CHAPTER

4

Programmer's Reference

Introduction

This chapter is intended for applications programmers and others who need information about how the SAS ODBC driver has been implemented. It provides information about the driver's support for ODBC functions, SQL grammar, and SQL data types.

For complete information about the ODBC standard, see the *Microsoft ODBC 2.0 Programmer's Reference and SDK Guide*.

Support for and Implementation of ODBC Functions

Microsoft's ODBC specification defines three levels of support for ODBC functions: CORE, LEVEL 1, and LEVEL 2. The SAS ODBC driver is ODBC 2.0 compliant and supports LEVEL 1 functionality, with the exception of those functions associated with cursors (SQLGetCursorName, SQLSetCursorName) and large data fields (SQLParamData, SQLPutData, SQLExtendedFetch), which are not supported. The following tables provide explanations of the functions that are not supported or whose implementation details may be noteworthy to applications developers.

CORE Functions

Table 4.1 CORE Functions

Function Name	Purpose	SAS ODBC Driver Implementation
SQLBindParameter	Assigns storage for a parameter in an SQL statement	Note that SQL_DATA_AT_EXEC is not supported because SAS does not support large data fields.
SQLCancel	Cancels an SQL statement	This function is used only in asynchronous mode, which SAS does not support. The SAS System's interprocess communication library does not provide any way to interrupt a transaction in process. SQLCancel does not cause an active statement to terminate immediately, and it does not halt an operation that is already in process. This is allowable within the function specification. This call always returns as successful.
SQLColAttributes	Describes the attributes of a column in the result set	A common reason for applications to call this function is to determine whether a column of data is a dollar amount. With SAS data sets or views, this is inferred from the FORMAT. See "Supported Data Types" on page 34 for more information about SAS FORMATs.
SQLGetCursorName	Returns the cursor name that is associated with a statement handle	SAS does not support cursors, so this function returns SQL_ERROR with SQLSTATE set to IM001 ("Driver does not support this function").
SQLPrepare	Prepares an SQL statement for later execution	Does not check syntax at this point; syntax checking is done later in the SQLExecute call by the server.
SQLSetCursorName	Specifies a cursor name	SAS does not support cursors, so this function returns SQL_ERROR with SQLSTATE set to IM001 ("Driver does not support this function").
SQLTransact	Commits or rolls back a transaction	Always returns SQL_SUCCESS for SQL_COMMIT. Returns an error for SQL_ROLLBACK because SAS does not support transactions.

LEVEL 1 Functions

Table 4.2 LEVEL 1 Functions

Function Name	Purpose	SAS ODBC Driver Implementation
SQLColumns	Returns the list of column names from specified tables	SAS uses a specially formatted query to query the virtual table DICTIONARY.COLUMNS.
SQLDriverConnect	Connects to a specific driver by connection string or requests that the Driver Manager and the driver display connection dialogs for the user	SAS makes use of the same dialogs that are used in configuration. If input was adequate, it continues with the connection rather than saving the parameters. However, in many cases input is not required. The connection to the host is made at this time.
SQLParamData	Returns the storage value that has been assigned to a parameter for which data will be sent at execution time	This function is used for large data fields, which SAS does not support, so the function returns SQL_ERROR with SQLSTATE set to IM001 ("Driver does not support this function").
SQLPutData	Sends part or all of a data value for a parameter	This function is used for large data fields, which SAS does not support, so the function returns SQL_ERROR with SQLSTATE set to IM001 ("Driver does not support this function").
SQLSpecialColumns	Retrieves information about the optimal set of columns that uniquely identifies a row in a specified table, or about the columns that are automatically updated when any value in the row is updated by a transaction	The SAS ODBC driver uses a query on the DICTIONARY.INDEXES view to obtain this information.
SQLStatistics	Retrieves statistics about a single table and the list of indexes that are associated with the table	The SAS ODBC driver uses a query on the DICTIONARY.INDEXES view to obtain this information.
SQLTables	Returns the list of table names stored in a specific data source	The SAS ODBC driver uses a query on the DICTIONARY.TABLES and DICTIONARY.MEMBERS views to obtain this information.

Support for SQL Grammar

Microsoft's ODBC specification defines three levels of support for SQL grammar: MINIMUM, CORE and EXTENDED. The SAS ODBC driver supports all of MINIMUM and some of the CORE SQL statements and the statement elements. See the *SAS Procedures Guide* for complete information about supported grammar.

Supported Data Types

Internally, the SAS System supports two data types for storing data:

CHAR	fixed-length character data, 200-character maximum
NUM	double-precision floating-point number

However, by using SAS *format* information, the SAS ODBC driver is able to represent other ODBC data types, both when responding to queries and in CREATE TABLE requests. (A SAS format is a string that describes how data should be printed. The SAS System associates format information with each column in a table.)

The following sections explain conventions for data type representation that the SAS ODBC driver follows.

For information about user-specified SQL options that can also affect data type representations, see “User-Specified SQL Options” on page 36. For more information about SAS formats, see *SAS Language Reference: Dictionary*.

Data Types Reported on Queries

When the SQLDescribeCol and SQLColAttributes functions are called against active queries, the SAS ODBC driver reports data types as follows:

□ When the SQLDescribeCol function is called, the SAS ODBC driver reports CHAR data types as SQL_CHAR. NUM data types are generally reported as SQL_DOUBLE.

However, the SAS System stores dates and times as numbers, and the SAS ODBC driver uses SAS format information to infer the following additional SQL data types from NUM data types:*

SAS Data Type	SQL Data Type
NUM FORMAT=DATE*n*.	SQL_DATE
NUM FORMAT=TIME*n*.	SQL_TIME
NUM FORMAT=DATETIME*n*.	SQL_TIMESTAMP

In each of the previous FORMAT= strings, *n* is a number that selects the printable representation by specifying a width for printing. The value of *n* is not relevant to the driver.

□ When the SQLColAttributes function is called, if a NUM column has a format of DOLLAR*n*., the SAS ODBC driver identifies it as financial data (having a column attribute of SQL_COLUMN_MONEY).

Creating or Comparing Date, Time, and Datetime Values

When you create or compare date, time and datetime values in SAS data sets from an ODBC application, you must consider the following:

□ A SAS time value is the number of seconds since the current day began. That is, 0 is 00:00:00 or 12:00:00 AM and 86399 is 11:59:59 PM.

* For a complete list of date and time formats that the SAS ODBC driver supports, see the table of formats listed by categories in *SAS Language Reference: Dictionary*.

Note: ODBC does not support negative time values or values greater than one day's worth of seconds. The SAS ODBC driver returns an error for time values that are less than 0 or greater than 86399 (the last second of the day). △

- □ A SAS date value is the number of days since January 1, 1960. That is, 0 is 01jan1960 and -1 is 31dec1959.
- □ A SAS datetime value is the number of seconds since midnight on January 1, 1960. That is, 0 is 01jan1960:00:00:00 and -1 is 31dec1959:11:59:59.

Both ODBC and SAS date, time and datetime literals are supported by the SAS ODBC driver.

CAUTION:
You can only compare equivalent literals against SAS date, time or datetime values since they each have a different unit of measure. △

For example, you cannot compare a SAS data set value that has been defined with a datetime format against a date literal using

```
select * where hiredate = {d'1995-01-02'}
```

or

```
select * where hiredate = '02jan1995'd
```

Instead, use a datetime literal such as

```
select * where hiredate = {ts'1995-01-02 00:00:00'}
```

or

```
select * where hiredate = '02jan1995:00:00:00'dt
```

Interpretation of Data Types in CREATE TABLE Requests

In CREATE TABLE requests, the SAS ODBC driver interprets certain column-type specifications by creating NUM variables and associating SAS formats with them, as shown in the following table:

Table 4.3 Correspondence of CREATE TABLE Data Types and SAS Data Types

CREATE TABLE Data Type Name	ODBC Data Type	SAS Data Type
char(*w*)	SQL_CHAR	CHAR(*w*)
num(*w*, *d*)	SQL_DOUBLE	NUM
num(*w*, *d*)	SQL_FLOAT	NUM
integer	SQL_INTEGER	NUM FORMAT=11.0
date9x	SQL_DATE	NUM FORMAT=DATE9.
datetime19x	SQL_TIMESTAMP	NUM FORMAT=DATETIME19.
time8x	SQL_TIME	NUM FORMAT=TIME8X

The data type names listed in the first column of the table are the values that are returned by SQLColAttributes (with the parameter SQL_COLUMN_TYPE_NAME) and by SQLGetTypeInfo. For all CREATE TABLE statements, the SAS ODBC driver translates these data type names into the respective SAS data types shown under the SAS Data Type heading. Do not try to use the ODBC data types directly in SAS.

In a CREATE TABLE statement, any FORMAT= specification is passed on to the SAS System unmodified, so a column within a table (or data set) can be created according to any exact specification that is required for its use within SAS. For example, in the following CREATE TABLE statement, variable **B**'s data type and format are passed directly to the SAS System.

```
CREATE TABLE
   SASUSER.TABLE1
      (A INTEGER,
       B NUM FORMAT=9.5,
       C CHAR(40) );
```

User-Specified SQL Options

This section describes two SQL options that affect how other default conversions of data types or data values can be made: **Infer INTEGER from FORMATS** and **Support VARCHAR**. A third SQL option, **Fuzz Numbers at N Places**, is important in comparison operations. You can specify these options in the SQL Options page of the SAS ODBC Driver Configuration dialog. (See "Naming Your Data Source and Specifying SQL Options" on page 14.)

Infer INTEGER from FORMAT Option

Even when no FORMAT string is specified for SAS data, the SAS System assigns a default width and number of decimal places to the data. If the SQL Option **Infer INTEGER from FORMAT** is selected, then the SAS ODBC driver reports SAS columns of NUM(*n*,0) data types as SQL_INTEGER, where *n* is less than 12. This can be important, because some PC products do not use indexes on floating-point columns. If those columns actually contain only integer values, then using this option enables these products to honor the index and to allow updates. See "Updating Attached Tables" on page 29 for more information.

Support VARCHAR Option

The SQL option **Support VARCHAR** causes the SAS ODBC driver to report the data type CHAR(*n*) as SQL_VARCHAR, where *n* is greater than 80. Because SAS is fixed width, CHAR fields are often specified at the maximum. For example, for a list of messages the text width might be specified as 200 characters, even though the average width is much less. Reporting it as SQL_VARCHAR enables some PC products to use less memory.

Fuzz Numbers at N Places Option

This option addresses a problem that arises from the conversion of floating-point numbers. Floating-point numbers are stored in different binary representations on different computer hardware. Even when data is transferred between different applications on the same type of hardware, the precision of floating-point numbers may be affected slightly due to conversion between ASCII and binary representations.

This effect is usually so slight that it is insignificant when a number is used in calculations. For example, the numbers 65.8 and 65.799999999999 are practically identical for mathematical purposes, and the difference between them might be the result of conversion between representations rather than any purposeful change in value.

However, such a slight difference in value can keep a number from comparing correctly. For example, many ODBC applications include a WHERE clause that lists every column in a record at its current value whenever the application performs an UPDATE. This is done to ensure that the record has not been changed since the last time it was read. Sometimes a comparison may fail because of the aforementioned problem with floating-point conversion.

To solve this problem, SAS "fuzzes" numbers (standardizes the degree of precision to use, overriding the hardware-specific representations). Instead of using exact comparisons, SAS checks to make sure that the numbers are acceptably close.

By default, the degree of precision is 12 decimal places. Given a number **N**, then if **N1** were to be checked for equality with **N**, the SAS ODBC driver would use the SQL BETWEEN function to determine whether **N1 > (N - (ABS(N * 10**-12))) AND N1 < (N + (ABS(N * 10**-12)))**.

If **N=0**, the driver checks for **BETWEEN -(10**-12) AND (10**-12)**.

SAS ODBC Driver Error Codes

See the *Microsoft ODBC 2.0 Programmer's Reference and SDK Guide* for information about the SQLSTATE values (return codes) and associated texts that can be returned for the SQLError function.

For explanations of messages that may be returned by your communications software, see the Appendix in this book.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® ODBC Driver User's Guide and Programmer's Reference, Version* 8, Cary, NC: SAS Institute Inc., 1999.

SAS® ODBC Driver User's Guide and Programmer's Reference, Version 8

ISBN 1–58025–505-1

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999