Related Functions

For information about	See
Discovering and listing values required to connect to a data source	**SQLBrowseConnect** (extension)
Connecting to a data source	**SQLConnect**
Returning data source names	**SQLDataSources** (extension)
Connecting to a data source using a connection string or dialog box	**SQLDriverConnect** (extension)

SQLError

ODBC 1.0

Core **SQLError** returns error or status information.

Syntax RETCODE **SQLError**(*henv*, *hdbc*, *hstmt*, *szSqlState*, *pfNativeError*, *szErrorMsg*, *cbErrorMsgMax*, *pcbErrorMsg*)

The **SQLError** function accepts the following arguments.

Type	Argument	Use	Description
HENV	*henv*	Input	Environment handle or SQL_NULL_HENV.
HDBC	*hdbc*	Input	Connection handle or SQL_NULL_HDBC.
HSTMT	*hstmt*	Input	Statement handle or SQL_NULL_HSTMT.
UCHAR FAR *	*szSqlState*	Output	SQLSTATE as null-terminated string. For a list of SQLSTATEs, see Appendix A, "ODBC Error Codes."
SDWORD FAR *	*pfNativeError*	Output	Native error code (specific to the data source).
UCHAR FAR *	*szErrorMsg*	Output	Pointer to storage for the error message text.
SWORD	*cbErrorMsgMax*	Input	Maximum length of the *szErrorMsg* buffer. This must be less than or equal to SQL_MAX_MESSAGE_LENGTH – 1.
SWORD FAR *	*pcbErrorMsg*	Output	Pointer to the total number of bytes (excluding the null termination byte) available to return in *szErrorMsg*. If the number of bytes available to return is greater than or equal to *cbErrorMsgMax*, the error message text in *szErrorMsg* is truncated to *cbErrorMsgMax* – 1 bytes.

Returns SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NO_DATA_FOUND, SQL_ERROR, or SQL_INVALID_HANDLE.

Diagnostics **SQLError** does not post error values for itself. **SQLError** returns SQL_NO_DATA_FOUND when it is unable to retrieve any error information, (in which case *szSqlState* equals 00000). If **SQLError** cannot access error values for any reason that would normally return SQL_ERROR, **SQLError** returns SQL_ERROR but does not post any error values. If the buffer for the error message is too short, **SQLError** returns SQL_SUCCESS_WITH_INFO but, again, does not return a SQLSTATE value for **SQLError**.

To determine that a truncation occurred in the error message, an application can compare *cbErrorMsgMax* to the actual length of the message text written to *pcbErrorMsg*.

Comments

An application typically calls **SQLError** when a previous call to an ODBC function returns SQL_ERROR or SQL_SUCCESS_WITH_INFO. However, any ODBC function can post zero or more errors each time it is called, so an application can call **SQLError** after any ODBC function call.

SQLError retrieves an error from the data structure associated with the rightmost non-null handle argument. An application requests error information as follows:

- To retrieve errors associated with an environment, the application passes the corresponding *henv* and includes SQL_NULL_HDBC and SQL_NULL_HSTMT in *hdbc* and *hstmt*, respectively. The driver returns the error status of the ODBC function most recently called with the same *henv*.
- To retrieve errors associated with a connection, the application passes the corresponding *hdbc* plus an *hstmt* equal to SQL_NULL_HSTMT. In such a case, the driver ignores the *henv* argument. The driver returns the error status of the ODBC function most recently called with the *hdbc*.
- To retrieve errors associated with a statement, an application passes the corresponding *hstmt*. If the call to **SQLError** contains a valid *hstmt*, the driver ignores the *hdbc* and *henv* arguments. The driver returns the error status of the ODBC function most recently called with the *hstmt*.
- To retrieve multiple errors for a function call, an application calls **SQLError** multiple times. For each error, the driver returns SQL_SUCCESS and removes that error from the list of available errors.

When there is no additional information for the rightmost non-null handle, **SQLError** returns SQL_NO_DATA_FOUND. In this case, *szSqlState* equals 00000 (Success), *pfNativeError* is undefined, *pcbErrorMsg* equals 0, and *szErrorMsg* contains a single null termination byte (unless *cbErrorMsgMax* equals 0).

The Driver Manager stores error information in its *henv*, *hdbc*, and *hstmt* structures. Similarly, the driver stores error information in its *henv*, *hdbc*, and *hstmt* structures. When the application calls **SQLError**, the Driver Manager checks if there are any errors in its structure for the specified handle. If there are errors for the specified handle, it returns the first error; if there are no errors, it calls **SQLError** in the driver.

The Driver Manager can store up to 64 errors with an *henv* and its associated *hdbcs* and *hstmts*. When this limit is reached, the Driver Manager discards any subsequent errors posted on the Driver Manager's *henv*, *hdbcs*, or *hstmts*. The number of errors that a driver can store is driver-dependent.

An error is removed from the structure associated with a handle when **SQLError** is called for that handle and returns that error. All errors stored for a given handle are removed when that handle is used in a subsequent function call. For example, errors on an *hstmt* that were returned by **SQLExecDirect** are removed when **SQLExecDirect** or **SQLTables** is called with that *hstmt*. The errors stored on a given handle are not removed as the result of a call to a function using an associated handle of a different type. For example, errors on an *hdbc* that were returned by **SQLNativeSql** are not removed when **SQLError** or **SQLExecDirect** is called with an *hstmt* associated with that *hdbc*.

For more information about error codes, see Appendix A, "ODBC Error Codes."

Related Functions None.

SQLExecDirect

ODBC 1.0

Core

SQLExecDirect executes a preparable statement, using the current values of the parameter marker variables if any parameters exist in the statement. **SQLExecDirect** is the fastest way to submit an SQL statement for one-time execution.

Syntax

RETCODE **SQLExecDirect**(*hstmt*, *szSqlStr*, *cbSqlStr*)

The **SQLExecDirect** function uses the following arguments.

Type	Argument	Use	Description
HSTMT	*hstmt*	Input	Statement handle.
UCHAR FAR *	*szSqlStr*	Input	SQL statement to be executed.
SDWORD	*cbSqlStr*	Input	Length of *szSqlStr*.

Returns

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NEED_DATA, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

Diagnostics

When **SQLExecDirect** returns either SQL_ERROR or SQL_SUCCESS_WITH_INFO, an associated SQLSTATE value may be obtained by calling **SQLError**. The following table lists the SQLSTATE values commonly returned by **SQLExecDirect** and explains each one in the context of this function; the notation "(DM)" precedes the descriptions of SQLSTATEs returned by the Driver Manager. The return code associated with each SQLSTATE value is SQL_ERROR, unless noted otherwise.

SQLSTATE	Error	Description
01000	General warning	Driver-specific informational message. (Function returns SQL_SUCCESS_WITH_INFO.)
01004	Data truncated	The argument *szSqlStr* contained an SQL statement that contained a character or binary parameter or literal and the value exceeded the maximum length of the associated table column. The argument *szSqlStr* contained an SQL statement that contained a numeric parameter or literal and the fractional part of the value was truncated. The argument *szSqlStr* contained an SQL statement that contained a date or time parameter or literal and a timestamp value was truncated.

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.