファレンス』を参照してください。
ind_storage	パーティションについての索引の記憶域パラメータを指定する文字列です。このパラメータには、SQL コマンド CREATE INDEX と同じ構文を使用する必要があります。詳細は、『Oracle9i SQL リファレンス』を参照してください。

例

```
Consolidator.PartitionMap("P_SAMPLE1", "5", "tablespace mobileadmin", "initrans 10
pctfree 70");
```

6.4.3.2 マップ表のパーティションの追加

参照パブリケーション・アイテムのマップ表に対応するパーティションを追加します。マップ表にデータがある場合、そのデータは作成するパーティションに転送されます。パーティションが正常に作成された後、SQL コマンド TRUNCATE TABLE を使用して、マップ表を切り捨て、冗長なデータを削除できます。

構文

```
public static void AddMapPartition
( String pub_item,
  int num_parts,
  String storage,
  String ind_storage) throws Throwable
```

AddMapPartition のパラメータを [表 6-35](#) に示します。

表 6-35 AddMapPartition のパラメータ

パラメータ	定義
pub_item	マップ表がパーティション化されるパブリケーション・アイテムです。
num_parts	パーティションの数です。
storage	記憶域パラメータを指定する文字列です。このパラメータには、SQL コマンド CREATE TABLE と同じ構文を使用する必要があります。詳細は、『Oracle9i Lite SQL リファレンス』を参照してください。
ind_storage	パーティションについての索引の記憶域パラメータを指定する文字列です。このパラメータには、SQL コマンド CREATE INDEX と同じ構文を使用する必要があります。詳細は、『Oracle9i Lite SQL リファレンス』を参照してください。

例

```
Consolidator.PartitionMap("P_SAMPLE1", "5", "tablespace mobileadmin", "intitrans 10
pctfree 40");
```

6.4.3.3 マップ表のパーティションの削除

指定された名前のパーティションを削除します。次の例のパラメータ `partition` は、パーティションの名前です。パーティション名は Consolidator が付けるため、ALL_PARTITIONS 表ビュー CV\$ALL_PARTITIONS の問合せによりパーティション名を取得する必要があります。

構文

```
public static void DropMapPartition( String partition) throws Throwable
```

例

```
Consolidator.DropMapPartition("MAP101_1");
```

6.4.3.4 マップ表の全パーティションの削除

指定されたパブリケーション・アイテムに対応するマップ表のすべてのパーティションを削除します。

構文

```
public static void DropAllMapPartitions( String pub_item) throws Throwable
```

例

```
Consolidator.DropAllMapPartitions("P_SAMPLE1");
```

6.4.3.5 マップ表のパーティションのマージ

データをパーティション間でマージします。パーティション名は Consolidator が付けるため、ALL_PARTITIONS 表ビュー CV\$ALL_PARTITIONS の問合せにより、パーティション名を取得する必要があります。

構文

```
public static void MergeMapPartitions  
    ( String from_partition,  
      String to_partiton) throws Throwable
```

例

```
Consolidator.MergeMapPartition("MAP101_1", "MAP101_2");
```

6.4.4 AlterPublicationItem を使用したパブリケーション・アイテムの変更

既存のパブリケーション・アイテムに列を追加できます。これらの新しい列は、次に同期される時点で、全サブスクライブ・クライアントにプッシュされます。これは変更済全パブリケーション・アイテムの完全リフレッシュで達成されます。

- 管理者は、複数の列を追加できます。
- この機能は、すべてのクライアント形式に対してサポートされます。
- クライアントは、サーバーにスナップショット情報をアップロードしません。これは、クライアントがクライアント・データベースでスナップショットを直接変更できないことを意味します。たとえば、EPOC 上で Mobile SQL を使用して表を変更することはできません。
- パブリケーション・アイテムのアップグレードは、高優先順位の同期中は遅延されます。これは、ワイヤレスなどの低帯域幅ネットワークの場合に必要です。パブリケーション・アイテムのアップグレードには常に、変更済パブリケーション・アイテムの完全リフレッシュが必要なためです。高優先順位フラグが設定されている間、高優先順位のクライアントは古いパブリケーション・アイテム形式を受信し続けます。
- サーバーは、変更されているパブリケーション・アイテムのバージョンを最大 2 個サポートする必要があります。

6.4.4.1 パブリケーション・アイテムの変更

これにより、既存のパブリケーション・アイテムに列を追加できます。WHERE 句を変更してもかまいませんが、サブスクリプション・パラメータを追加することはできません。

構文

```
public static void AlterPublicationItem
    (String name,
     String select_stmt)
    throws Throwable
```

AlterPublicationItem のパラメータを表 6-36 に示します。

表 6-36 AlterPublicationItem のパラメータ

パラメータ	説明
name	パブリケーション・アイテム名を指定する文字列です。
select_stmt	追加列の含まれた新規パブリケーション・アイテムの SELECT 文です。

例

```
Consolidator.AlterPublicationItem("P_SAMEPLE1", "select * from EMP");
```

6.4.5 複数表パブリケーション（ビュー）の高速リフレッシュおよび更新操作

Mobile サーバーでは、特定の条件を満たすビューと呼ばれる複合複数表パブリケーション・アイテムに対する高速リフレッシュ操作と更新操作をサポートしています。高速リフレッシュ中は増分変更の同期がとられ、完全リフレッシュ中は現在のデータがすべてリフレッシュされます。リフレッシュ・モードは、CreatePublicationItem API コールを使用してパブリケーション・アイテムを作成する際に設定されます。リフレッシュ・モードを変更するには、パブリケーション・アイテムを削除してから、適切なモードを使用してパブリケーション・アイテムを再作成する必要があります。

6.4.5.1 更新可能な親表

ビューを更新可能にするには、親表が必要です。親表は、ビューの任意の実表です。実表のビューの列リストには主キーが含まれており、これはビューの行セットで一意です。ビューを更新可能にする場合は、ビューでパブリケーション・アイテムを作成する前に、Mobile サーバーに適切なヒントとビューの親表を指定する必要があります。

6.4.5.2 親表のヒントと INSTEAD OF トリガーの使用

ビューに基づいてパブリケーション・アイテムを更新可能にするには、次の 2 つの方式を使用する必要があります。

- 親表のヒント
- INSTEAD OF トリガーまたは DML プロシージャのコールアウト

6.4.5.2.1 親表のヒントの作成

親表のヒントは、指定されたビューの親表を定義します。親表のヒントは、ParentHint 関数を使用して指定します。この関数の構文は、次のとおりです。

```
public static void ParentHint
    (String owner,
     Sting store,
     String owner_d,
     String store_d) throws Throwable
```

ParentHint のパラメータを表 6-37 に示します。

表 6-37 ParentHint のパラメータ

パラメータ	説明
owner	ビューの所有者を指定する文字列です。
store	ビューの名前を指定する文字列です。
owner_d	ベース・オブジェクトの所有者を指定する文字列です。
store_d	ベース・オブジェクトの名前を指定する文字列です。

例

```
Consolidator.ParentHint ("SAMPLE3", "ADDROLRL4P", "SAMPLE3", "ADDRESS");
```

6.4.5.2.2 INSTEAD OF トリガー

INSTEAD OF トリガーは、INSTEAD OF INSERT、INSTEAD OF UPDATE または INSTEAD OF DELETE の各コマンドを実行するために使用します。さらに INSTEAD OF トリガーは、ビューの実表に対して実行される操作にこれらの DML コマンドをマップします。INSTEAD OF トリガーは、Oracle データベースの機能です。INSTEAD OF トリガーの詳細は、Oracle データベースのドキュメントを参照してください。

6.4.5.3 ビューの高速リフレッシュ

パブリケーション・アイテムは、デフォルトでは高速リフレッシュ用に作成されます。高速リフレッシュでは、増分変更のみがレプリケートされます。高速リフレッシュの利点は、同期セッション間での変更がかざられている場合に大量のデータを持つデータ・ストアをレプリケートするときのオーバーヘッドが軽減され速度が向上することです。

ビューが次の条件を満たす場合、Mobile サーバーはビューの高速リフレッシュを実行します。

- 各ビューの実表には主キーが必要です。
- すべての実表の主キーは、すべてビューの列リストに含まれている必要があります。
- アイテムがビューで、アイテムの選択条件に複数の表が含まれている場合、選択条件定義に含まれているすべての表が主キーを持ち、対応するパブリケーション・アイテムを持っている必要があります。

ビューでは、親表に対してのみ一意の主キーが必要です。その他の表の主キーは重複してもかまいません。実表の主キー列ごとに、ビューの列名に関するヒントを Mobile サーバーに提供する必要があります。これは、PrimaryKeyHint を使用して実現できます。

6.4.5.3.1 PrimaryKeyHint

PrimaryKeyHint の構文は、次のとおりです。

```
public static void PrimaryKeyHint
(String publication_item,
String column,
String b_owner,
String b_store,
String b_column) throws Throwable
```

PrimaryKeyHint のパラメータを [表 6-38](#) に示します。

表 6-38 PrimaryKeyHint のパラメータ

パラメータ	説明
publication_item	主キーのヒントのマッピング先となるパブリケーション・アイテムの名前です。
column	パブリケーション・アイテムの列名です。
b_owner	親表の所有者です。
b_store	親表の名前です。
b_column	親表の主キー列の名前です。

例

```
Consolidator.PrimaryKeyHint("P_SAMPLE1", "Last Name", "SAMPLE1", "ADDROLRL4P", "First Name");
```

6.4.5.4 ビューの完全リフレッシュ

パブリケーション・アイテムは、Consolidator API の Complete Refresh コールを使用して、完全リフレッシュ用に作成できます。このモードを指定した場合、クライアント・データは同期のたびにサーバーの現在のデータで完全にリフレッシュされます。管理者は、API コールを介して、パブリケーション全体に完全リフレッシュを強制できます。完全リフレッシュ関数は、指定したクライアントに対するパブリケーションの完全リフレッシュを強制的に実行します。

6.4.5.4.1 CompleteRefresh

CompleteRefresh の構文は、次のとおりです。

```
public static void CompleteRefresh
    (String client_id,
     String publication) throws Throwable
```

CompleteRefresh のパラメータを表 6-39 に示します。

表 6-39 CompleteRefresh パラメータ

パラメータ	説明
client_id	Consolidator のクライアント名です。
publication	リフレッシュするパブリケーションの名前です。

6.4.6 仮想主キー

ベース・オブジェクトに主キーが定義されていないパブリケーション・アイテムに対して、仮想主キーを指定できます。仮想主キーは、2 つ以上の列に対して作成できますが、仮想主キーを割り当てるそれぞれの列に対して API を個別にコールする必要があります。仮想主キーの作成方法と削除方法を次に示します。

6.4.6.1 仮想主キー列の作成

これにより、仮想主キー列を作成できます。

構文

```
public static void CreateVirtualPKColumn
    (String owner,
     String store,
     String column) throws Throwable
```

CreateVirtualPKColumn のパラメータを表 6-40 に示します。

表 6-40 CreateVirtualPKColumn のパラメータ

パラメータ	説明
owner	実表またはビューの所有者を指定する文字列です。
store	実表またはビューを指定する文字列です。
column	主キー列を指定する文字列です。

例

```
Consolidator.CreateVirtualPKColumn("SAMPLE1", "DEPT", "DEPT_ID");
```

6.4.6.2 仮想主キー列の削除

これにより、仮想主キーを削除できます。

構文

```
public static void DropVirtualPKColumn
    (String owner,
     String store) throws Throwable
```

DropVirtualPKColumn のパラメータを表 6-41 に示します。

表 6-41 DropVirtualPKColumn のパラメータ

パラメータ	説明
owner	実表またはビューの所有者を指定する文字列です。
store	実表またはビューを指定する文字列です。

例

```
Consolidator.DropVirtualPKColumn("SAMPLE1", "DEPT");
```

6.4.7 パブリケーション・アイテムの問合せのキャッシング

この機能により、複雑なパブリケーション・アイテムの問合せをキャッシュできます。これは、Oracle 問合せエンジンで最適化できない問合せに適用されます。問合せを一時表にキャッシュすることにより、Consolidator テンプレートはスナップショットとより効率的に結合します。

データを一時表に格納すると MGP 操作に追加オーバーヘッドが発生します。Consolidator テンプレート内でパブリケーション・アイテムの問合せがうまく実行されるよう、問合せの最適化を試みた後に、一時表に格納するかどうかを判断してください。この方法で問合せを最適化できない場合に、キャッシングを使用します。

次の例は、MGP が構成フェーズで使用するテンプレートです。MGP は、このテンプレートを使用して、無効になったためにクライアントから削除する必要のあるクライアント・レコードを識別します。

```
UPDATE pub_item_map map
SET delete = true
WHERE client = <clientid>
AND NOT EXISTS (SELECT 'EXISTS' FROM
  (<publication item query>) snapshot
  WHERE map.pk = snapshot.pk);
```

この例では、<publication item query> が複雑になりすぎ、複数のネストした副問合せ、UNION、仮想列、CONNECT BY 句およびその他の複雑な関数を含んでいるために、問合せ最適化マイザが許容可能な計画を判断できません。その結果、MGP 構成フェーズで、パフォーマンスに重大な影響が発生する可能性があります。パブリケーション・アイテムの問合せのキャッシング機能を使用してパブリケーション・アイテムの問合せを一時表にキャッシュすることにより、問合せの構造が単純になり、テンプレートで問合せを効果的に結合できます。

6.4.7.1 パブリケーション・アイテムの問合せのキャッシングの有効化

次の API により、パブリケーション・アイテムの問合せのキャッシングが有効になります。

構文

```
public static void EnablePublicationItemQueryCache(String name)
    throws Throwable
```

EnablePublicationItemQueryCache のパラメータを表 6-42 に示します。

表 6-42 EnablePublicationItemQueryCache のパラメータ

パラメータ	説明
name	パブリケーション・アイテムの名前を指定する文字列です。

例

```
Consolidator.EnablePublicationItemQueryCache(  
    "P_SAMPLE1");
```

6.4.7.2 パブリケーション・アイテムの問合せのキャッシングの無効化

次の API により、パブリケーション・アイテムの問合せのキャッシングが無効になります。

構文

```
public static void DisablePublicationItemQueryCache(String name)  
    throws Throwable
```

DisablePublicationItemQueryCache のパラメータを表 6-43 に示します。

表 6-43 DisablePublicationItemQueryCache のパラメータ

パラメータ	説明
name	パブリケーション・アイテムの名前を指定する文字列です。

例

```
Consolidator.DisablePublicationItemQueryCache("P_SAMPLE1");
```

6.4.8 ユーザー定義の PL/SQL プロシージャのバインド

Mobile サーバーの同期プロセスは、多くの方法でカスタマイズできます。アプリケーション・ロジックは、PL/SQL プロシージャをパブリケーション・アイテムにバインドすることで、Mobile サーバーにアタッチできます。プロシージャは、Consolidator API の BeforeCompose、AfterCompose、BeforeApply および AfterApply の各メソッドを公開する必要があります。Mobile サーバーは、次の作業の前後にこれらのメソッドをコールします。

- クライアントの変更内容を Mobile Sync Client にかわってサーバーの表に適用する。
- 指定されたパブリケーション・アイテムに対する高速リフレッシュ変更を構成する。

Mobile サーバーは、現在の Mobile Sync ユーザー名の情報をこれらのメソッドに渡します。

ユーザー定義の PL/SQL プロシージャは、データのキャッシュや事前計算ができます。外部キー制約違反の問題も解決できます。詳細は、6.4.11 項「更新可能パブリケーション・アイテムの外部キー制約」を参照してください。これらのコールの使用の詳細は、6.4.12 項「構成と適用を使用する前後に行うコールバックのカスタマイズ」を参照してください。

6.4.9 レプリケーションをカスタマイズするためのキュー・インタフェース

アプリケーション開発者は、[CreateQueuePublicationItem API](#) を使用することで、レプリケーション・プロセスをプログラムにより管理できます。通常、MGP によりインキューとアウトキューのどちらも管理されますが、この API を使用すると、アプリケーション開発者は [6.4.9.3 項「キュー・インタフェースの PL/SQL プロシージャ」](#) に示されている PL/SQL パッケージを使用し、キューそのものを作成することにより、同期セッション中にキュー操作を管理できます。

6.4.9.1 キュー・インタフェース操作

クライアントからデータが到着すると、パブリケーション・アイテムのインキューに入れられます。データがコミットされると、Consolidator は、`UPLOAD_COMPLETE` を 1 回コールします。現行同期セッション内のすべてのレコードには、同一トランザクション識別子が付けられます。Consolidator には、どのパブリケーション・アイテムのインキューがこのトランザクション識別子を使用して新しいトランザクションを受け取ったかを示す [キュー制御表](#) (`C$INQ+name`) があります。この表を参照すると、処理を必要とするキューを判断できます。

Consolidator は、同期セッションのダウンロード・フェーズを開始する前に、`DOWNLOAD_INIT` をコールします。このプロシージャでは、クライアントに送信するデータを判断するために設定または変更する必要がある設定を、カスタマイズすることができます。Consolidator は、クライアントのサブスクリプションに基づいてダウンロードできるパブリケーション・アイテムのリストを検索します。パブリケーション・アイテムのリストとリフレッシュ・モード（完全リフレッシュの場合は「Y」、高速リフレッシュの場合は「N」）が一時表 (`C$PUB_LIST_Q`) に挿入されます。Consolidator は、`C$PUB_LIST_Q` を参照してクライアントにダウンロードするアイテムを判断するため、この一時表内でアイテムの削除またはリフレッシュ・ステータスの変更を行います。

インキュー同様、アウトキュー内の各レコードにトランザクション識別子 (`TRANID$$`) を関連付ける必要があります。Consolidator は、クライアントにより正常に適用された最後のトランザクションを示す `last_tran` パラメータを渡します。クライアントにまだダウンロードされていないアウトキュー内の新しいレコードには、`curr_tran` パラメータの値を使用してマークを付ける必要があります。`curr_tran` の値は、`last_tran` の値よりも常に大きくなります。ただし、必ずしも順次である必要はありません。Consolidator は、`TRANID$$` の値が `last_tran` よりも大きい場合にのみ、アウトキューからレコードをダウンロードします。データがダウンロードされると、Consolidator は、`DOWNLOAD_COMPLETE` をコールします。

6.4.9.2 キューの作成

SQL を使用して、Mobile サーバー・リポジトリ内にアウトキューを手動で作成する必要があります。また、インキューが存在しない場合は **Consolidator** で作成されますが、インキューも作成する必要がある場合もあります。リポジトリに接続し、次の文を実行して、次の構造を持つインキューとアウトキューを作成します。

アウトキュー

```
'CIM$'+name
(
  CLID$$CS  VARCHAR2 (30),
  ..
  publication_item_store_columns (c1..cN),
  ..
  TRANID$$  NUMBER (10),
  DMLTYPE$$ CHAR (1) CHECK (DMLTYPE$$ IN ('I','U','D'),
)
```

インキュー

```
'CFM$'+name
(
  CLID$$CS  VARCHAR2 (30),
  TRANID$$  NUMBER (10),
  SEQNO$$   NUMBER (10),

  DMLTYPE$$ CHAR (1) CHECK (DMLTYPE$$ IN ('I','U','D'),
  ..
  publication_item_store_columns (c1..cN),
  ..
)
```

Consolidator では、キュー制御表 C\$INQ および一時表 C\$PUB_LIST_Q が作成されます。キュー制御表を調べると、どのパブリケーション・アイテムが新しいトランザクションを受け取ったかを判断できます。

キュー制御表

```
'C$INQ'+name
(
  CLIENTID  VARCHAR2 (30),
  TRANID$$  NUMBER,
  STORE     VARCHAR2 (30),

)
```

一時表

```
'C$PUB_LIST_Q'
(
NAME    VARCHAR2 (30),
COMP_REF CHAR (1),
CHECK (COMP_REF IN ('Y', 'N'))

)
```

手動により作成されるキューのパラメータを表 6-44 に示します。

表 6-44 キュー・インタフェース作成のパラメータ

パラメータ	説明
CLID\$\$CS	クライアントを識別する一意の文字列です。
TRANID\$\$	トランザクションを識別する一意の番号です。
SEQNO\$\$	インキュー（CFM\$）のみにある、トランザクションごとの DML 言語操作を表す一意の数値です。
DMLTYPE\$\$	DML 命令のタイプをチェックします。 ■ I - 挿入 ■ D - 削除 ■ U - 更新 アウトキューの場合のみです。
STORE	キュー制御表（C\$INQ）のみにあるパブリケーション・アイテム名を表します。
NAME	一時表（C\$PUB_LIST_Q）のみにあるパブリケーション・アイテム名です。
COMP_REF	パブリケーション・アイテムのリフレッシュ・モードを判断する場合に使用されるフラグで、値は「Y」（はい）か「N」（いいえ）のいずれかです。

6.4.9.3 キュー・インタフェースの PL/SQL プロシージャ

次の PL/SQL パッケージ仕様により、キュー・インタフェースに必要なコールアウトが定義されます。

サンプル・コード

```
CREATE OR REPLACE PACKAGE CONS_QPKG AS
/*
```



```

    *      notifies that inq has new transaction
  */
PROCEDURE UPLOAD_COMPLETE(
    CLIENTID      IN      VARCHAR2,
    TRAN_ID       IN      NUMBER    -- IN queue tranid
);

/*
 *      init data for download
 */
PROCEDURE DOWNLOAD_INIT(
    CLIENTID      IN      VARCHAR2,
    LAST_TRAN     IN      NUMBER,
    CURR_TRAN     IN      NUMBER,
    HIGH_PRTY     IN      VARCHAR2
);

/*
 *      notifies when all the client's data is sent
 */
PROCEDURE DOWNLOAD_COMPLETE(
    CLIENTID      IN      VARCHAR2
);

END CONS_QPKG;
/

```

6.4.9.4 CreateQueuePublicationItem API

この API コールは、パブリケーション・アイテムをキューの形式で作成します。この API コールは、パブリケーション・アイテムを登録し、CFM\$name 表が存在しない場合は、この表をインキューとして作成します。

構文

```

public static void CreateQueuePublicationItem
( String name,
  String owner,
  String store,
  String select_stmt,
  String pk_columns,
  String cbk_owner,
  String cbk_name) throws Throwable

```

CreateQueuePublicationItem のパラメータを[表 6-45](#)に示します。

表 6-45 CreateQueuePublicationItem のパラメータ

パラメータ	説明
name	新しいパブリケーション・アイテムおよびキュー名を定義します。
owner	実表またはビューの所有者です。
store	実表またはビューの名前を指定する値です。
select_stmt	新しいパブリケーション・アイテムについての SELECT 文を指定する文字列です。この文には、サブスクリプション・パラメータを含めることができます。
pk_columns	仮想主キーを作成する、カンマで区切られたリストです。
cbk_owner	コールバック・パッケージの所有者を指定します。詳細は、 6.4.12 項「構成と適用を使用する前後に行うコールバックのカスタマイズ」 を参照してください。これは拡張機能です。
cbk_name	コールバック・パッケージ名を指定します。詳細は、 6.4.12 項「構成と適用を使用する前後に行うコールバックのカスタマイズ」 を参照してください。これは拡張機能です。

更新可能または高速リフレッシュ可能なキューを作成するには、owner.store の主キーを Consolidator に指定する必要があります。store に主キーがない場合は、pk_columns パラメータに主キーを指定してもかまいません。pk_columns が NULL の場合、Consolidator は store の主キーを使用します。

6.4.9.5 リポジトリの外部での PL/SQL パッケージの定義

必要な場合には、PL/SQL パッケージを Mobile サーバー・リポジトリ外でも定義できますが、正常に機能させるには、リポジトリ内で定義されたインキュー、アウトキュー、キュー制御表および一時表を参照させる必要があります。次の API コールは、プロシージャ名の取得、プロシージャの登録または削除に使用します。

6.4.9.5.1 RegisterQueuePkg

この API は、文字列「pkg」を現行プロシージャとして登録します。

構文

```
public String RegisterQueuePkg(String pkg) throws SQLException
```

例

```
Consolidator.RegisterQueuePkg("ASL.QUEUES_PKG");
```

6.4.9.5.2 GetQueuePkg

この API コールは、現在登録されているプロシージャの名前を返します。

構文

```
public String GetQueuePkg() throws SQLException
```

6.4.9.5.3 UnRegisterQueuePkg

この API コールは、現在登録されているプロシージャを削除します。

構文

```
public String UnRegisterQueuePkg() throws SQLException
```

6.4.10 Null Sync コールアウト

Mobile サーバーは、同期中に、クライアントが Null Sync を試行中かどうかを示すコールアウトを作成します。Null Sync とは、クライアントにアップロード対象の変更がないことを言います。このコールアウトは、Mobile サーバーのリポジトリ内に PL/SQL プロシージャを作成することで実装できます。プロシージャでは、次の指定を行う必要があります。

```
create or replace package CUSTOMIZE as procedure  
NullSync(p_Client IN varchar2, p_NullSync as boolean);  
end CUSTOMIZE;
```

6.4.11 更新可能パブリケーション・アイテムの外部キー制約

Oracle データベースとクライアントの間で更新可能モードにより表をレプリケートする際、表に参照整合性制約があると、外部キー制約違反が起きる場合があります。外部キー制約違反が発生した場合、サーバーはクライアント・トランザクションを拒否します。

6.4.11.1 外部キー制約違反の例

たとえば、2 つの表 EMP と DEPT に参照整合性制約があるとしします。DEPT 表の DeptNum (部門番号) 属性は、EMP 表の外部キーです。EMP 表の各従業員の DeptNum 値は、DEPT 表の有効な DeptNum 値である必要があります。

Mobile サーバー・ユーザーが DEPT 表に新しい部門を追加し、次に EMP 表でこの部門に新しい従業員を追加します。トランザクションはまず DEPT を更新し、次に EMP 表を更新します。しかし、データベース・アプリケーションでは、これらの操作を実行した順序を保存しません。

ユーザーが Mobile サーバーをレプリケートする際、Mobile サーバーは最初に EMP 表を更新します。この作業で、DeptNum に対して無効な外部キー値を持つ新規レコードを EMP に作成しようとします。Oracle データベースにより参照整合性違反が検出されます。Mobile サーバーはトランザクションをロールバックし、トランザクション・データを Mobile サーバーのエラー・キューに置きます。この場合、トランザクション内の操作が元と違う順序で実行されたために、外部キー制約違反が発生しました。

6.4.11.2 BeforeApply および AfterApply での制約違反の回避

PL/SQL プロシージャを使用すると、DEFERRABLE 制約を BeforeApply 関数および AfterApply 関数とともに使用することで、順序どおりではない操作による外部キー制約違反を回避できます。DEFERRABLE 制約は、INITIALLY IMMEDIATE または INITIALLY DEFERRED のいずれかにできます。DEFERRABLE INITIALLY IMMEDIATE 外部キー制約の動作は、通常の即時制約と同じです。どちらの制約をアプリケーションに適用しても、機能に違いはありません。

Mobile サーバーは、サーバーにクライアント・トランザクションを適用する前に BeforeApply 関数をコールし、トランザクションを適用した後で AfterApply 関数をコールします。BeforeApply 関数を使用して、制約を DEFERRED に設定し、参照整合性検査を遅延できます。トランザクションが適用された後、AfterApply 関数をコールして制約を IMMEDIATE に設定します。この時点で、クライアント・トランザクションが参照整合性に違反している場合は、ロールバックされエラー・キューに移動されます。

DEFERRABLE 制約を使用して外部キー制約違反を回避するには、次の手順を実行します。

1. 外部キー制約をすべて削除して DEFERRABLE 制約として作成し直します。
2. ユーザー定義の PL/SQL プロシージャを、参照整合性制約を持つ表が含まれたパブリケーションにバインドします。
3. PL/SQL プロシージャでは、次の例に示されているとおり、BeforeApply 関数で制約を DEFERRED に設定し、AfterApply 関数で IMMEDIATE に設定する必要があります。この例では、SAMPLE3 という表と address.14_fk という制約が使用されています。

```
procedure BeforeApply(clientname varchar2) is
cur integer;
begin
  cur := dbms_sql.open_cursor;
  dbms_sql.parse(cur, 'SET CONSTRAINT SAMPLE3.address14_fk
                     DEFERRED', dbms_sql.native);
  dbms_sql.close_cursor(cur);
end;
procedure AfterApply(clientname varchar2) is
cur integer;
begin
  cur := dbms_sql.open_cursor;
```

```

dbms_sql.parse(cur, 'SET CONSTRAINT SAMPLE3.address14_fk
                    IMMEDIATE', dbms_sql.native);
dbms_sql.close_cursor(cur);
end;
```

6.4.11.3 表の比率を使用した制約違反の回避

Mobile サーバーでは、表の比率を使用して、クライアント操作をマスター表に適用する順序を決定します。表の比率は、整数で表され、次のように実装されます。

1. 最初に、クライアントの INSERT 操作が、表の比率の低いものから高いものへと順に実行されます。
2. 次に、クライアントの DELETE 操作が、表の比率の高いものから低いものへと順に実行されます。
3. 最後に、クライアントの UPDATE 操作が、表の比率の低いものから高いものへと順に実行されます。

6.4.11.1 項「外部キー制約違反の例」にリストされた例では、DEPT 表に EMP 表よりも低い比率を割り当てることで、制約違反エラーを解決できます。たとえば、次のようになります。

```
(DEPT weight=1, EMP weight=2)
```

6.4.12 構成と適用を使用する前後に行うコールバックのカスタマイズ

パブリケーション・アイテムを作成する際、ユーザーは、MGP バックグラウンド・プロセスの適用と構成フェーズでコールされるカスタマイズ可能パッケージを指定できます。クライアント・データは MGP に処理される前にインキューに蓄積されます。データは MGP によって処理された後、アウトキューに蓄積されてから、Mobile Sync によりクライアントに取り込まれます。

これらのプロシージャにより、カスタマイズされたコードをプロセスに取り込むことができます。ユーザー・レベルおよびトランザクション・レベルでのカスタマイズを可能にするために、clientname と tranid が渡されます。

```
procedure BeforeApply(clientname varchar2)
```

このプロシージャは、クライアントのデータがすべて適用された後でコールする必要があります。

```
procedure AfterApply(clientname varchar2)
```

このプロシージャは、tranid を持つクライアントのデータが適用される前にコールする必要があります。

```
procedure BeforeTranApply(tranid number)
```

このプロシージャは、tranid を持つクライアントのデータが適用された後にコールする必要があります。

```
procedure AfterTranApply(tranid number)
```

このプロシージャは、アウトキューが構成される前にコールする必要があります。

```
procedure BeforeCompose(clientname varchar2)
```

このプロシージャは、アウトキューが構成された後にコールする必要があります。

```
procedure AfterCompose(clientname varchar2)
```

6.4.13 DML 操作のコールバックのカスタマイズ

パブリケーション・アイテムを作成すると、そのパブリケーション・アイテムのすべての DML 操作のかわりに、Mobile サーバー・リポジトリに格納されているカスタマイズ PL/SQL プロシージャをコールするように、Java を使用して指定できます。各パブリケーション・アイテムには、Mobile DML プロシージャは 1 つしかありません。プロシージャは、次のような構造で作成されます。

```
AnySchema.AnyPackage.AnyName(DML in CHAR(1), COL1 in TYPE, COL2 in TYPE, COLn.., PK1 in TYPE, PK2 in TYPE, PKn..)
```

DML 操作をカスタマイズするためのパラメータを表 6-46 に示します。

表 6-46 Mobile DML 操作のパラメータ

パラメータ	説明
DML	各行に対する DML 操作です。値は、削除を表す「D」、挿入を表す「I」または更新を表す「U」のいずれかです。
COL1 ... COLn	パブリケーション・アイテムに定義されている列のリストです。列名は、パブリケーション・アイテムの間合せに指定される順序と同じ順序で指定する必要があります。パブリケーション・アイテムが「SELECT * FROM example」を使用して作成されている場合、列の順序は表「example」に指定されている順序と同じである必要があります。
PK1 ... PKn	主キー列のリストです。列名は、実表または親表に指定されている順序と同じ順序で指定する必要があります。

たとえば、次の間合せにより定義されているパブリケーション・アイテム「example」に対して、DML プロシージャが必要になるとします。

```
select A,B,C from publication_item_example_table
```

「A」が「example」の主キー列とすると、DML プロシージャのシグネチャは次のようになります。

```
any_schema.any_package.any_name (DML in CHAR(1), A in TYPE, B in TYPE, C in TYPE,A_
OLD in TYPE)
```

実行時に、このプロシージャは、DML タイプの「I」、「U」または「D」でコールされます。挿入および削除操作の場合、A_OLD は NULL になります。更新の場合は、更新される行の主キーに設定されます。PL/SQL プロシージャを定義した後は、次の API コールを使用してプロシージャをパブリケーション・アイテムに連結できます。

```
Consolidator.AddMobileDmlProcedure("PUB_example", "example", "any_schema.any_
package.any_name")
```

「example」はパブリケーション・アイテム名で、「PUB_example」はパブリケーション名です。

この API コールの詳細は、『Consolidator Admin API Specification』を参照してください。

6.4.13.1 DML プロシージャの例

次の PL/SQL コード（一部）は、サンプル・パブリケーションのパブリケーション・アイテムに対する実際の DML プロシージャを定義します。次に説明された ORD_MASTER 表を使用して、問合せは次のように定義されています。

SQL 文

```
SELECT * FROM ord_master", where ord_master has a single column primary key on "ID"
```

ord_master 表

```
SQL> desc ord_master
```

Name	Null?	Type
-----	-----	-----
ID	NOT NULL	NUMBER(9)
DDATE		DATE
STATUS		NUMBER(9)
NAME		VARCHAR2(20)
DESCRIPTION		VARCHAR2(20)

コード例

```
CREATE OR REPLACE PACKAGE "SAMPLE11"."ORD_UPDATE_PKG" AS
  procedure UPDATE_ORD_MASTER(DML CHAR, ID NUMBER, DDATE DATE, STATUS
NUMBER, NAME VARCHAR2, DESCRIPTION VARCHAR2, ID_OLD NUMBER);
END ORD_UPDATE_PKG;
/
CREATE OR REPLACE PACKAGE BODY "SAMPLE11"."ORD_UPDATE_PKG" as
  procedure UPDATE_ORD_MASTER(DML CHAR, ID NUMBER, DDATE DATE, STATUS
NUMBER, NAME VARCHAR2, DESCRIPTION VARCHAR2, ID_OLD NUMBER) is
  begin
    if DML = 'U' then
      execute immediate 'update ord_master set id = :id, ddate = :ddate,
status = :status, name = :name, description = '||''''||'from
ord_update_pkg' || ''''||' where id = :id_old'
        using id, ddate, status, name, id_old;
    end if;
    if DML = 'I' then
      begin
        execute immediate 'insert into ord_master values(:id, :ddate,
:status, :name, '||''''||'from ord_update_pkg' || ''''||')'
          using id, ddate, status, name;
      exception
        when others then
          null;
      end;
    end if;
    if DML = 'D' then
      execute immediate 'delete from ord_master where id = :id'
        using id;
    end if;
  end UPDATE_ORD_MASTER;
end ORD_UPDATE_PKG;
/
```

この DML プロシージャを追加する API コールは、次のとおりです。

```
Consolidator.AddMobileDMLProcedure("T_SAMPLE11", "P_SAMPLE11-M", "SAMPLE11.ORD_UPDATE_
PKG.UPDATE_ORD_MASTER")
```

"T_SAMPLE11" はパブリケーション名で、"P_SAMPLE11-M" はパブリケーション・アイテム名です。

6.4.14 制限選択条件

制限選択条件は、パブリケーションにパブリケーション・アイテムを追加するときに、パブリケーション・アイテムに割り当てることができます。クライアントが高優先順位モードで同期される場合、選択条件はクライアントにダウンロードされるデータを制限するために使用されます。このパラメータは、NULL にできます。このパラメータは、上級者向けです。

6.5 同期エラーと競合

Mobile サーバーでは、サーバーが行を削除するのと同時にクライアントが更新する場合、Oracle データベースの同期との互換性エラーが発生します。NULL の使用違反や外部キー制約違反など、その他のエラーはすべて同期エラーです。

Mobile サーバーでは、同期エラーは自動的に解決されません。かわりに、Mobile サーバーは対応するトランザクションをロールバックし、トランザクション操作を Mobile サーバーのエラー・キューに移動します。Mobile サーバーのデータベース管理者は、後でこれらのトランザクション操作を変更して再実行したり、エラー・キューからページできます。

Mobile サーバーの同期の競合は、次の場合に発生します。

- クライアントとサーバーが同じ行を更新する場合。
- クライアントとサーバーが同じ主キー値を持つ行を作成する場合。
- サーバーが更新する行とクライアントが削除する行が同じ場合。

競合解決手法の詳細は、[6.5.3 項「エラー・キューを使用した競合の解決」](#)を参照してください。

6.5.1 バージョニング

Mobile サーバーでは、内部バージョニングを使用して同期の競合を検出します。バージョン番号は、各サーバー・レコードにかぎらず、各クライアント・レコードに対しても管理されます。クライアントの変更内容がサーバーに適用される際、Mobile サーバーはバージョンの不一致を検出し、ウィニング・ルールに従って競合を解決します。

6.5.2 ウィニング・ルール

Mobile サーバーは、ウィニング・ルールを使用して同期の競合を自動的に解決します。次のウィニング・ルールが組み込まれています。

- クライアント優先
- サーバー優先

クライアント優先の場合、Mobile サーバーはクライアントの変更内容を自動的にサーバーに適用します。サーバー優先の場合、Mobile サーバーはクライアントに対する変更内容を自動的に構成します。

Mobile サーバーの競合解決方式は、ウィニング・ルールを「クライアント優先」に設定し、データベース表に BEFORE INSERT、BEFORE UPDATE および BEFORE DELETE の各トリガーをアタッチすることで、カスタマイズできます。トリガーにより、新旧の行の値を比較して指定されたとおりにクライアントの変更内容を解決します。

6.5.3 エラー・キューを使用した競合の解決

作成したパブリケーション・アイテムごとに、対応するエラー・キューが別個に作成されます。このキューの目的は、未解決の競合が原因で失敗するトランザクションを格納することです。管理者は、エラー・キュー・データまたはサーバーのエラー・キューを変更することで競合の解決を試みることができ、続いて ExecuteTransaction API コールを介してトランザクションの再適用を試みることができます。管理者は、PurgeTransaction API コールを介してエラー・キューのパージを試みることもできます。Mobile サーバーのエラー・キューは C\$EQ であり、データは CEQ\$ に格納されます。

6.5.3.1 トランザクションの実行

トランザクション実行関数は、Mobile サーバーのエラー・キュー内のトランザクションを再実行します。

構文

```
public static void ExecuteTransaction
(String clientid,
 long tid) throws Throwable
```

ExecuteTransaction のパラメータを [表 6-47](#) に示します。

表 6-47 ExecuteTransaction のパラメータ

パラメータ	説明
clientid	Mobile Sync Client 名です。
tid	トランザクション ID です。これらはエラー・キューに示される生成済文字列です。

例

```
Consolidator.ExecuteTransaction("DAVIDL", 100002);
```

6.5.3.2 トランザクションのパージ

トランザクションのパージ関数は、Mobile サーバーのエラー・キューからトランザクションをパージします。

構文

```
public static void PurgeTransaction
    (String clientid,
     long tid) throws Throwable
```

PurgeTransaction のパラメータを表 6-48 に示します。

表 6-48 PurgeTransaction のパラメータ

パラメータ	説明
clientid	Mobile サーバー・ユーザー名です。
tid	トランザクション ID です。これらはエラー・キューに示される生成済文字列です。

例

```
Consolidator.PurgeTransaction("DAVIDL", 100001);
```

6.6 Oracle サーバーとクライアント間でのデータ型のマッピング

Mobile サーバーによって同期される Oracle データベースと Oracle9i Lite の表では、互換性のあるデータ型を使用している必要があります。Oracle データベースのデータ型は、Oracle9i Lite のデータ型と互換性があります。

6.6.1 Oracle Lite データ型

すべての Oracle9i Lite ベースのスナップショットは、同期時に Mobile Sync によって作成されます。Mobile サーバーは、Oracle データベースでのデータ精度に応じて自動的に Oracle9i Lite データ型を選択します。表 6-49 に、データ変換値を示します。Oracle データ型は左側の列に、Oracle9i Lite データ型は最上部行に表示されています。「Y」は無条件にサポートされ、「N」はサポートされないことを示します。1B、2B および 4B のデータ型は、OKAPI 専用のデータ型です。詳細は、『Oracle9i Lite (C および C++) オブジェクト・カーネル API リファレンス』を参照してください。

表 6-49 Oracle Lite データ型

Oracle データ型	1B	2B	4B	FLOAT	DOUBLE	DATETIME	LONG- VARIABLE	VARCHAR
INTEGER	Y	Y	Y	Y	Y	N	N	N
VARCHAR2	N	N	N	N	N	N	N	Y
VARCHAR	N	N	N	N	N	N	N	Y
CHAR	N	N	N	N	N	N	N	Y
SMALLINT	Y	Y	Y	Y	Y	N	N	N
FLOAT	Y	Y	Y	Y	Y	N	N	N
DOUBLE PRECISION	Y	Y	Y	Y	Y	N	N	N
NUMBER	Y	Y	Y	Y	Y	N	N	N
DATE	N	N	N	N	N	Y	N	N
LONG RAW	N	N	N	N	N	N	Y	N
LONG	N	N	N	N	N	N	N	Y
BLOB	N	N	N	N	N	N	Y	N
CLOB	N	N	N	N	N	N	N	N

システム・カタログ・ビュー

この付録は、Oracle Lite データベースのシステム・カタログ・ビューの参考資料です。この付録では、次に示すカタログ・ビューのセットについて説明します。

A.1 Oracle Lite データベースのカatalog・ビュー

Oracle Lite データベースのシステム・カatalogでは、次のビューが使用できます。

- [A.1.1 項「ALL_COL_COMMENTS」](#)
- [A.1.2 項「ALL_CONSTRAINTS」](#)
- [A.1.3 項「ALL_CONS_COLUMNS」](#)
- [A.1.4 項「ALL_INDEXES」](#)
- [A.1.5 項「ALL_IND_COLUMNS」](#)
- [A.1.6 項「ALL_OBJECTS」](#)
- [A.1.7 項「ALL_PARTITIONS」](#)
- [A.1.8 項「ALL_SEQUENCES」](#)
- [A.1.9 項「ALL_SYNONYMS」](#)
- [A.1.10 項「ALL_TABLES」](#)
- [A.1.11 項「ALL_TAB_COLUMNS」](#)
- [A.1.12 項「ALL_TAB_COMMENTS」](#)
- [A.1.13 項「ALL_USERS」](#)
- [A.1.14 項「ALL_VIEWS」](#)
- [A.1.15 項「CAT」](#)
- [A.1.16 項「COLUMN_PRIVILEGES」](#)
- [A.1.17 項「DATABASE_PARAMETERS」](#)
- [A.1.18 項「DUAL」](#)
- [A.1.19 項「TABLE_PRIVILEGES」](#)
- [A.1.20 項「USER_OBJECTS」](#)

注意： 以降に示す表で、アスタリスクの付いた列は Oracle Lite では使用されませんが、Oracle データベースと互換性があり、一般に NULL またはデフォルト値を返します。

A.1.1 ALL_COL_COMMENTS

このビューは、表の列に対するユーザーのコメントをリストします。このビューのパラメータを表 A-1 に示します。

表 A-1 ALL_COL_COMMENTS のパラメータ

列	データ型	NULL の使用	説明
OWNER	VARCHAR2(128)	×	表の所有者。
TABLE_NAME	VARCHAR2(128)	×	オブジェクトの名前。
COLUMN_NAME	VARCHAR2(128)	×	列の名前。
COMMENTS	VARCHAR2(4096)	○	列のコメントのテキスト。

A.1.2 ALL_CONSTRAINTS

このビューは、アクセス可能な表に対する制約の定義について次の情報を提供します。このビューのパラメータを表 A-2 に示します。

表 A-2 ALL_CONSTRAINTS のパラメータ

列	データ型	NULL の使用	説明
OWNER	VARCHAR2(128)	×	制約定義の所有者。
CONSTRAINT_NAME	VARCHAR2(128)	×	制約定義に対応付けられた名前。
CONSTRAINT_TYPE	VARCHAR2(128)	×	制約定義のタイプ： C（表に対する CHECK 制約） P（主キー） U（一意キー） R（参照整合性） V（ビューに対するチェック・オプション）
TABLE_NAME	VARCHAR2(128)	×	制約定義を持つ表の名前。

表 A-2 ALL_CONSTRAINTS のパラメータ (続き)

列	データ型	NULL の使用	説明
SEARCH_CONDITION	VARCHAR2(1000)	○	表検査のための検索条件のテキスト。
R_OWNER	VARCHAR2(128)	○	参照制約で使用する表の所有者。
R_CONSTRAINT_NAME	VARCHAR2(128)	○	参照される表に対する一意制約定義の名前。
DELETE_RULE	VARCHAR2(128)	○	参照制約の削除規則： 「NO ACTION」
STATUS	VARCHAR2(20)	×	制約のステータス： 「ENABLED」または 「DISABLED」

A.1.3 ALL_CONS_COLUMNS

このビューは、制約定義内のアクセス可能な列について次の情報を提供します。このビューのパラメータを[表 A-3](#) に示します。

表 A-3 ALL_CONS_COLUMNS のパラメータ

列	データ型	NULL の使用	説明
OWNER	VARCHAR2(128)	○	制約定義の所有者のユーザー名。
CONSTRAINT_NAME	VARCHAR2(128)	○	制約定義に対応付けられた名前。
TABLE_NAME	VARCHAR2(128)	○	制約定義を持つ表の名前。
COLUMN_NAME	VARCHAR2(128)	○	制約定義に指定されている列に対応付けられた名前。
POSITION	NUMBER(10)	○	定義内での列の元の位置。

A.1.4 ALL_INDEXES

このビューには、表に定義されているすべての索引の説明が含まれています。このビューのパラメータを表 A-4 に示します。

表 A-4 ALL_INDEXES のパラメータ

列	データ型	NULL の使用	説明
OWNER	VARCHAR2(128)	×	INDEX 定義の所有者。
INDEX_NAME	VARCHAR2(128)	×	INDEX 定義に対応付けられた名前。
TABLE_OWNER	VARCHAR2(128)	×	INDEX が定義されている表の所有者。
TABLE_NAME	VARCHAR2(128)	×	INDEX 定義を持つ表の名前。
TABLE_TYPE	VARCHAR2(10)	○	オブジェクトの型。
UNIQUENESS	VARCHAR2(128)	×	「UNIQUE」または「NONUNIQUE」を含む文字列。

A.1.5 ALL_IND_COLUMNS

このビューは、データベース内のすべての索引に対する索引キー列をリストします。このビューのパラメータを表 A-5 に示します。

表 A-5 ALL_IND_COLUMNS のパラメータ

列	データ型	NULL の使用	説明
INDEX_OWNER	VARCHAR2(128)	×	INDEX 定義の所有者。
INDEX_NAME	VARCHAR2(128)	×	INDEX 定義に対応付けられた名前。
TABLE_OWNER	VARCHAR2(128)	×	INDEX が定義されている表の所有者。
TABLE_NAME	VARCHAR2(128)	×	INDEX 定義を持つ表の名前。
COLUMN_NAME	VARCHAR2(128)	×	INDEX 定義に指定されている列に対応付けられた名前。
COLUMN_POSITION	NUMBER(10)	×	索引定義内での列の位置。

A.1.6 ALL_OBJECTS

このビューには、オブジェクト（表、ビュー、シノニム、索引および順序）の説明が含まれています。このビューのパラメータを[表 A-6](#)に示します。

表 A-6 ALL_OBJECTS のパラメータ

列	データ型	NULL の使用	説明
OWNER	VARCHAR2(128)	×	OBJECTS 定義の所有者。
OBJECT_NAME	VARCHAR2(128)	×	OBJECTS 定義に対応付けられた名前。
OBJECT_TYPE	VARCHAR2(128)	○	オブジェクトの型： TABLE、VIEW、INDEX、 SEQUENCE または SYNONYM
CREATED	DATE	○	OBJECTS の作成タイムスタンプ。
STATUS	VARCHAR2(128)	○	OBJECTS の状態です。 VALID、INVALID または N/A（常に有効）

A.1.7 ALL_PARTITIONS

このビューは、すべてのマップ表のパーティション、パーティションの削除またはマージに使用するパーティション ID およびパーティション間のクライアントの配分を表示します。このビューのパラメータを[表 A-7](#)に示します。

表 A-7 ALL_PARTITIONS のパラメータ

列	データ型	NULL の使用	説明
PUB_ITEM	VARCHAR2(30)	×	パブリケーション・アイテム名。
PARTITION_ID	NUMBER	×	パーティション ID。
CLID\$\$CS	VARCHAR2(30)	×	クライアント ID。

注意： このパーティション化のフォームは、Oracle サーバーによって提供されているパーティション機能とは関係なく、Oracle9i Lite 専用です。

A.1.8 ALL_SEQUENCES

このビューは、データベース内のすべての順序の説明をリストします。このビューのパラメータを表 A-8 に示します。

表 A-8 ALL_SEQUENCES のパラメータ

列	データ型	NULL の使用	説明
SEQUENCE_OWNER	VARCHAR2(128)	×	SEQUENCES 定義の所有者。
SEQUENCE_NAME	VARCHAR2(128)	×	SEQUENCES 定義に対応付けられた名前。
MIN_VALUE	NUMBER(10)	×	順序の最小値。
MAX_VALUE	NUMBER(10)	×	順序の最大値。
INCREMENT_BY	NUMBER(10)	×	順序の増分値。

A.1.9 ALL_SYNONYMS

このビューは、データベース内のすべてのシノニムをリストします。このビューのパラメータを表 A-9 に示します。

表 A-9 ALL_SYNONYMS のパラメータ

列	データ型	NULL の使用	説明
OWNER	VARCHAR2(128)	○	SYNONYMS 定義の所有者。
SYNONYM_NAME	VARCHAR2(128)	○	SYNONYMS 定義に対応付けられた名前。
TABLE_OWNER	VARCHAR2(128)	○	SYNONYMS が定義されている表の所有者。
TABLE_NAME	VARCHAR2(128)	○	SYNONYMS 定義を持つ表の名前。
DB_LINK	VARCHAR2(128)	○	予約済。

A.1.10 ALL_TABLES

このビューは、ユーザーがアクセスできる表について次の情報を提供します。このビューのパラメータを表 A-10 に示します。

表 A-10 ALL_TABLES のパラメータ

列	データ型	NULL の使用	説明
OWNER	VARCHAR2(128)	×	表の所有者のユーザー名。
TABLE_NAME	VARCHAR2(128)	×	表の名前。
TABLESPACE_NAME	VARCHAR2(128)	○	この表を含むカatalogまたはデータベース・ファイルの名前。
CLUSTER_NAME*	VARCHAR2(128)	○	この表が属するクラスタの名前（クラスタがある場合）。
PCT_FREE*	NUMBER(10)	○	ブロック内の空き領域の最小値（パーセント）。
PCT_USED*	NUMBER(10)	○	ブロック内の使用済領域の最小値（パーセント）。
INI_TRANS*	NUMBER(10)	○	トランザクション数の初期値。
MAX_TRANS*	NUMBER(10)	○	トランザクション数の最大値。
INITIAL_EXTENT*	NUMBER(10)	○	初期エクステントのサイズ（バイト単位）。
NEXT_EXTENT*	NUMBER(10)	○	2 次エクステントのサイズ（バイト単位）。
MIN_EXTENTS*	NUMBER(10)	○	セグメント内で使用できる最小エクステント数。
MAX_EXTENTS*	NUMBER(10)	○	セグメント内で使用できる最大エクステント数。
PCT_INCREASE*	NUMBER(10)	○	エクステント・サイズの増分パーセント。
BACKED_UP*	VARCHAR2(1)	○	最後の変更以降に表がバックアップされているかどうか。
NUM_ROWS*	NUMBER(10)	○	表の中の行数。

表 A-10 ALL_TABLES のパラメータ (続き)

列	データ型	NULL の使用	説明
BLOCKS*	NUMBER(10)	○	表に割り当てられているデータ・ブロック数。
EMPTY_BLOCKS*	NUMBER(10)	○	表に割り当てられているデータ・ブロックの中でデータを含まないブロック数。
AVG_SPACE*	NUMBER(10)	○	表に割り当てられているデータ・ブロック内の平均空き領域 (バイト単位)。
CHAIN_CNT*	NUMBER(10)	○	表の中で、あるデータ・ブロックから別のデータ・ブロックに連鎖されている行の数、または新規ブロックに移行された行で、古い ROWID を保持するためにリンクが必要な行の数。
AVG_ROW_LEN*	NUMBER(10)	○	表の行の平均長 (バイト単位)。

A.1.11 ALL_TAB_COLUMNS

このビューは、ユーザーがアクセスできる表、ビューおよびクラスタの列について次の情報を提供します。このビューのパラメータを表 A-11 に示します。

表 A-11 ALL_TAB_COLUMNS のパラメータ

列	データ型	NULL の使用	説明
OWNER	VARCHAR2(128)	×	表、ビューまたはクラスタの所有者のユーザー名。
TABLE_NAME	VARCHAR2(128)	×	表、ビューまたはクラスタの名前。
COLUMN_NAME	VARCHAR2(128)	×	列の名前。
DATA_TYPE	VARCHAR2(30)	○	列のデータ型。
DATA_LENGTH	NUMBER(10)	○	列の長さ (バイト単位)。

表 A-11 ALL_TAB_COLUMNS のパラメータ (続き)

列	データ型	NULL の使用	説明
DATA_PRECISION	NUMBER(10)	○	NUMERIC および DECIMAL データ型の場合は 10 進精度、FLOAT、REAL および DOUBLE データ型の場合はバイナリ精度、その他すべてのデータ型では NULL。
DATA_SCALE	NUMBER(10)	○	NUMERIC または DECIMAL データ型での小数点以下の桁数。
NULLABLE	VARCHAR2(1)	○	列で NULL を使用できるかどうかを示します。列に NOT NULL 制約が指定されている場合または列が主キーの一部である場合、値は N です。
COLUMN_ID	NUMBER(10)	×	作成された時点での列の順序番号。
DEFAULT_LENGTH	NUMBER(10)	○	列のデフォルト値の長さ。
DATA_DEFAULT	VARCHAR2(4096)	○	列のデフォルト値。
NUM_DISTINCT*	NUMBER(10)	○	表の各列内の個別値の数。
LOW_VALUE*	NUMBER(10)	○	HIGH_VALUE の説明を参照してください。
HIGH_VALUE*	NUMBER(10)	○	4 行以上の表の場合は、列内で 2 番目に低い値および 2 番目に高い値。3 行以下の表の場合は、一番低い値および一番高い値。この統計値は、値の最初の 32 バイトの内部表記の 16 進表記で表されます。

A.1.12 ALL_TAB_COMMENTS

このビューは、ユーザーが表およびビューに対して入力したコメントをリストします。このビューのパラメータを[表 A-12](#) に示します。

表 A-12 ALL_TAB_COMMENTS のパラメータ

列	データ型	NULL の使用	説明
OWNER	VARCHAR2(128)	×	TAB_COMMENTS 定義の所有者。
TABLE_NAME	VARCHAR2(128)	×	TAB_COMMENTS 定義を持つ表の名前。
TABLE_TYPE	VARCHAR2(128)	×	オブジェクトの型。
COMMENTS	VARCHAR2(4096)	×	コメントのテキスト。

A.1.13 ALL_USERS

このビューは、接続済データベースに作成されているすべてのスキーマについて次の情報を提供します。このビューのパラメータを[表 A-13](#) に示します。

表 A-13 ALL_USERS のパラメータ

列	データ型	NULL の使用	説明
USERNAME	VARCHAR2(30)	×	ユーザーの名前。
USER_ID*	NUMBER	×	ユーザーの ID 番号。
CREATED	DATE	×	ユーザー作成日付。

A.1.14 ALL_VIEWS

このビューは、ユーザーがアクセスできるビューについて次の情報を提供します。このビューのパラメータを[表 A-14](#) に示します。

表 A-14 ALL_VIEWS のパラメータ

列	データ型	NULL の使用	説明
OWNER	VARCHAR2(128)	×	ビューの所有者のユーザー名。
VIEW_NAME	VARCHAR2(128)	×	ビューの名前。
TEXT_LENGTH	NUMBER(10)	×	ビューのテキストの長さ。
TEXT	VARCHAR2(1000)	×	ビューのテキスト。

A.1.15 CAT

このビューは、ユーザーがアクセスできる表およびビューについて次の情報を提供します。このビューのパラメータを[表 A-15](#) に示します。

表 A-15 CAT のパラメータ

列	データ型	NULL の使用	説明
TABLE_NAME	VARCHAR2(128)	×	オブジェクトの名前。
TABLE_TYPE	VARCHAR2(128)	×	オブジェクトの型： TABLE または VIEW

A.1.16 COLUMN_PRIVILEGES

このビューは、ユーザーが権限付与者、権限受領者または所有者である場合、あるいは PUBLIC が権限受領者である場合の、列に対する権限付与について、次の情報を提供します。このビューのパラメータを[表 A-16](#) に示します。

表 A-16 COLUMN_PRIVILEGES のパラメータ

列	データ型	NULL の使用	説明
OWNER	VARCHAR2(128)	×	オブジェクトの所有者のユーザー名。
TABLE_NAME	VARCHAR2(128)	×	オブジェクトの名前。
COLUMN_NAME	VARCHAR2(128)	○	列の名前。
GRANTOR	VARCHAR2(128)	○	権限付与を実行したユーザーの名前。

表 A-16 COLUMN_PRIVILEGES のパラメータ (続き)

列	データ型	NULL の使用	説明
GRANTEE	VARCHAR2(128)	○	アクセス権が付与されたユーザーの名前。
GRANT_TYPE	VARCHAR2(128)	×	オブジェクトに対する権限。値は、SELECT、INSERT または DELETE です。
GRANTABLE	VARCHAR2(128)	○	GRANT OPTION を指定して権限が付与されている場合は YES、それ以外の場合は NO です。

A.1.17 DATABASE_PARAMETERS

このビューは、照合順序を制御する NLS_SORT パラメータの値をリストします。このビューのパラメータを表 A-17 に示します。

表 A-17 DATABASE_PARAMETERS のパラメータ

列	データ型	NULL の使用	説明
PARAMETER	VARCHAR2(30)	×	NLS_SORT
VALUE	VARCHAR2(128)	○	照合順序の文字列定数。値は、BINARY、FRENCH、GERMAN、CZECH または XCZECH のいずれかです。

A.1.18 DUAL

このビューは、単一行を返す問合せで使用できるダミー表です。たとえば、CURRENT_TIMESTAMP の選択に DUAL を使用できます。このビューのパラメータを表 A-18 に示します。

表 A-18 DUAL のパラメータ

列	データ型	NULL の使用	説明
DUMMY	VARCHAR2(1)	×	常に「X」。

A.1.19 TABLE_PRIVILEGES

このビューは、オブジェクトに対する権限付与について、ユーザーまたは PUBLIC が権限受領者である場合の情報を提供します。このビューのパラメータを[表 A-19](#)に示します。

表 A-19 TABLE_PRIVILEGES のパラメータ

列	データ型	NULL の使用	説明
OWNER	VARCHAR2(128)	×	オブジェクトの所有者のユーザー名。
TABLE_NAME	VARCHAR2(128)	×	オブジェクトの名前。
GRANTOR	VARCHAR2(128)	○	権限付与を実行したユーザーの名前。
GRANTEE	VARCHAR2(128)	○	アクセス権が付与されているユーザーの名前。
GRANT_TYPE	VARCHAR2(128)	×	オブジェクトに対する権限。値は、SELECT、INSERT または DELETE のいずれかです。
GRANTABLE	VARCHAR2(128)	○	GRANT OPTION を指定して権限が付与されている場合は YES、それ以外の場合は NO です。

A.1.20 USER_OBJECTS

このビューは、ユーザーがアクセスできるオブジェクトについて次の情報を提供します。このビューのパラメータを[表 A-20](#)に示します。

表 A-20 USER_OBJECTS のパラメータ

列	データ型	NULL の使用	説明
OWNER	VARCHAR2(128)	×	オブジェクトの所有者のユーザー名。
OBJECT_NAME	VARCHAR2(128)	×	オブジェクトの名前。
OBJECT_ID	NUMBER(10)	×	オブジェクトのオブジェクト識別子。
OBJECT_TYPE	VARCHAR2(128)	○	オブジェクトの型：TABLE、VIEW、INDEX、SEQUENCE または SYNONYM

表 A-20 USER_OBJECTS のパラメータ (続き)

列	データ型	NULL の使用	説明
CREATED	DATE	○	オブジェクトの作成タイムスタンプ。
LAST_DDL_TIME	DATE	○	DDL コマンド (GRANT および REVOKE を含む) の結果、オブジェクトが最後に変更されたタイムスタンプ。
CREATED_TIME	VARCHAR2(128)	○	オブジェクト (文字データ) の作成タイムスタンプ。
STATUS*	VARCHAR2(128)	○	オブジェクトのステータス: VALID、INVALID または N/A (常に有効)

Oracle Lite ユーティリティ

この付録では、次に示す Oracle Lite データベースのユーティリティの使用方法を説明します。表 B-1 に示すユーティリティの名前はアルファベット順です。

表 B-1 ツールとユーティリティ

ユーティリティ	説明
CREATEDB	Oracle Lite データベースの作成に使用します。
DECRYPDB	Oracle Lite データベースの復号化に使用します。
ENCRYPDB	Oracle Lite データベースの暗号化に使用します。
Mobile SQL	Mobile SQL は、Oracle Lite データベースに接続するための GUI インタフェースです。
ODBINFO	このユーティリティは、Oracle Lite データベースのバージョン・ナンバーとボリューム ID を見つけるために使用します。
REMOVEDB	Oracle Lite データベースの削除に使用します。

B.1 CE デバイスでのコマンドライン・ユーティリティの実行

Pocket PC でコマンドライン・アプリケーションを実行するには一連のボタンをクリックする必要がありますが、その操作はモデルごとに少し異なります。

B.1.1 Compaq 社の iPaq

「アクション」ボタン（画面の下の方ボタン）を押した状態で、画面右上隅をクリックしたまま押し続けます。これによってメニューが開き、「ファイル名を指定して実行」メニュー項目を選択するとダイアログがポップアップして、実行するプログラムのファイル名を入力するよう要求するプロンプトが表示されます。

B.1.2 HP 社の Jornada と Casio

デバイスの左側のダイヤルを押した状態で、画面右上隅をクリックしたまま押し続けます。同様のメニューがポップアップします。

B.2 CREATEDB

説明

データベースを作成するためのユーティリティです。

構文

```
CREATEDB DataSourceName DatabaseName [VoLiD]
```

キーワードとパラメータ

DataSourceName

データ・ソース名で、デフォルトのデータベース・ディレクトリの **ODBC.TXT** ファイルを検索するために使用します。

注意： 無効な DSN を指定すると、Oracle Lite はその DSN を無視し、カレント・ディレクトリにデータベースを作成します。このデータベースを ODBC 経由でアクセスするには、データベースが存在するディレクトリを指す DSN を作成する必要があります。

DatabaseName

作成するデータベース名です。フルパス名またはデータベース名のみを指定できます。データベース名のみを指定すると、データベースは、**ODBC.TXT** ファイルで指定されたデータ・ソース名のデータ・ディレクトリの下に作成されます。データベース名の拡張子は、常に **.ODB** です。指定した名前に **.ODB** が付いていない場合は、**.ODB** が付加されます。

VolID

VolID を指定すると、**POLITE.TXT** ファイルに指定されているデータベース ID のかわりに VolID がデータベース ID として使用されます。ID はデータベースごとに一意にする必要があります。

例

```
createdb polite db1
createdb polite c:\testdir\db2.odb 300
```

B.3 DECRYPDB

説明

このツールを使用すると、暗号化された Oracle Lite データベースを復号化できます。詳細は、[B.4 項「ENCRYPDB」](#)を参照してください。

構文

```
DECRYPDB DSN | NONE DBName [Password
```

キーワードとパラメータ**DSN**

復号化する Oracle Lite データベースのデータ・ソース名です。NONE を指定すると、DBName には（**.ODB** 拡張子を除く）フルパス名で入力する必要があります。

DBName

復号化するデータベース名です。DSN に NONE が指定されていると、DBName はフルパス名で入力する必要があります。

Password

オプションです。Oracle Lite データベースの暗号化に使用されたパスワードです。パスワードを指定しないと、DECRYPDB により入力を要求するプロンプトが表示されます。

コメント

Oracle Lite データベースに対してオープンされている接続がある場合は、データベースを復号化できません。

このユーティリティを別のプログラムからコールした場合、[表 B-2](#) に示す値が返されます。

表 B-2 ENCRYPDB のリターン・コード

リターン・コード	説明
EXIT_SUCCESS	成功。
EXIT_USAGE	コマンドライン引数の使用方法が適切でないか、引数にエラーがあります。
EXIT_PATH_TOO_LONG	パスが長すぎます。
EXIT_SYSCALL	復号化された新規コピーをディスク上に作成中に I/O エラーが発生しました。
EXIT_BAD_PASSWD	不適切なパスワードが指定されました。

詳細は、[B.4 項「ENCRYPDB」](#) のコメントを参照してください。

B.4 ENCRYPDB

説明

このツールにより、パスワードを使用して **Oracle Lite** データベースを暗号化したり、データベースのパスワードを変更できます。パスワードを使用すると、データベースへの許可されていないアクセスを防止し、データベースを暗号化できるので、データベース・ファイル内に格納されているデータを解釈できないようにすることが可能です。詳細は、[B.3 項「DECRYPDB」](#) を参照してください。

ENCRYPDB は、128 ビットの DES 準拠の暗号化スキームである CAST5 暗号化を使用しています。

構文

ENCRYPDB DSN | NONE DBName [New_Password [Old_Password]]

キーワードとパラメータ

DSN

暗号化する **Oracle Lite** データベースのデータ・ソース名です。NONE を指定すると、DBName には（.ODB 拡張子を除く）完全修飾されたデータベース名を入力する必要があります。DSN に NONE 以外の値を指定する場合、**ODBC.TXT** ファイル内のデータ・ソース名を指定する必要があります。

DBName

暗号化するデータベース名です。DSN に NONE が指定されていると、DBName はフルパス名で入力する必要があります。

New_Passwd と Old_Passwd

オプションです。データベースを暗号化する（または暗号化のために以前に使用された）パスワードです。このパスワードの長さは最大 128 文字です。パスワードを指定しないと、ENCRYPDB により入力を要求するプロンプトが表示されます。どちらのパスワードもオプションであるため、ユーティリティを起動するときのコマンドラインは次の 3 つの書式になります。

- パスワードなし。データベースがすでに暗号化されている場合は、ENCRYPDB はユーザーがデータベースのパスワードを変更しようとしていると解釈します。古いパスワードの入力を 1 回、新しいパスワードの入力を 2 回要求され、新しいパスワードを使用してデータベースが暗号化されます。データベースが暗号化されていない場合、ENCRYPDB によって、新しいパスワードの入力が 2 回要求され、新しいパスワードを使用してデータベースが暗号化されます。
- 片方のパスワードを指定。このパスワードは新しいパスワードとして解釈されます。データベースがすでに暗号化されている場合、ENCRYPDB によって、古いパスワードの入力が要求され、新しいパスワードを使用してデータベースが暗号化されます。
- 両方のパスワードを指定。ENCRYPDB では、最初のパスワードが新しいパスワードで、2 番目のパスワードが古いパスワードであると解釈します。

コメント

このユーティリティを別のプログラムからコールした場合、表 B-3 に示す値が返されます。

表 B-3 ENCRYPDB のリターン・コード

リターン・コード	説明
EXIT_SUCCESS	成功。
EXIT_USAGE	コマンドライン引数の使用方法が適切でないか、引数にエラーがあります。
EXIT_PATH_TOO_LONG	パスが長すぎます。
EXIT_SYSCALL	暗号化された新規コピーをディスク上に作成中に I/O エラーが発生しました。
EXIT_BAD_PASSWD	不適切なパスワードが指定されました。

デフォルトの Oracle Lite データベース (**POLITE.ODB**) は暗号化されていません。Oracle Lite データベースを暗号化すると、暗号化した Oracle Lite データベースに対して接続を確立しようとするユーザーはすべて、有効なパスワードを提供する必要があります。パスワードが提供されない場合、Oracle Lite はエラーを返します。Oracle Lite データベースに対してオープンされている接続がある場合は、データベースを暗号化できません。

Oracle Lite データベースを暗号化および復号化する場合、次の点を考慮する必要があります。

- 暗号化されたデータベースは、パスワードなしでは復号化できません。データベースは暗号化する前に、必ず安全な場所にバックアップを作成します。同じデータベースの別のユーザーは、パスワードを紛失したユーザーに対して新規のユーザー名でコピーを作成できます。それ以外の方法では、パスワードを紛失した場合、データベースのリカバリを行うことはできません。
- 他の Oracle Lite アプリケーションを実行中は、ENCRYPDB を実行しません。データベース・ファイルに他のアプリケーションが接続している場合、エラーが表示されます。
- データベースを暗号化した後、そのデータベースに接続するには、接続文字列にパスワードを含める必要があります。
- パスワードによってデータベース全体が暗号化されます。ユーザー固有のパスワードではありません。
- データベースを暗号化しても、サード・パーティが Oracle Lite データベースを削除するのを防ぐことはできません。つまり、removedb と rmdb は、パスワードをチェックせずにデータベースを削除します。許可されていないユーザーがファイル・システムを操作できないように保護するツールを使用します。
- 暗号化された Oracle Lite データベースに接続する ODBC アプリケーションは、有効なパスワードを指定する必要があります。パスワードは通常、アプリケーション内でコーディングされずに、実行時に入力が必要とされます。ほとんどの ODBC アプリケーションでは、Oracle Lite の ODBC ドライバが実行時にパスワードの入力を要求する場合、SQLConnect 関数ではなく、SQLDriverConnect 関数で DRIVER= オプションを指定できます。
- Oracle Lite のこのリリースに付属しているサンプル・アプリケーションはすべて、暗号化されていないデータベースに対して実行できます。
- DECRYPDB と ENCRYPDB を使用して（この順番で）、データベースのパスワードを変更できます。ただし、DECRYPDB によって Oracle Lite データベースが最初にプレーン・テキストで作成され、次に ENCRYPDB によって暗号化されます。プレーン・テキスト形式のデータベースが一時的に作成されるため、この方法はお勧めできません。
- 暗号化されたデータベースの場合、ユーザー名とパスワードはすべて **DSN.OPW** というファイルに書き込まれます。その後、各ユーザーは **.ODB** ファイルをアクセスする前に、パスワードを鍵として使用して **.OPW** ファイルのロックを解除できます。データベースをコピーまたはバックアップするときは、**.OPW** ファイルを含める必要があります。

B.4.1 暗号化された Oracle Lite データベースとの同期

暗号化された Oracle Lite データベースと同期するには、次の手順を実行します。

1. Mobile サーバー・リポジトリからユーザー・パスワードを取得します。
2. パスワードを大文字に変換します。たとえば、「manager」は「MANAGER」に変換します。
3. Mobile Sync (**msync.exe**) を起動して、同期を実行します。ユーザー名、パスワードおよび Mobile サーバーの URL を入力します。「適用」を選択し、次に「同期」を選択します。これにより、暗号化されていない Oracle Lite データベースが作成されます。
4. MANAGER などの変換した大文字のパスワードを使用して、ENCRYPDB ユーティリティを使用し、Oracle Lite データベースを暗号化します。
5. 同期を続行します。

B.5 Mobile SQL

Mobile SQL はアプリケーションで、ユーザーはこれを使用してローカル・データベースに対して SQL 文を実行できます。Mobile SQL は開発者用のツールで、コード例も含まれています。Mobile SQL を使用すると、基になる Oracle Lite データベース・エンジンの ODBC インタフェースと Oracle Lite OKAPI インタフェースにより提供されている関数にアクセスできます。

B.5.1 データベース・アクセス

Mobile SQL は、ODBC および OKAPI の両インタフェースを介してデータベースにアクセスします。関数のほとんどは ODBC を介して実行されますが、ODBC が処理できない関数は、OKAPI ファンクション・コールを使用して実装されています。

B.5.2 Mobile SQL の起動

Mobile SQL を起動するには、<ORACLE_HOME>%Mobile%SDK%wince をオープンし、Windows CE バージョンを表すフォルダを選択し、次に使用するデバイスのプロセッサを選択します。**mSQL.exe** ファイルをダブルクリックします。これで、標準 SQL コマンドを受け入れる GUI インタフェースが起動されます。詳細は、『Oracle9i Lite SQL リファレンス』を参照してください。

B.6 ODBINFO

説明

ODBINFO は、Oracle Lite データベースのバージョン・ナンバーとボリューム ID を見つけるために使用できます。また、ODBINFO はパラメータをいくつか表示し設定することもできます。

構文

変更を加えずに現在の情報を表示するには、次の構文を使用します。

```
odbinfo [-p passwd] DSN DBName
```

また、次の構文も使用できます。

```
odbinfo [-p passwd] NONE dbpath¥dbname.odb
```

たとえば、次のようになります。

```
odbinfo -p tiger polite polite
odbinfo NONE c:¥orant¥oldb40¥polite.odb
```

データベースを暗号化してある場合は、パスワードを含める必要があります。

オプションを設定または消去するには、DSN または NONE の前に「+」または「-」引数を 1 つ以上使用します。たとえば、次のようになります。

```
odbinfo +reuseoid -pagelog -fsync polite polite
```

パラメータ

ODBINFO では表 B-4 に示すパラメータを使用できます。

表 B-4 ODBInfo のパラメータ

パラメータ	説明
pagelog	デフォルトでは、実際に変更が <filename>.ODB に書き込まれる前に、コミットによって変更済データベース・ページのバックアップが <filename>.plg に作成されます。コミット中にアプリケーションまたはオペレーティング・システムに障害が発生した場合、トランザクションは次の接続時にクリーンな状態にロールバックされます。-pagelog を指定した場合、バックアップは作成されないため、障害が発生するとデータベースが破損する可能性があります。

表 B-4 ODBInfo のパラメータ（続き）

パラメータ	説明
fsync	<p>Oracle Lite は、通常、データベースに関連付けられている変更済バッファをコミット時にすべてディスクに書き込むように、オペレーティング・システムに対して強制します。このオプションが使用禁止になっていると（-fsync）、オペレーティング・システムは変更を後までメモリー内に保持しておくことができます。バッファがフラッシュされる前に（アプリケーションではなく）システムがクラッシュした場合、データベースも破損する可能性があります。</p> <p>odbinfo -fsync -pagelog を使用すると、多数の（自動コミットがオンに指定された）小規模なトランザクションや大規模な更新を伴うトランザクションを使用するアプリケーションのパフォーマンスが向上します。ただしデータベースが破損した場合、簡単にデータベースを修復したりデータをリカバリする方法はありません。このため、この 2 つのオプションは、(1) .ODB が定期的にバックアップされる場合、または (2) データベース内のデータが他のソースからリカバリできる場合にのみ、データベースの初期ロード中に消去するようにします。</p> <p>このオプションを使用しても、データベースをあまり更新しないアプリケーションには何の影響もありません。この場合は、トランザクション分離レベルを SINGLE USER に設定する方が効果があります。</p>
reuseoid	<p>Oracle Lite は、デフォルトでは、表にあるどの行の ROWID も表が削除されるまで再利用しません。削除されたオブジェクトにアクセスしようとすると、「スロットが削除されました」エラーが返されます。この場合、削除されたオブジェクトごとに 2 バイトのストレージが使用されるため、行が頻繁に挿入および削除される場合は、時間が経つとパフォーマンスが低下し使用ディスク領域が増えます。</p> <p>odbinfo +reuseoid を使用すると、削除された行の ROWID を新しい行で再利用できます。ただし、これでは削除されたオブジェクトが多数ある表の領域をすべて解放できないことがあります。最善の方法は、データベースの作成直後にこのオプションを設定することです。</p> <p>このオプションは、純粹にリレーショナルなアプリケーションの場合は安全です。ただし、ROWID を使用する SQL アプリケーションやオブジェクト間で直接ポインタを使用する JAC/OKAPI アプリケーションの場合は、オブジェクトが削除される前に、そのオブジェクトに対する参照がすべて NULL に設定されることを確認する必要があります。そうでないと、参照先のない参照が別の無関係のオブジェクトを指してしまう場合があります。</p>

表 B-4 ODBInfo のパラメータ（続き）

パラメータ	説明
compress	<p>このオプション（デフォルトは「on」）は、オブジェクトの実行長の圧縮を可能にします。実行長の圧縮には CPU 時間がほとんどかからないため、このオプションの選択を解除する（-compress）のは、次の場合のみです。</p> <ul style="list-style-type: none">■ オペレーティング・システム・レベルのファイル圧縮（ドライブスペースや NTFS 圧縮属性など）が使用される場合。この場合、同一データを 2 回圧縮しないので圧縮率が上がります。■ データベース内のほとんどのオブジェクトが高度に圧縮可能な状態（たとえば列がすべて NULL に設定されるなど）に頻繁に更新され、（ランダム・データを指定したバイナリ列のように）データがうまく圧縮できない場合。このような場合は、このオプション（+compress）を指定すると、表が非常に断片化される可能性があります。 <p>このオプションを変更しても、データベース内の既存のオブジェクトは圧縮も圧縮解除もされません。</p>

B.7 REMOVEDB

説明

データベースを削除するためのユーティリティです。

構文

REMOVEDB DataSourceName DatabaseName

キーワードとパラメータ

DataSourceName

削除するデータベースのデータ・ソース名です。

DatabaseName

削除するデータベースの名前です。フルパス名またはデータベース名のみを指定できます。データベース名のみを指定すると、データベースは、**ODBC.TXT** ファイルで指定されたデータ・ソース名のデータ・ディレクトリから削除されます。

例

removedb polite db1
removedb polite c:\testdir\db2.odbc

Windows CE 用 MSQL

この章では、Windows CE 用 Mobile SQL（MSQL）について説明します。内容は次のとおりです。

- [C.1 項「概要」](#)
- [C.2 項「接続」ページ](#)
- [C.3 項「表」ページ](#)
- [C.4 項「ビュー」ページ](#)
- [C.5 項「順序」ページ](#)
- [C.6 項「SQL」ページ](#)
- [C.7 項「ツール」ページ](#)

C.1 概要

Windows CE 用 MSQL は、Oracle Lite for Windows CE/Pocket PC に格納されているデータを
確認および変更するために使用する、Graphical User Interface（GUI）ベースのツールで
す。

MSQL のユーザー・インターフェースは、データの参照および変更を容易にする 6 つのタブま
たはページで構成されています。これらのタブまたはページ（「接続」、「ツール」タブなど）
は、データベース接続を必要としません。ただし、「表」、「ビュー」、「順序」および「SQL」
タブ（「データベース」タブまたは「データベース」ページとも呼ばれる）は、操作のため
にデータベース接続が必要です。

接続を確立する前に、これらの「データベース」タブを選択すると、MSQL は「ツール」
ページをアクティブにします。

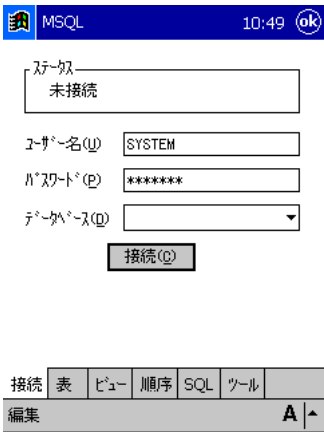
次の項では、Windows CE 用 MSQL について説明します。

C.2 「接続」ページ

このページでは、データベース接続を確立できます。

図 C-1 に示すとおり、ページの上部にデータベース接続の有無を示す接続状態が表示されま
す。

図 C-1 「接続」ページ



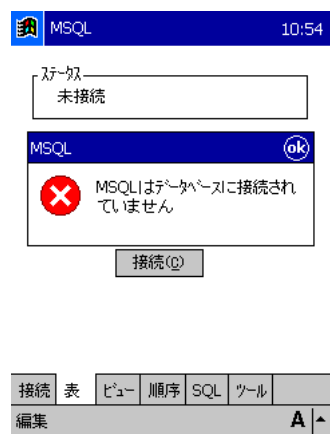
データベース接続情報が、ステータス・フィールドの下に表示されます。このページから接
続を確立する場合、「ユーザー名」、「パスワード」および「データベース」に該当する情報
を入力します。

「データベース」フィールドの下矢印をクリックすると、ドロップダウン・リストにデバイスで有効なすべての DSN が表示されます。このリストから、接続するデータベースを選択します。

接続を確立するには、DSN を選択して「接続」ボタンをクリックします。

接続が確立されると、「切断」ボタンが「接続」ボタンと置き換えられ、ページの上部に接続状態が表示されて、「表」ページがアクティブになります。MSQL をデータベースから切断すると、アラート・メッセージ「MSQL はデータベースに接続されていません」が表示され、図 C-2 に示すとおり、ステータス・フィールドに、その時点で MSQL がデータベースに接続していないことが示されます。

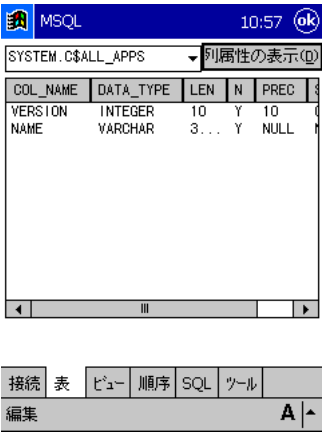
図 C-2 「MSQL はデータベースに接続されていません」メッセージ



C.3 「表」ページ

接続が確立されると、図 C-3 に示すとおり、MSQL は自動的に「表」ページをアクティブにします。

図 C-3 「表」 ページ



「表」 ページでは、データベースに格納されている表が表示されます。また、選択した表の内容も表示されます。選択した表の説明を表示するには、「列属性の表示」ボタンをクリックします。

このページでは、表データを変更できません。MSQL を使用して表データを変更するには、「SQL」タブを使用します。

このページで表示される最大行数は構成可能です（「ビュー」 ページも同様）。デフォルトでは、最大 20 行がこのページに表示されます。表示される行数を増減する場合、次のセクションを Windows CE デバイスの ¥ORACE ディレクトリに格納されている **polite.txt** ファイルに追加します。

```
[MSQL]
MAX_ROWS=100
```

C.4 「ビュー」 ページ

「ビュー」 ページでは、図 C-4 に示すとおり、データベースに格納されているビューが表示されます。また、選択したビューの内容も表示されます。選択した「ビュー」の内容を選択するには、「列属性の表示」ボタンを表示します。

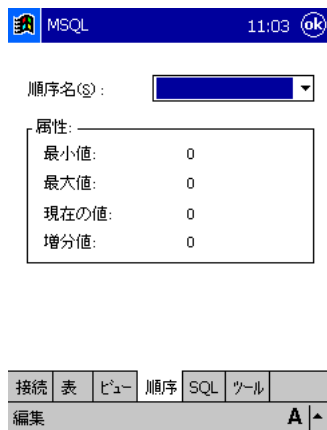
図 C-4 「ビュー」 ページ



C.5 「順序」 ページ

「順序」ページでは、データベースに格納されているシーケンスが表示されます。また、[図 C-5](#) に示すとおり、選択したシーケンスの内容も表示されます。

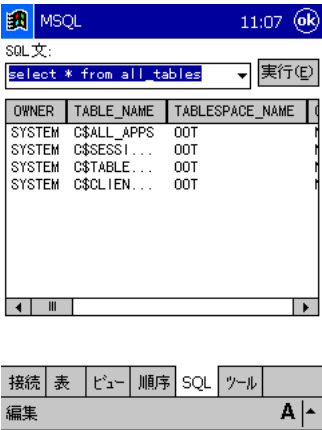
図 C-5 「順序」 ページ



C.6 「SQL」 ページ

「SQL」 ページを使用して、接続済データベースで汎用 SQL 文を実行します。図 C-6 に「SQL」 ページを示します。

図 C-6 「SQL」 ページ



また、「SQL」 ページから .sql スクリプト・ファイルを実行することもできます。「SQL 文」編集コントロールにファイルのフルパスを入力し、「実行」 ボタンをクリックします。SQL 文によってデータベース・エラーが発生した場合、これらのエラーは、Windows CE デバイスの ¥ORACE ディレクトリに格納されている **msql_err.txt** ファイルに書き込まれます。

.sql ファイルを実行する場合、MSQL は、SET、DEFINE、WHENEVER などの SQLPLUS コマンドを無視します。

C.7 「ツール」 ページ

「ツール」 ページでは、多くの機能を実行できます。図 C-7 に、「ツール」 ページを示します。

図 C-7 「ツール」 ページ

「ツール」 ページでは、次のことを実行できます。

- 新しいデータベースの作成。MSQL を使用してデータベースを作成する場合、データベース名と DSN 名を同じにしてください。
- 既存のデータベースの暗号化、データベースの復号化または既存のデータベースの整合性の検証。

既存のデータベースを暗号化する場合、データベースを選択し、データベースの新しいパスワードと以前のパスワードを入力します。その後、「暗号化」ボタンをクリックします。¥ORACE ディレクトリに **encryptdb.exe** プログラムがない場合、この操作は実行されません。

既存のデータベースを復号化する場合、データベースを選択し、「新しいパスワード」フィールドにパスワードを入力します。その後、「復号化」ボタンをクリックします。¥ORACE ディレクトリに **decryptdb.exe** がいない場合、この操作は実行されません。

既存のデータベースの整合性を検証する場合、データベースを選択して、「検証」ボタンをクリックします。¥ORACE ディレクトリに **validatedb.exe** プログラムがない場合、この操作は実行されません。

このプログラムによって生成されるメッセージを参照するには、¥ORACE ディレクトリの **valerr.txt** ファイルを確認します。

用語集

3 層 Web モデル (Three-Tier Web Model)

クライアント、中間層およびサーバーを含むインターネット・データベース構成。
Web-to-Go アーキテクチャは 3 層 Web モデルに準拠しています。

Apache Server

National Center for Supercomputing Applications (NCSA) から発表されたパブリック・ドメインの HTTP サーバー。

Java Servlet Development Kit

Java サーブレットの開発のために JavaSoft 社が提供しているツール。

Java Web Server Development Kit

Java Web Server Development Kit 1.0.1 は、JavaServer Pages (JSP) と Java サーブレットの開発のために JavaSoft 社が提供しているツールです。

JavaServer Pages

JavaServer Pages (JSP) とは、開発者がページの基になるコンテンツを変更せずにページのレイアウトを変更できるようにするテクノロジーです。JSP は HTML と Java コードを使用し、動的コンテンツとビジネス・ロジックを結び付けたプレゼンテーションを可能にします。

Java アプレット (Java Applets)

ブラウザで実行される小規模なアプリケーションで、動的コンテンツを追加することにより HTML ページの機能を拡張します。

Java サーブレット (Java Servlets)

Java で作成されているプロトコルで、プラットフォームに依存しないサーバー側コンポーネント。Java サーブレットは Java 対応のサーバーを動的に継承し、要求 - 応答方式を使用して作成されたサービスのための汎用フレームワークを提供します。

JDBC

Java Database Connectivity (JDBC) は Java クラスの標準セットで、リレーショナル・データに対してベンダーに依存しないアクセスを提供します。JDBC クラスは ODBC をモデルにしたもので、複数データベースへの同時接続、トランザクション管理、単純問合せ、バインド変数によるコンパイル済文の操作、ストアド・プロシージャへのコールなどの標準機能を提供します。JDBC では、静的 SQL と動的 SQL の両方がサポートされます。

MIME

Multipurpose Internet Mail Extensions (MIME) とは、メッセージの内容を記述するためにインターネット上で使用されるメッセージ形式です。MIME は、HTTP サーバーが配布対象ファイルのタイプを記述するために使用します。

MIME タイプ (MIME Type)

Multipurpose Internet Mail Extensions (MIME) により定義されているファイル形式。

Mobile Development Kit (Web-to-Go 用) (Mobile Development Kit for Web-to-go)

Mobile Development Kit (Web-to-Go 用) を使用すると、アプリケーション開発者は、Java サブレット、JavaServer Pages (JSP) または Java アプレットで構成される Web-to-Go アプリケーションの開発とデバッグを行えます。

Mobile サーバー (Mobile Server)

Mobile サーバーは、Web-to-Go の 3 層モデルのアプリケーション・サーバー層に常駐し、Web-to-Go 用 Mobile クライアントからのデータ変更要求を処理してデータベース・サーバー内のデータを変更します。Mobile サーバーは、Oracle HTTP Server、Apache Server およびスタンドアロンの Mobile サーバーとともに稼働するように構成できます。

ODBC

Open Database Connectivity (ODBC) は Microsoft 社の標準で、様々なプラットフォーム上でのデータベース・アクセスを可能にします。Web-to-Go 用 Mobile クライアント上では、トラブルシューティング用に ODBC サポートを使用可能にします。ODBC サポートを使用すると、ローカルな Oracle Lite データベースに格納されているクライアントのデータを表示できます。この情報を表示するには、Mobile SQL を使用します。

Oracle Lite

Oracle Lite は、Web-to-Go 用 Mobile クライアントのデータベース・コンポーネントです。クライアントがオフライン・モードのときは、アプリケーションとデータは Oracle Lite に格納されます。

Oracle データベース (Oracle Database)

Oracle データベースは、Mobile サーバーのデータベース・コンポーネントです。Web-to-Go 用 Mobile クライアントがオンライン・モードのときは、アプリケーションとデータは、Oracle データベースに格納されます。

SQL

Structured Query Language (SQL) は、リレーショナル・データベース・エンジンのほとんどで使用される非手続き型データベース・アクセス言語です。SQL 文はデータ・セットに対して実行される操作を記述します。SQL 文がデータベースに送られると、データベース・エンジンは指定されたタスクを実行するプロシージャを自動的に生成します。

Web-to-Go

Oracle Web-to-Go は、Web ベースのモバイル・データベース・アプリケーションを作成および配布するためのフレームワークです。Web-to-Go には、Web-to-Go 用 Mobile クライアント、Mobile サーバーおよび Oracle データベースで構成される 3 層データベース・アーキテクチャが含まれます。サーバーから一元管理され、Web-to-Go アプリケーションは Web-to-Go がサーバーに接続されたとき（オンライン）またはサーバーから切断されたとき（オフライン）に実行できます。オフラインのときは Web-to-Go はデータをローカルにキャッシュし、オンラインに戻ったときにそのデータをサーバーと同期します。

Web-to-Go 用 Mobile クライアント (Mobile Client for Web-to-go)

Web-to-Go 用 Mobile クライアントは、Web-to-Go の 3 層 Web モデルのクライアント層です。Mobile サーバーと Oracle Lite データベースが含まれます。オフライン・モードに切り替わると、Web-to-Go は、ユーザーのアプリケーションとデータを Oracle Lite にレプリケートします。オンラインに戻ると、Web-to-Go はデータ変更を Oracle データベースにレプリケートします。

WINDOW シーケンス (Window Sequence)

Web-to-Go がサポートする 2 つの順序のうちの 1 つで、オフライン・モードの Web-to-Go 用 Mobile クライアントに対して一意の主キー値を提供するために使用されるもの。WINDOW シーケンスには、一意の値範囲が含まれます。他のクライアントと値の範囲は重複しません。クライアントが順序の範囲内の値をすべて使用すると、Web-to-Go は新しい一意の値範囲を持つ順序を再び作成します。

一意キー (Unique key)

表の一意キーは、表の各列での一意の列または列グループです。一意キー制約を満たすには、一意キーの値が表の複数の行に出現することはできません。ただし、主キー制約とは異なり、単一列からなる一意キーは NULL 値を含むことができます。

位置付け DELETE (Positioned DELETE)

位置付け DELETE 文により、カーソルの現在行が削除されます。書式は次のとおりです。

```
DELETE FROM table
WHERE CURRENT OF cursor_name
```

位置付け UPDATE (Positioned UPDATE)

位置付け UPDATE 文により、カーソルの現在行が更新されます。書式は次のとおりです。

```
UPDATE table SET set_list  
WHERE CURRENT OF cursor_name
```

オフライン・モード (Offline Mode)

Web-to-Go 用 Mobile クライアントが Mobile サーバーから切断されている状態。オフライン・モードでは、クライアント・アプリケーションはローカルに実行され、データは Oracle Lite でアクセスおよび格納されます。「[オンライン・モード](#)」も参照。

オンライン・モード (Online Mode)

Web-to-Go 用 Mobile クライアントが Mobile サーバーに接続されている状態。「[オフライン・モード](#)」も参照。

外部キー (Foreign Key)

外部キーとは表またはビューに存在する列または列グループのことで、その値は別の表またはビューに存在する行を参照します。外部キーには、一般に、別の表の主キー値と一致する値が含まれます。「[主キー](#)」も参照。

結合 (Join)

2 つの異なる表またはビューに存在するキー（主キーと外部キーの両方）の間に確立された関係。結合は、リレーショナル・データベース内の重複したデータを排除するために正規化された表のリンクに使用します。結合リンクの一般的なものとしては、1 つの表の主キーを別の表の外部キーにリンクして、マスター・ディテール関連を確立するものがあります。結合は SQL 文の WHERE 句条件に対応します。

コントロール・センター (Control Center)

Mobile サーバー・コントロール・センターはブラウザ内で実行される Web ベースのアプリケーションで、これを使用すると Web-to-Go アプリケーションとそのユーザーの管理が容易になります。管理者はコントロール・センターを使用して、ユーザーまたはグループに対するアクセス権の付与と取消し、スナップショット・テンプレート変数の変更、Web-to-Go からのアプリケーションの削除などの機能を実行します。

サイト (Site)

Web-to-Go は、Web-to-Go 用 Mobile クライアント上の各ユーザーに対してデータベースを作成します。このデータベースはサイトと呼ばれます。1 つのクライアントに複数のサイトを含められますが、サイトは 1 人のユーザーに 1 つのみ可能です。ユーザーは、異なるクライアント上に複数のサイトを所有できます。

索引 (Index)

表内のそれぞれの行に対する高速アクセスを提供するデータベース・オブジェクト。索引を作成すると、表のデータに対して実行される問合せおよびソート操作を高速化できます。また、索引を使用して、一意キー制約や主キー制約などの制約を表に対して規定することもできます。

索引はいったん作成されると自動的にメンテナンスされ、データベース・エンジンにより可能なかぎりデータ・アクセスのために使用されます。

参照整合性 (Referential Integrity)

参照整合性は、レコードが追加、修正または削除されたときにメンテナンスされるマスター・ディテール・リレーション内の表間のリンクの精度として定義されます。

マスター・ディテール・リレーションを注意深く定義しておくことにより、参照整合性が高まります。データベース内の制約によって、データベース（クライアント / サーバー環境でのサーバー）レベルの参照整合性が規定されます。

参照整合性の目的は、孤立したレコード（マスター・レコードとの有効なリンクを持たないディテール・レコード）が作成されないようにすることです。参照整合性を規定する規則により、結果として孤立したレコードを作成するような、マスター・レコードの削除や更新、またはディテール・レコードの挿入や更新を予防できます。

実表 (Base Table)

ビューの基になるデータのソースで、表またはビューのいずれか。ビュー内のデータにアクセスするとき、実際は実表のデータにアクセスしています。

シノニム (Synonym)

表、ビュー、順序、スナップショットまたは別のシノニムに対する代替名（エイリアス）。

主キー (Primary Key)

表の主キーは、表内の各行を一意に識別するために使用される 1 つの列または列グループです。主キーを使用すると表のレコードにすばやくアクセスできます。また、主キーは 2 つの表またはビューの間の結合の基礎として頻繁に使用されます。それぞれの表に対して主キーは 1 つのみ定義できます。

主キー制約を満たすには、主キー値が表の 2 つ以上の列で使用されたり、主キーの一部の列に NULL 値が含まれないようにします。

順序 (Sequence)

連続した番号を生成するスキーマ・オブジェクト。順序を作成した後は、これを使用してトランザクション処理用の一意の順序番号を生成できます。これらの一意の整数には、主キー値を含むことができます。トランザクションで順序番号が生成される場合、トランザクションをコミットしたかロールバックしたかにかかわらず順序が即時増分されます。[「WINDOW シーケンス」](#) も参照。

スキーマ (Schema)

表、ビュー、索引、順序などを含む、名前の付いたデータベース・オブジェクトの集まり。

スナップショット (Snapshot)

スナップショットとは、Web-to-Go が Oracle データベースからリアルタイムで取得するアプリケーション・データのコピーで、オフラインになる前にクライアントにダウンロードされます。スナップショットは、データベース表全体のコピー、または表の行のサブセットのコピーです。ユーザーが初めてオンラインからオフラインに切り替えるとき、Web-to-Go はクライアント・マシン上にスナップショットを自動的に作成します。その後、オンラインまたはオフラインに切り替えるたびに、Web-to-Go はスナップショットの複雑さに応じて、スナップショットを最新のデータでリフレッシュするか、全体を再作成します。

整合性制約 (Integrity Constraint)

表の 1 つ以上の列に入力できる値を制限する規則。

接続 (Connected)

サーバーに接続されているユーザー、アプリケーションまたはデバイスを指す一般的な用語。Web-to-Go 用 Mobile クライアントは、オンライン・モードのときに接続されています。

切断 (Disconnected)

サーバーに接続されていないユーザー、アプリケーションまたはデバイスを指す一般的な用語。Web-to-Go 用 Mobile クライアントは、オフライン・モードのときに切断されています。

データベース・オブジェクト (Database Object)

データベース・オブジェクトとは、表、ビュー、順序、索引、スナップショットまたはシノニムなどの名前の付けられたデータベース構造体です。

データベース・サーバー (Database Server)

Web-to-Go の 3 層 Web モデルの 3 番目の層。アプリケーション・データを格納します。

同期 (Synchronization)

Web-to-Go 用 Mobile クライアントと Oracle データベースの間でデータをレプリケートするために Web-to-Go が使用するプロセス。Web-to-Go は、ユーザーがオフライン・モードに切り替わったときに、ユーザーのアプリケーションおよびデータを Oracle Lite にレプリケートします。オンラインに戻ると、Web-to-Go はデータ変更を Oracle データベースにレプリケートします。

トランザクション (Transaction)

リレーショナル・データベース内の選択されたデータに対して加えられる一連の変更。トランザクションは通常、INSERT、UPDATE、DELETE などの SQL 文を使用して実行します。トランザクションは、コミットされた（変更が永続的になる）とき、またはロールバックされた（変更が破棄された）ときに完了します。

トランザクションの前に問合せが実行されることがよくあります。問合せを使用して、変更対象の特定のレコードをデータベースから選択しておきます。「SQL」も参照。

パッケージ・ウィザード (Packaging Wizard)

パッケージ・ウィザードを使用すると、管理者が Web-to-Go アプリケーションを Mobile サーバー・リポジトリにパブリッシュできます。管理者は、パッケージ・ウィザードを使用して新しい Web-to-Go アプリケーションを作成したり、既存のアプリケーション定義を編集できます。

パブリケーション・アイテム (Publication Item)

パブリケーション・アイテムは、クライアントがアクセスできるデータ・サブセットを指定する SQL SELECT 文です。パブリケーション・アイテムは、通常クライアント・デバイス上のレプリカ表に対応します。パブリケーション・アイテムは、Mobile サーバー Admin API を使用して作成できます。この API には、パブリッシュ・サブスクライブ・モデルを実装する Java 関数が含まれています。これらの関数は、Java プログラムの内部から標準のファンクション・コールとしてコールできます。

ビュー (View)

1 つ以上の表（または他のビュー）から選択されたデータをカスタマイズして表したもの。ビューは「仮想的な表」のようなもので、複数の表（実表と呼ばれます）およびビューからのデータを関連させ、組み合わせることができます。ビューは表示されるデータの選択条件を指定できるため、一種の「格納された問合せ」といえます。

ビューは、表のように、行と列に編成されます。ただし、ビューには、データそのものは含まれません。ビューを使用すると、複数の表またはビューを 1 つのデータベース・オブジェクトとして扱うことができます。

表 (Table)

行と列に編成されたデータを格納するデータベース・オブジェクト。上手に設計されたデータベースでは、各表に単一のトピックに関する情報（たとえば、従業員や顧客の住所など）が格納されます。

マスター・ディテール・リレーション (Master-Detail Relationship)

1 つの表またはビュー（ディテール表またはビュー）の複数行が、別の表またはビュー（マスター表またはビュー）の単一のマスター行に関連付けられている場合に、マスター・ディテール・リレーションがデータベース内の表またはビューの間に存在するといえます。

マスター行およびディテール行は通常、ディテール表またはビュー内の外部キー列と一致するマスター表またはビュー内の主キー列により結合されます。

主キーの値を変更した場合、アプリケーションでは、外部キーの値が主キーの値と一致するように一連の新しいディテール・レコードを問い合わせる必要があります。たとえば、EMP 表内のディテール・レコードが、DEPT 表内のマスター・レコードと同期される場合、DEPT 内の主キーは DEPTNO で、EMP 内の外部キーは DEPTNO にします。「主キー」および「外部キー」も参照。

モードの切替え (Switching Modes)

Web-to-Go 用 Mobile クライアントがオフラインに切り替わったりオンラインに戻るために使用するプロセス。クライアントがオフライン・モードに切り替わると、オフラインで作業するために必要なすべてのアプリケーションとデータが Oracle Lite にダウンロードされます。クライアントがオンラインに戻ったときに、Oracle Lite に対するデータ変更を Oracle データベースと同期します。

Mobile サーバー・リポジトリ (Mobile Server Repository)

Mobile サーバー・リポジトリとは、仮想ファイル・システムです。このリポジトリは、すべてのアプリケーション・ファイルとアプリケーション定義を含む永続リソース・リポジトリです。

レジストリ (Registry)

レジストリには、Web-to-Go の一意の名前と値のペアが含まれます。レジストリの名前はすべて一意である必要があります。

レプリケーション (Replication)

分散データベース・システムを構成する複数のデータベース内で、データベース・オブジェクトをコピーしメンテナンスするプロセス。1つのサイトに適用された変更が取得されローカルに格納されてから、各リモート・サイトに転送され適用されます。レプリケーションは、共有データに対する高速のローカル・アクセスをユーザーに提供し、データ・アクセスの代替オプションを提供してアプリケーションの使用を保護します。1つのサイトが使用不能になっても、残りのサイトに対して問い合わせたり更新できます。

レプリケーションの競合 (Replication Conflict)

レプリケーションの競合は、同一のデータに対して矛盾する変更が加えられたときに発生します。Web-to-Go では、切断されているクライアント用の順序値を使用して、レプリケーションの競合を回避します。

ワークスペース (Workspace)

Mobile サーバーのワークスペースとは、Web-to-Go アプリケーションに対するアクセスをユーザーに提供する Web ページです。Web-to-Go は、ユーザーが Web-to-Go にログインした後に、ユーザーのブラウザ内にワークスペースを生成します。ワークスペースは、ユーザーが使用できるすべてのアプリケーションのアイコン、リンクおよび説明を表示します。アプリケーションを使用できるようになるのは、管理者がアプリケーションを Web-to-Go システムにパブリッシュし、ユーザーに対してアクセス権を付与した後です。

A

Active Data Objects for Windows CE, 1-16
ActiveSync
 概要, 3-6
 同期, 3-8
ADOCE, 1-16
ALL_CONS_COLUMNS
 システム・カタログ・オブジェクト, A-4
ALL_CONSTRAINTS
 システム・カタログ・オブジェクト, A-3
ALL_IND_COLUMNS
 システム・カタログ・オブジェクト, A-5
ALL_INDEXES
 システム・カタログ・オブジェクト, A-5
ALL_OBJECTS
 システム・カタログ・オブジェクト, A-6
ALL_SEQUENCES
 システム・カタログ・オブジェクト, A-7
ALL_SYNONYMS
 システム・カタログ・オブジェクト, A-7
ALL_TAB_COLUMNS
 システム・カタログ・オブジェクト, A-9
ALL_TAB_COMMENTS
 システム・カタログ・オブジェクト, A-11
ALL_TABLES
 システム・カタログ・オブジェクト, A-8
ALL_USERS
 システム・カタログ・オブジェクト, A-11
ALL_VIEWS
 システム・カタログ・オブジェクト, A-6, A-12
AlterPublicationItem, 6-50

C

CAT
 システム・カタログ・オブジェクト, A-12
C/C++ インタフェース, 3-30
CE デバイス上の Mobile クライアント, 3-6
COLUMN_PRIVILEGES
 システム・カタログ・オブジェクト, A-12
COM インタフェース, 3-24
Consolidator API, 6-2
CREATEDB
 使用方法, B-2
CreateSequencePartition, 6-27

D

DDL 操作, 6-25
DML 操作のコールバックのカスタマイズ, 6-66
doCompose メソッド, 6-35
DUAL
 システム・カタログ・オブジェクト, A-13

E

eMbedded Visual Tools のチュートリアル, 2-1

G

getDownloadInfo メソッド, 6-42

I

INSTEAD OF トリガー, 6-52

J

jar ファイル
 パッケージ・ウィザードによるパッケージ化, 5-21
Java インタフェース, 3-16
JDBC, 1-14
JDBC ドライバ, 1-14

M

Microsoft eMbedded Visual Tools, 1-14
Mobile Development Kit, 1-14
Mobile サーバー
 概要, 1-7, 6-24
Mobile サーバーからのファイルのインストール, 3-6
MyCompose, 6-32
 doCompose, 6-35
 needCompse メソッド, 6-33

N

needCompose メソッド, 6-33
Null Sync コールアウト, 6-63

O

openConnection, 6-11
Oracle Lite ユーティリティ, 1-13

P

PL/SQL, 6-57
Pocket PC 3.0, 1-14

R

REMOVEDB
 使用方法, B-10
Resource Manager クラス
 パスワードの変更, 6-25

S

setPassword, 6-25
Sync Discovery API, 6-42
 getDownloadInfo メソッド, 6-42

T

TABLE_PRIVILEGES
 システム・カタログ・オブジェクト, A-14

U

USER_OBJECTS
 システム・カタログ・オブジェクト, A-14

V

Visual Basic, 1-14
Visual Basic のチュートリアル, xv, 2-1
Visual C++, 1-14
Visual C++ for Windows CE, 1-14

W

Windows CE デバイスのセットアップ, 1-19

あ

アプリケーション
 パッケージ・ウィザードによる編集, 5-23
 パッケージ・ウィザードによる命名, 5-4
 パブリッシュ, 5-22
 ファイルのリスト表示, 5-7
「アプリケーション」パネル
 パッケージ・ウィザード, 5-4
暗号化されたデータベースの同期, 3-6, B-7

い

依存関係のヒントの削除, 6-31
依存関係のヒントの作成, 6-30, 6-52
インストール, 1-17

う

ウィニング・ルール, 6-69

え

エラー, 6-69

お

オブジェクト・カーネル API (OKAPI), 1-15

親表

 INSTEAD OF トリガー, 6-52

 更新可能, 6-51

 ヒント, 6-52

か

開発過程, 1-16

開発ツール, 1-14

外部キー制約, 6-63, 6-64

 違反, 6-63

仮想主キー, 6-54

き

キュー・インタフェース, 6-58

競合, 6-69

く

クライアント

 パブリケーションへのサブスクライブ, 6-23

クライアント・データベース

 作成, 6-25

こ

高速リフレッシュと更新, 6-51

固有のアプリケーション, 1-14

さ

索引

 パブリケーション・アイテム用の作成, 6-17

サブスクリプション, 6-2

 インスタンス化, 6-23

サブスクリプション・パラメータ, 6-2

 定義, 6-16

し

システム・カタログ・オブジェクト

 ALL_CONS_COLUMNS, A-4

 ALL_CONSTRAINTS, A-3

 ALL_IND_COLUMNS, A-5

 ALL_INDEXES, A-5

 ALL_OBJECTS, A-6

 ALL_SEQUENCES, A-7

 ALL_SYNONYMS, A-7

 ALL_TAB_COLUMNS, A-9

 ALL_TAB_COMMENTS, A-11

 ALL_TABLES, A-8

 ALL_USERS, A-11

 ALL_VIEWS, A-6, A-12

 CAT, A-12

 COLUMN_PRIVILEGES, A-12

 DUAL, A-13

 TABLE_PRIVILEGES, A-14

 USER_OBJECTS, A-14

システム・カタログ・ビュー, A-1

主キー索引, 6-25

す

スナップショット, 6-2

 インポート, 5-17

 パッケージ・ウィザードによる作成, 5-14

 編集, 5-18

「スナップショット」パネル

 パッケージ・ウィザード, 5-11

せ

制限選択条件, 6-69

宣言によるパブリケーション・アイテムの作成, 1-17

選択同期, 3-41

て

データ型

 Oracle Lite, 6-71

 マッピング, 6-71

データ・ソース名の作成, 3-6

「データベース」パネル

 パッケージ・ウィザード, 5-10

データベース・ユーティリティ

 CREATEDB, B-2

 REMOVEDB, B-10

テスト

 デバイスでの, 1-20

と

同期
 ActiveSync, 3-8
 エラー, 6-69
 競合, 6-69
トランザクション
 実行, 6-70
 ページ, 6-71

は

ページ、トランザクション, 6-71
バージョンング, 6-69
パスワードの変更, 6-25
パッケージ・ウィザード
 jar ファイルのパッケージ化, 5-21
 「アプリケーション」パネル, 5-4
 起動, 5-2
 スナップショットのインポート, 5-17
 スナップショットの編集, 5-18
 「スナップショット」パネル, 5-11, 5-14
 「データベース」パネル, 5-10
 ファイルのソート, 5-9
 「ファイル」パネル, 5-7
パブリケーション, 6-2
 クライアントのサブスクライブ, 6-23
 作成, 6-11
パブリケーション・アイテム, 6-2
 索引の作成, 6-17
 作成, 6-13
 パブリケーションへの追加, 6-18
パブリケーション・アイテムの作成, 1-17
パブリケーション・アイテムの間合せのキャッシング,
 6-56
パブリッシュ・サブスクライブ・モデル, 6-2

ひ

ビュー
 高速リフレッシュと更新, 6-51
ビューの完全リフレッシュ, 6-54
ビューの高速リフレッシュ, 6-53
ヒント, 6-52

ふ

「ファイル」パネル
 パッケージ・ウィザード, 5-7
フラッシュ・メモリー・カードへのデータベースの格
 納, 3-7
プラットフォームの選択, 5-4
プログラミング・インタフェース
 C/C++, 3-30
 COM, 3-24
 Java, 3-16
プログラムによるパブリケーション・アイテムの作成,
 1-18

ま

マップ表のパーティション API, 6-47

り

リフレッシュ
 完全, 6-54
 高速, 6-53
リモート・オブジェクトのシノニムのパブリッシュ,
 6-28
リモート・データベース・リンクのサポート, 6-28

れ

レプリケーション
 Windows CE デバイスでの, 1-19
レプリケート・シーケンス
 クライアントごとのパーティション化, 6-26
 作成, 6-25
レプリケート・シーケンスのパーティション化, 6-26