RDM Embedded 10.1

ODBC API Reference Guide

Trademarks

RDMe ODBC API Reference

The topics in this section describe each ODBC function in alphabetical order. Each function is defined as a C programming language function. Descriptions include the following:

- Purpose
- Conformance
- Syntax
- Arguments
- Return values

All of the following APIs conform to the ODBC 3.51 specification unless otherwise noted. RDM Embedded specific extensions to the APIs will be documented in the **Diagnostics** or **Comments** sections of the page. If function does not contain a **Diagnostics** or **Comments** section, the MSDN documentation applies to that specific function.

Unicode API Functions

The RDM Embedded ODBC API supports both ANSI and Unicode versions of all functions that accept pointers to character strings or SQLPOINTER in their arguments. The Unicode functions are implemented as functions with a suffix of "W", such as **SQLExecDirectW** and **SQLGetInfoW**.

SQLAllocHandle

Allocates an environment, connection, statement, or descriptor handle.

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQL_API SQLAllocHandle(
        SQLSMALLINT      HandleType,
        SQLHANDLE        InputHandle,
        SQLHANDLE       *OutputHandle)
```

Arguments

`HandleType`	`(input)`	The type of handle to be allocated by `SQLAllocHandle`. Must be one of the following values: • SQL_HANDLE_ENV • SQL_HANDLE_DBC • SQL_HANDLE_STMT • SQL_HANDLE_DESC
`InputHandle`	`(input)`	The input handle in whose context the new handle is to be allocated. If `HandleType` is SQL_HANDLE_ENV, this is SQL_NULL_HANDLE. If `HandleType` is SQL_HANDLE_DBC, this must be an environment handle, and if it is SQL_HANDLE_STMT or SQL_HANDLE_DESC, it must be a connection handle.
`OutputHandlePtr`	`(output)`	Pointer to a buffer in which to return the handle to the newly allocated data structure.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

This function allocates an environment, connection, or SQL statement handle and its associated resources.

> RDM Embedded does not require you to allocate an environment handle before allocating connection or statement handles, and doing so currently has no effect. The environment handle type is included only for ODBC compatibility.

For more information, reference MSDN documentation for SQLAllocHandle.

SQLBindCol

Binds application data buffers to columns in the result set.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQL_API SQLBindCol(
        SQLHSTMT         StatementHandle,
        SQLUSMALLINT     ColumnNumber,
        SQLSMALLINT      TargetType,
        SQLPOINTER       TargetValue,
        SQLLEN           BufferLength,
        SQLLEN          *StrLen_or_Ind)
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`ColumnNumber`	`(input)`	The column's location in the result data set. Columns are numbered sequentially from left to right, starting with 1, as they appear in the result data set.
`TargetType`	`(input)`	The data type of the value buffer that the column data being retrieved is to be stored in.
`TargetValue`	`(output)`	A pointer to a location in memory where the driver is to store column data when it is retrieved (fetched) from the result data set or where the application is to store column data that is to be written to a data source with a positioned UPDATE or DELETE operation.
`BufferLength`	`(input)`	The size of the buffer.
`StrLen_or_Ind`	`(output)`	A pointer to a location in memory where this function is to store either the size of the data value associated with the column or a special indicator value associated with the column data.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

This function binds application data buffers to columns in the result set.

For more information, reference MSDN documentation for SQLBindCol.

SQLBindParameter

Binds a buffer to a parameter marker in an SQL statement.

Conformance

Version Introduced: ODBC 2.0 Standards Compliance: ODBC

Syntax

```
SQLRETURN SQLBindParameter(
      SQLHSTMT        StatementHandle,
      SQLUSMALLINT    ParameterNumber,
      SQLSMALLINT     InputOutputType,
      SQLSMALLINT     ValueType,
      SQLSMALLINT     ParameterType,
      SQLULEN         ColumnSize,
      SQLSMALLINT     DecimalDigits,
      SQLPOINTER      ParameterValuePtr,
      SQLLEN          BufferLength,
      SQLLEN *        StrLen_or_IndPtr)
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`ParameterNumber`	`(input)`	Parameter number, ordered sequentially in increasing parameter order, starting at 1.
`InputOutputType`	`(input)`	The type of the parameter.
`ValueType`	`(input)`	The C data type of the parameter.
`ParameterType`	`(input)`	The size of the buffer.
`ColumnSize`	`(input)`	The size of the column or expression of the corresponding parameter marker.
`DecimalDigits`	`(input)`	The decimal digits of the column or expression of the corresponding parameter marker.
`ParameterValuePtr`	`(deferred input)`	A pointer to a buffer for the parameter's data.
`BufferLength`	`(input/output)`	Length of the `ParameterValuePtr` buffer in bytes.
`StrLen_or_IndPtr`	`(deferred input)`	A pointer to a buffer for the parameter's length.

Required Headers

```
#include "sqlext.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLBindParameter.

SQLCancel

Cancels the processing on a statement.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQL_API SQLCancel(
        SQLHSTMT         StatementHandle)
```

Arguments

`StatementHandle` `(input)` Statement handle.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional

	information is available from `SQLGetDiagRec` or `SQLGet-DiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

On RDM Embedded, `SQLCancel` supports the following type of processing.

- A function on a statement that needs data.

The following types of processing are not supported due to the driver limitations.

- A function running asynchronously on a statement.
- A function running on the statement on another thread.

For more information, reference MSDN documentation for SQLCancel.

SQLCloseCursor

Closes a cursor that has been opened on a statement and discards pending results.

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQL_API SQLCloseCursor(
        SQLHSTMT         StatementHandle)
```

Arguments

`StatementHandle` `(input)` Statement handle.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional

	information is available from `SQLGetDiagRec` or `SQLGet-DiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLCloseCursor.

SQLColAttribute

Returns descriptor information for a column in a result set. Descriptor information is returned as a character string, a 32-bit descriptor-dependent value, or an integer value.

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLColAttribute (
      SQLHSTMT         StatementHandle,
      SQLUSMALLINT     ColumnNumber,
      SQLUSMALLINT     FieldIdentifier,
      SQLPOINTER       CharacterAttributePtr,
      SQLSMALLINT      BufferLength,
      SQLSMALLINT *    StringLengthPtr,
      SQLPOINTER       NumericAttributePtr)
```

Arguments

StatementHandle	(input)	Statement handle.
ColumnNumber	(input)	The number of the record in the IRD from which the field value is to be retrieved. This argument corresponds to the column number of result data, ordered sequentially in increasing column order, starting at 1. Columns can be described in any order. Column 0 can be specified in this argument, but all values except SQL_DESC_TYPE and SQL_DESC_OCTET_LENGTH will return undefined values.
FieldIdentifier	(input)	The descriptor handle. This handle defines which field in the IRD should be queried (for example, SQL_COLUMN_TABLE_NAME).
CharacterAttributePtr	(output)	Pointer to a buffer in which to return the value in the FieldIdentifier field of the ColumnNumber row of the IRD, if the field is a character string. Otherwise, the field is unused. If CharacterAttributePtr is NULL, StringLengthPtr will still return the total number of bytes (excluding the null-termination character for character data) available to return in the buffer pointed to by CharacterAttributePtr.
BufferLength	(input)	If FieldIdentifier is an ODBC-defined field and CharacterAttributePtr points to a character string or binary buffer, this argument should be the length of *CharacterAttributePtr. If FieldIdentifier is an ODBC-defined field and *CharacterAttributePtr is an integer, this field is ignored. If the *CharacterAttributePtr is a Unicode string (when calling **SQLColAttributeW**), the BufferLength argument must be an even number. If FieldIdentifier is a driver-defined field, the application indicates the nature of the field to the Driver Manager by setting the BufferLength argument. BufferLength can have the following values: • If CharacterAttributePtr is a pointer to a pointer, BufferLength should have the value SQL_IS_POINTER. • If CharacterAttributePtr is a pointer to a character string, the BufferLength is the length of the buffer. • If CharacterAttributePtr is a pointer to a binary buffer, the application places the result of the SQL_LEN_BINARY_ATTR(length) macro in BufferLength. This places a negative value in BufferLength. • If CharacterAttributePtr is a pointer to a fixed-length data type, BufferLength must be one of the following: SQL_IS_INTEGER, SQL_IS_UNINTEGER, SQL_SMALLINT, or SQLUSMALLINT.
StringLengthPtr	(output)	Pointer to a buffer in which to return the total number of bytes (excluding the null-termination byte for character data) available to return in *CharacterAttributePtr. For character data, if the number of bytes available to return is greater than or equal to BufferLength, the descriptor information in *CharacterAttributePtr is truncated to BufferLength minus the length of a null-termination character and is null-terminated by the driver. For all other types of data, the value of BufferLength is ignored and the driver assumes the size of *CharacterAttributePtr is 32 bits.
NumericAttributePtr	(output)	Pointer to an integer buffer in which to return the value in the FieldIdentifier field of the ColumnNumber row of the IRD, if the field is a numeric descriptor type, such as SQL_DESC_COLUMN_LENGTH. Otherwise, the field is unused.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
rdmerdbc10	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls SQLGetDiagField to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls SQLGetDiagRec or SQLGetDiagField to retrieve additional information.
SQL_ERROR	Function failed. The application calls SQLGetDiagRec or SQLGetDiagField to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from SQLGetDiagRec or SQLGetDiagField. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLColAttribute.

SQLColumns

Rreturns the list of column names in specified tables. The driver returns this information as a result set on the specified `StatementHandle`.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQL_API SQLColumns(
        SQLHSTMT         StatementHandle,
        SQLCHAR         *CatalogName,
        SQLSMALLINT      NameLength1,
        SQLCHAR         *SchemaName,
        SQLSMALLINT      NameLength2,
        SQLCHAR         *TableName,
        SQLSMALLINT      NameLength3,
        SQLCHAR         *ColumnName,
        SQLSMALLINT      NameLength4)
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`CatalogName`	`(input)`	Catalog name. If a driver supports catalogs for some tables but not for others, such as when the driver retrieves data from different DBMSs, an empty string ("") indicates those tables that do not have catalogs. `CatalogName` cannot contain a string search pattern.
`NameLength1`	`(input)`	Length in characters of `*CatalogName`.
`SchemaName`	`(input)`	String search pattern for schema names. If a driver supports schemas for some tables but not for others, such as when the driver retrieves data from different DBMSs, an empty string ("") indicates those tables that do not have schemas.
`NameLength2`	`(input)`	Length in characters of `*SchemaName`.
`TableName`	`(input)`	String search pattern for table names.
`NameLength3`	`(input)`	Length in characters of `*TableName`.
`ColumnName`	`(input)`	String search pattern for column names.
`NameLength4`	`(input)`	Length in characters of `*ColumnName`.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

`SQLColumns` returns the information that pertain to the databases that are currently open. If no database is open on the data source, `SQLColumns` returns an empty result set.

For more information, reference MSDN documentation for SQLColumns.

SQLConnect

Establishes connections to RDM Embedded databases via RDMe ODBC.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQL_API SQLConnect(
        SQLHDBC          ConnectionHandle,
        const SQLCHAR   *ServerName,
        SQLSMALLINT      NameLength1,
        const SQLCHAR   *UserName,
        SQLSMALLINT      NameLength2,
        const SQLCHAR   *Authentication,
        SQLSMALLINT      NameLength3)
```

Arguments

`ConnectionHandle`	(input)	Connection handle.
`ServerName`	(input)	The name of the remote server. For the details, see the Comments section.
`NameLength1`	(input)	Length of `*ServerName` in characters. Specify SQL_NTS if `ServerName` is a null-terminated string.
`UserName`	(input)	This parameter is not used.
`NameLength2`	(input)	Length of `*UserName` in characters. Specify SQL_NTS if `UserName` is a null-terminated string.
`Authentication`	(input)	This parameter is not used.
`NameLength3`	(input)	Length of `*Authentication` in characters. Specify SQL_NTS if `Authentication` is a null-terminated string.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Diagnostics

SQLSTATE	Error	Description
01S02	Option value changed.	The RDMe ODBC Driver does not support the specified value for the ValuePtr argument in `SQLSetConnectAttr`. The specified value has been substituted by a similar value. The function returns SQL_SUCCESS_WITH_INFO.
08001	Unable to connect.	The RDMe ODBC driver was not able to connect to the data source.
08002	Connection already in use.	The specified connection handle is already used for an existing connection.
HY001	Driver memory allocation error	The RDMe ODBC driver failed to allocate memory required to complete the operation.
HY013	Memory management error	The function could not complete the operation due to the out-of-memory condition in the data source.

Comments

The value for the `ServerName` parameter specifies the name of the remote server for the data source. It is the TCP/IP host name of the computer where the remote server is running. The behavior of `SQLConnect` changes depending upon the combination of this value and the connection type, set with SQL_ATTR_RDM_CONN_TYPE attribute using `SQLSetConnectAttr`, as follows.

ServerName	Connection Type	Actual Connection	Actual Server Names	
NULL/empty	Not set	Local	TFS	SQL_ATTR_RDM_TFS_NAME
			Remote	N/A

ServerName	Connection Type	Actual Connection	Actual Server Names	
	Local	Local	TFS	SQL_ATTR_RDM_TFS_NAME
			Remote	N/A
	Remote	Remote	TFS	SQL_ATTR_RDM_TFS_NAME
			Remote	SQL_ATTR_RDM_REMOTE_NAME
Set	Not set	Remote	TFS	SQL_ATTR_RDM_TFS_NAME
			Remote	ServerName
	Local	Local	TFS	ServerName
			Remote	N/A
	Remote	Remote	TFS	SQL_ATTR_RDM_TFS_NAME
			Remote	ServerName

The `UserName`, `Authentication`, `NameLength2` and `NameLength3` parameters are currently not used in RDMe ODBC since RDMe 10.1 does not support user authentication. Values specified for those parameters will be ignored.

For more information, reference MSDN documentation for SQLConnect.

SQLCopyDesc

Copies descriptor information from one descriptor handle to another.

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLCopyDesc(
     SQLHDESC     SourceDescHandle,
     SQLHDESC     TargetDescHandle);
```

Arguments

`SourceDescHandle`	`(input)`	Source descriptor handle.
`TargetDescHandle`	`(input)`	Target descriptor handle. The TargetDescHandle argument can be a handle to an application descriptor or an IPD. TargetDescHandle cannot be set to a handle to an IRD, or **SQLCopyDesc** will return SQLSTATE HY016 (Cannot modify an implementation row descriptor).

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGet-`

	`DiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLCopyDesc.

SQLDescribeCol

Retrieves the basic result data set for a column

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQL_API SQLDescribeCol(
        SQLHSTMT         StatementHandle,
        SQLUSMALLINT     ColumnNumber,
        SQLCHAR         *ColumnName,
        SQLSMALLINT      BufferLength,
        SQLSMALLINT     *NameLengthPtr,
        SQLSMALLINT     *DataTypePtr,
        SQLULEN         *ColumnSizePtr,
        SQLSMALLINT     *DecimalDigitsPtr,
        SQLSMALLINT     *NullablePtr)
```

Arguments

StatementHandle	(input)	Statement handle.
ColumnNumber	(input)	Column number of result data, ordered sequentially in increasing column order, starting at 1.
ColumnName	(output)	Pointer to a null-terminated buffer in which to return the column name. This value is read from the SQL_DESC_NAME field of the IRD. If the column is unnamed or the column name cannot be determined, the driver returns an empty string. If ColumnName is NULL, NameLengthPtr will still return the total number of characters (excluding the null-termination character for character data) available to return in the buffer pointed to by ColumnName.
BufferLength	(input)	Length of the *ColumnName buffer, in characters.
NameLengthPtr	(output)	Pointer to a buffer in which to return the total number of characters (excluding the null termination) available to return in *ColumnName. If the number of characters available to return is greater than or equal to BufferLength, the column name in *ColumnName is truncated to BufferLength minus the length of a null-termination character.
DataTypePtr	(output)	Pointer to a buffer in which to return the SQL data type of the column. This value is read from the SQL_DESC_CONCISE_TYPE field of the IRD. This will be one of the values in SQL Data Types, or a driver-specific SQL data type. If the data type cannot be determined, the driver returns SQL_UNKNOWN_TYPE.
ColumnSizePtr	(output)	Pointer to a buffer in which to return the size (in characters) of the column on the data source. If the column size cannot be determined, the driver returns 0.
DecimalDigitsPtr	(output)	Pointer to a buffer in which to return the number of decimal digits of the column on the data source. If the number of decimal digits cannot be determined or is not applicable, the driver returns 0.
NullablePtr	(output)	Pointer to a buffer in which to return a value that indicates whether the column allows NULL values. This value is read from the SQL_DESC_NULLABLE field of the IRD. The value is one of the following: • SQL_NO_NULLS: The column does not allow NULL values. • SQL_NULLABLE: The column allows NULL values. • SQL_NULLABLE_UNKNOWN: The driver cannot determine if the column allows NULL values.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
rdmerdbc10	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLDescribeCol.

SQLDescribeParam

Retrieves description of a parameter marker

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQL_API SQLDescribeParam(
        SQLHSTMT         StatementHandle,
        SQLUSMALLINT     ParameterNumber,
        SQLSMALLINT     *DataTypePtr,
        SQLULEN         *ParameterSizePtr,
        SQLSMALLINT     *DecimalDigitsPtr,
        SQLSMALLINT     *NullablePtr)
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`ParamNumber`	`(input)`	Specifies the parameter number in the SQL statement text. Parameter numbers are numbered sequentially from left to right, starting with 1, as they appear in the SQL statement.
`DataTypePtr`	`(output)`	Pointer to a buffer in which to return the SQL data type of the parameter.
`ParameterSizePtr`	`(output)`	Pointer to a buffer in which to return the size, in characters, of the column or expression of the corresponding parameter marker as defined by the data source.
`DecimalDigitsPtr`	`(output)`	Pointer to a buffer in which to return the number of decimal digits of the column or expression of the corresponding parameter as defined by the data source.
`NullablePtr`	`(output)`	Pointer to a buffer in which to return a value that indicates whether the parameter allows NULL values.

Required Headers

```
#include "sqlext.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLDescribeParam.

SQLDescribeStmt

Retrieves the type of the SQL statement executed on the handle.

Conformance

Version Introduced: RDM Embedded API extension

Syntax

```
SQLRETURN SQL_API SQLDescribeStmt(
    SQLHSTMT          StatementHandle,
    SQLUSMALLINT     *pStmtType)
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`pStmtType`	`(output)`	The type of the SQL statement executed on the statement handle. Possible values returned through this parameter are described in the **Comments** section below.

Required Headers

```
#include "sqlrext.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGet-`

	`DiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Diagnostics

SQLSTATE	Error	Description
HY009	Invalid use of null pointer	A null pointer was specified for the `pStmtType` argument.
HY010	Function sequence error	This function was called before the statement was prepared.

Comments

The following table describes the types of SQL statements executed and the corresponding values of the statement type returned to this function through `pStmtType`.

SQL statement executed	Value of *pStmtType*
OPEN [DATABASE] <database>	SQL_RDM_STMT_OPEN
OPEN [DATABASE] <database-union>	SQL_RDM_STMT_DBUNION
CLOSE [DATABASE] <database>	SQL_RDM_STMT_CLOSE
SELECT	SQL_RDM_STMT_SELECT
INSERT	SQL_RDM_STMT_INSERT
UPDATE	SQL_RDM_STMT_UPDATE
DELETE	SQL_RDM_STMT_DELETE
BEGIN [WORK]	SQL_RDM_STMT_START
COMMIT [WORK]	SQL_RDM_STMT_COMMIT
SAVEPOINT <trans-id>	SQL_RDM_STMT_SAVEPOINT
RELEASE SAVEPOINT <trans-id>	SQL_RDM_STMT_RELEASE
ROLLBACK [WORK]	SQL_RDM_STMT_ROLLBACK
CREATE PROCEDURE <procedure>	SQL_RDM_STMT_CRPROC
DROP PROCEDURE <procedure>	SQL_RDM_STMT_DRPROC
EXECUTE <procedure>	SQL_RDM_STMT_EXECUTE
SET	SQL_RDM_STMT_SET
SET COLUMN STATS	SQL_RDM_STMT_SETCOLUMN
LOCK TABLE <table>	SQL_RDM_STMT_LOCK
UNLOCK TABLE <table>	SQL_RDM_STMT_UNLOCK
INITIALIZE [DATABASE] <database>	SQL_RDM_STMT_INITDB
CREATE CATALOG FOR <id>	SQL_RDM_STMT_CRCAT
IMPORT	SQL_RDM_STMT_IMPORT
EXPORT	SQL_RDM_STMT_EXPORT
Dynamic DDL statement[1]	SQL_RDM_STMT_DDL

[1]Any of the DDL statements supported by RDM Embedded, such as CREATE DATABASE and CREATE TABLE, returns SQL_RDM_STMT_DDL.

SQLDisconnect

Closes the connection associated with a specified handle

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQL_API SQLDisconnect(
        SQLHDBC          ConnectionHandle)
```

Arguments

`ConnectionHandle (input)` Connection handle.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional

	information is available from `SQLGetDiagRec` or `SQLGet-DiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLDisconnect.

SQLDriverConnect

Establishes connections to RDM Embedded databases via RDMe ODBC. Aceepts more connection options than SQLConnect.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQL_API SQLDriverConnect(
        SQLHDBC          ConnectionHandle,
        SQLHWND          WindowHandle,
        const SQLCHAR   *InConnectionString,
        SQLSMALLINT      StringLength1,
        SQLCHAR         *OutConnectionString,
        SQLSMALLINT      BufferLength,
        SQLSMALLINT     *StringLength2Ptr,
        SQLUSMALLINT     DriverCompletion)
```

Arguments

`ConnectionHandle`	(input)	Connection handle.
`WindowHandle`	(input)	Window handle.
`InConnectionString`	(input)	A full connection string (see the syntax in "Comments"), a partial connection string, or an empty string.
`StringLength1`	(input)	Length of `*InConnectionString`, in characters if the string is Unicode, or bytes if string is ANSI or DBCS.
`OutConnectionString`	(output)	Pointer to a buffer for the completed connection string. Upon successful connection to the target data source, this buffer contains the completed connection string. Applications should allocate at least 1,024 characters for this buffer.
`BufferLength`	(input)	Length of the `*OutConnectionString` buffer, in characters.
`StringLength2Ptr`	(output)	Pointer to a buffer in which to return the total number of characters (excluding the null-termination character) available to return in `*OutConnectionString`. If the number of characters available to return is greater than or equal to BufferLength, the completed connection string in `*OutConnectionString` is truncated to `BufferLength` minus the length of a null-termination character.
`DriverCompletion`	(input)	Flag that indicates whether the Driver Manager or driver must prompt for more connection information

Required Headers

```
#include "sqlext.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.
SQL_NO_DATA	No more data was available. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.

Comments

InConnectionString

RDM Embedded accepts the following connection string keywords.

Keyword	Attribute Value Description
DSN	When the application accesses the RDMe ODBC Driver through Microsoft ODBC Driver Manager (DM), this is the name of an ODBC data source that points to RDM Embedded. The value of this keyword is ignored if the application is linked directly with the RDMe ODBC Driver.
PORT	TCP/IP port number the TFServer is listening on. If PORT is not specified, RDM Embedded will use the default value (21553).

Keyword	Attribute Value Description
TFS[1]	Name of the TFServer. This keyword should be specified instead of DSN when an application attempts to specify our driver with the DRIVER keyword through the Microsoft ODBC DM, in which case the DM will remove the DSN keyword from the connection string. If TFS is specified, RDM Embedded will use its default TFServer name ("localhost").
REMOST_HOST[1]	Name of the remote server. If REMOTE_HOST is not specified, RDM Embedded will use its default remote server name ("localhost").
DATABASE	Name of the database to open upon successful connection to the data source.

For more information, reference MSDN documentation for SQLDriverConnect.

SQLEndTran

Requests a commit or rollback for active transactions

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLEndTran(
     SQLSMALLINT   HandleType,
     SQLHANDLE     Handle,
     SQLSMALLINT   CompletionType)
```

Arguments

`HandleType`	(input)	Handle type identifier. Contains either SQL_HANDLE_ENV (if `Handle` is an environment handle) or SQL_HANDLE_DBC (if `Handle` is a connection handle).
`Handle`	(input)	The handle, of the type indicated by `HandleType`, indicating the scope of the transaction.
`CompletionType`	(input)	One of the following two values: • SQL_COMMIT • SQL_ROLLBACK

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.

SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLEndTran.

SQLExecDirect

Prepares and executes an SQL statement

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQL_API SQLExecDirect(
        SQLHSTMT         StatementHandle,
        const SQLCHAR   *StatementText,
        SQLINTEGER       TextLength)
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`StatementText`	`(input)`	SQL statement to be executed.
`TextLength`	`(input)`	Length of *StatementText in characters.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any out-

	put arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGet-DiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.
SQL_NEED_DATA	More data is needed, such as when parameter data is sent at execution time or additional connection information is required. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information, if any.
SQL_NO_DATA	No more data was available. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.

For more information, reference MSDN documentation for SQLExecDirect.

SQLExecute

Executes a previously prepared SQL statement

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLExecute(
     SQLHSTMT     StatementHandle);
```

Arguments

`StatementHandle` `(input)` Statement handle.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional

	information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.
SQL_NEED_DATA	More data is needed, such as when parameter data is sent at execution time or additional connection information is required. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information, if any.
SQL_NO_DATA	No more data was available. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.

For more information, reference MSDN documentation for SQLExecute.

SQLExtendedTran

Performs a transaction operation

Conformance

Version Introduced: RDM Embedded API extension

Syntax

```
SQLRETURN SQL_API SQLExtendedTran(
    SQLSMALLINT      HandleType,
    SQLHANDLE        Handle,
    SQLSMALLINT      TransType,
    SQLSMALLINT      ReadOnly,
    const SQLCHAR   *TransactionID,
    SQLSMALLINT      StringLengthPtr)
```

Arguments

`HandleType`	(input)	Handle type identifier. Contains either SQL_HANDLE_ENV (if Handle is an environment handle) or SQL_HANDLE_DBC (if Handle is a connection handle).
`Handle`	(input)	The handle, of the type indicated by `HandleType`, indicating the scope of the transaction.
`OperationType`	(input)	One of the following operation types: • SQL_START -Start a transaction • SQL_SAVEPOINT -Create a save point in a transaction • SQL_RELEASE Release a save point in a transaction • SQL_COMMIT -Commit a transaction • SQL_ROLLBACK -Rollback a transaction
`ReadOnly`	(input)	Read-only mode, as follows: • SQL_TRUE Transaction is read-only. • SQL_FALSE Transaction is read-write If `OperationType` is not SQL_START or SQL_COMMIT, this parameter is ignored.
`TransactionID`	(input)	Character string that indicates the unique ID of the transaction operation. If Operation Code is SQL_COMMIT, this parameter is ignored. If `TransactionID` is not specified with SQL_START, an internal system value will be used. TransactionID is required with SQL_SAVEPOINT and SQL_RELEASE.
`StringLengthPtr`	(input)	The length of `TransactionID` in characters. SQL_NTS can be used to indicate that `TransactionID` is null-terminated.

Required Headers

```
#include "sqlrext.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Diagnostics

SQLSTATE	Error	Description
01000	General warning	General error returned from the RDM Embedded ODBC driver. The function returns SQL_SUCCESS_WITH_INFO.
08003	Connection not open	The RDM Embedded ODBC driver is not connected to a data source.
25000	Transaction already active	SQL_START is specified for `OperationType` and a transaction is already active on the connection associated with `Handle`.
25001	Read-only transaction already active	SQL_START, SQL_SAVEPOINT or SQL_RELEASE is specified for `OperationType` and a read-only transaction is active on the connection associated with `Handle`.
25S01	Transaction state unknown	When `HandleType` is SQL_HANDLE_ENV and one or more of the connections in Handle failed to complete the transaction.
HY001	Memory allocation error	Either the ODBC driver or the SQL engine failed to allocate sufficient memory to complete the required operation.
HY010	Function sequence error	This function was called after `SQLExecute` or `SQLExecDirect` but before all the parameters were bound.
HY012	Invalid transaction operation code	The value in `OperationType` is not one of the following: • SQL_START • SQL_SAVEPOINT • SQL_RELEASE • SQL_COMMIT • SQL_ROLLBACK
HY013	Memory management error	The RDM Embedded SQL engine failed to allocate sufficient memory to complete the required operation.
HY092	Invalid attribute/option identifier	The value in `HandleType` is not one of the following values: • SQL_HANDLE_ENV • SQL_HANDLE_DBC

Comments

`SQLExtendedTran` allows an RDM Embedded ODBC application to start a transaction, create a save point inside the current transaction, and roll back to the last save point, as well as committing and rolling back a transaction.

Starting a transaction

`SQLExtendedTran` can be called to start a regular ("read-write") transaction or a "read-only" transaction. The value specified for the `ReadOnly` parameter determines the type of transaction to start. To start a read-only transaction, specify SQL_TRUE for the `ReadOnly` parameter.

An application can optionally specify the unique ID of the transaction by using

	the `TransactionID` parameter. If NULL or an empty string is specified for the `TransactionID` parameter, an internal system ID will be used
Creating a save point	`SQLExtendedTran` can be used to create a save point in the current read-write transaction. An application must specify a unique ID of the save point using the `TransactionID` parameter. An application can roll back the database changes that are made after the save point by calling SQLExtendedTran with SQL_ROLLBACK and the ID that matches the one associated with the save point.
Releasing a save point	An application can manually release, or discard, any of the save points it created by calling `SQLExtendedTran` specifying SQL_RELEASE for `OperationType` and the ID associated with the save point. A call to `SQLEndTran` to either commit or abort the current transaction will automatically release all save points. A call to `SQLExtendedTran` to roll back to a save point will automatically release all the save points created after that save point.
Rolling back to a save point	An application can roll back a transaction to a particular save point by calling `SQLExtendedTran` specifying SQL_ROLLBACK for `OperationType` and the ID associated with the save point. All save points created after the specified save point will be released. If the specified ID is not associated with any existing save point, the entire transaction will be aborted.
Committing a transaction	An application can commit all changes in a transaction by calling `SQLEndTran` or `SQLExtendedTran` specifying SQL_COMMIT. All save points inside the transaction will be released. If no transaction is active, SQL_SUCCESS will be returned.
Aborting a transaction	An application can abort an entire transaction by calling `SQLEndTran` or `SQLExtendedTran` specifying SQL_ROLLBACK. For `SQLExtendedTran`, specify NULL or the ID of the transaction itself for `TransactionID`. All save points inside the transaction will be released. If no transaction is active, SQL_SUCCESS will be returned.

SQLFetch

Fetches data from a bound column to an application variable

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLFetch(
     SQLHSTMT     StatementHandle);
```

Arguments

`StatementHandle` `(input)` Statement handle.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional

	information is available from `SQLGetDiagRec` or `SQLGet-DiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLFetch.

SQLFetchScroll

Fetches the specified rowset of data from the result set and returns data for all bound columns.

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLFetchScroll(
      SQLHSTMT      StatementHandle,
      SQLSMALLINT   FetchOrientation,
      SQLLEN        FetchOffset);
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`FetchOrientation`	`(input)`	Type of fetch: • SQL_FETCH_NEXT • SQL_FETCH_PRIOR • SQL_FETCH_FIRST • SQL_FETCH_LAST • SQL_FETCH_ABSOLUTE • SQL_FETCH_RELATIVE • SQL_FETCH_BOOKMARK
`FetchOffset`	`(input)`	Number of the row to fetch. The interpretation of this argument depends on the value of the `FetchOrientation` argument.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.
SQL_NO_DATA	No more data was available. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.

Comments

Cursor Support

`SQLFetchScroll` in RDMe ODBC currently only supports the forward-only cursor. Therefore, the only accepted value for the `FetchOrientation` parameter is SQL_FETCH_NEXT. Specifying any other fetch type for `FetchOrientation` will result in the "Fetch type out of range" (SQLSTATE HY106) error. The value specified for the `FetchOffset` parameter is ignored.

For more information, reference MSDN documentation for SQLFetchScroll.

SQLForeignKeys

`SQLForeignKeys` can return:

- A list of foreign keys in the specified table (columns in the specified table that refer to primary keys in other tables).
- A list of foreign keys in other tables that refer to the primary key in the specified table.

The driver returns each list as a result set on the specified statement.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLForeignKeys(
     SQLHSTMT       StatementHandle,
     SQLCHAR *      PKCatalogName,
     SQLSMALLINT    NameLength1,
     SQLCHAR *      PKSchemaName,
     SQLSMALLINT    NameLength2,
     SQLCHAR *      PKTableName,
     SQLSMALLINT    NameLength3,
     SQLCHAR *      FKCatalogName,
     SQLSMALLINT    NameLength4,
     SQLCHAR *      FKSchemaName,
     SQLSMALLINT    NameLength5,
     SQLCHAR *      FKTableName,
     SQLSMALLINT    NameLength6);
```

Arguments

StatementHandle	(input)	Statement handle.
PKCatalogName	(input)	Primary key table catalog name. PKCatalogName cannot contain a string search pattern.
NameLength1	(input)	Length of *PKCatalogName, in characters.
PKSchemaName	(input)	Primary key table schema name. PKSchemaName cannot contain a string search pattern.
NameLength2	(input)	Length of *PKSchemaName, in characters.
PKTableName	(input)	Primary key table name. PKTableName cannot contain a string search pattern.
NameLength3	(input)	Length of *PKTableName, in characters.
FKCatalogName	(input)	Foreign key table catalog name. FKCatalogName cannot contain a string search pattern.
NameLength4	(input)	Length of *FKCatalogName, in characters.
FKSchemaName	(input)	Foreign key table schema name. FKSchemaName cannot contain a string search pattern.
NameLength5	(input)	Length of *FKSchemaName, in characters.
FKTableName	(input)	Foreign key table name. FKTableName cannot contain a string search pattern.
NameLength6	(input)	Length of *FKTableName, in characters.

Required Headers

```
#include "sqlext.h"
```

Libraries

Library Name	Description
rdmerdbc10	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls SQLGetDiagField to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls SQLGetDiagRec or SQLGetDiagField to retrieve additional information.
SQL_ERROR	Function failed. The application calls SQLGetDiagRec or SQLGetDiagField to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from SQLGetDiagRec or SQLGet-

	`DiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

`SQLForeignKeys` returns the information that pertains to the databases that are currently open. If no database is open on the data source, `SQLForeignKeys` returns an empty result set.

For more information, reference MSDN documentation for SQLForeignKeys.

SQLFreeHandle

Frees resources associated with a specific environment, connection, statement, or descriptor handle.

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLFreeHandle(
     SQLSMALLINT   HandleType,
     SQLHANDLE     Handle);
```

Arguments

`HandleType`	`(input)`	The type of handle to be freed by `SQLFreeHandle`. Must be one of the following values: • SQL_HANDLE_DBC • SQL_HANDLE_DESC • SQL_HANDLE_ENV • SQL_HANDLE_STMT If `HandleType` is not one of these values, `SQLFreeHandle` returns SQL_INVALID_HANDLE.
`Handle`	`(input)`	The handle to be freed.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGet-`

	`DiagField` to retrieve additional information from the header record.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGet-DiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGet-DiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLFreeHandle.

SQLFreeStmt

Stops processing associated with a specific statement, closes any open cursors associated with the statement, discards pending results, or, optionally, frees all resources associated with the statement handle.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLFreeStmt(
     SQLHSTMT       StatementHandle,
     SQLUSMALLINT   Option);
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`Option`	`(input)`	One of the following options:

SQL_ CLOSE: Closes the cursor associated with `StatementHandle` (if one was defined) and discards all pending results. The application can reopen this cursor later by executing a SELECT statement again with the same or different parameter values. If no cursor is open, this option has no effect for the application. SQLCloseCursor can also be called to close a cursor. For more information, see Closing the Cursor.

SQL_DROP: This option is deprecated. A call to `SQLFreeStmt` with an Option of SQL_DROP is mapped in the Driver Manager to `SQLFreeHandle`.

SQL_UNBIND: Sets the SQL_DESC_COUNT field of the ARD to 0, releasing all column buffers bound by `SQLBindCol` for the given `StatementHandle`. This does not unbind the bookmark column; to do that, the SQL_DESC_DATA_PTR field of the ARD for the bookmark column is set to NULL. Notice that if this operation is performed on an explicitly allocated descriptor that is shared by more than one statement, the operation will affect the bindings of all statements that share the descriptor.

SQL_RESET_PARAMS: Sets the SQL_DESC_COUNT field of the APD to 0, releasing all parameter buffers set by `SQLBindParameter` for the given `StatementHandle`. If this operation is performed on an explicitly allocated descriptor that is shared by more than one statement, this operation will affect the bindings of all the statements that share the descriptor.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLFreeStmt.

SQLGetConnectAttr

Returns the current setting of a connection attribute.

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLGetConnectAttr(
     SQLHDBC         ConnectionHandle,
     SQLINTEGER      Attribute,
     SQLPOINTER      ValuePtr,
     SQLINTEGER      BufferLength,
     SQLINTEGER *    StringLengthPtr);
```

Arguments

`ConnectionHandle`	`(input)`	Connection handle.
`Attribute`	`(input)`	Attribute to retrieve.
`ValuePtr`	`(output)`	A pointer to memory in which to return the current value of the attribute specified by `Attribute`.
`BufferLength`	`(input)`	If Attribute is an ODBC-defined attribute and ValuePtr points to a character string or a binary buffer, this argument should be the length of `*ValuePtr`. If Attribute is an ODBC-defined attribute and `*ValuePtr` is an integer, `BufferLength` is ignored.
`StringLengthPtr`	`(output)`	A pointer to a buffer in which to return the total number of bytes (excluding the null-termination character) available to return in `*ValuePtr`. If `*ValuePtr` is a null pointer, no length is returned. If the attribute value is a character string and the number of bytes available to return is greater than `BufferLength` minus the length of the null-termination character, the data in `*ValuePtr` is truncated to `BufferLength` minus the length of the null-termination character and is null-terminated by the driver.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

The attributes accepted by `SQLGetConnectAttr` are listed under `SQLSetConnectAttr`.

For more information, reference MSDN documentation for SQLGetConnectAttr.

SQLGetCursorName

Retrieves the cursor name of an SQL statement handle

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLGetCursorName(
     SQLHSTMT         StatementHandle,
     SQLCHAR *        CursorName,
     SQLSMALLINT      BufferLength,
     SQLSMALLINT *    NameLengthPtr);
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`CursorName`	`(output)`	Pointer to a buffer in which to return the cursor name.
`BufferLenght`	`(input)`	Length of `*CursorName`, in characters.
`NameLengthPrt`	`(output)`	Pointer to memory in which to return the total number of characters (excluding the null-termination character) available to return in `*CursorName`.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField`

	to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLGetCursorName.

SQLGetData

Retrieves data for a single column in the result set. It can be called multiple times to retrieve variable-length data in parts.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLGetData(
      SQLHSTMT       StatementHandle,
      SQLUSMALLINT   Col_or_Param_Num,
      SQLSMALLINT    TargetType,
      SQLPOINTER     TargetValuePtr,
      SQLLEN         BufferLength,
      SQLLEN *       StrLen_or_IndPtr);
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`Col_or_Param_Num`	`(input)`	For retrieving column data, it is the number of the column for which to return data. Result set columns are numbered in increasing column order starting at 1. The bookmark column is column number 0; this can be specified only if bookmarks are enabled. For retrieving parameter data, it is the ordinal of the parameter, which starts at 1.
`TargetType`	`(input)`	The type identifier of the C data type of the `*TargetValuePtr` buffer. For a list of valid C data types and type identifiers
`TargetValuePrt`	`(output)`	Pointer to the buffer in which to return the data.
`BufferLength`	`(input)`	Length of the `*TargetValuePtr` buffer in bytes.
`StrLen_or_IndPtr`	`(output)`	Pointer to the buffer in which to return the length or indicator value. If this is a null pointer, no length or indicator value is returned. This returns an error when the data being fetched is NULL.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLGetData.

SQLGetDescField

Returns the current setting or value of a single field of a descriptor record.

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLGetDescField(
     SQLHDESC         DescriptorHandle,
     SQLSMALLINT      RecNumber,
     SQLSMALLINT      FieldIdentifier,
     SQLPOINTER       ValuePtr,
     SQLINTEGER       BufferLength,
     SQLINTEGER *     StringLengthPtr);
```

Arguments

`DescriptorHandle`	`(input)`	Descriptor handle.
`RecNumber`	`(input)`	Indicates the descriptor record from which the application seeks information. Descriptor records are numbered from 0, with record number 0 being the bookmark record. If the `FieldIdentifier` argument indicates a header field, `RecNumber` is ignored. If `RecNumber` is less than or equal to SQL_DESC_COUNT but the row does not contain data for a column or parameter, a call to `SQLGetDescField` will return the default values of the fields.
`FieldIdentifier`	`(input)`	Indicates the field of the descriptor whose value is to be returned.
`ValuePtr`	`(output)`	Pointer to a buffer in which to return the descriptor information. The data type depends on the value of `FieldIdentifier`. If `ValuePtr` is NULL, `StringLengthPtr` will still return the total number of bytes (excluding the null-termination character for character data) available to return in the buffer pointed to by `ValuePtr`.
`BufferLength`	`(input)`	If `FieldIdentifier` is an ODBC-defined field and `ValuePtr` points to a character string or a binary buffer, this argument should be the length of `*ValuePtr`. If `FieldIdentifier` is an ODBC-defined field and `*ValuePtr` is an integer, `BufferLength` is ignored.
`StringLengthPtr`	`(output)`	Pointer to the buffer in which to return the total number of bytes (excluding the number of bytes required for the null-termination character) available to return in `*ValuePtr`.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLGetDescField.

SQLGetDescRec

Returns the current settings or values of multiple fields of a descriptor record. The fields returned describe the name, data type, and storage of column or parameter data.

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLGetDescRec(
      SQLHDESC        DescriptorHandle,
      SQLSMALLINT     RecNumber,
      SQLCHAR *       Name,
      SQLSMALLINT     BufferLength,
      SQLSMALLINT *   StringLengthPtr,
      SQLSMALLINT *   TypePtr,
      SQLSMALLINT *   SubTypePtr,
      SQLLEN *        LengthPtr,
      SQLSMALLINT *   PrecisionPtr,
      SQLSMALLINT *   ScalePtr,
      SQLSMALLINT *   NullablePtr);
```

Arguments

DescriptorHandle	(input)	Descriptor handle.
RecNumber	(input)	Indicates the descriptor record from which the application seeks information. Descriptor records are numbered from 1, with record number 0 being the bookmark record. The RecNumber argument must be less than or equal to the value of SQL_DESC_COUNT. If RecNumber is less than or equal to SQL_DESC_COUNT but the row does not contain data for a column or parameter, a call to SQLGetDescRec will return the default values of the fields.
Name	(output)	A pointer to a buffer in which to return the SQL_DESC_NAME field for the descriptor record. If Name is NULL, StringLengthPtr will still return the total number of characters (excluding the null-termination character for character data) available to return in the buffer pointed to by Name.
BufferLength	(input)	Length of the *Name buffer, in characters.
StringLengthPtr	(output)	A pointer to a buffer in which to return the number of characters of data available to return in the *Name buffer, excluding the null-termination character. If the number of characters was greater than or equal to BufferLength, the data in *Name is truncated to BufferLength minus the length of a null-termination character, and is null-terminated by the driver.
TypePtr	(output)	A pointer to a buffer in which to return the value of the SQL_DESC_TYPE field for the descriptor record.
SubTypePtr	(output)	For records whose type is SQL_DATETIME or SQL_INTERVAL, this is a pointer to a buffer in which to return the value of the SQL_DESC_DATETIME_INTERVAL_CODE field.
LengthPtr	(output)	A pointer to a buffer in which to return the value of the SQL_DESC_OCTET_LENGTH field for the descriptor record.
PrecisionPtr	(output)	A pointer to a buffer in which to return the value of the SQL_DESC_PRECISION field for the descriptor record.
ScalePtr	(output)	A pointer to a buffer in which to return the value of the SQL_DESC_SCALE field for the descriptor record.
NullablePtr	(output)	A pointer to a buffer in which to return the value of the SQL_DESC_NULLABLE field for the descriptor record.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
rdmerdbc10	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.
SQL_NO_DATA	No more data was available. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.

SQL_NO_DATA is returned if `RecNumber` is greater than the current number of descriptor records.

SQL_NO_DATA is returned if `DescriptorHandle` is an IRD handle and the statement is in the prepared or executed state but there was no open cursor associated with it.

For more information, reference MSDN documentation for SQLGetDescRec.

SQLGetDiagField

Retrieves current field value of a status record

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLGetDiagField(
     SQLSMALLINT     HandleType,
     SQLHANDLE       Handle,
     SQLSMALLINT     RecNumber,
     SQLSMALLINT     DiagIdentifier,
     SQLPOINTER      DiagInfoPtr,
     SQLSMALLINT     BufferLength,
     SQLSMALLINT *   StringLengthPtr)
```

Arguments

`HandleType`	`(input)`	A handle type identifier that describes the type of handle for which diagnostics are required. Must be one of the following: • SQL_HANDLE_ENV • SQL_HANDLE_DBC • SQL_HANDLE_STMT • SQL_HANDLE_DESC
`Handle`	`(input)`	A handle for the diagnostic data structure, of the type indicated by `HandleType`.
`RecNumber`	`(input)`	Indicates the status record from which the application seeks information. Status records are numbered from 1. If the `DiagIdentifier` argument indicates any field of the diagnostics header, `RecNumber` is ignored. If not, it should be more than 0.
`DiagIdentifier`	`(input)`	Indicates the field of the diagnostic whose value is to be returned.
`DiagInfoPtr`	`(output)`	Pointer to a buffer in which to return the diagnostic information. The data type depends on the value of `DiagIdentifier`. If `DiagInfoPtr` is NULL, `StringLengthPtr` will still return the total number of bytes (excluding the null-termination character for character data) available to return in the buffer pointed to by `DiagInfoPtr`.
`BufferLength`	`(input)`	If `DiagIdentifier` is an ODBC-defined diagnostic and `DiagInfoPtr` points to a character string or a binary buffer, this argument should be the length of `*DiagInfoPtr`. If `DiagIdentifier` is an ODBC-defined field and `*DiagInfoPtr` is an integer, `BufferLength` is ignored.
`StringLengthPtr`	`(output)`	Pointer to a buffer in which to return the total number of bytes (excluding the number of bytes required for the null-termination character) available to return in `*DiagInfoPtr`, for character data. If the number of bytes available to return is greater than or equal to `BufferLength`, the text in `*DiagInfoPtr` is truncated to `BufferLength` minus the length of a null-termination character.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warn-

	ing). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLGetDiagField.

SQLGetDiagRec

Retrieves current values of several common fields of a status record

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLGetDiagRec(
     SQLSMALLINT     HandleType,
     SQLHANDLE       Handle,
     SQLSMALLINT     RecNumber,
     SQLCHAR *       SQLState,
     SQLINTEGER *    NativeErrorPtr,
     SQLCHAR *       MessageText,
     SQLSMALLINT     BufferLength,
     SQLSMALLINT *   TextLengthPtr)
```

Arguments

`HandleType`	`(input)`	A handle type identifier that describes the type of handle for which diagnostics are required. Must be one of the following: • SQL_HANDLE_ENV • SQL_HANDLE_DBC • SQL_HANDLE_STMT • SQL_HANDLE_DESC
`Handle`	`(input)`	A handle for the diagnostic data structure, of the type indicated by `HandleType`.
`RecNumber`	`(input)`	Indicates the status record from which the application seeks information. Status records are numbered from 1.
`SQLState`	`(output)`	Pointer to a buffer in which to return a five-character SQLSTATE code (and terminating NULL) for the diagnostic record `RecNumber`. The first two characters indicate the class; the next three indicate the subclass. This information is contained in the SQL_DIAG_SQLSTATE diagnostic field.
`NativeErrorPtr`	`(output)`	Pointer to a buffer in which to return the native error code, specific to the data source. This information is contained in the SQL_DIAG_NATIVE diagnostic field.
`MessageText`	`(output)`	Pointer to a buffer in which to return the diagnostic message text string. This information is contained in the SQL_DIAG_MESSAGE_TEXT diagnostic field.
`BufferLength`	`(input)`	Length of the `*MessageText` buffer in characters. There is no maximum length of the diagnostic message text.
`TextLengthPtr`	`(output)`	Pointer to a buffer in which to return the total number of characters (excluding the number of characters required for the null-termination character) available to return in `*MessageText`. If the number of characters available to return is greater than `BufferLength`, the diagnostic message text in `*MessageText` is truncated to `BufferLength` minus the length of a null-termination character.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField`

	to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLGetDiagRec.

SQLGetEnvAttr

Returns the current setting of an environment attribute.

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLGetEnvAttr(
     SQLHENV         EnvironmentHandle,
     SQLINTEGER      Attribute,
     SQLPOINTER      ValuePtr,
     SQLINTEGER      BufferLength,
     SQLINTEGER *    StringLengthPtr)
```

Arguments

`EnvironmentHandle`	`(input)`	Environment handle.
`Attribute`	`(input)`	Attribute to retrieve.
`ValuePtr`	`(output)`	Pointer to a buffer in which to return the current value of the attribute specified by Attribute. If `ValuePtr` is NULL, `StringLengthPtr` will still return the total number of bytes (excluding the null-termination character for character data) available to return in the buffer pointed to by `ValuePtr`.
`BufferLength`	`(input)`	If `ValuePtr` points to a character string, this argument should be the length of `*ValuePtr`. If `*ValuePtr` is an integer, `BufferLength` is ignored. If the attribute value is not a character string, `BufferLength` is unused.
`StringLengthPtr`	`(output)`	A pointer to a buffer in which to return the total number of bytes (excluding the null-termination character) available to return in `*ValuePtr`. If `ValuePtr` is a null pointer, no length is returned. If the attribute value is a character string and the number of bytes available to return is greater than or equal to `BufferLength`, the data in `*ValuePtr` is truncated to `BufferLength` minus the length of a null-termination character and is null-terminated by the driver.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

The attributes accepted by `SQLGetEnvAttr` are listed under `SQLSetEnvAttr`.

For more information, reference MSDN documentation for SQLGetEnvAttr.

SQLGetFunctions

Returns information about whether a driver supports a specific ODBC function.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLGetFunctions(
     SQLHDBC           ConnectionHandle,
     SQLUSMALLINT      FunctionId,
     SQLUSMALLINT *    SupportedPtr)
```

Arguments

`ConnectionHandle`	`(input)`	Connection handle.
`FunctionId`	`(input)`	A `#define` value that identifies the ODBC function of interest; SQL_API_ODBC3_ALL_FUNCTIONS or SQL_API_ALL_FUNCTIONS.
`SupportedPtr`	`(output)`	If `FunctionId` identifies a single ODBC function, `SupportedPtr` points to a single SQLUSMALLINT value that is SQL_TRUE if the specified function is supported by the driver, and SQL_FALSE if it is not supported.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField`

	to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLGetFunctions.

SQLGetInfo

Returns general information about the driver and data source associated with a connection.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLGetInfo(
     SQLHDBC          ConnectionHandle,
     SQLUSMALLINT     InfoType,
     SQLPOINTER       InfoValuePtr,
     SQLSMALLINT      BufferLength,
     SQLSMALLINT *    StringLengthPtr)
```

Arguments

`ConnectionHandle`	`(input)`	Connection handle.
`InfoType`	`(input)`	Type of information.
`InfoValuePtr`	`(output)`	Pointer to a buffer in which to return the information. Depending on the `InfoType` requested, the information returned will be one of the following: a null-terminated character string, an SQLUSMALLINT value, an SQLUINTEGER bitmask, an SQLUINTEGER flag, or a SQLUINTEGER binary value. If the `InfoType` argument is SQL_DRIVER_HDESC or SQL_DRIVER_HSTMT, the `InfoValuePtr` argument is both input and output. (See the SQL_DRIVER_HDESC or SQL_DRIVER_HSTMT descriptors later in this function description for more information.) If `InfoValuePtr` is NULL, `StringLengthPtr` will still return the total number of bytes (excluding the null-termination character for character data) available to return in the buffer pointed to by `InfoValuePtr`.
`BufferLength`	`(input)`	Length of the `*InfoValuePtr` buffer. If the value in `*InfoValuePtr` is not a character string or if `InfoValuePtr` is a null pointer,the `BufferLength` argument is ignored. The driver assumes that the size of `*InfoValuePtr` is SQLUSMALLINT or SQLUINTEGER, based on the `InfoType`.
`StringLengthPtr`	`(output)`	Pointer to a buffer in which to return the total number of bytes (excluding the null-termination character for character data) available to return in `*InfoValuePtr`. For character data, if the number of bytes available to return is greater than or equal to `BufferLength`, the information in `*InfoValuePtr` is truncated to `BufferLength` bytes minus the length of a null-termination character and is null-terminated by the driver. For all other types of data, the value of `BufferLength` is ignored and the driver assumes the size of `*InfoValuePtr` is SQLUSMALLINT or SQLUINTEGER, depending on the InfoType.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
rdmerdbc10	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls SQLGetDiagField to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls SQLGetDiagRec or SQLGetDiagField to retrieve additional information.
SQL_ERROR	Function failed. The application calls SQLGetDiagRec or SQLGetDiagField to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from SQLGetDiagRec or SQLGetDiagField. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLGetInfo.

SQLGetStmtAttr

Returns the current setting of a statement attribute.

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLGetStmtAttr(
     SQLHSTMT        StatementHandle,
     SQLINTEGER      Attribute,
     SQLPOINTER      ValuePtr,
     SQLINTEGER      BufferLength,
     SQLINTEGER *    StringLengthPtr)
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`Attribute`	`(input)`	Attribute to retrieve.
`ValuePtr`	`(output)`	Pointer to a buffer in which to return the value of the attribute specified in Attribute. If `ValuePtr` is NULL, `StringLengthPtr` will still return the total number of bytes (excluding the null-termination character for character data) available to return in the buffer pointed to by `ValuePtr`.
`BufferLength`	`(input)`	If `Attribute` is an ODBC-defined attribute and `ValuePtr` points to a character string or a binary buffer, this argument should be the length of `*ValuePtr`. If `Attribute` is an ODBC-defined attribute and `*ValuePtr` is an integer, `BufferLength` is ignored
`StringLengthPtr`	`(output)`	A pointer to a buffer in which to return the total number of bytes (excluding the null-termination character) available to return in `*ValuePtr`. If `ValuePtr` is a null pointer, no length is returned. If the attribute value is a character string, and the number of bytes available to return is greater than or equal to `BufferLength`; the data in `*ValuePtr` is truncated to `BufferLength` minus the length of a null-termination character and is null-terminated by the driver.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

The attributes accepted by `SQLGetStmtAttr` are listed under `SQLSetStmtAttr`.

For more information, reference MSDN documentation for SQLGetStmtAttr.

SQLGetTypeInfo

Returns information about data types supported by the data source.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLGetTypeInfo(
     SQLHSTMT      StatementHandle,
     SQLSMALLINT   DataType)
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`DataType`	`(input)`	The SQL data type.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.

SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGet-DiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLGetTypeInfo.

SQLMoreResults

Determines whether more results are available on a statement containing SELECT, UPDATE, INSERT, or DELETE statements and, if so, initializes processing for those results.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLMoreResults(
     SQLHSTMT     StatementHandle)
```

Arguments

`StatementHandle` `(input)` Statement handle.

Required Headers

```
#include "sqlext.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or

	descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.
SQL_NO_DATA	No more data was available. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.

For more information, reference MSDN documentation for SQLMoreResults.

SQLNativeSql

Returns the SQL string as modified by the driver. SQLNativeSql does not execute the SQL statement.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLNativeSql(
     SQLHDBC           ConnectionHandle,
     const SQLCHAR *   InStatementText,
     SQLINTEGER        TextLength1,
     SQLCHAR       *   OutStatementText,
     SQLINTEGER        BufferLength,
     SQLINTEGER    *   TextLength2Ptr)
```

Arguments

`ConnectionHandle`	`(input)`	Connection handle.
`InStatementText`	`(input)`	SQL text string to be translated.
`TextLength1`	`(input)`	Length in characters of `*InStatementText` text string.
`OutStatementText`	`(output)`	Pointer to a buffer in which to return the translated SQL string. If `OutStatementText` is NULL, `TextLength2Ptr` will still return the total number of characters (excluding the null-termination character for character data) available to return in the buffer pointed to by `OutStatementText`.
`BufferLength`	`(input)`	Number of characters in the `*OutStatementText` buffer.
`TextLength2`	`(output)`	Pointer to a buffer in which to return the total number of characters (excluding null-termination) available to return in `*OutStatementText`. If the number of characters available to return is greater than or equal to `BufferLength`, the translated SQL string in `*OutStatementText` is truncated to `BufferLength` minus the length of a null-termination character.

Required Headers

```
#include "sqlext.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLNativeSql.

SQLNumParams

Determines the number of parameters in a prepared statement

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLNumParams(
     SQLHSTMT        StatementHandle,
     SQLSMALLINT *   ParameterCountPtr)
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`ParameterCountPtr`	`(output)`	Pointer to a buffer in which to return the number of parameters in the statement.

Required Headers

```
#include "sqlext.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.

SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGet-DiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLNumParams.

SQLNumResultCols

Returns the number of columns in a result set.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLNumResultCols(
     SQLHSTMT        StatementHandle,
     SQLSMALLINT *   ColumnCountPtr)
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`ColumnCountPtr`	`(output)`	Pointer to a buffer in which to return the number of columns in the result set.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.

SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGet-DiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLNumResultsCol.

SQLParamData

SQLParamData is used together with SQLPutData to supply parameter data at statement execution time, and with SQLGetData to retrieve streamed output parameter data.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLParamData(
     SQLHSTMT       StatementHandle,
     SQLPOINTER *   ValuePtrPtr)
```

Arguments

StatementHandle	(input)	Statement handle.
ValuePtrPtr	(output)	Pointer to a buffer in which to return the address of the ParameterValuePtr buffer specified in SQLBindParameter (for parameter data) or the address of the TargetValuePtr buffer specified in SQLBindCol (for column data), as contained in the SQL_DESC_DATA_PTR descriptor record field.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
rdmerdbc10	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls SQLGetDiagField to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls SQLGetDiagRec or SQLGetDiagField to retrieve additional information.

SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLParamData.

SQLPrepare

Prepares an SQL string for execution.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLPrepare(
     SQLHSTMT      StatementHandle,
     SQLCHAR *     StatementText,
     SQLINTEGER    TextLength);
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`StatementText`	`(input)`	SQL text string.
`TextLength`	`(input)`	Length of `*StatementText` in characters.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any out-

	put arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGet-DiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLPrepare.

SQLPrimaryKeys

Returns the column names that make up the primary key for a table. The driver returns the information as a result set. This function does not support returning primary keys from multiple tables in a single call.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLPrimaryKeys(
     SQLHSTMT       StatementHandle,
     SQLCHAR *      CatalogName,
     SQLSMALLINT    NameLength1,
     SQLCHAR *      SchemaName,
     SQLSMALLINT    NameLength2,
     SQLCHAR *      TableName,
     SQLSMALLINT    NameLength3);
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`CatalogName`	`(input)`	Catalog name. `CatalogName` cannot contain a string search pattern.
`NameLength1`	`(input)`	Length in characters of `*CatalogName`.
`SchemaName`	`(input)`	Schema name. `SchemaName` cannot contain a string search pattern.
`NameLength2`	`(input)`	Length in characters of `*SchemaName`.
`TableName`	`(input)`	Table name. This argument cannot be a null pointer. `TableName` cannot contain a string search pattern.
`NameLength3`	`(input)`	Length in characters of `*TableName`.

Required Headers

```
#include "sqlext.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

`SQLPrimaryKeys` returns the information that pertains to the databases that are currently open. If no database is open on the data source, `SQLPrimaryKeys` returns an empty result set.

For more information, reference MSDN documentation for SQLPrimaryKeys.

SQLProcedureColumns

Returns the parameter and column information about the specified procedures.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLProcedureColumns(
     SQLHSTMT      StatementHandle,
     SQLCHAR *     CatalogName,
     SQLSMALLINT   NameLength1,
     SQLCHAR *     SchemaName,
     SQLSMALLINT   NameLength2,
     SQLCHAR *     ProcName,
     SQLSMALLINT   NameLength3,
     SQLCHAR *     ColumnName,
     SQLSMALLINT   NameLength4);
```

Arguments

`StatementHandle`	(input)	Statement handle.
`CatalogName`	(input)	Procedure catalog name. `CatalogName` cannot contain a string search pattern.
`NameLength1`	(input)	Length in characters of `*CatalogName`.
`SchemaName`	(input)	String search pattern for procedure schema names.
`NameLength2`	(input)	Length in characters of `*SchemaName`.
`ProcName`	(input)	String search pattern for procedure names.
`NameLength3`	(input)	Length in characters of `*ProcName`.
`ColumnName`	(input)	String search pattern for column names.
`NameLength4`	(input)	Length in characters of `*ColumnName`.

Required Headers

```
#include "sqlext.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

`SQLProcedureColumns` returns the information that pertains to the databases that are currently open. If no database is open on the data source, `SQLProcedureColumns` returns an empty result set.

For more information, reference MSDN documentation for SQLProcedureColumns.

SQLProcedures

Returns the list of procedure names stored in a specific data source. Procedure is a generic term used to describe an executable object, or a named entity that can be invoked using input and output parameters.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLProcedures(
     SQLHSTMT       StatementHandle,
     SQLCHAR *      CatalogName,
     SQLSMALLINT    NameLength1,
     SQLCHAR *      SchemaName,
     SQLSMALLINT    NameLength2,
     SQLCHAR *      ProcName,
     SQLSMALLINT    NameLength3);
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`CatalogName`	`(input)`	Procedure catalog.
`NameLength1`	`(input)`	Length in characters of `*CatalogName`.
`SchemaName`	`(input)`	String search pattern for procedure schema names.
`NameLength2`	`(input)`	Length in characters of `*SchemaName`.
`ProcName`	`(input)`	String search pattern for procedure names.
`NameLength3`	`(input)`	Length in characters of `*ProcName`.

Required Headers

```
#include "sqlext.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

`SQLProcedures` returns the information that pertains to the databases that are currently open. If no database is open on the data source, `SQLProcedures` returns an empty result set.

For more information, reference MSDN documentation for SQLProcedures.

SQLPutData

Allows an application to send data for a parameter or column to the driver at statement execution time. This function can be used to send character or binary data values in parts to a column with a character, binary, or data source-specific data type (for example, parameters of the SQL_LONGVARBINARY or SQL_LONGVARCHAR types).

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLPutData(
      SQLHSTMT     StatementHandle,
      SQLPOINTER   DataPtr,
      SQLLEN       StrLen_or_Ind);
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`DataPtr`	`(input)`	Pointer to a buffer containing the actual data for the parameter or column. The data must be in the C data type specified in the `ValueType` argument of `SQLBind-Parameter` (for parameter data) or the `TargetType` argument of `SQLBindCol` (for column data).
`StrLen_or_Ind`	`(input)`	Length of `*DataPtr`. Specifies the amount of data sent in a call to `SQLPutData`. The amount of data can vary with each call for a given parameter or column.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description

SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

`SQLPutData` can only be called to insert or update column values. Calling `SQLPutData` on any other type of SQL statement returns a "data-at-exec params only allowed with INSERT VALUES/UPDATE" error (SQLSTATE "RX021").

Also, `SQLPutData` can only be used with columns of the following data types: SQL_LONGVARCHAR, SQL_LONGWVARCHAR and SQL_LONGVARBINARY. Calling `SQLPutData` on a column of any other data type returns a "data-at-exec params only allowed for blob (long var...) columns" error ("SQLSTATE RX022").

For more information, reference MSDN documentation for SQLPutData.

SQLRowCount

Gets row count in a table following an INSERT, UPDATE, or DELETE

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLRowCount (
       SQLHSTMT StatementHandle,
       SQLINTEGER *RowCount)
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`RowCount`	`(output)`	Points to a buffer in which to return a row count.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.

SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGet-DiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

If the last SQL statement executed was not an **INSERT**, **UPDATE**, **DELETE**, **IMPORT** or **EXPORT**, `SQLRow-Count` will return `-1` to `*RowCount`.

For more information, reference MSDN documentation for SQLRowCount.

SQLSetConnectAttr

Sets attributes that govern aspects of connections.

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLSetConnectAttr(
     SQLHDBC       ConnectionHandle,
     SQLINTEGER    Attribute,
     SQLPOINTER    ValuePtr,
     SQLINTEGER    StringLength);
```

Arguments

`ConnectionHandle`	(input)	Connection handle.
`Attribute`	(input)	Attribute to set, listed in "Comments."
`ValuePtr`	(input)	Pointer to the value to be associated with Attribute. Depending on the value of `Attribute`, `ValuePtr` will be a 32-bit unsigned integer value or will point to a null-terminated character string. Note that if the `Attribute` argument is a driver-specific value, the value in `ValuePtr` may be a signed integer.
`StringLength`	(input)	If Attribute is an ODBC-defined attribute and `ValuePtr` points to a character string or a binary buffer, this argument should be the length of `*ValuePtr`. For character string data, this argument should contain the number of bytes in the string. If `Attribute` is an ODBC-defined attribute and `ValuePtr` is an integer, `StringLength` is ignored.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.
SQL_NO_DATA	No more data was available. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.

Comments

Attribute	ODBC	ValuePtr Contents
SQL_ATTR_ACCESS_MODE[1]	1.0	An SQLUINTEGER value that specifies the access mode of the database. RDM Embedded accepts the following values: SQL_MODE_READ_ONLY = the database will be opened with the read-only mode. It is equivalent of specifying "r" as the core-level open mode. SQL_MODE_READ_WRITE = the database will be opened for read/write access. It is equivalent of specifying "s" as the core-level open mode. This is the default. SQL_MODE_EXCLUSIVE = the database will be opened with the exclusive-access mode. Only one user can access the database.
SQL_ATTR_ASYNC_ENABLE	3.0	An SQLUINTEGER value that enables the connection-level asynchronous execution support. RDM Embedded supports the following values: SQL_ASYNC_ENABLE_OFF = Connection-level asynchronous execution is not supported.
SQL_ATTR_AUTO_IPD	3.0	A read-only SQLUINTEGER value that specifies whether or not the automatic population of the implementation parameter descriptor (IPD) is supported. RDM Embedded supports the following values: SQL_TRUE = Automatic population of the IPD is supported. An application can set the SQL_ATTR_ENABLE_AUTO_IPD statement attribute to enable the automatic population of the IPD. This is a read-only attribute and cannot be set by **SQLSetConnectAttr**.
SQL_ATTR_AUTOCOMMIT[2]	1.0	An SQLUINTEGER value that specifies whether transactions are committed automatically or manually. RDM Embedded supports the following values: SQL_AUTOCOMMIT_ON = Transactions are committed automatically. This is the default value when RDM Embedded is accessed through Microsoft ODBC Driver Manager. SQL_AUTOCOMMIT_OFF = Transactions need to be

Attribute	ODBC	ValuePtr Contents
		manually committed by calling **SQLEndTran**. This is the default when RDM Embedded is accessed directly from an ODBC application.
SQL_ATTR_CONNECTION_DEAD	3.5	A read-only SQLUINTEGER value that indicates whether or not the connection to the data source is active. RDM Embedded supports the following values: SQL_CD_TRUE = The connection to the RDM Embedded runtime engine has been established. SQL_CD_FALSE = The connection to the RDM Embedded runtime engine has been terminated. Note that the returned value of this attribute may not accurately indicate a "lost" connection. This is a read-only attribute and cannot be set by **SQLSetConnectAttr**.
SQL_ATTR_CONNECTION_TIMEOUT	1.0	An SQLUINTEGER value that specifies the number of seconds a request made by the driver waits until it returns to the application. RDM Embedded supports the following values: 0 = there is no timeout.
SQL_ATTR_CURRENT_CATALOG[1]	2.0	A character string that specifies the name(s) of the databases to open at the time of connection. RDM Embedded supports multiple databases to be open simultaneously. The database names are to be delimited by semicolons.
SQL_ATTR_LOGIN_TIMEOUT	1.0	An SQLUINTEGER value that specifies the number of seconds the driver waits for the login request until returning to the application. RDM Embedded supports the following values: 0 = there is no timeout.
SQL_ATTR_METADATA_ID	3.0	An SQLUINTEGER value that specifies how the string arguments of catalog functions are to be treated. RDM Embedded supports the following values: SQL_FALSE = the string arguments are not treated as identifiers. Case is significant. They can contain search patterns.
SQL_ATTR_ODBC_CURSORS	2.0	An SQLULEN value that tells the Driver Manager whether to use the ODBC cursor library. RDM Embedded supports the following values: SQL_CUR_USE_DRIVER = use the RDM Embed-

Attribute	ODBC	ValuePtr Contents
		ded ODBC driver's cursor.
SQL_ATTR_TXN_ISOLATION	1.0	An SQLUINTEGER value that indicates the transaction isolation level of the current connection. RDM Embedded supports the following values: SQL_TXN_SERIALIZABLE = transactions are serializable. Dirty reads, non-repeatable reads and phantoms are not allowed.
SQL_ATTR_RDM_DBCAT[1]	RDMe	A SQLPOINTER value that specifies the pointer to the RDM Embedded database catalog buffer. More than catalog buffers can be specified. Calling **SQLSetConnectAttr** with this attribute multiple times will append the specified value to the existing catalog buffer list. **SQLGetConnectAttr** with this attribute will return the catalog buffer list. The number of pointers in the list can be retrieved by using the SQL_ATTR_RDM_DBCAT_COUNT attribute.
SQL_ATTR_RDM_DBCAT_COUNT	RDMe	A read-only SQLUINTEGR value that indicates the number of catalog buffer pointers stored in the catalog buffer list. This is a read-only attribute and cannot be set by **SQLSetConnectAttr**.
SQL_ATTR_RDM_DEFER_BLOB	RDMe	An SQLUINTEGER value that indicates whether the reading of BLOB data is deferred. The following values are supported. SQL_FALSE = the reading of BLOB data is not deferred. It means the entire BLOB value is retrieved when a BLOB column is fetched whether or not the column is bound. A call to **SQLGetData** will simply copy the already-retrieved data into the application buffer. This is the default. SQL_TRUE = the reading of BLOB data is deferred. It means the value of a BLOB column is not retrieved when the column is fetched but not bound. A call to **SQLGetData** will retrieve the actual column value.
SQL_ATTR_RDM_REMOTE_CONN[1][3]	RDMe	An SQLUINTEGER value that indicates the type of connection an application makes with the RDM Embedded core engine. The following values are supported.

Attribute	ODBC	ValuePtr Contents
		SQL_CT_UNKNOWN = Whether RDMe ODBC uses the embedded core engine or a remote server is unknown. This is the default. SQL_CT_LOCAL = the RDMe ODBC Driver will use the embedded core engine. This is the default. SQL_CT_REMOTE = the RDMe ODBC Driver will use the RPC mechanism to connect to the specified remote server that accesses the core engine through TCP/IP. The name of the remote server can be specified with the SQL_ATTR_RDM_REMOTE_NAME attribute
SQL_ATTR_RDM_REMOTE_NAME[1]	RDMe	A character string that indicates the name of the remote RDM Embedded host that the driver will connect to for database access. The default is "localhost."
SQL_ATTR_RDM_RESERVE_BYTES[1]	RDMe	An SQLUINTEGER value that indicates the number of bytes RDM Embedded SQL will reserve at connection time. The default value is 0.
SQL_ATTR_RDM_TFS_PORT[1]	RDMe	An SQLUSMALLINT value that indicates the port number the target TFS is listening on. The default is 21553.
SQL_ATTR_RDM_TFSINIT_DBKEEP[4]	RDMe	**Windows only:** A 32-bit integer that indicates whether the embedded TFServer starts the dbkeep module (for in-memory management). If SQL_TRUE, the driver starts dbkeep as part of the embedded TFServer initialization. If SQL_FALSE, the driver does not start dbkeep. The default is SQL_TRUE. The application should set this attribute to SQL_FALSE when it uses an embedded TFServer and a dbkeep module is already running on the computer hosting the application.
SQL_ATTR_RDM_TFSINIT_DISKLESS[4]	RDMe	A 32-bit integer that indicates whether the embedded TFServer uses the "diskless" mode. If SQL_TRUE, the driver uses the "diskless" mode. If SQL_FALSE, the driver does not use the "diskless" mode. The default is SQL_FALSE.
SQL_ATTR_RDM_TFSINIT_DOCROOT[4]	RDMe	A character string that contains the documentation root (docroot) of the embedded TFServer. If the value of this attribute is not set and the RDME_DOCROOT environment variable is set, the value in RDME_DOCROOT will be used. If neither is set, the application's working directory will be used.
SQL_ATTR_RDM_TFSINIT_LOGFILE[4]	RDMe	A character string that contains the name of the log file where the activities of the embedded TFServer will be

Attribute	ODBC	ValuePtr Contents
		logged. The default is NULL, in which case activities will not be logged.
SQL_ATTR_RDM_TFSINIT_PORT[4]	RDMe	A 32-bit integer that contains the port number of the embedded TFServer. The default value is port 21553.
SQL_ATTR_RDM_TFS_READONLY[4]	RDMe	A 32-bit integer that indicates whether the embedded TFServer is in read-only mode. If SQL_TRUE, the embedded TFServer will only allow read operations on its databases. If SQL_FALSE, both read and write operations will be allowed. The default is SQL_FALSE.
SQL_ATTR_RDM_TFSINIT_STDOUT[4]	RDMe	A charater string that contains the name of the file the stdout of the embedded TFServer will be redirected to. The default is NULL, in which case theconsole output will be directed to stdout.
SQL_ATTR_RDM_TFSINIT_VERBOSE[4]	RDMe	A 32-bit integer that indicates whether the verbose mode is enabled. If SQL_TRUE, more detailed message will be printed out to the TFSever console (or a file if the application uses the SQL_ATTR_RDM_TFS_STDOUT attribute). If SQL_FALSE, such detailed messages will not be displayed. The default is SQL_FALSE.

[1]Those attributes can be set before and after connecting to RDM Embedded SQL. However, the values set after a connection has been established will not take effect until the existing connection has been closed and a new one has been established.

[2]Those attributes can be set before and after connecting to RDM Embedded SQL. However, the values set or retrieved before a connection has been established may not reflect the value stored in the RDM Embedded SQL engine. The application should always set/get those attribute values while the connection to the data source is on.

[3] If the SQL_ATTR_RDM_CONN_TYPE attribute is set to SQL_CT_UNKNOWN at the time of connection, **SQLConnect** or **SQLDriverConnect** will determine the connection type by checking the values of SQL_ATTR_RDM_REMOTE_NAME, SQL_ATTR_RDM_TFS_NAME, and in the case of **SQLConnect**, the value of first argument. Once the connection type has been determined, that type will be stored in the SQL_ATTR_RDM_CONN_TYPE attribute. For details, see **SQLConnect** and **SQLDriverConnect**.

[4] Values set with those attributes will have no effect if the application does not use an embedded TFServer ("tfss" or "tfst").

For more information, reference MSDN documentation for SQLSetConnectAttr.

SQLSetCursorName

Associates a cursor name with an active statement.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLSetCursorName (
       SQLHSTMT     StatementHandle,
       SQLCHAR     *CursorName,
       SQLSMALLINT  BufferLength)
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`CursorName`	`(input)`	Cursor name. For efficient processing, the cursor name should not include any leading or trailing spaces in the cursor name, and if the cursor name includes a delimited identifier, the delimiter should be positioned as the first character in the cursor name.
`NameLength`	`(input)`	Length in characters of `*CursorName`.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField`

	to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLCursorName.

SQLSetDescField

Sets the value of a single field of a descriptor record.

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLSetDescField(
     SQLHDESC      DescriptorHandle,
     SQLSMALLINT   RecNumber,
     SQLSMALLINT   FieldIdentifier,
     SQLPOINTER    ValuePtr,
     SQLINTEGER    BufferLength);
```

Arguments

`DescriptorHandle`	`(input)`	Descriptor handle.
`RecNumber`	`(input)`	Indicates the descriptor record containing the field that the application seeks to set. Descriptor records are numbered from 0, with record number 0 being the bookmark record. The `RecNumber` argument is ignored for header fields.
`FieldIdentifier`	`(input)`	Indicates the field of the descriptor whose value is to be set.
`ValuePtr`	`(input)`	Pointer to a buffer containing the descriptor information, or a 4-byte value. The data type depends on the value of `FieldIdentifier`. If `ValuePtr` is a 4-byte value, either all four of the bytes are used or just two of the four are used, depending on the value of the `FieldIdentifier` argument.
`BufferLength`	`(input)`	If `FieldIdentifier` is an ODBC-defined field and `ValuePtr` points to a character string or a binary buffer, this argument should be the length of `*ValuePtr`. For character string data, this argument should contain the number of bytes in the string.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLSetDescField.

SQLSetDescRec

Sets multiple descriptor fields that affect the data type and buffer bound to a column or parameter data.

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLSetDescRec(
      SQLHDESC      DescriptorHandle,
      SQLSMALLINT   RecNumber,
      SQLSMALLINT   Type,
      SQLSMALLINT   SubType,
      SQLLEN        Length,
      SQLSMALLINT   Precision,
      SQLSMALLINT   Scale,
      SQLPOINTER    DataPtr,
      SQLLEN *      StringLengthPtr,
      SQLLEN *      IndicatorPtr);
```

Arguments

`DescriptorHandle`	`(input)`	Descriptor handle. This must not be an IRD handle.
`RecNumber`	`(input)`	Indicates the descriptor record that contains the fields to be set. Descriptor records are numbered from 0, with record number 0 being the bookmark record. This argument must be equal to or greater than 0. If `RecNumber` is greater than the value of SQL_DESC_COUNT, SQL_DESC_COUNTis changed to the value of RecNumber.
`Type`	`(input)`	The value to which to set the SQL_DESC_TYPE field for the descriptor record.
`SubType`	`(input)`	For records whose type is SQL_DATETIME or SQL_INTERVAL, this is the value to which to set the SQL_DESC_DATETIME_INTERVAL_CODE field.
`Length`	`(input)`	The value to which to set the SQL_DESC_OCTET_LENGTH field for the descriptor record.
`Precision`	`(input)`	The value to which to set the SQL_DESC_PRECISION field for the descriptor record.
`Scale`	`(input)`	The value to which to set the SQL_DESC_SCALE field for the descriptor record.
`DataPtr`	`(input/output)`	The value to which to set the SQL_DESC_OCTET_LENGTH_PTR field for the descriptor record. `StringLengthPtr` can be set to a null pointer to set the SQL_DESC_OCTET_LENGTH_PTR field to a null pointer.
`IndicatorPtr`	`(input/output)`	The value to which to set the SQL_DESC_INDICATOR_PTR field for the descriptor record. `IndicatorPtr` can be set to a null pointer to set the SQL_DESC_INDICATOR_PTR field to a null pointer.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
rdmerdbc10	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls SQLGetDiagField to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls SQLGetDiagRec or SQLGetDiagField to retrieve additional information.
SQL_ERROR	Function failed. The application calls SQLGetDiagRec or SQLGetDiagField to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from SQLGetDiagRec or SQLGetDiagField. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

For more information, reference MSDN documentation for SQLSetDescRec.

SQLSetEnvAttr

Sets attributes that govern aspects of environments.

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLSetEnvAttr(
     SQLHENV      EnvironmentHandle,
     SQLINTEGER   Attribute,
     SQLPOINTER   ValuePtr,
     SQLINTEGER   StringLength);
```

Arguments

`EnvironmentHandle`	(input)	Environment handle.
`Attribute`	(input)	Attribute to set, listed in "Comments."
`ValuPtr`	(input)	Pointer to the value to be associated with `Attribute`. Depending on the value of `Attribute`, `ValuePtr` will be a 32-bit integer value or point to a null-terminated character string.
`StringLength`	(input)	If `ValuePtr` points to a character string or a binary buffer, this argument should be the length of `*ValuePtr`. For character string data, this argument should contain the number of bytes in the string. If `ValuePtr` is an integer, `StringLength` is ignored.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGet-`

	`DiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

Attribute and ***ValuePtr***

RDM Embedded accepts the following attribute values.

Attribute	ODBC	ValuePtr contents
SQL_ATTR_CONNECTION_POOLING	3.0	A 32-bit SQLUINTEGER value that enables or disables connection pooling at the environment level. RDM Embedded accepts the following values: SQL_CP_OFF = Connection pooling is turned off.
SQL_ATTR_CP_MATCH	3.0	A 32-bit SQLUINTEGER value that determines how a connection is chosen from a connection pool. RDM Embedded does not support setting this value since it does not support connection pooling.
SQL_ATTR_ODBC_VERSION	3.0	A 32-bit integer that determines whether certain functionality exhibits ODBC 2.x behavior or ODBC 3.x behavior. RDM Embedded accepts the following values. SQL_OV_ODBC3 = The driver exhibits the following ODBC 3.x behavior: • The driver returns and expects ODBC 3.x codes for date, time, and timestamp • The driver returns ODBC 3.x SQLSTATE codes when `SQLGetDiagField`, or `SQLGetDiagRec` is called • The `CatalogName` argument in a call to `SQLTables` accepts a search pattern. • The driver does not support C data type extensibility.

Attribute	ODBC	ValuePtr contents
		RDM Embedded does not require that this attribute be explicitly set before an application calls other functions that have an SQLHENV argument.
SQL_ATTR_OUTPUT_NTS	3.0	A 32-bit integer that determines how the driver returns string data. If SQL_TRUE, the driver returns string data null-terminated. If SQL_FALSE, the driver does not return string data null-terminated.

For more information, reference MSDN documentation for SQLSetEnvAttr.

SQLSetError

Registers user-defined status/error handling functions.

Conformance

Version Introduced: RDM Embedded API extension

Syntax

```
SQLRETURN SQLSetError(
    SQLSMALLINT     HandleType,
    SQLHANDLE       Handle,
    SQLRETURN       ErrorCode,
    ECALLBACK       ErrorHandler);
```

Arguments

`HandleType`	`(input)`	Handle type identifier. The following values are accepted. • SQL_HANDLE_DBC • SQL_HANDLE_DESC • SQL_HANDLE_STMT
`Handle`	`(input)`	The handle, of the type indicated by `HandleType`, indicating the scope of the transaction.
`ErrorCode`	`(input)`	The ODBC error code for which to register the error handler.
`ErrorHandler`	`(input)`	A pointer to the error handler function. (See Error Handler Prototype below). The application can specify NULL in order to unregister the error handler.

Required Headers

```
#include "sqlrext.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Error Handler Prototype

```
int32_t SQL_API errHandler(
    int16_t    HandleType,
    SQLHANDLE  Handle,
    SQLRETURN  error_code);
```

Return Codes

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Diagnostics

SQLSTATE	Error	Description
HY001	Memory allocation error	Either the ODBC driver or the SQL engine failed to allocate sufficient memory to complete the required operation.
HY010	Function sequence error	This function was called after `SQLExecute` or `SQLExecDirect` but before all the parameters were bound.

SQLSetStmtAttr

Sets attributes related to a statement

Conformance

Version Introduced: ODBC 3.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLSetStmtAttr(
       SQLHSTMT StatementHandle,
       SQLINTEGER Attribute,
       SQLPOINTER ValuePtr,
       SQLINTEGER StringLength)
```

Arguments

`StatementHandle`	`(input)`	Statement handle.
`Attribute`	`(input)`	Specifies the statement option type. Possible values are described in "Comments" below.
`ValuePtr`	`(input)`	Pointer to the value to be associated with `Attribute`. Depending on the value of `Attribute`, `ValuePtr` will be a 32-bit unsigned integer value, or will point to a null-terminated character string.
`StringLength`	`(input)`	If Attribute is an ODBC-defined attribute and `ValuePtr` points to a character string or a binary buffer, this argument should be the length of `*ValuePtr`. If `Attribute` is an ODBC-defined attribute and `ValuePtr` is an integer, `StringLength` is ignored.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description

SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

Attributes and ***ValuePtr***

RDM Embedded accepts the following attribute values.

Attribute	ODBC	ValuePtr contents
SQL_ATTR_APP_PARAM_DESC	3.0	The handle to the APD. The handle will replace the handle currently associated with the APD. SQL_NULL_DESC can be specified to dissociate the current handle from the APD; in that case, the APD will revert to the implicitly allocated internal handle. This attribute cannot be set to an implicitly allocated handle except for the handle that was originally implicitly allocated for this APD.
SQL_ATTR_APP_ROW_DESC	3.0	The handle to the ARD. The handle will replace the handle currently associated with the APD. SQL_NULL_DESC can be specified to dissociate the current handle from the APD; in that case, the ARD will revert to the implicitly allocated internal handle. This attribute cannot be set to an implicitly allocated handle except for the handle that was originally implicitly allocated for this ARD.
SQL_ATTR_ASYNC_ENABLE	3.0	An SQLULEN value that specifies whether functions on this statement handle are executed asynchronously. The possible values are: SQL_ASYNC_ENABLE_OFF = Disable statement-level asynchronous execution support (default). The following values are not supported: SQL_ASYNC_ENABLE_ON

Attribute	ODBC	ValuePtr contents
SQL_ATTR_CONCURRENCY	2.0	A SQLULEN value that specifies the cursor concurrency. The possible values are: SQL_CONCUR_READ_ONLY = Cursor is read-only; updates are not allowed (default). SQL_CONCUR_LOCK = Cursor uses the lowest-level locking necessary to perform updates. The following values are not supported: SQL_CONCUR_ROWVER SQL_CONCUR_VALUES If an unsupported value is specified, `SQLSetStmtAttr` substitutes SQL_CONCUR_LOCK for it and returns 01S02 ("option value changed") at execution time.
SQL_ATTR_CURSOR_SCROLLABLE	3.0	An SQLULEN value that specifies whether the cursor is scrollable. The possible values are: SQL_NONSCROLLABLE = The cursor is not scrollable (default). Only SQL_FETCH_NEXT is supported for `SQLFetchScroll`. The following values are not supported: SQL_SCROLLABLE
SQL_ATTR_CURSOR_SENSITIVITY		An SQLULEN value that specifies whether the cursors on the statement handle makes visible the changes made to a result set by another cursor. The possible values are: SQL_UNSPECIFIED = Whether the cursors on the statement handle makes visible the changes made to a result set by another cursor is unspecified (default). The following values are not supported: SQL_INSENSITIVE SQL_SENSITIVE
SQL_ATTR_CURSOR_TYPE	2.0	An SQLULEN value that specifies the cursor type. The possible values are: SQL_CURSOR_FORWARD_ONLY = The cursor scrolls forward-only (default). The following values are not supported: SQL_CURSOR_STATIC SQL_CURSOR_KEYSET_DRIVEN SQL_CURSOR_DYNAMIC

Attribute	ODBC	ValuePtr contents
		RDMe ODBC only supports forward-only cursors.
SQL_ATTR_ENABLE_AUTO_IPD	3.0	An SQLULEN value that specifies whether the IPD is populated automatically. The possible values are: SQL_TRUE = The IPD is automatically populated after `SQLPrepare`. SQL_FALSE = The IPD is not automatically populated after `SQLPrepare`(). The application should call `SQLDescribeParam` to obtain the IPD information (default).
SQL_ATTR_FETCH_BOOKMARK_PTR	3.0	Not supported. Setting this attribute has no effect.
SQL_ATTR_IMP_PARAM_DESC	3.0	The handle to the IPD. This is a read-only attribute. The application can call `SQLGetStmtAttr` to retrieve the value of the attribute.
SQL_ATTR_IMP_ROW_DESC	3.0	The handle to the IRD. This is a read-only attribute. The application can call `SQLGetStmtAttr` to retrieve the value of the attribute.
SQL_ATTR_KEYSET_SIZE	2.0	An SQLULEN value that specifies the number of rows in the keyset for a keyset-driven cursor. RDMe 10.1 does not support this attribute. Setting this attribute has no effect.
SQL_ATTR_MAX_LENGTH	1.0	An SQLULEN value that specifies the maximum amount of data the RDMe ODBC driver returns from a binary or character column. The default value is 0. In that case, RDMe ODBC Driver attempts to return all available data. If the specified length is less than the length of the available data, the data is truncated and SQL_SUCCESS is returned.
SQL_ATTR_MAX_ROWS	1.0	An SQLULEN value that specifies the maximum number of rows returned by a **select** statement. The default value is 0. In that case, RDMe ODBC Driver attempts to return all rows. If set to non-zero, this value sets the maximum value for the cursor row count.
SQL_ATTR_METADATA_ID	3.0	An SQLULEN value that specifies how the string arguments of catalog functions are treated. The possible values are: SQL_FALSE = the string arguments of catalog functions are not treated as identifiers. Arguments are case-sensitive and may contain a string search pattern (default).

Attribute	ODBC	ValuePtr contents
		The following values are not supported: SQL_TRUE
SQL_ATTR_NOSCAN	1.0	An SQLULEN value that indicates whether the RDMe ODBC Driver should scan SQL strings for escape sequences. The possible values are: SQL_NOSCAN_OFF = SQL strings are scanned for escape sequences (default). SQL_NOSCAN_ON = SQL strings are not scanned for escape sequences. They are passed directly to the data source.
SQL_ATTR_PARAM_BIND_OFFSET_PTR	3.0	An SQLULEN pointer that points to an offset value added to pointers to change dynamic parameter binding. If non-null, the pointer will be dereferenced, and the value contained will be added to each of the dereferenced fields in the APD. The affected fields are: SQL_DESC_DATA_PTR SQL_DESC_INDICATOR_PTR SQL_DESC_OCTET_LENGTH_PTR The default value is NULL.
SQL_ATTR_PARAM_BIND_TYPE	3.0	An SQLULEN vaule that specifies the binding orientation to be used for dynamic parameters. The possible values are: SQL_PARAM_BIND_BY_COLUMN = Perform column-wise binding (default). Row-wise binding is not supported.
SQL_ATTR_PARAM_OPERATION_PTR	3.0	Not supported. Setting this attribute has no effect.
SQL_ATTR_PARAM_STATUS_PTR	3.0	Not supported. Setting this attribute has no effect.
SQL_ATTR_PARAMS_PROCESSED_PTR	3.0	Not supported. Setting this attribute has no effect.
SQL_ATTR_PARMSET_SIZE	3.0	Not supported. Setting this attribute has no effect.
SQL_ATTR_QUERY_TIMEOUT	1.0	Not supported. Setting this attribute has no effect.
SQL_ATTR_RETRIEVE_DATA	2.0	An SQULEN value that specifies whether `SQLFetch` and `SQLFetchScroll` retrieve data after it positions the cursor. The possible values are: SQL_RD_ON = Data are retrieved (default). The following values are not supported: SQL_RD_OFF
SQL_ATTR_ROW_ARRAY_SIZE	3.0	An SQLULEN value that specifies the number of rows returned by each call to SQLFetch or SQLFetchScroll. The default value is 1.

Attribute	ODBC	ValuePtr contents
SQL_ATTR_ROW_BIND_OFFSET_PTR	3.0	An SQLULEN pointer that points to an offset value added to pointers to change column binding. If non-null, the pointer will be dereferenced, and the value contained will be added to each of the dereferenced fields in the ARD. The affected fields are: SQL_DESC_DATA_PTR SQL_DESC_INDICATOR_PTR SQL_DESC_OCTET_LENGTH_PTR The default value is NULL.
SQL_ATTR_ROW_BIND_TYPE	1.0	An SQLULEN value that specifies the binding orientation to be used for column binding The possible values are: SQL_BIND_BY_COLUMN = Perform column-wise binding (default). Row-wise binding is not supported by RDMe 10.1.
SQL_ATTR_ROW_NUMBER	2.0	An SQLULEN value that indicates the number of the current row in the entire result set. If the value cannot be determined or there is no current row, 0 will be returned. This is a read-only attribute. The application can call `SQLGetStmtAttr` to retrieve the value of the attribute.
SQL_ATTR_ROW_OPERATION_PTR	3.0	Not supported. Setting this attribute has no effect.
SQL_ATTR_ROW_STATUS_PTR	3.0	An SQLUSMALLINT pointer that points to an array of SQLUSMALLINT values that will contain the row status values after `SQLFetch` or `SQLFetchScroll` was called. The array has the same number of elements as the number of rows in the rowset. The default value of this attribute is NULL. In that case, row status values will not be returned to the application.
SQL_ATTR_ROWS_FETCHED_PTR	3.0	An SQLULEN pointer that points to a buffer in which the number of rows fetched by `SQLFetch` or `SQLFetchScroll` is stored. The default value of this attribute is NULL. In that case, the number of result rows will not be returned to the application.
SQL_ATTR_SIMULATE_CURSOR	2.0	An SQLULEN value that specifies whether RDMe ODBC Driver guarantees positioned update and delete statements affect only one row. The possible values are:

Attribute	ODBC	ValuePtr contents
		SQL_SC_NON_UNIQUE = The driver does not guarantee that simulated positioned update and delete statements affect only one row. SQL_SC_TRY_UNIQUE = The driver tries to guarantee that simulated positioned update and delete statements affect only one row. SQL_SC_UNIQUE = The driver guarantees that simulated positioned update and delete statements affect only one row (default). SQL_SC_UNIQUE is the default value for RDMe ODBC since RDMe provides the native SQL support that guarantees that simulated positioned update and delete statements affect only one row. Specifying SQL_SC_TRY_UNIQUE or SQL_SC_NON_UNIQUE will return SQL_SUCCESS_WITH_INFO ("01S02").
SQL_ATTR_USE_BOOKMARKS	2.0	An SQLULEN value that specifies whether bookmarks are used with a cursor. The possible values are: SQL_UB_OFF = Off (default). The following values are not supported: SQL_UB_ON

For more information, reference MSDN documentation for SQLSetStmtAttr.

SQLSpecialColumns

Returns the information about the set of columns that uniquely identifies a row in the table.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: Open Group

Syntax

```
SQLRETURN SQLSpecialColumns(
     SQLHSTMT      StatementHandle,
     SQLSMALLINT   IdentifierType,
     SQLCHAR *     CatalogName,
     SQLSMALLINT   NameLength1,
     SQLCHAR *     SchemaName,
     SQLSMALLINT   NameLength2,
     SQLCHAR *     TableName,
     SQLSMALLINT   NameLength3,
     SQLSMALLINT   Scope,
     SQLSMALLINT   Nullable);
```

Arguments

StatementHandle	(input)	Statement handle for retrieved results.
IdentifierType	(input)	Type of column to return. Must be one of the following values: SQL_BEST_ROWID: Returns the optimal column or set of columns that, by retrieving values from the column or columns, allows any row in the specified table to be uniquely identified. A column can be either a pseudo-column specifically designed for this purpose (as in Oracle ROWID or Ingres TID) or the column or columns of any unique index for the table. SQL_ROWVER: Returns the column or columns in the specified table, if any, that are automatically updated by the data source when any value in the row is updated by any transaction (as in SQLBase ROWID or Sybase TIMESTAMP).
CatalogName	(input)	Catalog name for the table. If a driver supports catalogs for some tables but not for others, such as when the driver retrieves data from different DBMSs, an empty string ("") denotes those tables that do not have catalogs. CatalogName cannot contain a string search pattern.
NameLength1	(input)	Length in characters of *CatalogName.
SchemaName	(input)	Schema name for the table. If a driver supports schemas for some tables but not for others, such as when the driver retrieves data from different DBMSs, an empty string ("") denotes those tables that do not have schemas. SchemaName cannot contain a string search pattern.
NameLength2	(input)	Length in characters of *SchemaName.
TableName	(input)	Table name. This argument cannot be a null pointer. TableName cannot contain a string search pattern.
NameLenght3	(input)	Length in characters of *TableName.
Scope	(input)	Minimum required scope of the rowid. The returned rowid may be of greater scope. Must be one of the following: SQL_SCOPE_CURROW: The rowid is guaranteed to be valid only while positioned on that row. A later reselect using rowid may not return a row if the row was updated or deleted by another transaction. SQL_SCOPE_TRANSACTION: The rowid is guaranteed to be valid for the duration of the current transaction. SQL_SCOPE_SESSION: The rowid is guaranteed to be valid for the duration of the session (across transaction boundaries).
Nullable	(input)	Determines whether to return special columns that can have a NULL value. Must be one of the following: SQL_NO_NULLS: Exclude special columns that can have NULL values. Some drivers cannot support SQL_NO_NULLS, and these drivers will return an empty result set if SQL_NO_NULLS was specified. Applications should be prepared for this case and request SQL_NO_NULLS only if it is absolutely required. SQL_NULLABLE: Return special columns even if they can have NULL values.

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
rdmerdbc10	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls SQLGetDiagField to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls SQLGetDiagRec or SQLGetDiagField to retrieve additional information.
SQL_ERROR	Function failed. The application calls SQLGetDiagRec or SQLGetDiagField to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from SQLGetDiagRec or SQLGetDiagField. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

SQLSpecialColumns returns the information that pertains to the databases that are currently open. If no database is open on the data source, SQLSpecialColumns returns an empty result set.

For more information, reference MSDN documentation for SQLSpecialColumns.

SQLStatistics

Retrieves a list of statistics about a single table and the indexes associated with the table. The driver returns the information as a result set.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLStatistics(
     SQLHSTMT        StatementHandle,
     SQLCHAR *       CatalogName,
     SQLSMALLINT     NameLength1,
     SQLCHAR *       SchemaName,
     SQLSMALLINT     NameLength2,
     SQLCHAR *       TableName,
     SQLSMALLINT     NameLength3,
     SQLUSMALLINT    Unique,
     SQLUSMALLINT    Reserved);
```

Arguments

`StatementHandle`	`(input)`	Statement handle for retrieved results.
`CatalogName`	`(input)`	Catalog name.
`NameLength1`	`(input)`	Length in characters of `*CatalogName`.
`SchemaName`	`(input)`	Schema name. If a driver supports schemas for some tables but not for others, such as when the driver retrieves data from different DBMSs, an empty string ("") indicates those tables that do not have schemas. `SchemaName` cannot contain a string search pattern.
`NameLength2`	`(input)`	Length in characters of `*SchemaName`.
`TableName`	`(input)`	Table name. This argument cannot be a null pointer. `SchemaName` cannot contain a string search pattern.
`NameLenght3`	`(input)`	Length in characters of `*TableName`.
`Unique`	`(input)`	Type of index: SQL_INDEX_UNIQUE or SQL_INDEX_ALL.
`Reserved`	`(input)`	

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
rdmerdbc10	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls SQLGetDiagField to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls SQLGetDiagRec or SQLGetDiagField to retrieve additional information.
SQL_ERROR	Function failed. The application calls SQLGetDiagRec or SQLGetDiagField to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from SQLGetDiagRec or SQLGetDiagField. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

SQLStatistics returns the information that pertains to the databases that are currently open. If no database is open on the data source, SQLStatistics returns an empty result set.

For more information, reference MSDN documentation for SQLStatistics.

SQLTables

Returns the list of table, catalog, or schema names, and table types, stored in a specific data source.

Conformance

Version Introduced: ODBC 1.0 Standards Compliance: ISO 92

Syntax

```
SQLRETURN SQLTables(
     SQLHSTMT       StatementHandle,
     SQLCHAR *      CatalogName,
     SQLSMALLINT    NameLength1,
     SQLCHAR *      SchemaName,
     SQLSMALLINT    NameLength2,
     SQLCHAR *      TableName,
     SQLSMALLINT    NameLength3,
     SQLCHAR *      TableType,
     SQLSMALLINT    NameLength4);
```

Arguments

`StatementHandle`	`(input)`	Statement handle for retrieved results.
`CatalogName`	`(input)`	Catalog name.
`NameLength1`	`(input)`	Length in characters of `*CatalogName`.
`SchemaName`	`(input)`	String search pattern for schema names.
`NameLength2`	`(input)`	Length in characters of `*SchemaName`.
`TableName`	`(input)`	String Search pattern for table names.
`NameLenght3`	`(input)`	Length in characters of `*TableName`.
`TableType`	`(input)`	List of table types to match.
`NameLength4`	`(input)`	Length in characters of `*TableType`

Required Headers

```
#include "sql.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Comments

`SQLTables` returns the information that pertains to the databases that are currently open. If no database is open on the data source, `SQLTables` returns an empty result set.

For more information, reference MSDN documentation for SQLTables.

SQLTransactStatus

Retrieves the type and status of the transaction possibly executed on the handle.

Conformance

Version Introduced: RDM Embedded API extension

Syntax

```
SQLRETURN SQL_API SQLTransactStatus(
    SQLSMALLINT      HandleType,
    SQLHANDLE        Handle,
    SQLSMALLINT     *pActive)
```

Arguments

`HandleType`	`(input)`	Handle type identifier. The following values are accepted. Contains either SQL_HANDLE_ENV (if Handle is an environment handle) or SQL_HANDLE_DBC (if Handle is a connection handle).
`Handle`	`(input)`	The handle, of the type indicated by `HandleType`, indicating the scope of the transaction.
`pActive`	`(output)`	The status of the transaction on the handle. For the possible return values for `pActive`, see the table in the **Comments** section below.

Required Headers

```
#include "sqlrext.h"
```

Libraries

Library Name	Description
`rdmerdbc10`	RDMe ODBC API Library

See Library Naming Conventions section for full library name and a list of library dependencies.

Returns

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warn-

	ing). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.

Diagnostics

SQLSTATE	Error	Description
08003	Connection not open	The Handle argument specifies a connection handle and is not connected to the data source.
25S01	Transaction state unknown	The Handle argument specifies an environment handle, and one or more of the connection handles associated with it failed to retrieve the status of its transaction.
HY009	Invalid use of null pointer	A null pointer was specified for the `pActive` argument.

Comments

The following table describes the possible return values of the status of the transaction through the `pActive` argument.

Value of `pActive`	Description
SQL_TXN_INACTIVE	No transaction is active on the specified handle.
SQL_TXN_REGULAR	A regular (i.e. updatable) transaction is active on the specified handle.
SQL_TXN_READONLY	A read-only transaction (i.e. snapshot) is active on the specified handle.

Return Codes ODBC

Each function in ODBC returns a code, known as its return code, which indicates the overall success or failure of the function.

For example, the following code calls `SQLFetch` to retrieve the rows in a result set. It checks the return code of the function to determine if the end of the result set was reached (SQL_NO_DATA), if any warning information was returned (SQL_SUCCESS_WITH_INFO), or if an error occurred (SQL_ERROR).

```
SQLRETURN   rc;
SQLHSTMT    hstmt;

while ((rc=SQLFetch(hstmt)) != SQL_NO_DATA) {
   if (rc == SQL_SUCCESS_WITH_INFO) {
      // Call function to display warning information.
   } else if (rc == SQL_ERROR) {
      // Call function to display error information.
      break;
   }
   // Process row.
}
```

The return code SQL_INVALID_HANDLE always indicates a programming error and should never be encountered at run time. All other return codes provide run-time information, although SQL_ERROR may indicate a programming error.

The following table defines the return codes.

Return Code	Description
SQL_SUCCESS	Function completed successfully. The application calls `SQLGetDiagField` to retrieve additional information from the header record.
SQL_SUCCESS_WITH_INFO	Function completed successfully, possibly with a nonfatal error (warning). The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_ERROR	Function failed. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information. The contents of any output arguments to the function are undefined.
SQL_INVALID_HANDLE	Function failed due to an invalid environment, connection, statement, or descriptor handle. This indicates a programming error. No additional information is available from `SQLGetDiagRec` or `SQLGetDiagField`. This code is returned only when the handle is a null pointer or is the wrong type, such as when a statement handle is passed for an argument that requires a connection handle.
SQL_NO_DATA	No more data was available. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional information.
SQL_NEED_DATA	More data is needed, such as when parameter data is sent at execution time or additional connection information is required. The application calls `SQLGetDiagRec` or `SQLGetDiagField` to retrieve additional

	information, if any.