ODBC Guide

Zen v16

Activate Your Data™

目次

このドキュメントについて

このドキュメントでは、Zen の ODBC 仕様のサポートについて説明します。

このドキュメントの読者

このドキュメントは、ODBC アーキテクチャや ODBC ドライバー コンポーネントの基本概念について理解ができていることと、Microsoft ODBC Software Development Kit を使用できる環境にあることを前提としています。

また、最新のデータベース原理とその用語、C 言語、およびコンパイラやリンカーなどの開発環境についての知識と経験があることを前提にしています。

メモ： 特段の記述がない限り、本書における Zen 製品へのすべてのリファレンスは、現行バージョンについて述べています。

詳細情報

ODBC 仕様の詳細については、Microsoft ODBC ドキュメントを参照してください。

ODBC 仕様のサポート

以下のトピックでは、Zen の Open Database Connectivity（ODBC）仕様のサポートについて説明します。

- ODBC 仕様のサポート
- ODBC API サポート
- ODBC 属性サポート
- ODBC 記述子フィールドのサポート

Zen で ODBC 構成オプションを設定する方法については、DSN のセットアップおよび接続文字列を参照してください。

ODBC 仕様のサポート

ODBC（Open Database Connectivity）とは、Microsoft によって開発された、データベース マネージメント システム（DBMS）にアクセスするための標準 API です。この標準は、長年にわたって進化し続けてきました。Zen リレーショナル インターフェイスは、コア、レベル 1、およびレベル 2 のインターフェイス サポート レベルに対する ODBC v3.51 仕様をサポートしています（レベル 3 はサポート対象外です）。

ODBC インターフェイス サポートの例外

コア レベル

SQL_BEST_ROWID

リレーショナル インターフェイスでは、テーブル内の行を識別する最適な列セットとして固有のインデックスが使用されます。

IDENTITY 列のあるテーブルに新しい列を挿入した場合、リレーショナル インターフェイスは IDENTITY 列に割り当てられた値を返しません。IDENTITY 列の値は @@IDENTITY 変数を使用することによって判断できます。『*SQL Engine Reference*』の @@IDENTITY および @@BIGIDENTITY を参照してください。

レベル 2

以下はサポート対象外です。

- SQL_ATTR_LOGIN_TIMEOUT
- SQL_BEST_ROWID（上記の説明を参照）
- SQL_ROWVER

ODBC API サポート

次の表は、リレーショナル インターフェイスによりサポートされる ODBC API 関数、および ODBC サポート レベルの一覧を示します。ODBC API の詳細については、Microsoft の ODBC ドキュメントを参照してください。

ODBC 関数	ODBC サポート レベル
SQLAllocHandle	コア
SQLBindCol	コア
SQLBindParameter	コア
SQLBrowseConnect	レベル 1
SQLBulkOperations	レベル 1
SQLCancel	コア
SQLCloseCursor	コア
SQLColAttribute	コア
SQLColumnPrivileges	レベル 2
SQLColumns	コア
SQLConnect	コア
SQLCopyDesc	コア
SQLDataSources	コア
SQLDescribeCol	コア
SQLDescribeParam	レベル 2
SQLDisconnect	コア

ODBC 関数	ODBC サポート レベル
SQLDriverConnect	コア
SQLDrivers	コア
SQLEndTran	コア
SQLExecDirect	コア
SQLExecute	コア
SQLExtendedFetch	コア
SQLFetch	コア
SQLFetchScroll	コア
SQLForeignKeys	レベル 2
SQLFreeHandle	コア
SQLFreeStmt	コア
SQLGetConnectAttr	コア
SQLGetCursorName	コア
SQLGetData	コア
SQLGetDescField	コア
SQLGetDescRec	コア
SQLGetDiagField	コア
SQLGetDiagRec	コア
SQLGetEnvAttr	コア
SQLGetFunctions	コア
SQLGetInfo	コア
SQLGetStmtAttr	コア
SQLGetTypeInfo	コア
SQLMoreResults	レベル 1
SQLNativeSql	コア
SQLNumParams	コア
SQLNumResultCols	コア

ODBC 関数	ODBC サポート レベル
SQLParamData	コア
SQLPrepare	コア
SQLPrimaryKeys	レベル 1
SQLProcedureColumns	レベル 1
SQLProcedures	レベル 1
SQLPutData	コア
SQLRowCount	コア
SQLSetConnectAttr	コア
SQLSetCursorName	コア
SQLSetDescField	コア
SQLSetDescRec	コア
SQLSetEnvAttr	コア
SQLSetPos	レベル 1
SQLSetStmtAttr	コア
SQLStatistics	コア
SQLTablePrivileges	レベル 2
SQLTables	コア

ODBC API サポートの例外

次のセクションには、ODBC API サポートの例外に関する詳細が記述されています。

SQLGetData

アプリケーションで SQLGetData を呼び出して SQL_C_NUMERIC 構造体にデータを返した場合、ODBC 標準では、SQL_DESC_SCALE フィールドはゼロに設定され、SQL_DESC_PRECISION フィールドにはドライバー定義の精度が使用されます。

Zen は、メタデータに定義されている小数位およびドライバー定義の精度（桁数）の値を使用します。次の例を考えて見ましょう。この例では、小数位は 2 に設定されます。

```
CREATE TABLE testnum (col1 NUMERIC(10,2))
INSERT INTO testnum VALUES (10.34)
SELECT * FROM testnum
```

SELECT ステートメントは 10.00 ではなく 10.34 を返します。

SQLGetTypeInfo

SQLGetTypeInfo を使用すると、リレーショナル インターフェイスによって指定されたネイティブ データ型名（タイプ名）のリストが生成されます。たとえば、SQL_CHAR は CHARACTER に割り当てられます。CREATE TABLE ステートメントまたは ALTER TABLE ステートメント内の列のデータ型名、あるいはプロシージャのパラメーターや、プロシージャおよびトリガーで宣言された変数のデータ型名には、この関数から戻される名前を使用してください。

サポートされている ODBC データ型の一覧については、使用できるデータ型を参照してください。

SQLGetInfo

リレーショナル インターフェイスは、SQL_DRIVER_VER と SQL_DBMS_VER に対して同一の値を返します。このバージョン値は次の形式で返されます。

```
aa.bb.cccc ddd
```

この値は、次の表で説明されているように 4 つの構成要素として解釈できます。

要素	値	説明
aa	メジャー バージョン	データベース エンジンのメジャー バージョン。
bb	マイナー バージョン	データベース エンジンのマイナー バージョン。一般に、サービス パックで更新されます。
cccc	ビルド番号	リリースを特定するビルドの詳細。
ddd	ポイント ビルド	ビルドのマイナー アップデート。

次の表では、SQLGetInfo で一般的に返されるその他の値についてまとめています。

項目	値の例
SQL_DRIVER_NAME	W3ODBCCI.DLL
SQL_DRIVER_VER	10.00.0147 012
SQL_DRIVER_ODBC_VER	03.51
SQL_DBMS_NAME	Zen
SQL_DBMS_VER	10.00.0147 012
SQL_ODBC_VER	03.52.0000
SQL_ODBC_API_CONFORMANCE	SQL_OAC_LEVEL2
SQL_ODBC_INTERFACE_CONFORMANCE	SQL_OIC_LEVEL2

SQLSpecialColumns

Zen リレーショナル インターフェイスでは、テーブル内の行を一意に識別する最適な列セットとして固有のインデックスが使用されます。新しい行が挿入されたとき、リレーショナル インターフェイスは IDENTITY 列の値を返しません。IDENTITY 列の値は @@IDENTITY 変数を使用することによって判断できます。『*SQL Engine Reference*』の @@IDENTITY および @@BIGIDENTITY を参照してください。

ODBC 属性サポート

リレーショナル インターフェイスは ODBC v3.51 の属性サポートを提供していますが、次に挙げる例外があります。

接続属性サポート

次の表は、ODBC 接続属性サポートの例外を示します。

fOption	解説
SQL_ATTR_AUTO_IPD	デフォルト値は SQL_TRUE です。Pervasive ODBC ドライバーは、この属性値を SQL_FALSE に設定することを許可していません。

fOption	解説
SQL_ATTR_CONNECTION_TIMEOUT	デフォルト値は 0 です。それ以外の値はサポートされていません。
SQL_ATTR_METADATA_ID	デフォルト値は SQL_FALSE です。Pervasive ODBC ドライバーは、この属性値を SQL_TRUE に設定することを許可していません。

ステートメント属性のサポート

次の表は、ODBC ステートメント属性サポートの例外を示します。

fOption（数値）	解説
SQL_ATTR_ENABLE_AUTO_IPD(15)	デフォルト値は SQL_TRUE です。Pervasive ODBC ドライバーは、この属性値を SQL_FALSE に設定することを許可していません。
SQL_ATTR_METADATA_ID(10014)	デフォルト値は SQL_FALSE です。Pervasive ODBC ドライバーは、この属性値を SQL_TRUE に設定することを許可していません。
SQL_ATTR_PARAM_BIND_TYPE(18)	SQL_PARAM_BIND_BY_COLUMN のみサポートされます。
SQL_ATTR_QUERY_TIMEOUT(0)	SQLSetStmtAttr および SQLSetConnectAttr を介してサポートされます。SQLExecDirect、SQLExecute、SQLFetch および SQLExtendedFetch にのみ適用できます。DDL ステートメントに適用してはいけません。

ODBC 記述子フィールドのサポート

Zen リレーショナル エンジンは ODBC v 3.51 記述子フィールド サポートを提供していますが、次の例外があります。

オプション	解説
SQL_DESC_BIND_TYPE	アプリケーション パラメーター記述子（APD）では、SQL_BIND_BY_COLUMN のみサポートされます。

オプション	解説
SQL_DESC_ROWVER	ODBC インターフェイス サポートの例外 *を参照してください。*

記述子フィールドおよびビット数値

一部の記述子フィールドは ODBC のさまざまな SQLSet および SQLGet 関数を介して設定できますが、これらの関数が 64 ビット値対応に変更されている一方、それ以外の関数はまだ 32 ビット値対応であることに注意してください。64 ビット ODBC ドライバーを使用している場合、このようなフィールドを設定および取得する際には、適切なサイズの変数を使用するようにしてください。詳細については、Microsoft の ODBC ドキュメントを参照してください。

説明の要点は、SQL_ROWSET_SIZE は SQLGetStmtOption と SQLGetStmtAttr の両方でサポートされるということです。64 ビット ODBC ドライバーを使用し、SQLGetStmtOption または SQLGetStmtAttr を呼び出した場合、属性パラメーターが SQL_ROWSET_SIZE に設定されている場合には、*ValuePtr に 64 ビット値が返されます。

SQLSetStmtOption の各種オプション

このセクションでは、以下の SQLSetStmtOption の各種オプションに対する Zen のサポートについて説明します。

- SQL_BIND_TYPE
- SQL_CONCURRENCY
- SQL_CURSOR_TYPE
- SQL_RETRIEVE_DATA
- SQL_ROWSET_SIZE
- SQL_USE_BOOKMARKS

次の表は、各オプションで有効な設定値を示します。

オプション	ODBC カーソル ライブラリ	現在の Zen ODBC ドライバー
SQL_BIND_TYPE	SQL_BIND_BY_COLUMN または行方向のバインドを示す長さ	SQL_BIND_BY_COLUMN または行方向のバインドを示す長さ

オプション	ODBC カーソル ライブラリ	現在の Zen ODBC ドライバー
SQL_CONCURRENCY	SQL_CONCUR_READ_ONLY または SQL_CONCUR_ VALUES （SQL_CONCUR_ROWVER の場合、ライブラリは SQL_CONCUR_VALUES を代用し、SQL_SUCCESS_WITH_INFO を返して、SQLSTATE に 01S02 を設定する） （SQL_CONCUR_LOCK の場合、ライブラリは SQLSTATE = S1C00 の SQL_ERROR を返す）	SQL_CONCUR_READ_ONLY、SQL_CONCUR_ROWVER または SQL_CONCUR_LOCK （SQL_CONCUR_VALUES の場合、ドライバーは自動的に SQL_CONCUR_ROWVER を代用する）
SQL_CURSOR_TYPE	SQL_CURSOR_FORWARD_ONLY または SQL_CURSOR_STATIC （SQL_CURSOR_KEYSET_DRIVEN および SQL_CURSOR_DYNAMIC の場合、ライブラリは SQL_CURSOR_STATIC を代用し、SQL_SUCCESS_WITH_INFO を返して、SQLSTATE に 01S02 を設定する）	SQL_CURSOR_FORWARD_ONLY または SQL_CURSOR_STATIC または SQL_CURSOR_DYNAMIC （SQL_CURSOR_KEYSET_DRIVEN の場合、ライブラリは SQL_CURSOR_STATIC を代用し、SQL_SUCCESS_WITH_INFO を返して、SQLSTATE に 01S02 を設定する）
SQL_RETRIEVE_DATA	SQL_RD_ON （SQL_RD_OFF の場合、ライブラリは SQLSTATE = S1C00 の SQL_ERROR を返す）	SQL_RD_ON または SQL_RD_OFF
SQL_ROWSET_SIZE	行セットの行数を示す値。これは最大行セット サイズを限度とします。	行セットの行数を示す値。これは最大行セット サイズを限度とします。
SQL_USE_BOOKMARKS	SQL_UB_ON または SQL_UB_OFF	SQL_UB_ON または SQL_UB_OFF

Zen ODBC リファレンス

このリファレンスでは、リレーショナル インターフェイスおよび ODBC の接続文字列、メタデータ バージョン、制限、SQL 文法などの情報について、以下のトピックで説明します。

- データ ソース名接続文字列キーワード
- 開いたテーブルを閉じる
- SQL 文法のサポート
- 使用できるデータ型
- 無限の表現

データ ソース名接続文字列キーワード

DSN への接続に使用される接続文字列には、ドライバーの定義済みキーワードをいくつでも含むことができます。これらのキーワードを使用することにより、ドライバーはデータ ソースへ接続するのに十分な情報を得ることができます。ドライバーは、データ ソースとの接続に必要なキーワードを定義します。

Zen の接続文字列およびキーワードの完全な説明については、ODBC 接続文字列を参照してください。

開いたテーブルを閉じる

SQLFreeStmt に SQL_CLOSE オプションを指定して呼び出すと、SQLSTATE は変更されますが、hStmt が使用している開いたテーブルは閉じません。hStmt が現在使用しているテーブルを閉じるには、SQLFreeStmt に SQL_DROP オプションを指定して呼び出す必要があります。

次の例では、Emp と Dept のテーブルは開いたままになります。

```
SQLPrepare(hStmt, "SELECT * FROM Emp, Dept", SQL_NTS)
SQLExecute(hStmt)
SQLFetch until SQL_No_Data_Found
SQLFreeStmt(hStmt, SQL_CLOSE)
```

その次に SQLPrepare が hStmt で呼び出されたとき、前のステートメントで使用したテーブルが閉じられます。たとえば、次の呼び出しを実行すると、Emp と Dept の両方のテーブルは Zen によって閉じられます。

```
SQLPrepare(hStmt, "SELECT * FROM Customer",SQL_NTS)
```

その後、次の呼び出しによって Customer テーブルを閉じます。

```
SQLFreeStmt(hStmt, SQL_DROP)
```

SQL 文法のサポート

ODBC v 2.5 仕様は、最小、コア、拡張の 3 つの SQL 文法レベルを提供します。レベルが高くなるほど、より完全なデータ定義の実装とデータ操作言語のサポートが提供されます。

リレーショナル インターフェイスでは、最小 SQL 文法に加え、多数のコアおよび拡張のステートメント文法を完全サポートしています。リレーショナル インターフェイスがサポートしている SQL 文法は、次の表に要約されています。ステートメント文法は、『*SQL Engine Reference*』に記載されています。

SQL ステートメント文法	最小	コア	拡張
ALTER TABLE			X
CREATE GROUP			X
CREATE INDEX		X	
CREATE TABLE			X
CREATE TABLE			X
CREATE TRIGGER			X
CREATE VIEW		X	
DELETE（位置付け）	X		
DELETE（検索済み）	X		
DROP GROUP			X
DROP INDEX		X	
DROP PROCEDURE			X

SQL ステートメント文法	最小	コア	拡張
DROP TABLE	X		
DROP TRIGGER			X
DROP VIEW		X	
GRANT		X	
INSERT	X		
JOIN LEFT OUTER（SELECT）			X
REVOKE		X	
SELECT（INTO 付き）	X		
概算数値リテラル		X	
BETWEEN 述語		X	
相関名		X	
日付演算			X
日付リテラル			X
正確な数値リテラル		X	
拡張述部			X
IN 述語		X	
セット関数		X	
時刻リテラル			X
タイムスタンプ リテラル			X
サブクエリ		X	
SET SECURITY			X
UPDATE（位置付け）	X		
UPDATE（検索済み）	X		
UNION			X

SQL ステートメント内のデリミター付き識別子

列名およびテーブル名に非標準文字が含まれる場合、列名とテーブル名はデリミター付き識別子として表記できます。識別子がキーワードである場合は、デリミターを付ける必要があります。

識別子のデリミター文字は二重引用符です。

例

```
SELECT "last-name" FROM "non-standard-tbl"
```

ハイフンは標準文字ではありません。

```
SELECT "password" FROM my_pword_tbl
```

"password" は SET PASSWORD ステートメントのキーワードです。

使用できるデータ型

次の表は、Zen によってサポートされる ODBC のリレーショナル データ型に関する情報を示します。SRDE は、**SQLGetData** または **SQLBindCol** を呼び出すときに別のデータ型変換が指定されない限り、リレーショナル データ型を ODBC のデフォルトのデータ型に変換します。データ型の変換については、Microsoft の ODBC ドキュメントに記載されているデータ型を参照してください。

以下のデータ型に関する詳しい情報は、『*SQL Engine Reference*』の Zen で使用できるデータ型を参照してください。

- Zen メタデータの型コード
- サイズ
- 作成 / 追加パラメーター
- 各データ型固有の注記

リレーショナル型	ODBC 型（コード）[1]
AUTOTIMESTAMP	SQL_TIMESTAMP(93)
BFLOAT4	SQL_REAL(7)
BFLOAT8	SQL_DOUBLE(8)

リレーショナル型	ODBC 型（コード）[1]
BIGIDENTITY	SQL_BIGINT(-5)
BIGINT	SQL_BIGINT(-5)
BINARY	SQL_BINARY(-2)
BIT	SQL_BIT(-7)
CHAR	SQL_CHAR(1)
CURRENCY	SQL_DECIMAL(3)
DATE	SQL_DATE(91)
DATETIME	SQL_TIMESTAMP(93)
DECIMAL	SQL_DECIMAL(3)
DOUBLE	SQL_DOUBLE(8)
IDENTITY	SQL_INTEGER(4)
INTEGER	SQL_INTEGER(4)
LONGVARBINARY	SQL_LONGVARBINARY(-4)
LONGVARCHAR	SQL_LONGVARCHAR(-1)
MONEY	SQL_DECIMAL(3)
NCHAR	SQL_WCHAR(-8)
NLONGVARCHAR	SQL_WLONGVARCHAR(-10)
NUMERIC	SQL_NUMERIC(2)
NUMERICSA	SQL_NUMERIC(2)
NUMERICSLB	SQL_NUMERIC(2)
NUMERICSLS	SQL_NUMERIC(2)
NUMERICSTB	SQL_NUMERIC(2)
NUMERICSTS	SQL_NUMERIC(2)
NVARCHAR	SQL_WVARCHAR(-9)
REAL	SQL_REAL(7)
SMALLIDENTITY	SQL_SMALLINT(5)
SMALLINT	SQL_SMALLINT(5)

リレーショナル型	ODBC 型（コード）[1]
TIME	SQL_TIME(92)
TIMESTAMP	SQL_TIMESTAMP(93)
TIMESTAMP2	SQL_TIMESTAMP(93)
TINYINT	SQL_TINYINT(-6)
UBIGINT	SQL_BIGINT(-5)
UINTEGER	SQL_INTEGER(4)
UNIQUEIDENTIFIER	SQL_GUID(-11)
USMALLINT	SQL_SMALLINT(5)
UTINYINT	SQL_TINYINT(-6)
VARCHAR	SQL_VARCHAR(12)

[1] SQL_FLOAT および SQL_VARBINARY は Zen ではサポートされません。

無限の表現

Zen で無限を表すには、次の表のように、4 バイト（C 言語の float 型）または 8 バイト（C 言語の double 型）の形式で、16 進数または文字として表現できます。

値	Float 16 進数	Float 文字	Double 16 進数	Double 文字
正の最大数			0x7FEFFFFFFFFFFFFF	
負の最大数			0xFFEFFFFFFFFFFFFF	
正の無限数	0x7F800000	1E999	0x7FF0000000000000	1E999
負の無限数	0xFF800000	-1E999	0xFFF0000000000000	-1E999

トランザクション

START TRANSACTION ステートメントはストアド プロシージャ外ではサポートされません。これは、ODBC 標準が、すべてのステートメントはデフォルトでトランザクション内にあることを条件としているためです。ODBC 標準にはトランザクションを

開始する API がありません。『*SQL Engine Reference*』の START TRANSACTION を参照してください。

ODBC は、各 SQL ステートメントを SQL 独自のトランザクション内に置くか、それともアプリケーションが各トランザクションの終了時を指定するかどうかを決定するアプリケーションを提供します。ODBC は、トランザクション内にないどのステートメントより前に、自動的にトランザクションを開きます。したがって、指定された接続の最初のステートメント、あるいは COMMIT または ROLLBACK 後の最初のステートメントにより、ODBC は自動的に新しいトランザクションを開始します。

ODBC 標準内では、SQLSetConnectOption を使って、各ステートメントを独自のトランザクション内に置くか、またはアプリケーションが複数のステートメントを 1 つのトランザクションにまとめるかどうかを指定します。

SQLSetConnectOption にオプション SQL_AUTOCOMMIT、値 SQL_AUTOCOMMIT_ON（これがデフォルトです）を指定して呼び出すと、各ステートメントは独自のトランザクション内に置かれます。このように使用すると、トランザクションはステートメントの実行を開始するときに開始され、ステートメントの実行の完了時点でエラーが発生していない場合は自動的にコミットされ、エラーが発生した場合はロール バックされるようになります。

SQLSetConnectOption にオプション SQL_AUTOCOMMIT、値 SQL_AUTOCOMMIT_OFF を指定して呼び出すと、アプリケーションがステートメントを 1 つのトランザクションにまとめることができます。このように使用すると、トランザクションは実行される最初のステートメントを開始するときに開始されます。その後、トランザクションをいつ、どのように終了するかは、アプリケーションが SQLTransact を呼び出すか、'COMMIT WORK' または 'ROLLBACK WORK' ステートメントを実行するかによって決まります。アプリケーションが 1 つのトランザクションを終了すると、次のステートメントの実行時に別のトランザクションが自動的に開始されます。

DSN のセットアップおよび接続文字列

以下のトピックでは、Zen でのドメイン ソース名および接続文字列の管理について説明します。

- ODBC データベース アクセス
- Zen DSN セットアップ
- Zen エンジン DSN セットアップ
- DSN セットアップを介したデータベースの作成
- ODBC 接続文字列

ODBC データベース アクセス

標準 ODBC では、ODBC を使用するアプリケーションはオペレーティング システムで定義されているデータ ソース名（DNS）を介してデータベースにアクセスします。Zen では、DSN 接続文字列または DSN レス接続文字列を利用することができます。Zen はデータベース エンジンとの通信用の ODBC ドライバーを提供します。これらのドライバーは、DSN と関連付けられているか、接続文字列で指定されています。

以下のセクションでは、Zen ODBC ドライバーの一覧を示し、DSN データベース アクセスおよび DSN レスの接続文字列アクセスについて簡単に説明します。

Zen ODBC ドライバー名

データベース エンジンとの通信には、Zen ODBC ドライバーを利用します。DSN を作成するときに適切なドライバーと関連付けが行われます。接続文字列を利用している場

合は、適切なドライバーを指定する必要があります。次の表に、Zen ODBC ドライバーの一覧を示します。

ドライバー名	ビット数	備考
Pervasive ODBC Unicode Interface	32 ビットと 64 ビット	• Windows オペレーティング システムでのみ利用できます。[1] • ローカルまたはリモートの名前付きデータベースへ接続します。 • 32 ビット ODBC アドミニストレーターでは、ワイド文字データを扱う 32ビット アプリケーション向けの 32 ビット DSN を作成します。 • 64 ビット ODBC アドミニストレーターでは、ワイド文字データを扱う 64ビット アプリケーション向けの 64 ビット DSN を作成します。
Pervasive ODBC Interface	64 ビット	• 64 ビット DSN を作成します。 • ローカルまたはリモートの名前付きデータベースへ接続します。 • 64 ビット アプリケーション向け。
Pervasive ODBC Client Interface	32 ビット	• 32 ビット クライアント DSN を作成します。 • ローカルまたはリモートの名前付きデータベース、あるいはエンジン DSN へ接続します。 • インターフェイス GUI では、名前付きデータベースとエンジン DSN の両方を一覧に表示します。 • 32 ビット アプリケーション向け。
Pervasive ODBC Engine Interface	32 ビット	• 32 ビット エンジン DSN を作成します。[2] • ローカルの名前付きデータベースへ接続します。 • 32 ビット アプリケーション向け。 • 非推奨

ドライバー名	ビット数	備考

[1] Linux では通常、システム エンコードは UTF-8 です。このエンコードを使用すると、SQL テキストにワイド文字データを含めることができます。UTF8 を使用する SQL テキストは既存の Pervasive ODBC Client Interface ドライバーと互換性があるので、Linux で ODBC Unicode ドライバーは必要ありません。

[2] 新規または修正を施す 32 ビット アプリケーションは、ローカルでもリモートでも、エンジン DSN を使用するのではなく、名前付きデータベースに接続するか、クライアント DSN を使用する必要があります。この代わりに、Pervasive ODBC Client Interface を指定することによってアプリケーションが DSN レス接続を使用するという方法もあります。エンジン DSN の使用を避けることで、将来エンジン DSN が Zen でサポートされなくなってもアプリケーションを維持することができます。

DSN 接続

Zen はファイル DSN をサポートしません。ユーザー DSN またはシステム DSN を使用する必要があります。そのコンピューター上のすべてのユーザーが利用可能であることから、一般的にはシステム DSN が使用されます。

ODBC アプリケーションが DSN の使用を想定している場合は、DSN はデータベースを識別する必要があります。

Zen *Unicode* DSN は、ローカルまたはリモートの名前付きデータベースを指します。これは、ワイド文字データを扱う Windows 32 ビットまたは 64 ビットアプリケーション向けです。

また、Zen は 32 ビットおよび 64 ビット用の非 Unicode DSN も提供します。これらも、ローカルまたはリモートの名前付きデータベースを指します。32 ビット DSN はクライアント DSN と呼ばれます。64 ビット オペレーティング システムの DSN は、単に 64 ビット DSN と呼ばれ、「クライアント」の名称は付きません。これは、64 ビット アプリケーション向けです。

ODBC アドミニストレーターを使用した DSN の構成および設定については、Zen DSN セットアップを参照してください。

メモ： Zen は引き続き 32 ビット版の**エンジン** DSN の提供を行います。**エンジン DSN の使用は推奨されていません。**新規または修正を施す 32 ビット アプリケーションは、ローカルでもリモートでも、エンジン DSN を使用するより、名前付きデータベースに接続してください。エンジン DSN の使用を避けることで、将来エンジン DSN が Zen で

サポートされなくなってもアプリケーションを維持することができます。エンジン DSN はローカルの名前付きデータベースのみを指します。クライアント DSN はエンジン DSN を指すこともできます。

DSN を使用しない接続（DSN レス接続）

DSN を利用する代わりに、アプリケーションは Zen ドライバー名を直接指定することによって、DSN レス接続を使用することができます（ODBC 接続文字列を参照してください）。

Btrieve API や、その他 ADO.NET などの SQL アクセス方法を利用してのみ Zen データベースにアクセスするアプリケーションでは、DSN は必要ありません。これらのアクセス方法では、接続に名前付きデータベースを用います。これは、ODBC アプリケーションではオプションです。

Zen Java ユーティリティは DSN を必要としません。たとえば、ZenCC は、名前付きデータベースでは ODBC ではなく JDBC を使用します。

Zen DSN セットアップ

このダイアログは、ODBC アドミニストレーターから利用できます。以下の Zen ODBC インターフェイスのいずれかを使用して、DSN を設定することができます。

- **Pervasive ODBC Unicode Interface**
  32 ビット ODBC アドミニストレーターを用いた場合は、32 ビット DSN を作成します。64 ビット ODBC アドミニストレーターを用いた場合は、64 ビット DSN を作成します。
- **Pervasive ODBC Client Interface**
  32 ビット DSN 用。
- **Pervasive ODBC Interface**
  64 ビット DSN 用。

ODBC アドミニストレーター

Windows 64 ビット オペレーティング システムには、ODBC アドミニストレーター用の実行可能ファイルが 2 種類含まれています。1 つは 32 ビット DSN 用で、もう 1 つは 64

ビット DSN 用です。各 ODBC アドミニストレーターは、自身のビット数に合致するシステム DSN のみを列挙します。つまり、64 ビット ODBC アドミニストレーターは 64 ビット システム DSN を列挙します。逆もまた同様です。Windows コントロール パネルから ODBC アドミニストレーターを起動した場合は、64 ビット バージョンが実行されます。

Zen Control Center（ZenCC）の［ツール］メニューには、32 ビットまたは 64 ビットの ODBC アドミニストレーターを起動するためのオプションが個々に含まれています。ここで留意する点は、ODBC アドミニストレーターが既に開かれている場合、Windows はそれをデフォルトとするということです。つまり、32 ビット ODBC アドミニストレーターが開いているときに 64 ビット用を起動しようとすると、Windows は 32 ビット バージョンを表示します（逆も同様）。言い換えると、ODBC アドミニストレーターは同時に 1 つのバージョンしか実行されないということです。これは Zen の制限ではなく、Windows オペレーティング システムの制限です。

データ ソース名

ODBC クライアント サーバー アーキテクチャは特定のデータ セットをそれぞれの名前で呼び出すので、わかりやすい名前で参照することができます。

接続をセットアップするデータ ソースの名前（データ ソース名または DSN と呼びます）を入力します。この DSN はデータ ソースの識別に役立ちます。

データベース エンジンでの DSN の使用に関する詳細については、ODBC データベース アクセスを参照してください。

説明

必要に応じて、DSN の説明を入力します。説明は、DSN、データベース、またはアプリケーションの識別に役立ちます。

サーバー名 /IP

データベース エンジンが実行されているマシンを指定します。クライアントを接続させるサーバー マシンのマシン名または IP アドレスを入力します。

転送のヒント

使用する転送プロトコルまたは最初に試行する転送プロトコルを指定します。デフォルトは TCP（TCPIP のみ試行）です。

データベース名

［**データベース名**］をクリックし、［**リストの取得**］をクリックしたら、リストの中から接続したいデータベースを選択します。このリストは、サーバー名 /IP に指定されたサーバー上のデータベースを返します。

任意で、［**作成**］をクリックして新しいデータベースを作成できます。

読み取り専用 DSN

このチェック ボックスをオンにすると、DSN 接続文字列に OpenMode=1 設定が含まれるようになります。この設定は次の構成で使用できます。

- Pervasive ODBC Unicode Interface（32 ビットおよび 64 ビット）
- Pervasive ODBC Interface（64 ビット）

データベース設定の詳細

次のデータベース設定の詳細については、DSN セットアップを介したデータベースの作成を参照してください。

- 辞書のロケーション
- データ ファイルのロケーション
- 整合性の設定
- バウンド

エンジン DSN

このオプションは、32 ビット クライアント DSN ダイアログにのみ表示されます。これは、他の Zen ドライバーの DSN ダイアログのいずれにも存在しません。

［**エンジン DSN**］をクリックし、［**リストの取得**］をクリックしたら、リストの中からクライアントを接続させたいエンジン DSN を選択します。このリストは、**サーバー名 / IP** に指定されたサーバー上のエンジン DSN を返します。

任意で、［**作成**］をクリックして新しいエンジン DSN を作成したり、［**変更**］をクリックして既存のエンジン DSN を変更することができます。

Zen エンジン DSN セットアップも参照してください。

メモ： 新規または修正を施す 32 ビット アプリケーションは、ローカルでもリモートでも、エンジン DSN ではなく名前付きデータベースに接続する必要があります。この代わりに、アプリケーションは DSN レス接続を使用することもできます（DSN を使用しない接続（DSN レス接続）を参照してください）。エンジン DSN の使用を避けることで、将来エンジン DSN が Zen でサポートされなくなってもアプリケーションを維持することができます。

詳細な接続属性

以下の接続属性は、32 ビット クライアント DSN、64 ビット クライアント DSN、および Unicode DSN に適用されます。

- 配列フェッチを有効にする
- TCP/IP ポート番号
- エンコード変換

（エンジン DSN に適用される接続属性については、**エンジン DSN 用の詳細な接続属性**を参照してください。）

配列フェッチを有効にする

配列フェッチは、クライアント マシン上の結果セットのためのメモリ キャッシュです。配列フェッチが有効な場合、最新の結果セットのデータはクライアント マシンのローカル メモリにキャッシュされます。それによって、その後のクエリのパフォーマンスが向上します。複数のクエリを実行する場合は、配列フェッチをオンにしておくことをお勧めします。

配列フェッチのキャッシュに使用されるバッファーのデフォルト サイズは 64 KB です。この値には、1 から 64 KB までの任意の数値を設定できます。

TCP/IP ポート番号

この設定を使用して Zen ODBC 通信を行うネットワーク ポート番号を変更することができます。サーバー エンジンのネットワーク レイヤーにも同様な設定があります。両方の設定を同時に行って同一のポート番号に変更しないと、クライアントとサーバーは通信できません。

注意！ サーバーの対応するポート番号を変更しないのであれば、クライアントのポート番号も変更しないでください。サーバーおよびクライアントが同一のポート番号を使用しないと通信することができません。『*Advanced Operations Guide*』の TCP/IP ポートを参照してください。

一般的に、このポート番号を変更する必要があるのは、このポートを既に使用している別のネットワーク サービスがあり、ほかのアプリケーションより Zen アプリケーションのポート番号を変更する方が容易な場合です。

ポートに関する詳細については、『*Getting Started with Zen*』のデフォルトの通信ポートの変更を参照してください。

エンコード変換

エンコード変換とは、文字データのエンコードを、データベース中に存在するエンコードから、クライアントに存在するエンコードへ変換する操作のことを言います（その逆も同様）。これにより、データベースとクライアントが異なるエンコードを使用している場合でも、クライアントは特定の条件下で、データベースのテキストの読み取りと書き込みが可能になります。両方のエンコードが同じである場合、変換の必要がないことは明らかです。変換の有効性は、クライアント上とサーバー上の文字セットがどれくらい一致しているかによります。つまり、共通している文字が多いほど、変換は有効になります。変換できない文字は疑問符に置き換えられます。たとえば、データベースが OEM コード ページ 850、クライアントが ANSI コード ページ 1252 を利用していた場合、文字は変換されますが、一部の図形記号は変換されません。

データベースの接続文字列や DSN は、自動的に変換を取り決めるようにするか、異なるコード ページのエンコード間で OEM/ANSI 変換を実施するか、またはあらゆる変換を無効にするかを設定することができます。Unicode ドライバーを使用している場合は、自動変換がデフォルトです。それ以外の Zen ドライバーの場合は、「変換なし」がデフォルトです。自動変換は、DSN セットアップ画面で指定するか、または ODBC 接続文字列で PvTranslate 属性を使用して指定できます。

次の表は、クライアントとドライバー エンコードのさまざまな組み合わせについて、文字エンコード変換の操作をまとめています。1 列目は、アプリケーションが ANSI ま

たは Unicode のいずれであるかを示します。2 列目は、Zen ドライバーがクライアントドライバー（クライアント 32 ビット /64 ビット ドライバー）または Unicode ドライバーのいずれであるかを示します。（クライアント ドライバーと Unicode ドライバーは、ODBC データベース アクセスを参照してください。）3 列目は、Microsoft ODBC Driver Manager が Zen ODBC ドライバーにアプリケーションを接続して、テキスト変換を実行することができるかどうかを示します。残り 3 列は、特定のエンコード設定（列 4）における、SQL テキストまたは CHAR ユーザー データ（それぞれ列 5 と列 6）に対する Zen ドライバーのテキスト処理について示します。データベースからデータを取得するときは、変換が逆になります。設定オプションの説明をまとめた表は次のとおりです。

アプリケーション	Zen ドライバー	Microsoft ドライバー マネージャー テキスト処理	DSN または接続文字列の変換設定	Zen ドライバー SQL テキスト処理	Zen ドライバー CHAR データ処理
ANSI	クライアント	変換なし	なし	変換なし	変換なし
ANSI	クライアント	変換なし	OEM/ANSI	クライアント エンコードを OEM へ	クライアント エンコードを OEM へ
ANSI	クライアント	変換なし	自動	クライアント エンコードをデータベース エンコードへ	クライアント エンコードをデータベース エンコードへ
ANSI	Unicode	クライアント エンコードを SQL テキストの UCS-2 へ	自動	UCS-2 を UTF-8 へ	クライアント エンコードをデータベース エンコードへ
Unicode	クライアント	UCS-2 をクライアント エンコードへ	なし	変換なし	変換なし
Unicode	クライアント	UCS-2 をクライアント エンコードへ	OEM/ANSI	クライアント エンコードを OEM へ	クライアント エンコードを OEM へ
Unicode	クライアント	UCS-2 をクライアント エンコードへ	自動	クライアント エンコードをデータベース エンコードへ	クライアント エンコードをデータベース エンコードへ

アプリケーション	Zen ドライバー	Microsoft ドライバー マネージャー テキスト処理	DSN または 接続文字列の 変換設定	Zen ドライバー SQL テキスト処理	Zen ドライバー CHAR データ処理
Unicode	Unicode	変換なし	自動	UCS-2 を UTF-8 へ	UCS-2 をデータベース エンコードへ

メモ： Zen クライアント ドライバーを使用している場合、Unicode の SQL テキストは常に、Microsoft Driver Manager によってクライアント エンコードに変換されます。これにより、SQL クエリ テキスト内の NCHAR リテラルはクライアントの文字セットに限定されます。SQL クエリ テキスト内の NCHAR リテラルを保持するには、Zen Unicode ドライバーを使用します。

DSN エンコード変換オプション

エンコード変換オプションは、Zen データベース エンジンと ODBC を使用する Zen クライアント アプリケーション間で文字データをどのように変換するかを指定します。このオプションはクライアントが 32 ビットまたは 64 ビット DSN を設定する場合にのみ使用可能です。Unicode DSN はデフォルトで " 自動 " に設定されています。

自動

この設定は、エンジン マシン上のデータベースのエンコードがクライアント マシン上の OS エンコードと異なる場合は文字データ エンコードを自動的に変換するよう、Zen ODBC クライアントに指示します。Unicode ドライバーは、自動的に変換するようにデフォルトで設定されています。

文字データの変換は、要求に応じてクライアントで行われます。エンジン マシン上のデータベース エンコードがクライアント マシン上の OS エンコードと同じ場合は、文字データ変換は不要です。

" 自動 " を指定するには、クライアントとサーバーがバージョン 10.1 以上である必要があります。

『*Advanced Operations Guide*』の**データベース コード ページとクライアント エンコード**も参照してください。

なし

この設定では、クライアントおよびサーバー間で文字データの変換は行われません（クライアントとサーバーが同じオペレーティング システムのエンコードを使用していることが前提です)。

Zen v10 SP1 より前のバージョンでは、"OEM/ANSI 変換 " は単一選択で、**選択解除**または**選択**の 2 つの状態があるだけでした。現在、**選択解除**状態は " なし " と表示されるようになり、Unicode ドライバー以外の Zen ODBC ドライバーでは、これがデフォルトとなっています。

OEM/ANSI 変換

この設定により、アプリケーションは Zen エンジンのすべての OEM 文字セットの文字データを格納または取得することができ、アプリケーションの ANSI Windows 文字セットを使用してデータを操作および表示することができます。

Zen ODBC ドライバー トランスレーター DLL が、2 つの文字セット間で必要な変換をすべて行います。この機能は、DSN ごとにオン / オフを切り替えることができます。データベースとやり取りする文字データは ODBC ドライバーによって、OEM 文字セットと ANSI 文字セット間で正しく変換されます。

アプリケーションで SQLDriverConnect を使用してデータ ソースに接続する場合は、接続文字列オプション TRANSLATIONDLL=*path_and_DLL_name* を使って、トランスレーター DLL を指定することもできます。Zen 用のトランスレーター DLL 名は W32BTXLT.DLL です。

メモ： OEM から ANSI への変換オプションは、クライアント DSN と 64 ビット DSN のみで使用できます。（また、この変換オプションをローカルのエンジン DSN で利用することもできます。リモート クライアント接続をエンジン DSN に設定しているときには使用できません。エンジン DSN の使用は推奨されておらず、新しいアプリケーションには利用できないということを頭に入れておいてください。）

データベース コード ページとエンコード変換の相互の影響

次の表で、データベース コード ページと DSN エンコード間の相互の影響を説明します。コード ページの説明については、DSN セットアップを介したデータベースの作成を参照してください。

データベースのコード ページ	接続エンコード変換	Zen ODBC ドライバー
サーバーのデフォルト	なし (PSQL v10 SP1 より前のバージョンのデフォルトの動作と同じです。)	データやメタデータの変換は行われません。サーバー上の OS エンコードとクライアント上の OS エンコードが一致していることが前提です。 データ変換の互換性のためには、クライアント マシンで使用するエンコードが、データベースにおけるデータとメタデータのエンコードと一致する必要があります。
特定のコード ページ	なし (PSQL v10 SP1 より前のバージョンのデフォルトの動作と同じです。)	データやメタデータの変換は行われません。サーバー上の OS エンコードとクライアント上の OS エンコードが一致していることが前提です。 データ変換の互換性のためには、クライアント マシンで使用するエンコードが、データベースにおけるデータとメタデータのエンコードと一致する必要があります。
サーバーのデフォルト または 特定のコード ページ	OEM/ANSI	データベース コード ページを無視し、データおよびメタデータを、データベースの OEM エンコードからクライアント アプリケーションの ANSI Windows エンコードに変換します。
サーバーのデフォルト	自動	データおよびメタデータを、サーバーのデフォルトの OS エンコードから、クライアントの OS エンコードに変換します。
特定のコード ページ	自動	データおよびメタデータを、データベース コード ページから、クライアントの OS エンコードに変換します。

Zen エンジン DSN セットアップ

エンジン DSN は 32 ビットのみです。

Windows 64 ビット オペレーティング システムには、ODBC アドミニストレーター用の実行可能ファイルが 2 種類含まれています。1 つは 32 ビット DSN 用で、もう 1 つは 64 ビット DSN 用です。各 ODBC アドミニストレーターは、自身のビット数に合致するシステム DSN のみを列挙します。つまり、64 ビット ODBC アドミニストレーターは 64 ビット システム DSN を列挙します。逆もまた同様です。Windows コントロール パネルから ODBC アドミニストレーターを起動した場合は、64 ビット バージョンが実行されます。**エンジン DSN は 32 ビットでのみ使用できるため、64 ビット バージョンでは表示されません。**

Zen Control Center（ZenCC）の［ツール］メニューには、32 ビットまたは 64 ビットの ODBC アドミニストレーターを起動するためのオプションが個々に含まれています。ここで留意する点は、ODBC アドミニストレーターが既に開かれている場合、Windows はそれをデフォルトとするということです。つまり、32 ビット ODBC アドミニストレーターが開いているときに 64 ビット用を起動しようとすると、Windows は 32 ビット バージョンを表示します（逆も同様）。言い換えると、ODBC アドミニストレーターは同時に 1 つのバージョンしか実行されないということです。これは Windows オペレーティング システムの制限であり、Zen の制限では**ありません**。

メモ： 新規または修正を施す 32 ビット アプリケーションは、ローカルでもリモートでも、エンジン DSN ではなく名前付きデータベースに接続する必要があります。この代わりに、アプリケーションは DSN レス接続を使用することもできます（DSN を使用しない接続（DSN レス接続）を参照してください）。エンジン DSN の使用を避けることで、将来エンジン DSN が Zen でサポートされなくなってもアプリケーションを維持することができます。

データ ソース名

ODBC クライアント サーバー アーキテクチャは特定のデータ セットをそれぞれの名前で呼び出すので、わかりやすい名前で参照することができます。

接続をセットアップするデータ ソースの名前（データ ソース名または DSN と呼びます）を入力します。この DSN はデータ ソースの識別に役立ちます。

データベース エンジンでの DSN の使用に関する詳細については、ODBC データベース アクセスを参照してください。

説明

必要に応じて、DSN の説明を入力します。説明は、DSN、データベース、またはアプリケーションの識別に役立ちます。

データベース名

DSN に関連付けるデータベースを選択します。オプションで［**作成**］をクリックすると、新規データベースを作成できます。

データベース設定の詳細

次のデータベース設定の詳細については、DSN セットアップを介したデータベースの作成を参照してください。

- 辞書のロケーション
- データ ファイルのロケーション
- 整合性の設定
- バウンド

エンジン DSN 用の詳細な接続属性

エンジン DSN 用の接続属性には以下のものがあります。

- DSN オープン モード
- エンコード変換

メモ： エンジン DSN の使用は推奨されていません。新規または修正を施すアプリケーションは、接続モードがローカルでもリモートでも、クライアント DSN を使用してください。

DSN オープン モード

エンジン DSN 用の DSN オープン モード オプションを使用すると、指定した DSN を介してテーブルを開く際に適用する特性を指定できます。これらのオプションは互いに排他的で、1 つしか選択することはできません。

これらのオプションは Btrieve の Open（0）オペレーションで使用できるオープン モードに直接対応しています。DSN のオープン モードを設定することにより、その DSN によって開かれるテーブル（Btrieve ファイルに対応）のデフォルトの動作を設定することになります。

オープン モード	生成される ODBC 接続文字列	SQLSetConnectOption 呼び出し
ノーマル	OPENMODE=0	SQLSetConnectOption(pSubDbc, SQL_ACCESS_MODE, SQL_MODE_READ_WRITE);
アクセラレイティド	OPENMODE=-1	SQLSetConnectOption は無視されます。
リードオンリー	OPENMODE=1	SQLSetConnectOption(pSubDbc, SQL_ACCESS_MODE, SQL_MODE_READ_ONLY);
エクスクルーシブ	OPENMODE=-4	SQLSetConnectOption は無視されます。

ノーマル

デフォルトはノーマル モードです。ノーマル モードでテーブルを開くと、データベースに定義されている権限に従って読み込み / 書き込みアクセスが許可されます。

このモードが選択された場合、ODBC 接続文字列には OPENMODE=0 が含まれ、データベースに接続した際に次の ODBC 関数呼び出しが実行されます。

```
SQLSetConnectOption(pSubDbc, SQL_ACCESS_MODE, SQL_MODE_READ_WRITE);
```

アクセラレイティド

アクセラレイティド モードでテーブルを開くと、データベース エンジンが現在のユーザーに対し関数のロギングを行えなくなり、追加 / 更新のパフォーマンスが向上します。アクセラレイティド モードのロギングを変更しても、同じテーブルにアクセスしている他のユーザーには影響がありません。

注意！ データベース エンジンは、クライアントがアクセラレイティド モードを使用している間は、クライアントのトランザクション アトミシティ、トランザクション一貫性保持、およびアーカイブ ログの安全性を保証できません。この制約があるのは、ログからの復元が必要な場合に、完全な復元を行うために十分な情報がログに含まれていない可能性があるからです。なぜなら、ログは、1 つのデータ ファイル上で行った操

作の部分的な記録でしかないからです。

たとえば、アクセラレイティド モードを使用して挿入を実行するクライアントと、ノーマル モードを使用して更新を実行するクライアントが同じファイルにアクセスしているときにシステム障害が発生した場合、トランザクション ログには、データ ファイルにまだ存在しないレコードに対する更新が含まれている可能性があります。これは、メモリ内のアクセラレイティドの挿入操作は一度もディスクにフラッシュされていませんが、トランザクショナルな更新操作はトランザクション ログに書き込まれているためです。

この操作の組み合わせを含むアーカイブ ログをロール フォワードしようとすると、失敗します。

このモードが選択された場合、ODBC 接続文字列には OPENMODE=-1 が含まれ、ODBC ドライバーは SQLSetConnectOption 呼び出しを無視します。このモードを指定するのに SQLSetConnectOption は使用できません。

リードオンリー

リード オンリー モードでテーブルを開いた場合、データベース構造やデータベース内のデータを変更する操作は許可されません。

このモードが選択された場合、ODBC 接続文字列には OPENMODE=1 が含まれ、データベースに接続した際に次の ODBC 関数呼び出しが実行されます。

```
SQLSetConnectOption(pSubDbc, SQL_ACCESS_MODE, SQL_MODE_READ_WRITE);
```

エクスクルーシブ

テーブルをエクスクルーシブ モードで開くと、そのテーブルに対するほかの接続は許可されません。そのテーブルに現在ほかのユーザーがアクセスしている場合、エクスクルーシブ モードで開くことはできません。後でもう一度試す必要があります。

このモードが選択された場合、ODBC 接続文字列には OPENMODE=-4 が含まれ、ODBC ドライバーは SQLSetConnectOption 呼び出しを無視します。このモードを指定するのに SQLSetConnectOption は使用できません。

エンコード変換

エンコード変換オプションはクライアント DSN や 64 ビット DSN 用のオプションと同じです。エンコード変換を参照してください。

DSN セットアップを介したデータベースの作成

次の表は、データベースの作成ダイアログ上にあるコントロールについての説明です。

要素	説明
データベース名	データベース一覧に表示されるデータベースの名前。たとえば、Zen Control Center で表示されるデータベース名です。 **メモ：既存**のデータベース名と同じ名前にすることはできません。
整合性の設定	データベースに整合性制約（セキュリティ、RI、トリガー）を設定するかどうかを指定します。これらの制約は、データ ファイルへの ODBC/SQL アクセスだけでなく、Btrieve アクセスにも適用されます。 追加情報については、『*Advanced Operations Guide*』の Btrieve およびリレーショナル制約間の相互作用を参照してください。
バウンド	データベースが、バインドされているかどうかを示します。データベースをバインドすると、DDF またはデータ ファイルが別のデータベースによって使用されることを防ぎ、データ ファイルが同一データベース内で複数の別のテーブル定義を持つことを防ぎます。 バウンド データベースの詳細については、『*Advanced Operations Guide*』の バウンド データベースと整合性の設定を参照してください。
長いメタデータ（V2 メタデータ）	データベースで使用するメタデータに対し、バージョン 1（V1）またはバージョン 2（V2）のどちらかを指定します。 データベース エンジンでは、メタデータでバージョン 1（V1）とバージョン 2（V2）という 2 つのバージョンをサポートします。メタデータのバージョンはデータベースのプロパティです。このプロパティはそのデータベース内の全テーブルに対して適用され、dbnames.cfg ファイルに記録されます。データベースでは、メタデータ バージョン 1 を使用するテーブルとメタデータ バージョン 2 を使用するテーブルを一緒に使用することはできません。2 つのバージョンのメタデータはそれぞれ情報をやり取りすることができません。 追加情報については、『*SQL Engine Reference*』の Zen メタデータを参照してください。

要素	説明
コード ページ	データベースのデータおよびメタデータに適用されるコード ページを指定します。このプロパティは DBNAMES.CFG に格納されます。 デフォルトのコード ページは " サーバーのデフォルト " で、データベース エンジン実行中のサーバーのオペレーティング システム コード ページを意味します。 データベース コード ページとクライアント エンコードは別個のものですが、相互に関連しています。『*Advanced Operations Guide*』の**データベース コード ページとクライアント エンコード**を参照してください。
Btrieve セキュリティ ポリシー	トランザクショナル インターフェイスで使用するセキュリティ モデルを指定します。『*Advanced Operations Guide*』の MicroKernel エンジンのセキュリティ モデルを参照してください。
辞書のロケーション	この場所は、辞書ファイル（DDF）が存在する物理的な保管場所を指定します。この場所は、接続しているサーバーと同じサーバーで、データベース エンジンが実行されているサーバーにある必要があります。場所の形式は、サーバー マシンで直接作業しているような形式にする必要があります。 *drive:¥path* という形式で入力します。*drive* はサーバーのドライブ名です。
データ ファイルのロケーション	この場所は、データ ファイルが存在する物理的な保管場所を指定します。［**追加**］ボタンをクリックすると、データ ファイルの場所をリストに追加することができます。［**削除**］ボタンをクリックすると、データ ファイルの場所をリストから削除することができます。データ ファイルの場所は、データベース エンジンが起動している同じサーバー上でなければなりません。 ［辞書のロケーション］についても同じ方法で場所を指定してください。

ODBC 接続文字列

このセクションでは、Zen でサポートされる ODBC 接続文字列について説明します。この情報は、接続文字列を指定することができるデータベース アクセス ツールを使用する上級ユーザー、および Zen にアクセスする ODBC アプリケーションの開発者向けです。

ODBC ドライバー パラメーター

Zen データベース エンジンに接続するには、Zen ODBC ドライバーを使用するように指定する必要があります。使用可能なドライバーの詳細については、Zen ODBC ドライバー名を参照してください。

ODBC の **Driver** パラメーターを使用して、適切なドライバーを指定します。たとえば、次のように指定します。

```
Driver={Pervasive ODBC Unicode Interface}
Driver={Pervasive ODBC Interface}
Driver={Pervasive ODBC Client Interface}
Driver={Pervasive ODBC Engine Interface}
```

Driver パラメーター

Driver パラメーターで指定された特定のドライバーには、サーバー、ポート、データベースなどの名前を付けるための追加の属性パラメーターがあります。これらの共通パラメーターに加えて、各ドライバー固有のパラメーターもあります。次の表は、さまざまなドライバーで利用できるドライバー パラメーターを示しています。属性は、ODBC 関数の SQLDriverConnect または SQLConnect を介して含めることができます。

Unicode の接続文字列パラメーター	説明
ServerName=*server*[.*port*]	接続するコンピューターのマシン名または IP アドレスを指定します。必須。*Port* は下位互換性のために用意されています。デフォルトのポートを使用しない場合は、使用するポート番号を指定することができます。IPv6 アドレスを使用したり、ODBC 接続にポート番号を追加したりする場合は、IPv6-literal.net 名または UNC で正しく動作する名前を使用します。『*Getting Started with Zen*』のドライブ ベースの形式を参照してください。
TransportHint=TCP	使用する転送プロトコルまたは最初に試行する転送プロトコルを指定します。デフォルトは TCP（TCPIP のみ試行）です。省略可能。
DBQ=[@]*db_name*	接続する内部データベース名を指定します。DSN ではありません。必須。 @ 文字は省略可能です。この文字はなんの意味も持たず、以前のバージョンとの互換性のためだけにサポートされています。

Unicode の接続文字列パラメーター	説明
TCPPort=*port*	サーバーを探す TCP/IP ポートを指定します。省略可能。『*Getting Started with Zen*』の デフォルトの通信ポートの変更も参照してください。
ArrayFetchOn=1 | 0	結果セットをクライアント上にキャッシュするかどうかを指定します。デフォルトは 1（キャッシュする）です。省略可能。
ArrayBufferSize=*size*	クライアント キャッシュのサイズを KB 単位で指定します。デフォルトは 8 KB です。省略可能。
PvTranslate=auto	クライアントがデータベース エンジンに接続するときのデータ エンコードの処理方法を指定します（詳細については、エンコード変換を参照してください）。 Unicode ドライバーの PvTranslate はデフォルトで " 自動 " に設定されています。これにより、明示的に PvTranslate を " 自動 " に設定しなくても、ワイド文字データを含んでいる NCHAR 列や NCHAR リテラルを使用できます。 属性を " 自動 " に設定した場合、クライアントとサーバーは互換性のあるエンコードを自動的に確立します。データ変換は、必要に応じクライアントで行われます。
UID=*user_name*	データベースのセキュリティが有効になっている場合、ユーザー名を指定します。セキュリティの設定により、省略可能。『*Advanced Operations Guide*』の Zen セキュリティを参照してください。
PWD=*password*	データベースのセキュリティが有効になっている場合、パスワードを指定します。セキュリティの設定により、省略可能。『*Advanced Operations Guide*』の Zen セキュリティを参照してください。

- **例 A**

  ServerMain というリモート サーバー上にある、ワイド文字データを含んでいる SOMEDATA というデータベースに対し、TCP/IP ポート 1590 を使用して接続します。

```
Driver={Pervasive ODBC Unicode Interface}; ServerName=ServerMain.1590;DBQ=SOMEDATA;
TransportHint=TCP;
```

- **例 B**

  データベース セキュリティが有効になっているローカル サーバー上の、ワイド文字データを含んでいる EuropeRegion4 という名前のデータベースに接続します。

```
Driver={Pervasive ODBC Unicode Interface}; DBQ=EuropeRegion4;UID=tonyawu7;PWD=HR19lb8w;
```

64 ビット接続文字列パラメーター	説明
ServerName=*server*[.*port*]	接続するコンピューターのマシン名または IP アドレスを指定します。必須。*Port* は下位互換性のために用意されています。デフォルトのポートを使用しない場合は、使用するポート番号を指定することができます。IPv6 アドレスを使用したり、ODBC 接続にポート番号を追加したりする場合は、IPv6-literal.net 名または UNC で正しく動作する名前を使用します。『*Getting Started with Zen*』の**ドライブ ベースの形式**を参照してください。
TransportHint=TCP	使用する転送プロトコルまたは最初に試行する転送プロトコルを指定します。デフォルトは TCP（TCPIP のみ試行）です。省略可能。
DBQ=[@]*db_name*	接続する内部データベース名を指定します。DSN ではありません。必須。 @ 文字は省略可能です。この文字はなんの意味も持たず、以前のバージョンとの互換性のためだけにサポートされています。
TCPPort=*port*	サーバーを探す TCP/IP ポートを指定します。省略可能。『*Getting Started with Zen*』の **デフォルトの通信ポートの変更**も参照してください。
ArrayFetchOn=1 | 0	結果セットをクライアント上にキャッシュするかどうかを指定します。デフォルトは 1（キャッシュする）です。省略可能。
ArrayBufferSize=*size*	クライアント キャッシュのサイズを KB 単位で指定します。デフォルトは 8 KB です。省略可能。

64 ビット接続文字列パラメーター	説明
PvTranslate=auto	クライアントがデータベース エンジンに接続するときのデータ エンコードの処理方法を指定します。この属性は省略することも、値を " 自動 " に設定して自動変換を示すこともできます（詳細については、**エンコード変換**を参照してください）。 属性を " 自動 " に設定した場合、クライアントとサーバーは互換性のあるエンコードを自動的に確立します。データ変換は、必要に応じクライアントで行われます。DSN では "OEM/ANSI" より " 自動 " が優先されることに注意してください。 この属性が指定されていない場合、ODBC は文字データを変換しません。これはデフォルトの動作です。旧来どおり "OEM/ANSI" 設定が適用されます。OEM/ANSI 変換を参照してください。
UID=*user_name*	データベースのセキュリティが有効になっている場合、ユーザー名を指定します。セキュリティの設定により、省略可能。『*Advanced Operations Guide*』の Zen セキュリティを参照してください。
PWD=*password*	データベースのセキュリティが有効になっている場合、パスワードを指定します。セキュリティの設定により、省略可能。『*Advanced Operations Guide*』の Zen セキュリティを参照してください。

例

64 ビット アプリケーションを使用して、acctdomestic という名前のローカル データベースに接続します。

```
Driver={Pervasive ODBC Interface};DBQ=acctdomestic;
```

32 ビット接続文字列パラメーター	説明
ServerName=*server*[.*port*]	接続するコンピューターのマシン名または IP アドレスを指定します。必須。*Port* は下位互換性のために用意されています。デフォルトのポートを使用しない場合は、使用するポート番号を指定することができます。IPv6 アドレスを使用したり、ODBC 接続にポート番号を追加したりする場合は、IPv6-literal.net 名または UNC で正しく動作する名前を使用します。『*Getting Started with Zen*』のドライブ ベースの形式を参照してください。
ServerDSN=*dsn_name*	接続するエンジン DSN を指定します。DBQ が指定されていなければ必須。
TransportHint=TCP	使用する転送プロトコルまたは最初に試行する転送プロトコルを指定します。デフォルトは TCP（TCPIP のみ試行）です。省略可能。
DBQ=[@]*db_name*	接続する内部データベース名を指定します。DSN ではありません。必須。 @ 文字は省略可能です。この文字はなんの意味も持たず、以前のバージョンとの互換性のためだけにサポートされています。
TCPPort=*port*	サーバーを探す TCP/IP ポートを指定します。省略可能。『*Getting Started with Zen*』の デフォルトの通信ポートの変更も参照してください。
ArrayFetchOn=1 | 0	結果セットをクライアント上にキャッシュするかどうかを指定します。デフォルトは 1（キャッシュする）です。省略可能。
ArrayBufferSize=*size*	クライアント キャッシュのサイズを KB 単位で指定します。デフォルトは 8 KB です。省略可能。

32 ビット接続文字列パラメーター	説明
PvTranslate=auto	クライアントがデータベース エンジンに接続するときのデータ エンコードの処理方法を指定します。この属性は省略することも、値を " 自動 " に設定して自動変換を示すこともできます（詳細については、エンコード変換を参照してください）。 属性を " 自動 " に設定した場合、クライアントとサーバーは互換性のあるエンコードを自動的に確立します。データ変換は、必要に応じクライアントで行われます。DSN では "OEM/ANSI" より " 自動 " が優先されることに注意してください。 この属性が指定されていない場合、ODBC は文字データを変換しません。これはデフォルトの動作です。旧来どおり "OEM/ANSI" 設定が適用されます。OEM/ANSI 変換を参照してください。
UID=*user_name*	データベースのセキュリティが有効になっている場合、ユーザー名を指定します。セキュリティの設定により、省略可能。『*Advanced Operations Guide*』の Zen セキュリティを参照してください。
PWD=*password*	データベースのセキュリティが有効になっている場合、パスワードを指定します。セキュリティの設定により、省略可能。『*Advanced Operations Guide*』の Zen セキュリティを参照してください。

- **例 A**

  TCP/IP ポート 1585 を使用して、AncientLore というリモート サーバー上にある Atlantis というデータベースに接続します。

```
Driver={Pervasive ODBC Client Interface};
ServerName=AncientLore.1585;DBQ=Atlantis;
```

- **例 B**

  データベース セキュリティが有効になっている SalesSvr というリモート サーバー上にある、DomSales というデータベースに接続します。

```
Driver={Pervasive ODBC Client Interface};
ServerName=SalesSvr;DBQ=DomSales;UID=alexjame;PWD=k7Jb9xRR;
```

- **例 C**

  MyServer という名前のリモート サーバー上にある、mydata というエンジン DSN に接続し、自動エンコード サポートを確立します。

```
Driver={Pervasive ODBC Client Interface};
ServerName=MyServer;ServerDSN=mydata;PvTranslate=auto;
```

接続文字列	説明
DBQ=[@]*db_name*	接続する内部データベース名を指定します。DSN ではありません。必須。 @ 文字は省略可能です。この文字はなんの意味も持たず、以前のバージョンとの互換性のためだけにサポートされています。
UID=*user_name*	データベースのセキュリティが有効になっている場合、ユーザー名を指定します。セキュリティの設定により、省略可能。『*Advanced Operations Guide*』の Zen セキュリティを参照してください。
PWD=*password*	データベースのセキュリティが有効になっている場合、パスワードを指定します。セキュリティの設定により、省略可能。『*Advanced Operations Guide*』の Zen セキュリティを参照してください。
OPENMODE=-4 | -1 | 0 | 1	現在の接続で開くファイルのデフォルトのオープン モードを指定します。デフォルトは 0（ノーマル）です。ローカル接続のみで使用でき、リモート クライアント接続では使用できません。省略可能。 ファイル オープン モードの詳細については、DSN オープン モードを参照してください。
TRANSLATIONDLL=*path_and_DLL_name*	OEM/ANSI 変換に使用する DLL のフル パス名を指定します。詳細については、OEM/ANSI 変換を参照してください。

- **例**

  DATA5 という名前のローカル データベースに接続します。

```
Driver={Pervasive ODBC Engine Interface};DBQ=DATA5;
```

メモ： エンジン DSN の使用は推奨されていません。新規または修正を施すアプリケーションは、接続モードがローカルでもリモートでも、クライアント DSN を使用してください。