ODBC Guide

Zen v15

Activate Your Data™

Contents

About This Document v

Who Should Read this Manual . . . v
For More Information . . . v

ODBC Specification Support 1

ODBC Specification Supported . . . 1
Exceptions to ODBC Interface Support . . . 1
ODBC API Support . . . 2
Exceptions to ODBC API Support . . . 4
ODBC Attribute Support . . . 6
Connection Attribute Support . . . 6
Statement Attribute Support . . . 6
ODBC Descriptor Field Support . . . 7
Descriptor Fields and Bitness Values . . . 7
SQLSetStmtOption Options . . . 7

Zen ODBC Reference 27

Data Source Name Connection String Keywords . . . 27
Closing an Open Table . . . 27
SQL Grammar Support . . . 28
Delimited Identifiers in SQL Statements . . . 29
Supported Data Types . . . 30
Representation of Infinity . . . 32
Transactions . . . 32

DSN Setup and Connection Strings 35

ODBC Database Access . . . 35
Zen ODBC Driver Names . . . 35
DSN Connections . . . 36
DSN-less Connections . . . 37
Zen DSN Setup . . . 37
ODBC Administrator . . . 37
Data Source Name . . . 38
Description . . . 38
Server Name/IP . . . 38
Transport Hint . . . 38

Database Name 38
Read-Only DSN 39
Database Configuration Details 39
Engine DSN. 39
Advanced Connection Attributes 39
Zen Engine DSN Setup 44
Data Source Name. 45
Description 45
Database Name 45
Database Configuration Details 45
Engine DSN Advanced Connection Attributes 45
Create Database Through DSN Setup 48
ODBC Connection Strings 49
ODBC Driver Parameter 49

About This Document

This documentation covers Zen support for the ODBC specification.

Who Should Read this Manual

This manual assumes you have a general understanding of ODBC architecture and ODBC driver components, and have access to the Microsoft ODBC Software Development Kit.

This document also assumes you have a working understanding of modern database principles and terminology, the C language, and your development environment (compiler and linker).

We would appreciate your comments and suggestions about this document. Your feedback can determine what we write about the use of our products and how we deliver information to you. Please post your feedback in the community forum on the Actian Zen website.

Note: Unless otherwise noted, all references in this book to the Zen product refer to the current version.

For More Information

For complete information on the ODBC specification, see the Microsoft ODBC documentation.

ODBC Specification Support

The following topics overview Zen support for the Open Database Connectivity Specification (ODBC) specification:

- ODBC Specification Supported
- ODBC API Support
- ODBC Attribute Support
- ODBC Descriptor Field Support

For instructions on setting up ODBC configuration options in Zen, see DSN Setup and Connection Strings.

ODBC Specification Supported

ODBC (Open Database Connectivity) is a standard API originally developed by Microsoft for accessing database management systems (DBMS). The standard has evolved over the years. The Zen relational interface supports the ODBC v3.51 specifications for Core, Level 1, and Level 2 interface support levels (Level 3 is not supported).

Exceptions to ODBC Interface Support

Core Level

SQL_BEST_ROWID

The relational interface uses unique indexes as the optimal set of columns that identifies a row in the table.

When a new row is inserted into a table that includes an IDENTITY column, the relational interface does not return the value assigned to the IDENTITY column. You may determine the value for an IDENTITY column through the use of the @@IDENTITY variable. See @@IDENTITY and @@BIGIDENTITY in *SQL Engine Reference*.

Level 2

The following are not supported:

- SQL_ATTR_LOGIN_TIMEOUT

- SQL_BEST_ROWID (see discussion above)
- SQL_ROWVER

ODBC API Support

The following table lists the ODBC API functions supported by the relational interface and the ODBC support level. For detailed information on the ODBC API, see the Microsoft ODBC documentation.

ODBC Function	ODBC Support Level
SQLAllocHandle	Core
SQLBindCol	Core
SQLBindParameter	Core
SQLBrowseConnect	Level 1
SQLBulkOperations	Level 1
SQLCancel	Core
SQLCloseCursor	Core
SQLColAttribute	Core
SQLColumnPrivileges	Level 2
SQLColumns	Core
SQLConnect	Core
SQLCopyDesc	Core
SQLDataSources	Core
SQLDescribeCol	Core
SQLDescribeParam	Level 2
SQLDisconnect	Core
SQLDriverConnect	Core
SQLDrivers	Core
SQLEndTran	Core
SQLExecDirect	Core

ODBC Function	ODBC Support Level
SQLExecute	Core
SQLExtendedFetch	Core
SQLFetch	Core
SQLFetchScroll	Core
SQLForeignKeys	Level 2
SQLFreeHandle	Core
SQLFreeStmt	Core
SQLGetConnectAttr	Core
SQLGetCursorName	Core
SQLGetData	Core
SQLGetDescField	Core
SQLGetDescRec	Core
SQLGetDiagField	Core
SQLGetDiagRec	Core
SQLGetEnvAttr	Core
SQLGetFunctions	Core
SQLGetInfo	Core
SQLGetStmtAttr	Core
SQLGetTypeInfo	Core
SQLMoreResults	Level 1
SQLNativeSql	Core
SQLNumParams	Core
SQLNumResultCols	Core
SQLParamData	Core
SQLPrepare	Core
SQLPrimaryKeys	Level 1
SQLProcedureColumns	Level 1

ODBC Function	ODBC Support Level
SQLProcedures	Level 1
SQLPutData	Core
SQLRowCount	Core
SQLSetConnectAttr	Core
SQLSetCursorName	Core
SQLSetDescField	Core
SQLSetDescRec	Core
SQLSetEnvAttr	Core
SQLSetPos	Level 1
SQLSetStmtAttr	Core
SQLStatistics	Core
SQLTablePrivileges	Level 2
SQLTables	Core

Exceptions to ODBC API Support

The following section contains details on the exceptions to ODBC API support.

SQLGetData

If your application calls SQLGetData to return data into an SQL_C_NUMERIC structure, the ODBC standard specifies that the SQL_DESC_SCALE field is set to zero and the SQL_DESC_PRECISION field uses the driver-defined precision.

Zen use the values for scale and driver-defined precision as defined in the metadata. Consider the following example, where scale is set to two:

```
CREATE TABLE testnum (col1 NUMERIC(10,2))

INSERT INTO testnum VALUES (10.34)

SELECT * FROM testnum
```

The SELECT statement returns 10.34, not 10.00.

SQLGetTypeInfo

SQLGetTypeInfo generates a list of native data type names (type_name) specified by the relational interface. For example, SQL_CHAR is mapped to CHARACTER. Use the names which are returned from this function for the data type names for columns in a CREATE TABLE or ALTER TABLE statement or for parameters for procedures or declared variables in procedures and triggers.

See Supported Data Types for a list of supported ODBC data types.

SQLGetInfo

The relational interface returns identical values for SQL_DRIVER_VER and SQL_DBMS_VER. This version value is returned in the following format:

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

This value can be interpreted as four components as explained in the following table:

Part	Value	Description
aa	Major version	The major version of the database engine
bb	Minor version	The minor version of the database engine, which is typically updated in a service pack
cccc	Build number	The build further specifies the release
ddd	Point build	A minor update to the build

The following table summarizes formats of other values typically returned by SQLGetInfo.

Item	Example Value
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

The Zen relational interface uses unique indexes as the optimal set of columns that uniquely identifies a row in the table. When a new row is inserted, the relational interface does not return the values for IDENTITY columns. You may determine the value for an IDENTITY column through the use of the @@IDENTITY variable. See @@IDENTITY and @@BIGIDENTITY in *SQL Engine Reference*.

ODBC Attribute Support

The relational interface provides ODBC v3.51 attribute support, with the exceptions listed here.

Connection Attribute Support

The following table lists the exceptions to ODBC connection attribute support:

fOption	Comments
SQL_ATTR_AUTO_IPD	The default value is SQL_TRUE. The Pervasive ODBC Driver does not allow setting this attribute value to SQL_FALSE.
SQL_ATTR_CONNECTION_TIMEOUT	The default value is 0. No other value is supported.
SQL_ATTR_METADATA_ID	The default value is SQL_FALSE. The Pervasive ODBC Driver does not allow setting this attribute's value to SQL_TRUE

Statement Attribute Support

The following table lists the exceptions to ODBC statement attribute support:

fOption (numerical value)	Comments
SQL_ATTR_ENABLE_AUTO_IPD (15)	*The default value is SQL_TRUE. Pervasive ODBC Driver does not allow setting this attribute value to SQL_FALSE.*
SQL_ATTR_METADATA_ID (10014)	*The default value is SQL_FALSE. Pervasive ODBC Driver does not allow setting this attribute's value to SQL_TRUE.*
SQL_ATTR_PARAM_BIND_TYPE (18)	*Only SQL_PARAM_BIND_BY_COLUMN is supported.*

fOption (numerical value)	Comments
SQL_ATTR_QUERY_TIMEOUT (0)	Supported through SQLSetStmtAttr and SQLSetConnectAttr. Applies only to SQLExecDirect, SQLExecute, SQLFetch, and SQLExtendedFetch. Does not apply to DDL statements.

ODBC Descriptor Field Support

The Zen relational interface provides ODBC v3.51 descriptor field support except for the options listed in the following table.

Option	Comments
SQL_DESC_BIND_TYPE	For application parameter descriptors (APDs), only SQL_BIND_BY_COLUMN is supported.
SQL_DESC_ROWVER	*See Exceptions to ODBC Interface Support.*

Descriptor Fields and Bitness Values

Note that some of the descriptor fields that can be set through the various ODBC SQLSet and SQLGet functions have been changed to accommodate 64-bit values while others are still 32-bit values. If you are using the 64-bit ODBC driver, ensure that you use the appropriate sized variable when setting and retrieving these fields. For more information, refer to the Microsoft ODBC documentation.

A point of clarification is that SQL_ROWSET_SIZE is supported by both SQLGetStmtOption *and* SQLGetStmtAttr. If you are using the 64-bit ODBC driver and you call either SQLGetStmtOption or SQLGetStmtAttr, a 64-bit value is returned in *ValuePtr when that attribute parameter is set to SQL_ROWSET_SIZE.

SQLSetStmtOption Options

The section discusses the Zen support for the following SQLSetStmtOption options:

- SQL_BIND_TYPE
- SQL_CONCURRENCY
- SQL_CURSOR_TYPE
- SQL_RETRIEVE_DATA

- SQL_ROWSET_SIZE
- SQL_USE_BOOKMARKS

The following tables indicate valid set values for each option.

Option	ODBC Cursor Library	Current Zen ODBC Drivers
SQL_BIND_TYPE	SQL_BIND_BY_COLUMN or a length to indicate row-wise binding	SQL_BIND_BY_COLUMN or a length to indicate row-wise binding
SQL_CONCURRENCY	SQL_CONCUR_READ_ONLY or SQL_CONCUR_ VALUES (for SQL_CONCUR_ROWVER the library substitutes SQL_CONCUR_VALUES, returns SQL_SUCCESS_WITH_INFO and sets SQLSTATE to 01S02) (for SQL_CONCUR_LOCK the library returns SQL_ERROR returned with SQLSTATE of S1C00)	SQL_CONCUR_READ_ONLY or SQL_CONCUR_ROWVER or SQL_CONCUR_LOCK (for SQL_CONCUR_VALUES the driver automatically substitutes SQL_CONCUR_ROWVER)
SQL_CURSOR_TYPE	SQL_CURSOR_FORWARD_ONLY or SQL_CURSOR_STATIC (for SQL_CURSOR_KEYSET_DRIVEN and SQL_CURSOR_DYNAMIC the library substitutes SQL_CURSOR_STATIC, returns SQL_SUCCESS_WITH_INFO and sets SQLSTATE to 01S02)	SQL_CURSOR_FORWARD_ONLY or SQL_CURSOR_STATIC or SQL_CURSOR_DYNAMIC (for SQL_CURSOR_KEYSET_DRIVEN the driver substitutes SQL_CURSOR_STATIC, returns SQL_SUCCESS_WITH_INFO and sets SQLSTATE to 01S02)
SQL_RETRIEVE_DATA	SQL_RD_ON (for SQL_RD_OFF the library returns SQL_ERROR returned with SQLSTATE of S1C00)	SQL_RD_ON or SQL_RD_OFF
SQL_ROWSET_SIZE	Any value indicating number of rows in the rowset as long as it does not exceed maximum rowset size.	Any value indicating number of rows in the rowset as long as it does not exceed maximum rowset size.
SQL_USE_BOOKMARKS	SQL_UB_ON or SQL_UB_OFF	SQL_UB_ON or SQL_UB_OFF

Zen ODBC Reference

This reference covers information for the relational interface and ODBC, including connection strings, metadata versions, limits, and SQL grammar under the following topics:

- Data Source Name Connection String Keywords
- Closing an Open Table
- SQL Grammar Support
- Supported Data Types
- Representation of Infinity

Data Source Name Connection String Keywords

A connection string used to connect to a DSN may include any number of driver-defined keywords. Using these keywords, the driver has enough information to connect to the data source. The driver defines which keywords are required to connect to the data source.

See ODBC Connection Strings for a complete discussion of Zen connection strings and the keywords.

Closing an Open Table

Calling SQLFreeStmt with the SQL_CLOSE option changes the SQLSTATE but does not close the open tables used by the hStmt. To close the tables currently used by hStmt, SQLFreeStmt must be called with the SQL_DROP option.

In the following example, the Emp and Dept tables remain open:

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

When SQLPrepare is subsequently called on the hStmt, the tables used in the previous statement are closed. For example, when the following call is made, both the Emp and Dept tables are closed by Zen:

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

The following call would then close the table Customer:

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

SQL Grammar Support

The ODBC v2.5 specification provides three levels of SQL grammar support: Minimum, Core, and Extended. Each higher level provides more fully implemented data definition and data manipulation language support.

The relational interface fully supports the minimum SQL grammar, as well as many core and extended grammar statements. The relational interface support for SQL grammar is summarized in the following table. The grammar statements are documented in *SQL Engine Reference*.

SQL Grammar Statement	Minimum	Core	Extended
ALTER TABLE			X
CREATE GROUP			X
CREATE INDEX		X	
CREATE TABLE			X
CREATE TABLE			X
CREATE TRIGGER			X
CREATE VIEW		X	
DELETE (positioned)	X		
DELETE (searched)	X		
DROP GROUP			X
DROP INDEX		X	
DROP PROCEDURE			X
DROP TABLE	X		
DROP TRIGGER			X
DROP VIEW		X	
GRANT		X	
INSERT	X		
JOIN LEFT OUTER (Select)			X

SQL Grammar Statement	Minimum	Core	Extended
REVOKE		X	
SELECT (with INTO)	X		
Approximate Numeric Literal		X	
Between Predicate		X	
Correlation Name		X	
Date Arithmetic			X
Date Literal			X
Exact Numeric Literal		X	
extended predicates			X
In Predicate		X	
Set Functions		X	
Time Literal			X
Time Stamp Literal			X
Subqueries		X	
SET SECURITY			X
UPDATE (positioned)	X		
UPDATE (searched)	X		
UNION			X

Delimited Identifiers in SQL Statements

Column names and table names can occur as delimited identifiers if they contain non-standard characters. If an identifier is a keyword, it must delimited.

The delimiter character for identifiers is the double-quote.

Examples

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

The hyphen is a non-standard character.

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

"Password" is a keyword in the SET PASSWORD statement.

Supported Data Types

The following table shows information about the relational data types supported by Zen for ODBC. The SRDE converts the relational data types to ODBC default types unless another data type conversion is specified when **SQLGetData** or **SQLBindCol** is called. For a discussion of data type conversions, refer to data types in the Microsoft ODBC documentation.

See Zen Supported Data Types in *SQL Engine Reference* for the following information about the data types:

- Zen metadata type code
- Size
- Create and Add parameters
- Notes specific to each data type

Relational Type	ODBC Type (*code*)[1]
AUTOTIMESTAMP	SQL_TIMESTAMP(93)
BFLOAT4	SQL_REAL(7)
BFLOAT8	SQL_DOUBLE(8)
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)

Relational Type	ODBC Type (*code*)[1]
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)
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 and SQL_VARBINARY are not supported by Zen.

Representation of Infinity

When Zen is required by an application to represent infinity, it can do so in either a 4-byte (C float type) or 8-byte (C double type) form, and in either a hexadecimal or character representation, as shown in the following table:

Table 1 Infinity Representation

Value	Float Hexadecimal	Float Character	Double Hexadecimal	Double Character
Maximum Positive			0x7FEFFFFFFFFFFFFF	
Maximum Negative			0xFFEFFFFFFFFFFFFF	
Infinity Positive	0x7F800000	1E999	0x7FF0000000000000	1E999
Infinity Negative	0xFF800000	-1E999	0xFFF0000000000000	-1E999

Transactions

The START TRANSACTION statement is not supported outside of a stored procedure because the ODBC standard specifies that every statement is by default inside a transaction. The ODBC standard does not have an API to start a transaction. See START TRANSACTION in *SQL Engine Reference*.

ODBC provides for the application to decide if each SQL statement is in its own transaction or if the application will specify when each transaction is completed. ODBC automatically opens a transaction prior to any statement that is not in a transaction. Thus, with the first statement of a given connection, or with the first statement after a COMMIT or ROLLBACK, ODBC automatically starts a new transaction.

Within the ODBC standard, SQLSetConnectOption is used to specify whether each statement is in its own transaction or the application groups statements within a transaction.

Each statement is in its own transaction if SQLSetConnectOption is called specifying the option SQL_AUTOCOMMIT and the value SQL_AUTOCOMMIT_ON (this is the default). This usage means that a transaction is started at the beginning of executing a statement and the transaction is either automatically committed, if no error occurs, or rolled back, if error occurred, upon completion of statement execution.

The application can group statements in a transaction if SQLSetConnectOption is called specifying the option SQL_AUTOCOMMIT and the value SQL_AUTOCOMMIT_OFF value. This usage means that a transaction is started at the beginning of the first statement executed. The application then decides when and how to end the transaction by calling SQLTransact or executing a 'COMMIT WORK' or 'ROLLBACK WORK' statement. When the application ends one transaction, another transaction is automatically started on execution of the next statement.

DSN Setup and Connection Strings

The following topics cover the administration of domain source names and connection strings in Zen:

- ODBC Database Access
- Zen DSN Setup
- Zen Engine DSN Setup
- Create Database Through DSN Setup
- ODBC Connection Strings

ODBC Database Access

The ODBC standard specifies that applications using ODBC connect to databases through data source names (DSNs) defined in the operating system. With Zen, you can use DSN connections or DSN-less connection strings. Zen provides ODBC drivers for communication with the database engine. These drivers are associated with a DSN or specified in the connection string.

The following sections list the Zen ODBC drivers and briefly introduce DSN database access and DSN-less connection string access.

Zen ODBC Driver Names

Communication with the database engine is through a Zen ODBC driver. The setup will associate the appropriate driver during DSN creation. If you are using connection strings, you must specify the appropriate driver. The following table lists the Zen ODBC drivers.

Driver Name	Bitness	Remarks
Pervasive ODBC Unicode Interface	32- and 64-bit	• Available only on Windows operating systems[1] • Connects to a local or remote named database • With the 32-bit ODBC Administrator, creates 32-bit DSNs for use by 32-bit applications that use wide character data • With the 64-bit ODBC Administrator, creates 64-bit DSNs for use by 64-bit applications that use wide character data

Driver Name	Bitness	Remarks
Pervasive ODBC Interface	64-bit	• Setup creates 64-bit DSNs • Connects to a local or remote named database • For use by 64-bit applications
Pervasive ODBC Client Interface	32-bit	• Setup creates 32-bit Client DSNs • Connects to a local or remote named database or an Engine DSN • Interface GUI lists both named databases and Engine DSNs • For use by 32-bit applications
Pervasive ODBC Engine Interface	32-bit	• Setup creates 32-bit Engine DSNs[2] • Connects to a local named database • For use by 32-bit applications • Deprecated

[1]On Linux and macOS, the system encoding is usually UTF-8, which allows SQL text to contain wide character data. SQL text using UTF-8 is compatible with the existing Pervasive ODBC Client Interface driver, so an ODBC Unicode driver on Linux or macOS is not required.

[2]New or revised 32-bit applications, local or remote, should connect to a named database or use a Client DSN instead of using Engine DSNs. Alternately, applications could use DSN-less connections by specifying "Pervasive ODBC Client Interface." Avoiding the use of Engine DSNs positions your application for the future when Engine DSNs will no longer be supported in Zen.

DSN Connections

Zen does not support File DSNs. You must use User or System DSNs. System DSNs are generally preferred, because they are available to all users on a given computer.

If your ODBC application expects to use DSNs, then the DSN must identify the database.

A Zen *Unicode* DSN points to a local or remote named database and is for use with Windows 32-bit or 64-bit applications that use wide character data.

Zen also provides both 32- and 64-bit non-Unicode DSNs. These also point to a local or remote named database. The 32-bit DSN is referred to as a Client DSN. A DSN on a 64-bit operating system is simply referred to as a 64-bit DSN, without the Client designation and is for use by a 64-bit application.

For setting up and configuring a DSN with the ODBC Administrator, see Zen DSN Setup.

Note: Zen also continues to provide an additional 32-bit *Engine* DSN. **Engine DSNs are deprecated.** New or revised 32-bit applications, local or remote, should connect to a named database rather than using Engine DSNs. Avoiding the use of Engine DSNs positions your application for the future when Engine DSNs will no longer be supported in Zen. An Engine DSN points only to a local named database. Client DSNs may also point to an Engine DSN.

DSN-less Connections

As an alternative to DSNs, applications can use DSN-less connections by specifying the Zen driver name directly (see ODBC Connection Strings).

DSNs are not required for applications that access Zen databases only through the Btrieve API or through other SQL access methods, such as ADO.NET. Those access methods use named databases for the connection, which is also an option for ODBC applications.

The Zen Java utilities do not require DSNs. For example, ZenCC uses JDBC with named databases, not ODBC.

Zen DSN Setup

This dialog is available via the ODBC Administrator and allows you to set up a DSN using any of the following Zen ODBC interfaces.

- **Pervasive ODBC Unicode Interface**
  With the 32-bit ODBC Administrator, creates 32-bit DSNs. With the 64-bit ODBC Administrator, creates 64-bit DSNs.
- **Pervasive ODBC Client Interface**
  For 32-bit DSNs
- **Pervasive ODBC Interface**
  For 64-bit DSNs

ODBC Administrator

Windows 64-bit operating systems contain two different executable files for ODBC Administrator, one for 32-bit DSNs and one for 64-bit DSNs. Each ODBC Administrator lists the system DSNs that only match its bitness. That is, the 64-bit ODBC Administrator lists 64-bit system DSNs, and vice versa. If you start ODBC Administrator from the Windows Control Panel, the 64-bit version is run.

Zen Control Center (ZenCC) contains separate options in the Tools menu to start the 32-bit or the 64-bit ODBC Administrator. Note that, if an ODBC Administrator is already open, Windows defaults to it. That is, if the 32-bit ODBC Administrator is open and you attempt to start the 64-bit one, Windows displays the 32-bit version (and vice versa). In other words, only one version of the ODBC Administrator runs at a time. This is a limitation of the Windows operating system rather than Zen.

Data Source Name

The ODBC client-server architecture calls for the naming of each specific data set so that it can be referred to by a well-known name.

Enter a name (called a data source name, or DSN) for the data source to which you wish to set up a connection. This DSN will help you identify the data source.

For additional information about using DSNs with the database engine, see ODBC Database Access.

Description

Enter a description of the DSN, if desired, to help identify the DSN, database, or application.

Server Name/IP

Specifies the machine on which the database engine is running. Enter a machine name or IP address of the server machine to which you want the client to connect.

Transport Hint

Specify the transport protocol to use, or which to try first. Default is TCP.

Database Name

Click **Database Name**, click **Get List**, then select in the list the database to which you wish to connect. The list returns the databases on the server specified for Server Name/IP.

Optionally you may create a new database by clicking **Create**.

Read-Only DSN

Selecting this check box causes the DSN connection string to include the setting OpenMode=1. This setting is available for the following configuratins:

- Pervasive ODBC Unicode Interface (32- and 64-bit)
- Pervasive ODBC Interface (64 bit)

Database Configuration Details

See Create Database Through DSN Setup for the following database configuration details:

- Dictionary Location
- Data File Location
- Integrity Enforced
- Bound

Engine DSN

This option appears only on the 32-bit Client DSN dialog. It is not present on any of the other Zen driver DSN dialogs.

Click **Engine DSN**, click **Get List**, then select in the list the Engine DSN to which you want the client to connect. The list returns the Engine DSNs on the server specified for Server Name/IP.

Optionally, you may create a new Engine DSN by clicking **Create**, or modify an existing Engine DSN by clicking **Modify**.

See also Zen Engine DSN Setup.

Note: New or revised 32-bit applications, local or remote, should connect to a named database, not to an Engine DSN. Alternately, applications could use DSN-less connections (see DSN-less Connections). Avoiding the use of Engine DSNs positions your application for the future when Engine DSNs will no longer be supported in Zen.

Advanced Connection Attributes

The following connection attributes apply to 32-bit Client DSNs, 64-bit DSNs, and Unicode DSNs.

- Enable Array Fetch

- TCP/IP Port Number
- Encoding Translation

(For connection attributes that apply to Engine DSNs, see Engine DSN Advanced Connection Attributes.)

Enable Array Fetch

An array fetch is a memory cache on the client machine for result sets. When array fetch is enabled, data from the latest result set is cached to the client machine's local memory, thereby speeding performance on subsequent queries. We recommend you leave array fetch turned on if you are doing multiple queries.

The default size of the buffer used to cache an array fetch is 64 KB. Its value can be set anywhere from 1 to 64 KB.

TCP/IP Port Number

You can use this setting to change the network port number on which Zen transmits ODBC communications. The network layer on the server engine has a similar setting. You must change both settings at the same time, and you must change them both to the same port number, or else your client and server cannot communicate.

Caution! Do not change the client port number unless you also change the corresponding port number on the server. If the server and client are not using the same port number, they cannot communicate. See TCP/IP Port in *Advanced Operations Guide*.

Generally, the only reason you would need to change this port number is if you have another network service that is already using this port, and it is easier to change the port number for your Zen applications than for the other application.

For additional information about ports, see Changing the Default Communication Ports in *Getting Started with Zen*.

Encoding Translation

Encoding translation refers to the operation of converting the encoding of character data from that present in the database to the encoding present at the client and vice versa. This allows a client to read and write text from and to the database, under certain conditions, even if the database and client are using different encodings. Obviously, if the two encodings are the same, no translation is needed. The effectiveness of the translation is dependent on the amount of overlap between the character sets on the client and server, i.e., the greater number of characters they have in common.

Characters that cannot be translated are replaced by question marks. For example, if the database uses OEM code page 850 and the client uses ANSI code page 1252, the letters will translate, but some graphics symbols will not.

The database connection string or DSN can be configured to automatically negotiate the translation, perform OEM/ANSI translation between different code page encodings, or disable any translation. Automatic translation is the default when using the Unicode driver; no translation is the default for all other Zen ODBC drivers. You can specify automatic translation in the DSN setup, or in the ODBC connection string with the PvTranslate attribute.

The following table summarizes the operation of text encoding translation for various combinations of client and driver encodings. Your application may be either ANSI or Unicode, indicated in column one. The Zen driver is either the Client driver (Client 32-bit or 64-bit driver) or the Unicode driver, column two. (The client and Unicode drivers are described in ODBC Database Access, above.) The Microsoft ODBC Driver Manager connects your application to the Zen ODBC driver and may perform text conversions, as indicated in column three. The three remaining columns describe the Zen driver text handling for a given encoding configuration (column four) for SQL text or CHAR user data (columns five and six, respectively). When data is retrieved from the database, the translation is reversed. Following the table are descriptions of these configuration options.

Application	Zen Driver	Microsoft Driver Manager Text Handlling	Translation Configuration in DSN or Connection String	Zen Driver SQL Text Handling	Zen Driver CHAR Data Handling
ANSI	Client	no translation	none	no translation	no translation
ANSI	Client	no translation	OEM/ANSI	Client encoding to OEM	Client encoding to OEM
ANSI	Client	no translation	Auto	Client encoding to database encoding	Client encoding to database encoding
ANSI	Unicode	Client encoding to UCS-2 for SQL text	Auto	UCS-2 to UTF-8	Client encoding to database encoding
Unicode	Client	UCS-2 to client encoding	none	no translation	no translation
Unicode	Client	UCS-2 to client encoding	OEM/ANSI	Client encoding to OEM	Client encoding to OEM

Application	Zen Driver	Microsoft Driver Manager Text Handlling	Translation Configuration in DSN or Connection String	Zen Driver SQL Text Handling	Zen Driver CHAR Data Handling
Unicode	Client	UCS-2 to client encoding	Auto	Client encoding to database encoding	Client encoding to database encoding
Unicode	Unicode	no translation	Auto	UCS-2 to UTF-8	UCS-2 to database encoding

Note: When using the Zen Client driver, Unicode SQL text is always converted to the client encoding by the Microsoft Driver Manager. This restricts NCHAR literals in a SQL query string to the character set of the client. To preserve NCHAR literals in SQL query text, use the Zen Unicode driver.

DSN Encoding Translation Options

The encoding translation options specify how character data is translated between the Zen database engine and a Zen client application that uses ODBC. This option is only available when configuring a Client 32-bit or 64-bit DSN. The Unicode DSN defaults to automatic.

Automatic

This setting instructs the Zen ODBC client to automatically translate character data encoding when the database encoding on the engine machine differs from the OS encoding on the client machine. Automatic is the default for the Unicode driver.

Character data translation, if required, occurs on the client. (No character data translation is required if the database encoding on the engine machine is the same as the OS encoding on the client machine.)

"Automatic" requires that the client and the server be version 10.1 or greater.

See also Database Code Page and Client Encoding in *Advanced Operations Guide*.

None

This setting means that no character data is translated between the client and server. (The assumption is that the client and server use the same operating system encoding.)

Note that, in versions prior to Zen v10.10, OEM/ANSI Conversion was a single choice and had two states: *not selected* or *selected*. The *not selected* state is now labeled None and is the default for Zen ODBC drivers other than the Unicode driver.

OEM/ANSI Conversion

This setting allows applications to store or retrieve character data in any OEM character set in the Zen engine, while allowing the data to be manipulated and displayed using the ANSI Windows character set in the application.

The Zen ODBC driver translation DLL can perform all necessary translations between the two character sets. This feature can be turned on or off for each DSN. Any character data that is passed to and from the database is correctly translated by the ODBC driver between the OEM and ANSI character sets.

If your application connects to the data source using SQLDriverConnect, you can also specify the translation DLL using the connection string option TRANSLATIONDLL=*path_and_DLL_name*. The translation DLL name for Zen is W32BTXLT.DLL.

Note: The OEM-to-ANSI translation option is available only for Client and 64-bit DSNs. (You can also use this translate option with a local Engine DSN. It is not available when setting up a remote Client connection to an Engine DSN. Keep in mind that Engine DSNs are deprecated and should not be used for new applications.)

Interaction Between Database Code Page and Encoding Translation

The following table explains the interaction between database code page and DSN encoding translation. See Create Database Through DSN Setup for a discussion of code page.

If Database Code Page Is	And the Connection Encoding Translation Is	Then the Zen ODBC Driver
Server Default	None (The equivalent default behavior in versions prior to PSQL v10.10.)	Performs no translation of data or metadata. The assumption is that the OS encoding on the server matches the OS encoding on the client. For compatible data interpretation, the encoding used by the client machine must match the encoding of the data and metadata in the database.

If Database Code Page Is	And the Connection Encoding Translation Is	Then the Zen ODBC Driver
A specific code page	None (The equivalent default behavior in versions prior to PSQL v10.10.)	Performs no translation of data or metadata. The assumption is that the OS encoding on the server matches the OS encoding on the client. For compatible data interpretation, the encoding used by the client machine must match the encoding of the data and metadata in the database.
Server Default *or* A specific code page	OEM/ANSI	Ignores database code page and translates data and metadata from the OEM encoding of the database to ANSI Windows encoding for the client application.
Server Default	Automatic	Translates data and metadata from the default OS encoding on the server to the OS encoding on the client.
A specific code page	Automatic	Translates data and metadata from the database code page to the OS encoding on the client.

Zen Engine DSN Setup

Note that Engine DSNs are 32-bit only.

Windows 64-bit operating systems contain two different executable files for ODBC Administrator, one for 32-bit DSNs and one for 64-bit DSNs. Each ODBC Administrator lists only the system DSNs that match its bitness. That is, the 64-bit ODBC Administrator lists 64-bit system DSNs, and vice versa. If you start ODBC Administrator from the Windows Control Panel, the 64-bit version is run. **The 64-bit version does not list Engine DSNs because they are 32-bit only**.

Zen Control Center (ZenCC) contains separate options in the Tools menu to start the 32-bit or the 64-bit ODBC Administrator. Note that, if an ODBC Administrator is already open, Windows defaults to it. That is, if the 32-bit ODBC Administrator is open and you attempt to start the 64-bit one, Windows displays the 32-bit version (and vice versa). In other words, only one version of the ODBC Administrator runs at a time. This is a limitation of the Windows operating system, *not* Zen.

Note: New or revised 32-bit applications, local or remote, should connect to a named database, not to an Engine DSN. Alternately, applications could use DSN-less connections (see DSN-less Connections). Avoiding the use of Engine DSNs positions your application for the future when Engine DSNs will no longer be supported in Zen.

Data Source Name

The ODBC client-server architecture calls for the naming of each specific data set so that it can be referred to by a well-known name.

Enter a name (called a data source name, or DSN) for the data source to which you wish to set up a connection. This DSN will help you identify the data source.

For additional information about using DSNs with the database engine, see ODBC Database Access.

Description

Enter a description of the DSN, if desired, to help identify the DSN, database, or application.

Database Name

Select a database with which you want to associate the DSN. Optionally, you may create a new database by clicking **Create**.

Database Configuration Details

See Create Database Through DSN Setup for the following database configuration details:

- Dictionary Location
- Data File Location
- Integrity Enforced
- Bound

Engine DSN Advanced Connection Attributes

The connection attributes for Engine DSNs include the following:

- DSN Open Mode

- Encoding Translation

Note: Engine DSNs are deprecated. New or updated applications should use Client DSNs in local or remote connection mode.

DSN Open Mode

The DSN Open Mode options for Engine DSNs allow you to specify one of several characteristics that go into effect when tables are opened through the specified DSN. These options are mutually exclusive—you are not permitted to select more than one.

These options correspond directly to the Btrieve open modes allowed in the Open (0) operation. By setting an Open Mode for a DSN, you are setting the default behavior for tables (corresponding to Btrieve files) opened through that DSN.

Open Mode	ODBC Connection String Generated	SQLSetConnectOption Call
Normal	OPENMODE=0	SQLSetConnectOption(pSubDbc, SQL_ACCESS_MODE, SQL_MODE_READ_WRITE);
Accelerated	OPENMODE=-1	SQLSetConnectOption is ignored
Read-only	OPENMODE=1	SQLSetConnectOption(pSubDbc, SQL_ACCESS_MODE, SQL_MODE_READ_ONLY);
Exclusive	OPENMODE=-4	SQLSetConnectOption is ignored

Normal

Normal mode is the default. Opening a table in Normal mode allows read/write access according to the permissions defined in the database.

If this mode is selected, the ODBC connection string includes OPENMODE=0, and the following ODBC function call is executed when you connect to the database:

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

Accelerated

Opening a table in Accelerated mode provides increased insert/update performance by disabling database engine logging functions for the current user. The changes to logging in Accelerated mode do not affect other users accessing the same table.

Caution! The database engine cannot guarantee transaction atomicity, transaction durability, or archival log safety for any client during use of Accelerated mode by any client. The reason for this restriction is that in the event a restore from log is needed, the log may not contain adequate information to complete the restore, because it is only a partial record of operations on a data file.

For example, if a system failure occurs while the same file is being accessed by a client performing inserts using Accelerated mode and a client performing updates using Normal mode, it is possible for the transaction log to contain updates to records that do not yet exist in the data files, because the Accelerated insert operation in memory was never flushed to disk, while the transactional update operation was written to the transaction log.

An attempt to roll forward an archival log containing this combination of operations will fail.

When this mode is selected, the ODBC connection string includes OPENMODE=-1, and the SQLSetConnectOption call is ignored by the ODBC driver. You cannot use SQLSetConnectOption to specify this mode.

Read-Only

When a table is opened in read-only mode, operations that modify the database structure or the data in the database are not permitted.

If this mode is selected, the ODBC connection string includes OPENMODE=1, and the following ODBC function call is executed when you connect to the database:

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

Exclusive

When a table is opened in exclusive mode, no other connections to the table are permitted. If other users are currently accessing the given table, it cannot be opened in Exclusive mode. You must try again later.

When this mode is selected, the ODBC connection string includes OPENMODE=-4, and the SQLSetConnectOption call is ignored by the ODBC driver. You cannot use SQLSetConnectOption to specify this mode.

Encoding Translation

The encoding translation options are the same as for Client and 64-bit DSNs. See Encoding Translation.

Create Database Through DSN Setup

The following table explains the controls on the Create Database dialog.

Element	Description
Database Name	The name for the database that you want to appear in the database listings. For example, the database name you see in Zen Control Center. **Note:** The database name cannot be the same as an *existing* database name.
Integrity Enforced	Specifies whether integrity constraints (security, RI, and triggers) are enforced on the database. These constraints apply to Btrieve access to the data files as well as ODBC/SQL access. For additional information, see Interactions Between Btrieve and Relational Constraints in *Advanced Operations Guide*.
Bound	Indicates whether or not the database is bound. Binding a database prevents the DDFs or data files from being used in another database and prevents a data file from having two or more different table definitions within the same database. For more information about bound databases, refer to Bound Database versus Integrity Enforced in *Advanced Operations Guide*.
Long Metadata (V2 metadata)	Specifies whether you want the database to use version 1 (V1) or version 2 (V2) metadata. The database engine supports two versions of metadata, referred to as version 1 (V1) and version 2 (V2). Metadata version is a property of the database, which means that it applies to all tables within that database and is recorded in the dbnames.cfg file. A database cannot use some tables with V1 metadata and others with V2 metadata. Metadata from the two versions cannot interact. For additional information, see Zen Metadata in *SQL Engine Reference*.
Code Page	Specifies the code page that applies to the database data and metadata.This property is stored in DBNAMES.CFG. The default code page is "server default," meaning the operating system code page on the server where the database engine is running. Note that database code page and client encoding are separate but interrelated. See Database Code Page and Client Encoding in *Advanced Operations Guide*.
Btrieve security policy	Specifies the security model to use for the transactional interface.See Security Models for the MicroKernel Engine in *Advanced Operations Guide*.
Dictionary Location	This location specifies where the dictionary files (DDFs) reside on physical storage. This location must be on the same server to which you are connected (and where the database engine is running). The location must be formatted as though you are working directly at the server machine. Enter a path in the form *drive:\path*, where *drive* is a drive letter on the server.

Element	Description
Data File Location	This location specifies where the data files reside on physical storage. The **Add** button lets you add locations to the list. The **Delete** button lets you remove locations from the list. The locations must be on the same server where the database engine is running. Specify the location in the same manner as for the dictionary locations.

ODBC Connection Strings

This section describes the ODBC connection strings supported by Zen. This information is provided for advanced users using a database access tool that allows connection strings to be specified and for developers writing ODBC applications to access Zen.

ODBC Driver Parameter

You must specify the Zen ODBC driver to use to connect to the Zen database engine. See Zen ODBC Driver Names for a description of the available drivers.

Use the ODBC **Driver** parameter to specify the appropriate driver. For example:

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

Driver Parameters

The specific driver specified by the Driver parameter has additional attribute parameters for naming the server, port, database, etc. In addition to these common parameters, each driver has parameters specific to it. The tables below show, for the different drivers, the driver parameters

that may be used. The attributes can be included with the ODBC function SQLDriverConnect or with SQLConnect.

Unicode Connection String Parameters	Description
ServerName=*server*[.*port*]	Specify the machine name or IP address of the computer to which you wish to connect. Required. *Port* is provided from backwards compatibility and allows you to specify the port number to use if you are not using the default port. When using IPv6 addresses and appending a port number in ODBC connections, use an IPv6-literal.net name or UNC-safe name. See Drive-based Formats in *Getting Started with Zen*.
TransportHint=TCP	Specify the transport protocol to use, or which to try first. Default is TCP. Optional.
DBQ=[@]*db_name*	Specify the internal database name (not DSN) to which you wish to connect. Required. The @ character is optional. It has no particular effect and is supported only for backward compatibility.
TCPPort=*port*	Specify the TCP/IP port on which to find the server. Optional. See also Changing the Default Communication Ports in *Getting Started with Zen*.
ArrayFetchOn=1 | 0	Specify whether to cache result sets on the client. Default is 1, "On." Optional.
ArrayBufferSize=*size*	Specify the size of the client cache, in KB. Default is 8 KB. Optional.
PvTranslate=auto	Specify how to handle the data encoding when the client connects to the database engine. (See Encoding Translation for more information.) PvTranslate defaults to "auto" for the Unicode driver. This allows you to use NCHAR columns and NCHAR literals with wide character data without having to explicitly set PvTranslate to "auto." With the attribute is set to "auto," the client and server automatically establish compatible encoding. Data translation, if required, occurs on the client.
UID=*user_name*	If security is enabled for the given database, specify the user name. Optional, depending on security. See Zen Security in *Advanced Operations Guide*.

Unicode Connection String Parameters	Description
PWD=*password*	If security is enabled for the given database, specify the password. Optional, depending on security. See Zen Security in *Advanced Operations Guide*.

- **Example A**

  Connect to a database named SOMEDATA that contains wide character data on a remote server named ServerMain using TCP/IP port 1590:

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

- **Example B**

  Connect to a database named EuropeRegion4 that contains wide character data on a local server named with database security turned on:

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

64-Bit Connection String Parameters	Description
ServerName=*server*[.*port*]	Specify the machine name or IP address of the computer to which you wish to connect. Required. *Port* is provided from backwards compatibility and allows you to specify the port number to use if you are not using the default port. When using IPv6 addresses and appending a port number in ODBC connections, use an IPv6-literal.net name or UNC-safe name. See Drive-based Formats in *Getting Started with Zen*.
TransportHint=TCP	Specify the transport protocol to use, or which to try first. Default is TCP. Optional.
DBQ=[@]*db_name*	Specify the internal database name (not DSN) to which you wish to connect. Required. The @ character is optional. It has no particular effect and is supported only for backward compatibility.
TCPPort=*port*	Specify the TCP/IP port on which to find the server. Optional. See also Changing the Default Communication Ports in *Getting Started with Zen*.
ArrayFetchOn=1 | 0	Specify whether to cache result sets on the client. Default is 1, "On." Optional.

64-Bit Connection String Parameters	Description
ArrayBufferSize=*size*	Specify the size of the client cache, in KB. Default is 8 KB. Optional.
PvTranslate=auto	Specify how to handle the data encoding when the client connects to the database engine. The attribute can either be absent, or have a value set to "auto" to indicate automatic translation. (See Encoding Translation for more information.) With the attribute is set to "auto," the client and server automatically establish compatible encoding. Data translation, if required, occurs on the client. Note that "auto" overrides the "OEM/ANSI" setting for a DSN. If the attribute is absent, ODBC does not translate any character data. This is the default behavior. The "OEM/ANSI" setting may still apply. See OEM/ANSI Conversion.
UID=*user_name*	If security is enabled for the given database, specify the user name. Optional, depending on security. See Zen Security in *Advanced Operations Guide*.
PWD=*password*	If security is enabled for the given database, specify the password. Optional, depending on security. See Zen Security in *Advanced Operations Guide*.

Example

Connect to a local named database acctdomestic with a 64-bit application:

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

32-Bit Connection String Parameters	Description
ServerName=*server*[.*port*]	Specify the machine name or IP address of the computer to which you wish to connect. Required. *Port* is provided from backwards compatibility and allows you to specify the port number to use if you are not using the default port. When using IPv6 addresses and appending a port number in ODBC connections, use an IPv6-literal.net name or UNC-safe name. See Drive-based Formats in *Getting Started with Zen*.
ServerDSN=*dsn_name*	Specify the Engine DSN to which you wish to connect. Required, unless DBQ is specified.

32-Bit Connection String Parameters	Description
TransportHint=TCP	Specify the transport protocol to use, or which to try first. Default is TCP. Optional.
DBQ=[@]*db_name*	Specify the internal database name (not DSN) to which you wish to connect. Required. The @ character is optional. It has no particular effect and is supported only for backward compatibility.
TCPPort=*port*	Specify the TCP/IP port on which to find the server. Optional. See also Changing the Default Communication Ports in *Getting Started with Zen*.
ArrayFetchOn=1 | 0	Specify whether to cache result sets on the client. Default is 1, "On." Optional.
ArrayBufferSize=*size*	Specify the size of the client cache, in KB. Default is 8 KB. Optional.
PvTranslate=auto	Specify how to handle the data encoding when the client connects to the database engine. The attribute can either be absent, or have a value set to "auto" to indicate automatic translation. (See Encoding Translation for more information.) With the attribute is set to "auto," the client and server automatically establish compatible encoding. Data translation, if required, occurs on the client. Note that "auto" overrides the "OEM/ANSI" setting for a DSN. If the attribute is absent, ODBC does not translate any character data. This is the default behavior. The "OEM/ANSI" setting may still apply. See OEM/ANSI Conversion.
UID=*user_name*	If security is enabled for the given database, specify the user name. Optional, depending on security. See Zen Security in *Advanced Operations Guide*.
PWD=*password*	If security is enabled for the given database, specify the password. Optional, depending on security. See Zen Security in *Advanced Operations Guide*.

- **Example A**

  Connect to a database named Atlantis on a remote server named AncientLore using TCP/IP port 1585:

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

- **Example B**

  Connect to a database named DomSales on a remote server named SalesSvr with database security turned on:

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

- **Example C**

  Connect to an Engine DSN named mydata on a remote server named MyServer and establish automatic encoding support:

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

Connection String	Description
DBQ=[@]*db_name*	Specify the internal database name (not DSN) to which you wish to connect. Required. The @ character is optional. It has no particular effect and is supported only for backward compatibility.
UID=*user_name*	If security is enabled for the given database, specify the user name. Optional, depending on security. See Zen Security in *Advanced Operations Guide*.
PWD=*password*	If security is enabled for the given database, specify the password. Optional, depending on security. See Zen Security in *Advanced Operations Guide*.
OPENMODE=-4 | -1 | 0 | 1	Specify the default file open mode for files opened with the current connection. Default is 0, "Normal." Can be used only with local connections, not remote client connections. Optional. For more information on file open modes, see DSN Open Mode.
TRANSLATIONDLL=*path_and_DLL_name*	Specify the full path name of the DLL to use for OEM/ANSI translation. For more information, see OEM/ANSI Conversion.

- **Example**

Connect to a local database named DATA5:

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

Note: Engine DSNs are deprecated. New or updated applications should use Client DSNs in local or remote connection mode.