HP NonStop ODBC/MX Client Drivers User Guide for SQL/MX Release 3.2.1

HP Part Number: 734873-002
Published: November 2013
Edition: J06.16 and subsequent J-series RVUs; H06.27 and subsequent H-series RVUs

Contents

About this document....5
Intended audience....5
Document organization....5
New and changed information in this edition....5
Notation conventions....5
General syntax notation....5
Related Documentation....7
NonStop SQL/MX customer library....7
Publishing history....9
HP encourages your comments....9
1 Overview....10
Architecture....10
Client components....12
Server components....12
Driver managers....12
Data sources....13
Client data source configuration....13
Server data source configuration....13
Default data source....13
Tracing....14
Object naming and mapping....14
ANSI name type....14
ANSI names for SQL/MX objects....14
ANSI alias names for SQL/MP objects....14
Unsupported ODBC APIs....14
Unsupported ODBC data types....15
Considerations and limitations....15
2 Installing the drivers....17
Installation requirements....17
Installation procedures....17
Verifying the installation....21
Uninstalling the driver....22
3 Configuring data sources....24
Configuring a client data source....24
Configuring the drivers for IPv6....26
Configuring a secure ODBC connection using NonStop SSL....26
Installing a NonStop SSL Server process for ODBC/MX....26
Installing and configuring the Remote Proxy Client....27
Configuring a client data source for NonStop SSL....28
DataSource configuration in Windows driver....29
MXODSN and ODBCDSN file formats....30
Managing data sources....33
Managing transactions....34
Setting AutoCommit....34
Tracing....35
4 Compatibility and considerations....37
Considerations....37
ODBC data types....38
Unsigned data types....39

Partial DATE or TIME values....40
Microsoft escape clauses....41
Stored Procedures....41
Transaction and cursor behavior....41
Timestamp values with fraction....42
SQL Interval Behaviour property....46
SQL Datetime Retrieval property....48
5 Error messages....49
A Sample ODBC application....53
Compiling and linking the sample application....53
Example for compiling a threaded application....53
Testing the sample....54
Sample application code....54
....55
Glossary....61
Index....62

About this document

This manual describes how to install, configure and use the ODBC/MX client drivers. These drivers allow HP-UX, Linux32, Linux64, OSS and Windows applications developed for Open Database Connectivity (ODBC) to access data from an SQL/MX database on a NonStop system.

Intended audience

This manual is intended for the following audience:

- ODBC application programmers who develop applications to access data from an SQL/MX database
- Administrators installing and configuring the driver on the HP-UX, Linux, OSS, and Windows platforms

Document organization

Chapter Name	Description
Chapter 1: Overview	Provides an overview of the ODBC/MX client drivers.
Chapter 2: Installing the drivers	Describes procedures to install the ODBC/MX client drivers.
Chapter 3: Configuring data sources	Describes steps to configure the client data sources.
Chapter 4: Compatibility and considerations	Lists the compatibility for the drivers.
Chapter 5: Error messages	Lists the error codes and messages.
Appendix A: Sample ODBC application	Provides a sample ODBC application.

New and changed information in this edition

Changes to the 734873–002 manual:

- Added information about `AutoCommit` in "Managing transactions" (page 34).
- Added information about Linux 64 bit driver across the manual.

This manual combines the following manuals:

- *Open System Services ODBC/MX Client Driver for SQL/MX Release 3.2.1*
- *Linux ODBC/MX Client Driver for SQL/MX Release 3.2.1*
- *ODBC/MX Driver for Windows Manual for SQL/MX Release 3.2.1*

This manual also contains added information on configuring secure ODBC connection for NonStop SSL, HP-UX driver, and Unicode support in Windows ODBC/MX client driver.

Notation conventions

General syntax notation

This list summarizes the notation conventions for syntax presentation in this manual.

UPPERCASE LETTERS

Uppercase letters indicate keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

SELECT

Italic Letters

Italic letters, regardless of font, indicate variable items that you supply. Items not enclosed in brackets are required. For example:

```
file-name
```

Computer Type

Computer type letters within text indicate case-sensitive keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

```
myfile.sh
```

Bold Text

Bold text in an example indicates user input typed at the terminal. For example:

```
ENTER RUN CODE

?123
CODE RECEIVED:      123.00
```

The user must press the Return key after typing the input.

[ ] Brackets

Brackets enclose optional syntax items. For example:

```
DATETIME [start-field TO] end-field
```

A group of items enclosed in brackets is a list from which you can select one item or none. The items in the list can be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
DROP schema [CASCADE]
                    [RESTRICT]

DROP schema [ CASCADE | RESTRICT ]
```

{ } Braces

Braces enclose required syntax items. For example:

```
FROM { grantee[, grantee]...}
```

A group of items enclosed in braces is a list from which you are required to select one item. The items in the list can be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
INTERVAL { start-field TO end-field }
         { single-field }

INTERVAL { start-field TO end-field  | single-field }
```

| Vertical Line

A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
{expression | NULL}
```

... Ellipsis

An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times. For example:

```
ATTRIBUTE[S] attribute [, attribute]...
```

```
{, sql-expression}...
```

An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times. For example:

```
expression-n…
```

Punctuation

Parentheses, commas, semicolons, and other symbols not previously described must be typed as shown. For example:

```
DAY (datetime-expression)

@script-file
```

Quotation marks around a symbol such as a bracket or brace indicate the symbol is a required character that you must type as shown. For example:

```
"{" module-name [, module-name]... "}"
```

Item Spacing

Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or a comma. For example:

```
DAY (datetime-expression)

DAY(datetime-expression)
```

If there is no space between two items, spaces are not permitted. In this example, no spaces are permitted between the period and any other items:

```
myfile.sh
```

Line Spacing

If the syntax of a command is too long to fit on a single line, each continuation line is indented three spaces and is separated from the preceding line by a blank line. This spacing distinguishes items in a continuation line from items in a vertical list of selections. For example:

```
match-value [NOT] LIKE pattern

   [ESCAPE esc-char-expression]
```

Related Documentation

This manual is part of the HP NonStop SQL/MX library of manuals.

NonStop SQL/MX customer library

The manuals in the SQL/MX customer library are listed here for your convenience.

- **Introductory guides**

SQL/MX Comparison Guide for SQL/MP Users	Describes differences between NonStop SQL/MP and NonStop SQL/MX databases.
SQL/MX Quick Start Guide	Describes basic techniques for using SQL in the SQL/MX conversational interface (MXCI). Includes information about installing the sample database.

- **Installation guides**

SQL/MX Installation and Upgrade Guide	Describes how to plan for install and upgrade an SQL/MX database.
SQL/MX Management Manual	Describes how to manage an SQL/MX database.
NSM/web Installation Guide	Describes how to install NSM/web and troubleshoot NSM/web installations.

- **Reference manuals**

SQL/MX Reference Manual	Describes the syntax of SQL/MX statements, MXCI commands, functions, and other SQL/MX language elements.
SQL/MX Messages Manual	Describes SQL/MX messages.
SQL/MX Glossary	Defines SQL/MX terminology.

- **Connectivity manuals**

SQL/MX Connectivity Service Manual	This manual is an installation guide and describes how to install and manage SQL/MX Connectivity Service (MXCS), which enables ODBC and other connectivity APIs to use NonStop SQL/MX.
SQL/MX Connectivity Service Administrative Command Reference	Describes the SQL/MX Administrative Command Library (MACL) available with the SQL/MX conversational interface (MXCI).
SQL/MX Remote Conversational Interface (RMXCI) Guide	Describes how to use SQL/MX Remote Conversational Interface to run the RMXCI commands, and SQL statements interactively or from script files.
HP NonStop JDBC Type 4 Driver 3.0 Programmer's Reference	Describes the NonStop JDBC Type 4 Driver functionality, which allows Java programmers to remotely develop applications deployed on client workstations to access NonStop SQL/MX databases.
ODBC/MX Client Drivers User Guide	Describes how to install and configure the ODBC/MX client drivers. These products enable applications developed for the Open Database Connectivity (ODBC) application programming interface to access data from an SQL/MX database on a NonStop system.

- **Migration guides**

SQL/MX Database and Application Migration Guide	Describes how to migrate databases and applications to NonStop SQL/MX, and how to manage different versions of NonStop SQL/MX.
NonStop NS-Series Database Migration Guide	Describes how to migrate NonStop SQL/MX, NonStop SQL/MP, Enscribe databases and applications to HP Integrity NonStop NS-series systems.

- **Data management guides**

SQL/MX Data Mining Guide	Describes the SQL/MX data structures and operations for data mining.
SQL/MX Report Writer Guide	Describes how to produce formatted reports using data from an SQL/MX database.
DataLoader/MX Reference Manual	Describes the features and functions of the DataLoader/MX product, a tool to load SQL/MX databases.

- **Application development guides**

SQL/MX Programming Manual for C and COBOL	Describes how to embed SQL/MX statements in ANSI C and COBOL programs.
SQL/MX Query Guide	Describes how to understand query execution plans and write optimal queries for an SQL/MX database.
SQL/MX Queuing and Publish/Subscribe Services	Describes how NonStop SQL/MX integrates transactional queuing and publish/subscribe services into its database infrastructure.
SQL/MX Guide to Stored Procedures in Java	Describes how to use stored procedures that are written in Java within NonStop SQL/MX.

- **Online help**

Reference Help	Overview and reference entries from the SQL/MX Reference Manual.
Messages Help	Individual messages grouped by source from the SQL/MX Messages Manual.
Glossary Help	Terms and definitions from the SQL/MX Glossary.
NSM/web Help	Context-sensitive help topics that describe how to use the NSM/web management tool.
Visual Query Planner Help	Context-sensitive help topics that describe how to use the Visual Query Planner graphical user interface.
SQL/MX Database Manager Help	Contents and reference entries from the *SQL/MX Database Manager User Guide*.

Publishing history

Part Number	Product Version	Publication Date
734873-001	ODBC/MX Client Drivers User Guide for SQL/MX Release 3.2.1	August 2013
734873-002	ODBC/MX Client Drivers User Guide for SQL/MX Release 3.2.1	November 2013

HP encourages your comments

HP encourages your comments concerning this document. We are committed to providing documentation that meets your needs. Send any errors found, suggestions for improvement, or compliments to:

docsfeedback@hp.com

Include the document title, part number, and any comment, error found, or suggestion for improvement you have concerning this document.

1 Overview

The following ODBC/MX client drivers enable ODBC applications running on various platforms, to access a NonStop SQL/MX database:

- Linux 32–bit and 64–bit drivers
- HP-UX 64–bit driver
- Windows 32–bit and 64–bit drivers for ANSI and Unicode applications
- OSS 32–bit and 64–bit libraries

The HP-UX, Linux, OSS, Windows ANSI and Unicode ODBC/MX client drivers are collectively referred as drivers. In this manual the HP-UX, Linux, and OSS drivers are collectively referred as Unix drivers.

The drivers implement the ODBC 3.5 set of APIs. The ODBC client applications (applications) send the requests to access data from an SQL/MX database through these drivers.

This chapter describes the following:

- Architecture (page 10)
- Client components (page 12)
- Server components (page 12)
- Driver managers (page 12)
- Data sources (page 13)
- Tracing (page 14)
- Object naming and mapping (page 14)
- Unsupported ODBC APIs (page 14)
- Unsupported ODBC data types (page 15)
- Considerations and limitations (page 15)

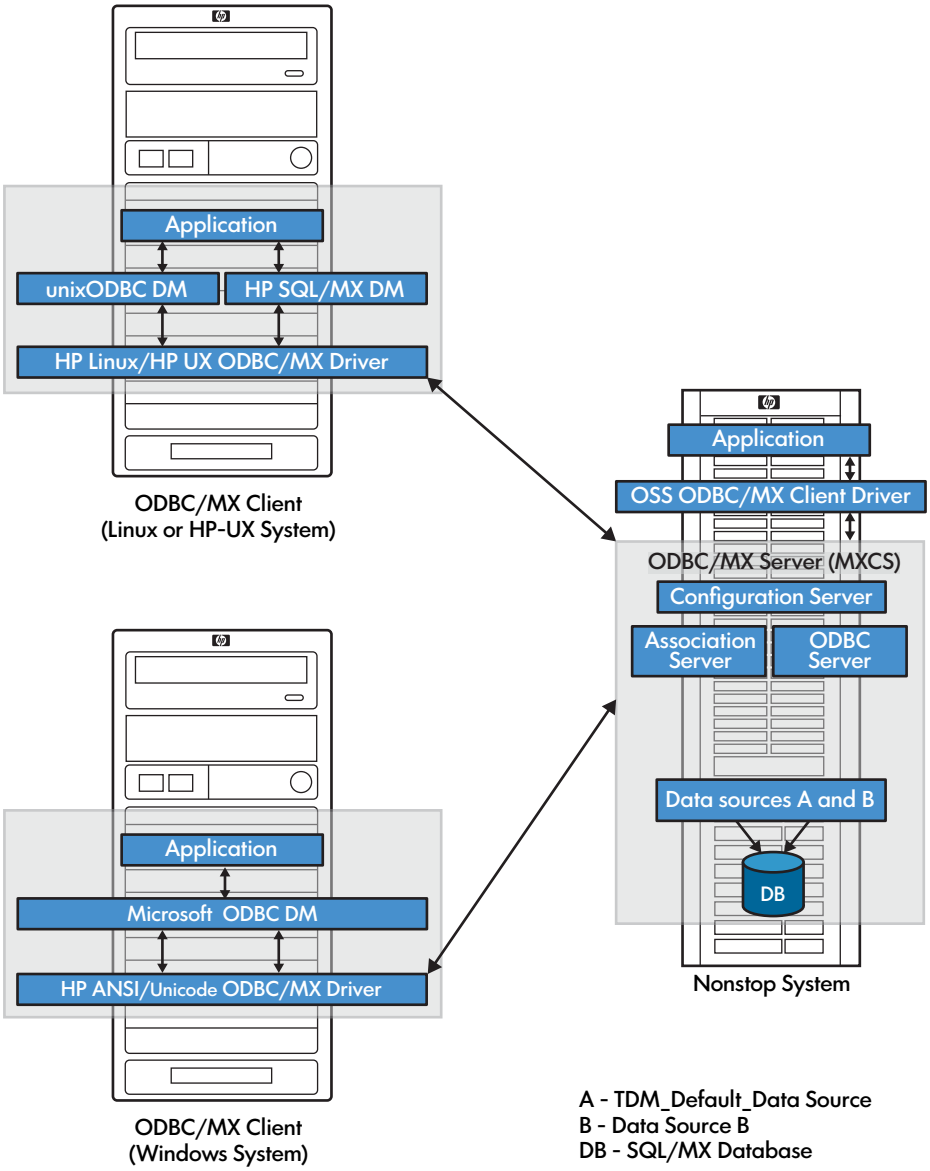
Architecture

The applications interact with the drivers to access an SQL/MX database.

The Unix drivers are thread-safe, and you can write multi-threaded C/C++ applications, which use the POSIX thread library.

Figure 1 (page 11) illustrates the architecture, which comprise client and server components.

Figure 1 Architecture



The application calls the `SQLConnect()` ODBC API to open a connection to a data source, which represents NonStop SQL on the host. The NonStop SQL/MX Connectivity Service (MXCS) provides support for remote connections to the SQL/MX database. For example, support for remote connections from ODBC and JDBC applications. The drivers forward the request to MXCS, which authenticates the connection parameters. After successful authentication, MXCS assigns an ODBC server instance to the application. MXCS matches the client data source name with server data source name. If the client data source name matches the server data source name, MXCS attempts a connection. MXCS connects to the data source based on the following conditions:

- The data source must be configured and started.
- If the data source is not configured on the server, then MXCS forwards the connect request to the default data source, `TDM_Default_DataSource`.

After the connection is established, MXCS forwards the application requests to the SQL/MX database, and routes the responses from the database to the driver.

The communication between the driver and MXCS is over TCP/IP sockets in the client-server parlance. Configure the IP address and the port number for the HP-UX and Linux drivers in the

MXODSN file, and for the OSS driver in the ODBCDSN file. Use the Microsoft ODBC Data Source Administrator to configure the parameters for the Windows driver.

Client components

The common client components are the ODBC applications, which are user-written or third-party products that use the ODBC APIs, and the driver for the application platform. Install the following client components for HP-UX, Linux, and Windows platforms:

HP-UX

- unixODBC driver manager (only required for standalone driver, libmxodbc_ia64_drvr.sl), which is the open source driver manager implementation that you can download from the unixODBC website http://www.unixODBC.org.
- SQL/MX driver manager, which is the driver manager implementation from HP and includes a set of APIs, which internally makes multiple ODBC API calls.

Linux 32

- unixODBC driver manager (only required for standalone driver, libmxodbc_l_drvr.so), which is the open source driver manager implementation that you can download from the unixODBC website http://www.unixODBC.org.
- SQL/MX driver manager, which is the driver manager implementation from HP and includes a set of APIs, which internally makes multiple ODBC API calls.

Linux 64

- unixODBC driver manager (only required for standalone driver, libmxodbc64_l_drvr.so), which is the open source driver manager implementation that you can download from the unixODBC website http://www.unixODBC.org.
- SQL/MX driver manager, which is the driver manager implementation from HP and includes a set of APIs, which internally makes multiple ODBC API calls.

Windows

- Microsoft ODBC Driver Manager version 3.5.1 or later, which manages ODBC API requests to ODBC drivers.
- Microsoft ODBC Data Source Administrator.

Server components

The following server components are required on the NonStop system:

- SQL/MX database release 3.2.1
- MXCS

Driver managers

The HP-UX or Linux applications can use either the unixODBC driver manger versions 2.2.14, or 2.3.1 and later, or the SQL/MX driver manager. When compiling, the applications can directly link with the unixODBC library or can use connectors to communicate with the driver manager. The driver manager uses the information provided by the applications and routes the ODBC API calls to the driver.

For more information about unixODBC driver manager, see the unixODBC website http://www.unixODBC.org.

The Windows applications use the Microsoft ODBC Driver Manager version 3.5.1 or later.

Data sources

Data sources are logical groupings of connections to the SQL/MX database. There are two types of data sources.

- Client data source, commonly known as Data Source Name (DSN). The DSN is configured on the client workstation.
- Server data source, which you must define and start on the NonStop system.

The driver uses the client data source configuration to route the ODBC API requests to MXCS.

For more information about data sources, see the *HP NonStop SQL/MX Connectivity Service Manual for SQL/MX Release 3.x*.

Client data source configuration

The client data sources are configured on the client workstation.

HP-UX, Linux32 and Linux 64

- Configure the client data sources in the `MXODSN` file when the driver uses the SQL/MX driver manager.
- Configure the client data sources in the `odbc.ini` and `odbcinst.ini` files when the driver uses the unixODBC driver manager.

OSS

You must configure the data source in the `ODBCDSN` file.

Windows

You must configure the client data sources with Microsoft ODBC Data Source Administrator.

The driver accesses the configuration details when it sends a connect request to MXCS. If the data source is not configured on the server, MXCS connects the application to the default data source, `TDM_Default_DataSource`.

Server data source configuration

The server data sources are configured on a NonStop server. For more information about configuring and managing server data sources, see the *HP NonStop SQL/MX Connectivity Service Manual for SQL/MX Release 3.x*.

Default data source

`TDM_Default_DataSource` is the pre-configured default server data source. You can stop and start `TDM_Default_DataSource` with any one of the following, but you cannot delete it:

- NSM/web
- SQL/MX Administrative Command Library (MACL, `mode mxcs` invoked from mxci)
- SQL/MX Remote Conversational Interface (RMXCI)
- SQL/MX Database Manager (available from SQL/MX Release 3.2 onwards)

MXCS establishes a connection with `TDM_Default_DataSource` when there is a mismatch in the client and server data source names.

For more information about MACL, see the *SQL/MX Connectivity Service Administrative Command Reference* .

Tracing

HP-UX, Linux 32 and Linux 64

The trace information is captured in a log file. You can enable or disable tracing and specify the trace file name in `MXODSN` file. When using the unixODBC driver manager, you can enable the driver tracing by setting the following environment variables:

- HPODBC_TRACE_LEVEL. For trace values, see "`MXODSN` and `ODBCDSN` file formats" (page 30).
- HPODBC_TRACEFILE_NAME to set the trace file name. The PID and current time are appended to this name to form the final trace file name.
- HPODBC_TRACEFILE_SIZE to set the maximum trace file size. The default file size is 1 GB.

OSS

The trace information is captured in an OSS log file. You can enable or disable tracing and specify the trace file name in `ODBCDSN` file.

Windows

You can trace the entry and exit calls from the Microsoft driver manager to the driver, contents of the buffers (except password buffers), and activity between the driver and transport layer. To get the tracing help, click the **Help** button in the Microsoft ODBC Data Source Administrator panel.

NOTE: For information about MXCS tracing, see *SQL/MX Connectivity Service Administrative Command Reference* manual.

Object naming and mapping

The drivers can access SQL/MX database objects and SQL/MP objects that have an alias.

ANSI name type

Only ANSI name types are supported for SQL/MX database objects. You can configure the default catalog and schema names to qualify table names in the queries. The name format is:

```
catalog.schema.object-name
```

ANSI names have a maximum of 128 characters for each part of the name (not including the dots).

ANSI names for SQL/MX objects

ANSI names for SQL/MX database objects are registered in the metadata in SQL/MX tables.

ANSI alias names for SQL/MP objects

Only ANSI name types are supported for SQL/MP database objects. Alias names are ANSI standard names that map to physical Guardian names for existing SQL/MP objects.

To add ANSI alias names for SQL/MP tables, use the command `CREATE SQLMP ALIAS`, which stores the alias name in the SQL/MX catalog. For more information about `CREATE SQLMP ALIAS` command, see the *SQL/MX 3.x Reference Manual*.

Unsupported ODBC APIs

The drivers are ODBC 3.5 compliant. Table 1 (page 15) lists the unsupported ODBC APIs.

Table 1 Unsupported ODBC APIs

Category	Unsupported APIs
Installer – Data Sources	• SQLManageDataSources • SQLRemoveDefaultDataSource • SQLConfigDataSource • SQLCreateDataSource • SQLValidDSN • SQLRemoveDSNFromIni • SQLWriteDSNToIni
Installer – Drivers	• SQLConfigDriver • SQLRemoveDriver • SQLInstallDriver • SQLInstallDriverEx • SQLValidDSN • SQLGetInstalledDrivers
Installer – Driver Manager	• SQLRemoveDriverManager • SQLInstallDriverManager
Installer – Installer Errors	• SQLInstallerError • SQLPostInstallerError
Installer – Profile Strings	• SQLWritePrivateProfileString • SQLGetPrivateProfileString
Installer – Translator	• SQLInstallTranslator • SQLInstallTranslatorEx • SQLRemoveTranslator • SQLGetTranslator
Installer – Config Mode	• SQLSetConfigMode • SQLGetConfigMode
Bulk operations	SQLBulkOperations
Scroll operations	• SQLFetchScroll supported only for SQL_FETCH_NEXT attribute • SQLSetScrollOption

Unsupported ODBC data types

The drivers do not support the BINARY, VARBINARY and LONG VARBINARY data types.

Considerations and limitations

The Unix drivers do not support the following:

- Module File Caching (MFC)
- GUI for configuring the driver manager

The following are the considerations and limitations for the Windows Unicode driver:

- You cannot execute DDL statements with character set qualifiers `_UCS2` or `N`.
- Prefix `_UCS2` or `N` to the string literal for the following columns to execute DML statements:
  - CHAR CHARACTER SET UCS2 column
  - NCHAR column — the default NATIONAL character set is UCS2
  - VARCHAR CHARACTER SET UCS2 column
  - NCHAR VARYING column — the default NATIONAL character set is UCS2

  For example,

  ```
  insert into CAT.SCH.TAB (unicodechar_col) values (_UCS2'abcd');
  ```
- The driver does not support KANJI and KSC5601 as NATIONAL character set. Prefix `_kanji` or `_ksc5601` to the string literal for NCHAR column if SQL/MX is installed with KANJI and KSC5601 as NATIONAL character set.
- SQL statements and `SQLBindParameter` values must be in UTF-16 format.
- The driver considers C variable of type SQL_C_DEFAULT as SQL_C_CHAR instead of SQL_C_WCHAR to match with the SQL data type of the column or parameter.
- Consider the following byte length for IN or OUT WCHAR parameter buffer length:
  - Number of characters * 2 for double byte Unicode characters
  - Number of characters * 4 for supplementary Unicode characters
- When the trace is on, the Unicode characters in the SQL statements may not get displayed properly in the log file.

2 Installing the drivers

The hardware and software requirements for the drivers are described in the softdoc or README file delivered with the products, either through the product CD or through the Scout website for NonStop servers in the HP NonStop eServices portal (https://onepoint.nonstop.compaq.com/).

This chapter describes the following:

- "Installation requirements"
- "Installation procedures"
- "Verifying the installation"

Installation requirements

HP-UX

A standard HP–UXia64 installation is required for the driver. If you want the applications to use the unixODBC driver manager, then download and install unixODBC driver manager release 2.2.14, or 2.3.1 and later. For installing unixODBC, see the unixODBC website http://www.unixODBC.org.

Linux 32

The Linux driver is LSB 4.1 compliant and is tested on RHEL 5.1, 6.0 and ubuntu 11.4. A standard Linux installation is required for the driver. If you want the applications to use the unixODBC driver manager, then download and install unixODBC driver manager release 2.2.14, or 2.3.1 and later. For installing unixODBC, see the unixODBC website http://www.unixODBC.org.

Linux 64

The 64 Bit Linux driver is LSB 4.1 compliant and is tested on 64 Bit RHEL 6.2, 6.3 and A standard Linux installation is required for the driver. If you want the applications to use the unixODBC driver manager, then download and install unixODBC driver manager release 2.2.14, or 2.3.1 and later. For installing unixODBC, see the unixODBC website http://www.unixODBC.org.

Windows

The following table lists the installation requirements for the Windows driver:

Component	Installation requirement
Memory	Minimum 32 MB
Disk space	Minimum 30 MB additional free space
Operating System	**ANSI driver:** Microsoft Windows NT 4.0, Windows 98, Windows 2000, Windows 2003 or Windows XP, Windows Vista, and Windows 7 **Unicode driver:** Windows 2003, Windows XP Service Pack2 or later, Windows Vista, and Windows 7
Driver Manager	Microsoft ODBC version 3.51 or later
Browser	Microsoft Internet Explorer 4.0 or later; HP recommends version 8.0

Installation procedures

HP-UX

Perform the following steps to install the driver:

1. Download the driver from NonStop system to the HP–UXia64 workstation in binary mode (You can use FTP utility for downloading). The driver is packaged as a tar file in the Software Update Tape (SUT) and is available on the NonStop system when DSM/SCM is used to install the SUT. The file name is `ODBC64HI`, and is located in the volume `$SYSTEM.ZMXODBC`.
2. Log on to the HP–UXia64 workstation as a root user.
3. Go to download directory, and enter the following command to extract the files:

   ```
   tar -xvf ODBC64HI
   ```

   The files are extracted to `HPUXia64_ODBC_64` directory. The install scripts and the README file are also extracted with the driver.
4. Go to the `HPUXia64_ODBC_64` directory.
5. To start the installation, enter the following command:

   ```
   ./runme.sh
   ```
6. Enter `1` and press `Enter`. If you have already installed the driver, a warning message appears.

The library files are installed in the `/usr/lib/hpux64` directory and a sample `MXODSN` file is copied to the `/etc/mxodbc` directory. Upon successful installation, the following message is displayed:

```
Installed !
```

Linux 32

Perform the following steps to install the driver:

1. Download the driver from NonStop system to the Linux workstation in binary mode (You can use FTP utility for downloading). The driver is packaged as a tar file in the Software Update Tape (SUT) and is available on the NonStop system when DSM/SCM is used to install the SUT. The file name is `LODBCTAR`, and is located in the volume `$SYSTEM.ZMXODBC`.
2. Log in to the Linux workstation as a root user.
3. Go to download directory, and enter the following command to extract the files:

   ```
   tar -xvf LODBCTAR
   ```

   The files are extracted to `Linux_ODBC_32` directory. The install scripts and the README file are also extracted with the driver.
4. Go to the `Linux_ODBC_32` directory.
5. To start the installation, enter the following command:

   ```
   ./runme.sh
   ```
6. Enter `1` and press `Enter`. If you have already installed the driver, a warning message appears.

The library files are installed in the `/usr/lib` directory and a sample `MXODSN` file is copied to the `/etc/hpodbc` directory. Upon successful installation, the following message is displayed:

```
Installed !
```

Linux 64

Perform the following steps to install the driver:

1. Download the driver from NonStop system to the Linux workstation in binary mode (You can use FTP utility for downloading). The driver is packaged as a tar file in the Software Update Tape (SUT) and is available on the NonStop system when DSM/SCM is used to install the SUT. The file name is `LODBC64`, and is located in the volume `$SYSTEM.ZMXODBC`.
2. Log in to the Linux workstation as a root user.

3. Go to download directory, and enter the following command to extract the files:
   `tar -xvf LODBC64`

   The files are extracted to `64Linux_ODBC_321` directory. The install scripts and the README file are also extracted with the driver.
4. Go to the `64Linux_ODBC_321` directory.
5. To start the installation, enter the following command: `./runme.sh`
6. Enter `1` and press `Enter`. If you have already installed the driver, a warning message appears.

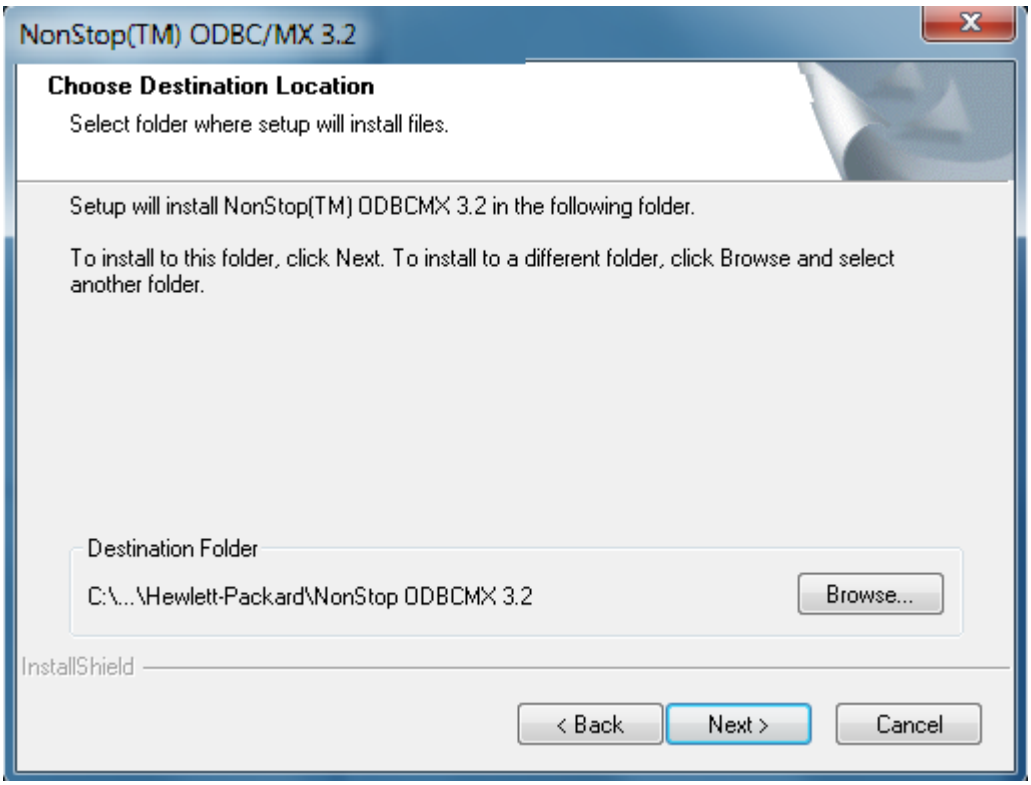
   The library files are installed in the `/usr/lib64` directory and a sample `MXODSN` file is copied to the `/etc/hpodbc` directory. Upon successful installation, the following message is displayed:

   ```
   Installed !
   ```

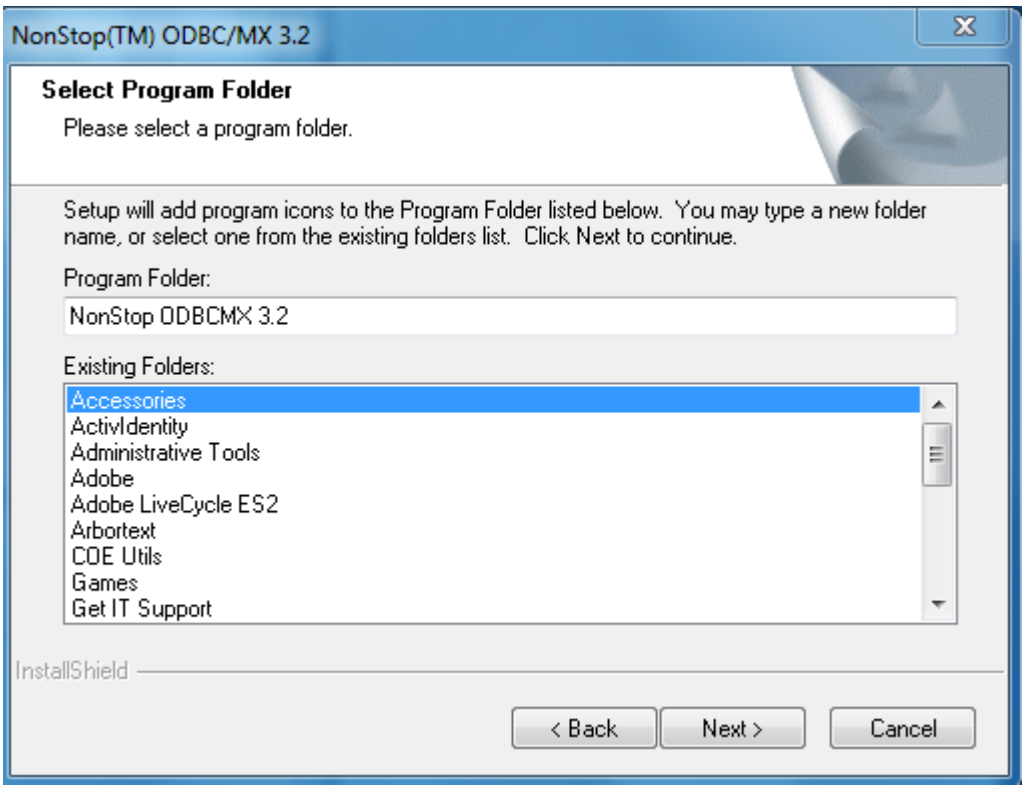
Windows

Perform the following steps to install the ANSI or Unicode drivers:

1. Close all other applications.
2. Log in to the Windows workstation as an administrator.
3. Create a temporary installation folder to download the installation file. For example, `C:\ODBCTEMP`.
4. Use FTP to download the following files from the `$SYSTEM.ZMXODBC` installation subvolume:
   - ANSI `TDMODBC` or the Unicode `ODBCW32` file for 32–bit driver install
   - ANSI `NSODBC64` or the Unicode `ODBCW64` file for 64–bit driver install
5. Rename the files with .EXE extension. For example, `TDMODBC.EXE`.
6. Double click `TDMODBC.EXE`, `NSODBC64.EXE`, `ODBCW32.EXE`, or `ODBCW64.EXE` to launch the Installation Wizard.
7. Click the **Next** button on the **Welcome** Screen.
8. Accept the License Agreement and click **Next**.
9. You can retain the default or change the **Destination Folder** on the following screen by clicking the **Browse** button. Click **Next**.

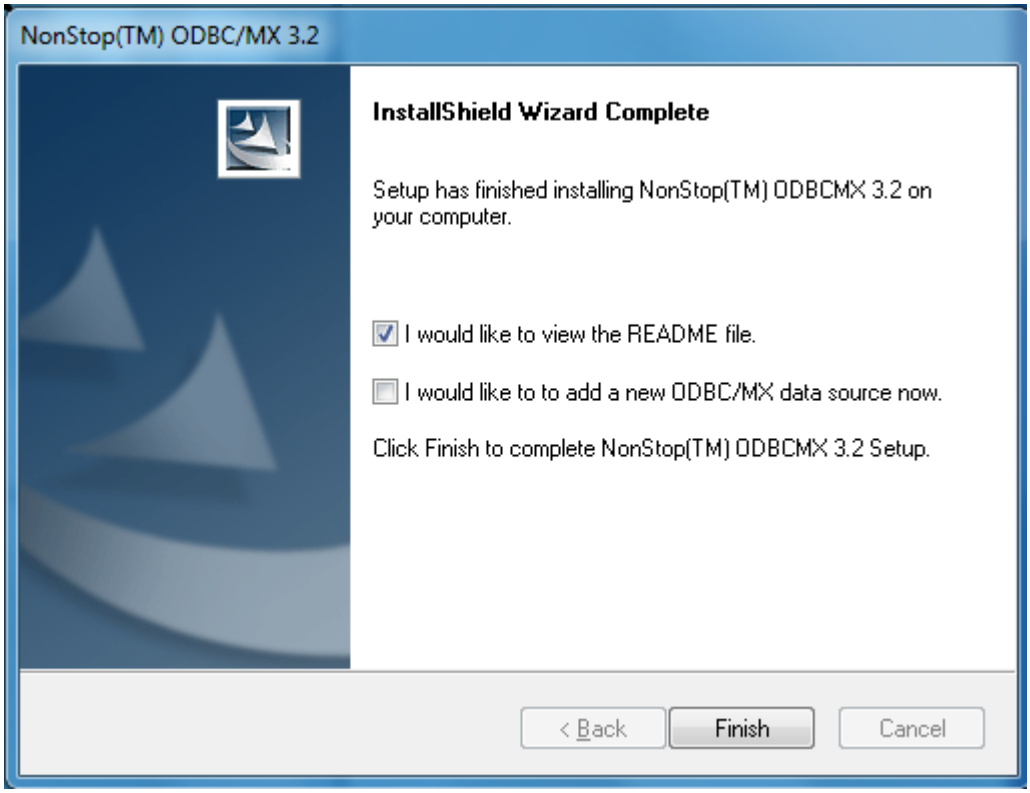


10. You can retain the default or provide a different **Program Folder** on the following screen:



Click **Next**.

11. You can see the installation progress in the **Setup Status** progress bar.
After a successful installation, the following screen appears:



12. Click **Finish**.

NOTE:

- HP recommends that you restart the PC before you use the ODBC/MX client interface.
- When the installation is complete, delete the temporary installation folder.

Verifying the installation

HP-UX

After a successful installation, the following files are installed:

- `/usr/lib/hpux64/libmxodbc_ia64.sl`
- `/usr/lib/hpux64/libmxodbc_ia64_drvr.sl`
- `/usr/lib/hpux64/libhpsecClient64.sl`
- `/etc/mxodbc/MXODSN`

The checksum of the shared libraries must match the checksum in the `md5sum` file. If the library is not present in the default location (`/usr/lib/hpux64`), ensure that the location of the driver (`libmxodbc_ia64.sl`, `libmxodbc_ia64_drvr.sl`, `libsecClient64.sl`) is set in the environment variable `SHLIB_PATH`.

To set the SHLIB_PATH variable, enter the following command:

```
export SHLIB_PATH = < directory, where the driver is located >
```

Linux 32

After a successful installation, the following files are installed:

- `/usr/lib/libmxodbc.so`
- `/usr/lib/libmxodbc.so.3`
- `/usr/lib/libmxodbc.so.3.2.1`
- `/usr/lib/libmxodbc_l_drvr.so`

- /usr/lib/libmxodbc_l_drvr.so.3
- /usr/lib/libmxodbc_l_drvr.so.3.2.1
- /usr/lib/libhpsecClient.so
- /etc/hpodbc/MXODSN

The checksum of the shared libraries must match the checksum in the md5sum file.

Linux 64

After a successful installation, the following files are installed:

- /usr/lib64/libmxodbc64.so
- /usr/lib64/libmxodbc64.so.3
- /usr/lib64/libmxodbc64.so.3.2.1
- /usr/lib64/libmxodbc64_l_drvr.so
- /usr/lib64/libmxodbc64_l_drvr.so.3
- /usr/lib64/libmxodbc64_l_drvr.so.3.2.1
- /usr/lib64/libhpsecClient64.so
- /etc/hpodbc/MXODSN

The checksum of the shared libraries must match the checksum in the md5sum file.

Windows

After a successful installation of ANSI driver, the following files are installed in Windows System folder:

- tdm_oadm0300.dll,
- tdm_odbc0300.dll
- tdm_odbcDrvMsg_intl0300.dll
- tdm_OdbcTrace0300.dll
- tdm_ores0300.dll
- tdm_tcpipv40300.dll
- tdm_tcpipv60300.dll
- tdm_translation.dll

After a successful installation of Unicode driver, the following files are installed in Windows System folder:

- tdm_oadmw0300.dll,
- tdm_odbcw0300.dll
- tdm_odbcDrvMsg_intlw0300.dll
- tdm_OdbcTracew0300.dll
- tdm_oresw0300.dll
- tdm_tcpipv4w0300.dll
- tdm_tcpipv6w0300.dll
- tdm_translationw.dll

Uninstalling the driver

HP-UX

Perform the following steps to uninstall the driver:

1. Log in to the HP-UXia64 workstation as a root user.
2. Go to the `HPUXia64_ODBC_64` directory.
3. To uninstall the driver, enter the following command:
   `./runme.sh`
4. Enter `2` and press `Enter`. Enter `yes` if you want to continue with uninstall operation.
5. Enter `yes` if you want to remove the configuration files.
6. Upon successful uninstallation, the following message is displayed:
   `UnInstalled !`

Linux 32

Perform the following steps to uninstall the driver:

1. Log in to the Linux workstation as a root user.
2. Go to the `Linux_ODBC_32` directory.
3. To uninstall the driver, enter the following command:
   `./runme.sh`
4. Enter `2` and press `Enter`. Enter `yes` if you want to continue with uninstall operation.
5. Enter `yes` if you want to remove the configuration files.
6. Upon successful uninstallation, the following message is displayed:
   `UnInstalled !`

Linux 64

Perform the following steps to uninstall the driver:

1. Log in to the Linux workstation as a root user.
2. Go to the `64Linux_ODBC_321` directory.
3. To uninstall the driver, enter the following command:
   `./runme.sh`
4. Enter `2` and press `Enter`. Enter `yes` if you want to continue with uninstall operation.
5. Enter `yes` if you want to remove the configuration files.
6. Upon successful uninstallation, the following message is displayed:
   `UnInstalled !`

Windows

You can uninstall the ANSI or Unicode client interface using one of the following methods:

- Select `Start -> Programs -> Release -> Remove Release`.
  `Release` can be `NonStop ODBCMX <version>` or `NonStop ODBCMX <version> Unicode`.
  Click **Yes**.
- Run `TDMODBC.EXE` or `NSODBC64.EXE` for ANSI driver, and `ODBCW32.EXE` or `ODBCW64.EXE` for Unicode driver, and select `Uninstall`.

NOTE: The uninstall operation deletes the driver, but it does not delete the driver manager or data sources. You cannot use these data sources until you reinstall the driver.

3 Configuring data sources

This chapter describes the various procedures to configure the SQL/MX data sources.

- "Configuring a client data source"
- "Configuring the drivers for IPv6"
- "Configuring a secure ODBC connection using NonStop SSL"
- "MXODSN and ODBCDSN file formats"
- "Managing data sources"
- "Managing transactions" (page 34)
- "Tracing"

Configuring a client data source

Unix

The MXODSN file on HP-UX and Linux, or the ODBCDSN file on OSS, which is a text file, is a repository of all client data sources.

You can place the MXODSN or ODBCDSN file in any one of the following locations; the search order for these files are also mentioned:

1. Directory specified in DSN_PATH environment variable
2. ODBC application directory
3. Default system location
   - On HP-UX, `/etc/mxodbc`
   - On Linux32 and Linux64, `/etc/hpodbc`
   - On OSS, `/G/system/system`, which is the OSS name for the Guardian location

To configure a client data source, edit the MXODSN or ODBCDSN file. Change this file so that the attributes are set to the correct values for your system (for example, names of data sources and attributes describing them).

The DSN in the MXODSN or ODBCDSN file must exactly match the server data source name (data source names are case sensitive). If the data source names do not match, the application gets a warning that connection is established with the default data source, `TDM_Default_DataSource`.

If you have not defined the CATALOG and SCHEMA attributes in the MXODSN or ODBCDSN files, the driver uses the default values.

Windows

Perform the following steps to add a client data source:

1. Select `Start -> Control Panel -> Administrative Tools -> Data Sources (ODBC)`.
2. Click **Add** to add a new data source. To configure a data source, get the list of data source names by clicking either the **System DSN** or the **User DSN** tab.
3. Choose **NonStop(TM) ODBCMX < Release >** from the list, and then click **Finish**. `< Release >` is *version* or `Unicode` *version*.
4. Enter the details in the **Create Data Source** screens.

Perform the following steps to configure a client data source:

1. Select `Start -> Control Panel -> Administrative Tools -> Data Sources (ODBC)`.
2. Click either the **System DSN** or the **User DSN** tab to get the list of data sources.

3. Select the data source name that you want to configure and click **Configure**.
4. Each tab in the configuration view displays different fields in the data source configuration. Make the required changes.

   For more information about the Microsoft ODBC Data Source Administrator screens and tabs, click **Help**.

NOTE:

- While creating a new data source, ensure the following special characters are not part of the data source name:

  \, /, :, *, ?, “, <, >, |
- When creating a new client data source, ensure that it matches that of a server data source name and have the matching case. If a client data source name does not exist on the server, the default server data source, `TDM_Default_DataSource` is used.

The following connection string is used to configure the client data source:

```
"DSN=< DataSource name>;catalog=<catalog_name>; schema=<schema_name>;
Uid=<userid>;Pwd=<password>;FETCHBUFFERSIZE=<value in
bytes>; SQLFRACTIONVALUE=<value>;ATTR_CONNECTION_TIMEOUT=<value in seconds>;
SQL_LOGIN_TIMEOUT=<value in seconds>;SQL_QUERY_TIMEOUT=<value in seconds>”;
```

For more information about the attributes, see `MXODSN` and `ODBCDSN` file formats.
Table 2 explains the driver connection attributes specific to Windows driver:

Table 2 Connection attributes

Attribute	Description
`DSN`	Name of the data source as returned by `SQLDataSources` API or the data sources dialog box of `SQLDriverConnect` API.
`FILEDSN`	Name of a `.dsn` file. The driver builds the connection string for the data source with the information stored in this file.
`SAVEFILE`	The file name of a `.dsn` file. The attribute values of keywords used for the successful connection are saved in this file.
`DRIVER`	Description of the driver as returned by the `SQLDrivers` API. For example, Rdb or SQL Server.
`ERRORMSGLANG`	Client error message language. This is equal to SYSTEM_DEFAULT. The default is ENGLISH.
`CTRLINFERNCHAR`	Used for setting Infer charset, default is SYSTEM_DEFAULT, which is the client local character set.
`TRANSLATIONDLL`	Translates the data from one character set to other. Set the `TRANSLATIONDLL` in either of the following ways: • `TRANSLATIONDLL=`*`tdm_translation.dll`* • Click **Browse** to select a DLL. The DLL you specify is used to map the character set for a particular language.
`TRANSLATIONOPTION`	Options supported by the translation DLL.

NOTE: If DSN attribute is not provided in the connection string, the driver uses the SERVERDSN as the default DSN.

Configuring the drivers for IPv6

HP-UX, Linux 32 and Linux 64

You can directly enter the IPv6 address instead of IPv4 address in the `MXODSN` or `odbc.ini` (when using the unixODBC driver manager) configuration files. The driver recognizes the IPv6 address format and connects to the server.

The following is a sample from the configuration file:

```
Server = TCP: 2620:0:a17:e03e:a00:8eff:fe08:cd65:3033
```

OSS

Perform the following steps for the driver to use IPv6:

1. Ensure that the TCPIP DEFINE is set to use IPv6 when you start the application. The value, IPv6 is case sensitive.
2. Add the TCPIP^PROCESS^NAME DEFINE.
   - Delete the existing DEFINE using the following command:

     ```
     del_define =TCPIP^PROCESS^NAME
     ```

   - Add the DEFINE using the following command:

     ```
     PARAM TCPIP^PROCESS^NAME $<TCPSAM-process-name>
     add_define =TCPIP^PROCESS^NAME, class MAP,
     file "<system-name>.$<TCPSAM-process-name>"
     ```

     For more information, see the *TCP/IPv6 Configuration and Management Manual*.
3. Start the OSH or OSS process. If the process is started directly, the environmental variable TCPIP_PROCESS_NAME must point to IPv6 process.

Windows

Perform the following steps for the ANSI or Unicode driver to use IPv6:

1. Select `Start -> Programs -> Release`.

   `Release` can be `NonStop ODBCMX <version>` or `NonStop ODBCMX <version> Unicode`.
2. Create or modify a DSN.
3. Select the **Network** tab to enter the IPv6 address in the **MXCS IP Address** field.

IPv6 is not supported for the driver through HP SSL. For more information about SSL, see the *HP NonStop SSL Reference Manual*.

NOTE: The SSL ODBC/MX and MXOAS processes must run on the same TCP/IP process. `TargetPort` must not be specified with HP NonStop SSL REMOTE PROXY SETUP macro.

Configuring a secure ODBC connection using NonStop SSL

Perform the following tasks to encrypt the ODBC connection by using NonStop SSL with Windows driver:

1. Install a NonStop SSL Server process for the ODBC/MX Association Server for the target MXCS subsystem.
2. Install the RemoteProxy software and configure Remote Proxy to route ODBC connections to the NonStop SSL Server process.
3. Modify the connection attributes of client data source for Windows drivers to connect to Remote Proxy.

Installing a NonStop SSL Server process for ODBC/MX

Installing a NonStop SSL Server process is restricted to the `super.super` user.

Perform the following steps on the NonStop server to install a NonStop SSL Server process for ODBC/MX:

1. Log on as `super.super`.
2. Enter the following command to change to the NonStop SSL directory:

   ```
   $SYSTEM STARTUP 1> volume $system.znsssl
   ```

3. Enter the following command at the TACL prompt to start the setup:

   ```
   $SYSTEM ZNSSSL 10> run $SYSTEM.ZNSSSL.SETUP
   ```

4. Enter `[7]` to select ODBC/MX server in run mode.
5. Enter the home terminal. The default value is `$YMIOP.#CLCI`.
6. Enter the CPU on which you want to run SSL. The default value is CPU 3.
7. Enter the SSL process name. The default value is `$ODBS3`.
8. Enter the TCP/IP process name for the subnet on which the ODBC/MX MXCS service runs. The default value is `$ZTC0`.
9. Enter a port number for the SSL ODBC/MX connection. The default port number is 8402.
10. Enter `y` or `n` to specify whether the startup and error messages must be sent to EMS.
11. Enter a name for the SCF IN file for the ODBCMXS configuration. The default value is `ODBSIN3`.
12. Enter a name for the SCF configuration file for the ODBCMXS configuration. The default value is `ODBSCF3`.
13. The following message indicates that SSL configuration successfully created the script files. These script files are used to configure and start the process as a kernel managed persistent process.

    ```
    **** SETUP has completed successfully ****
    Files created: <SCF IN file> and <SCF configuration file>.
    ```

14. Enter the following command to configure NonStop SSL as a persistent process:

    ```
    $SYSTEM ZNSSSL 11> SCF/IN <SCF IN filename>
    ```

    For example, if you use the default name, the command is `SCF/IN ODBSIN3`.
15. Enter the following command to start the process :

    ```
    $SYSTEM ZNSSSL 12> SCF START PROCESS $ZZKRN.#SSL-ODBCMXS-3
    ```

16. Enter the following command to check whether the process started correctly:

    ```
    $SYSTEM ZNSSSL 13> SHOWLOG ODBSLOG *
    ```

    The log file must contain a message similar to the following sample:

    ```
    $ODBS3|05Mar12 09:54:19.89|20|ODBC/MX server proxy started on target host 127.0.0.1,
    target port 23,
    source port 8402 $ODBS3|05Mar12 09:54:19.93|50|Performed action
    CombinedAction(CreateSocket,ListenSocketAction) successfully.
    ```

    The host NonStop SSL Server process is now running and available for client access.

For more information about installing the NonStop SSL Server process for ODBC/MX, see the *HP NonStop SSL Reference Manual*.

Installing and configuring the Remote Proxy Client

You must obtain the following information from your administrator before installing the Remote Proxy Client:

- The IP address, host name, and port number of the SSL server running on the NonStop system.
- The port number of the MXCS Association Server that you want to connect.

NOTE: The MXCS Association Server port number must not be in use by any other program or service on your client workstation.

For information about obtaining the port number, see the *SQL/MX Connectivity Service Administrative Command Reference*.

Perform the following steps to install and configure the Remote Proxy client on Windows workstation:

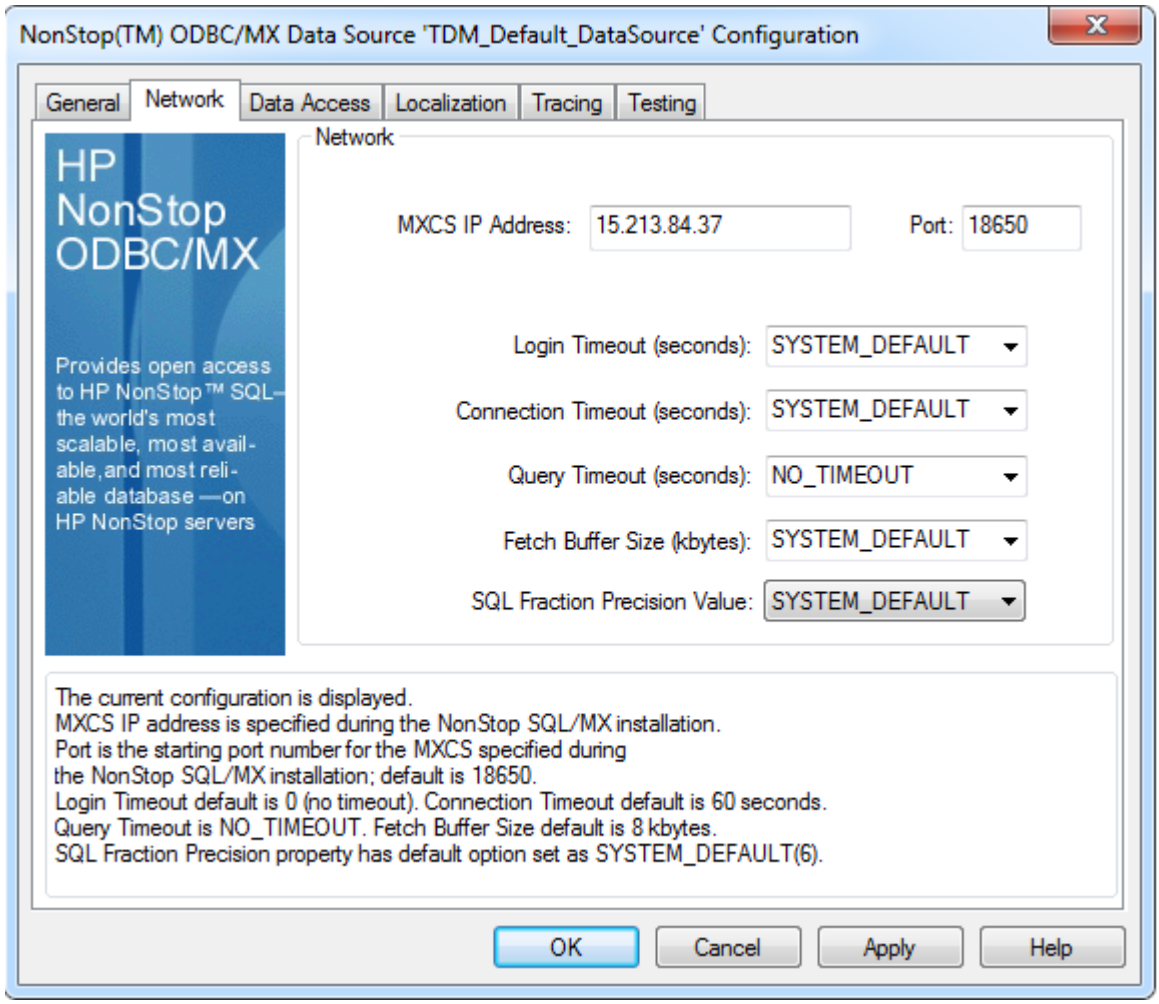
1. Download the `$SYSTEM.ZNSSSL.PROXYEXE` file in binary format and rename it to `PROXY.EXE`.
2. Run `PROXY.EXE` to start the RemoteProxy installation program, and follow the installation instructions in the wizards.
3. Double-click the **NonStop SSL RemoteProxy** icon in your system tray. The RemoteProxy configuration window is displayed.
4. Select **Session –> New**. The HP NonStop SSL RemoteProxy dialog is displayed.
    1. Select **ODBCMX Client** in the **Protocol** menu.
    2. Enter the IP address or host name of the system where the MXCS Association Server process is running, in the `Target Information` field.
    3. Enter the port number of the SSL server in the `Target (Connecting) Port` field. For example, 8402.
    4. Enter the port number of the MXCS Association Server in the `Local (Accepting) Port` field.
5. Click **Start** to start the Remote Proxy session.
6. Check the startup messages to verify that the Remote Proxy session started successfully.

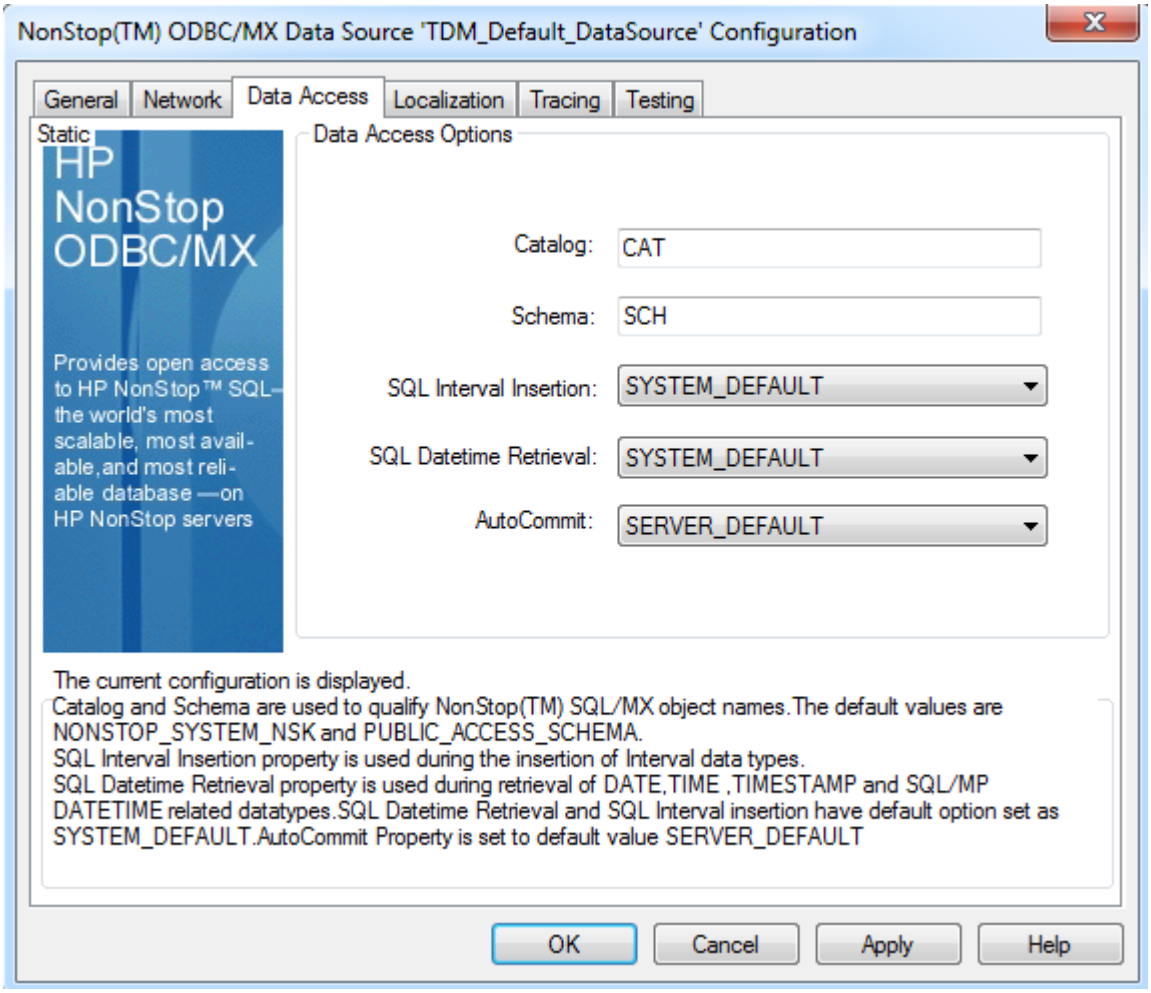
Configuring a client data source for NonStop SSL

Perform the following steps to configure a client data source on your workstation:

1. Select `Start -> Control Panel -> Administrative Tools -> Data Sources (ODBC).`
2. Click either the **System DSN** or the **User DSN** tab to get the list of data sources.
3. Select the data source name that you want to configure and click **Configure**.
4. Change the `MXCS IP Address` in **Network** tab to `localhost` in the screen shot after table 3.
5. Go to **Testing** tab and click **Test Connection** to test the connection. Upon successful connection, `Connected Successfully` message is displayed.

DataSource configuration in Windows driver





MXODSN and ODBCDSN file formats

The following table describes the various sections in the MXODSN and ODBCDSN files.

Table File sections

File Section	Description
[ODBC]	This section lists the tracing options.
[ODBC Data Sources]	This section lists the data sources.
Data source specification	Each DSN listed in the [ODBC Data Sources] section must have this section that describes the DSN.
[DataSourceName]	Associates a driver with the DSN.

Table 4 describes the MXODSN and ODBCDSN attributes.

Table 4 MXODSN and ODBCDSN attributes

File Section	Attributes	Description
[ODBC]	TraceStart	TraceStart can have two values, 0 or 1. If set to 0, tracing is off. If set to

Table 4 MXODSN and ODBCDSN attributes *(continued)*

File Section	Attributes	Description
		1, tracing is on. By default, tracing is off.
	`TraceFlags`	`TraceFlags` indicates the trace level. It can have the following values: • ERROR to trace failed SQL calls and communication problems. • WARNING to trace the warnings. • CONFIG to trace configuration calls. • INFO to trace details about calls made. • DEBUG to trace the internal details about calls made for debugging an issue.
	`TraceFile`	`TraceFile` attribute indicates the trace file name. The trace file is located in the ODBC application directory. For example, if you specify the `TraceFile` as `Tracefile = trfetch`, the trace file is created as `trfetch.6947088_1285815329` on Linux systems.
[ODBC Data Sources]	`DSN = description` [DataSourceName]	List of data sources, in the format `DSN = description`. [DataSourceName] attribute indicates the driver.
[*data-source-name*] , one section for each DSN	`Description`	`Description` attribute describes the data source.
	`DataLang`	`DataLang` attribute indicates the character set for SQL commands. The default is 0.
	`FetchBufferSize`	`FetchBufferSize` indicates the buffer size to fetch rows. The range is between 0 and 256 Kb. If set to SYSTEM_DEFAULT, the buffer size is 8 Kb (SYSTEM_DEFAULT). For a single row, the value is 0.
	`Server`	`Server` attribute indicates the IP address or domain name where MXCS is running. The format is TCP:*number1*`:`*number2*.
	`SQLFractionValue`	`SQLFractionValue` indicates the fraction value for SQL_C_TIMESTAMP to be in nanoseconds or microseconds. If set to 9, the fraction value is in nanoseconds. If set to SQL_DEFAULT, not specified, or set to a value other than 9, the fraction value is in microseconds. For more information, see "Timestamp values with fraction" (page 42)
	`SQLIntervalBehaviour`	`SQLIntervalBehaviour` includes zeroes at the beginning or at end of an interval datatype for micro or nano

Table 4 MXODSN and ODBCDSN attributes *(continued)*

File Section	Attributes	Description
		seconds. `SQLIntervalBehaviour` has two values, SYSTEM_DEFAULT and MSDN_DEFAULT. If not specified, the default is SYSTEM_DEFAULT. For more information, see "SQL Interval Behaviour property" (page 46).
	`SQLDatetimeRetrieval`	`SQLDatetimeRetrieval` specifies how the SQL/MP Datetime data type is retrieved from the driver. `SQLDatetimeRetrieval` has two values, SYSTEM_DEFAULT and MSDN_DEFAULT. If not specified, the default is SYSTEM_DEFAULT. For more information, see "SQL Datetime Retrieval property" (page 48).
	`SQL_ATTR _CONNECTION _TIMEOUT`	`SQL_ATTR_CONNECTION_TIMEOUT` indicates the wait time to close an idle connection. If set to SYSTEM_DEFAULT, the value is 60 seconds. If set to -1, use the `ConnTimeout` value set on the MXCS server data source. If set to NO_TIMEOUT, the driver waits until a ODBC server becomes available or MXCS is stopped.
	`SQL_LOGIN _TIMEOUT`	`SQL_LOGIN_TIMEOUT` indicates the wait time to establish a connection with MXCS. If set to SYSTEM_DEFAULT, the value is 60 seconds. If set to NO_TIMEOUT, the driver waits until a connection is established with MXCS.
	`SQL_QUERY _TIMEOUT`	`SQL_QUERY_TIMEOUT` indicates the wait time to close the cursor and return control to the ODBC application. If set to SYSTEM_DEFAULT, the value is 60 seconds. If set to NO_TIMEOUT, the driver waits till the query completes.
	`Catalog`	`Catalog` indicates the catalog name. If not specified, the driver uses the MXCS defined catalog name.
	`Schema`	`Schema` indicates the schema name. If not specified, the driver uses the MXCS defined schema name.
	`AUTOCOMMIT`	The values for `AutoCommit` can be `SERVER_DEFAULT`, `ON`, and `OFF`. For more information, see "Managing transactions" (page 34).
[DataSourceName]	`Driver = driver`	List of drivers. This value must match the driver name defined by `DataSourceName` attribute in the [ODBC Data Sources] section.

The following sample `MXODSN` file defines connections to three client data sources; `TDM_Default_DataSource`, `DS1`, and `DS2`:

In this example, the attributes `Catalog` and `Schema` for data source `DS1` are defined as `DS1CAT` and `DS1SCH`.

```
[ODBC]
TraceFlags = 6
TraceStart = 0
TraceFile = trlog
[ODBC Data Sources]
TDM_Default_DataSource = NonStop ODBC/MX 3.x
DS1 = NonStop ODBC/MX 3.x
DS2 = NonStop ODBC/MX 3.x
DataSourceName = Driver
[TDM_Default_DataSource]
Description = Default Data Source
Catalog = CAT
Schema = SCH
DataLang = 0
FetchBufferSize = SYSTEM_DEFAULT
Server = TCP:xxx.xxx.xxx.xxx:xxxx
SQL_ATTR_CONNECTION_TIMEOUT = SYSTEM_DEFAULT
SQL_LOGIN_TIMEOUT = SYSTEM_DEFAULT
SQL_QUERY_TIMEOUT= NO_TIMEOUT
[DS1]
Description = Sample Data Source 1
Catalog = DS1CAT
Schema = DS1SCH
Server = TCP:xxx.xxx.xxx.xxx:xxxx
SQLIntervalBehaviour = SYSTEM_DEFAULT
SQLDatetimeRetrieval = SYSTEM_DEFAULT
[DS2]
Description = Sample Data Source 2
Catalog = DS2CAT
Schema = DS2SCH
SQLIntervalBehaviour = SYSTEM_DEFAULT
[DataSourceName]
Driver = NonStop ODBC/MX
```

Managing data sources

Unix

To add a new DSN entry in the `MXODSN` or `ODBCDSN` file, include the DSN and its description. If the values of the attributes are not specified in the `MXODSN` or `ODBCDSN` file, the driver uses the default values when creating a connection.

Perform the following steps to modify or delete a DSN entry:

1. Modify the attributes or remove the DSN entry, and save the file.
2. Restart the application to reflect the changes.

Windows

To add a new data source or configure an existing data source, see "Configuring a client data source".

Perform the following steps to delete a client data source:

1. Select `Start -> Control Panel -> Administrative Tools -> Data Sources (ODBC)`.
2. Click either the **System DSN** or the **User DSN** tab to get the list of data sources.
3. Select the data source name that you want to delete and click **Remove**.
4. Click **Yes** in the confirmation dialog box to remove the data source.

Managing transactions

The `AutoCommit` property indicates whether SQL/MX must automatically commit the transaction or not.

The following are the valid values for `AutoCommit` property:

- **ON**
- **OFF**
- **SERVER_DEFAULT** specifies to use the server side `AutoCommit` settings. This option is useful for existing applications where `AutoCommit` is set on the server side data source and the user wants to persist with the same settings without making any changes to the client application or client side data source. This is the default option.

`AutoCommit` property can be set in the following ways:

- Using `SQLSetConnectAttr` API, the values are `ON/OFF`.
- Using DSN properties, the values are `SERVER_DEFAULT/ON/OFF`.
- Using connection string without DSN, the values are `ON/OFF/SERVER_DEFAULT`.

NOTE: The value set using `SQLSetConnectAttr` API takes the highest precedence.

Setting `AutoCommit`

Windows driver

You can set the `AutoCommit` property from the **Data Access** tab.

`Unix` drivers

AUTOCOMMIT (case sensitive) is the keyword to set `AutoCommit` property in property file (MXODSN for Linux and HP-UX, ODBCDSN for OSS).

Example 1 Depicting AUTOCOMMIT set to SERVER_DEFAULT

```
[TDM_Default_DataSource]
Description                    = Default Data Source
Catalog                        = CAT
Schema                         = SCH
DataLang                       = 0
FetchBufferSize                = SYSTEM_DEFAULT
Server                         = TCP:hostname/portnumber
SQL_ATTR_CONNECTION_TIMEOUT    = SYSTEM_DEFAULT
SQL_LOGIN_TIMEOUT              = SYSTEM_DEFAULT
SQL_QUERY_TIMEOUT             = NO_TIMEOUT
SQLFractionValue               = SYSTEM_DEFAULT
SQLDatetimeRetrieval          = MSDN_DEFAULT
AUTOCOMMIT       = SERVER_DEFAULT
```

Using connection string without DSN

You can set AUTOCOMMIT as part of the Connection string.

"AutoCommit=ON/OFF/SERVER_DEFAULT".

Example 2 Depicting AUTOCOMMIT set to ON

```
Driver={NonStop(TM)ODBCMX 3.2};
ServerDSN = TDM_Default_DataSource;
SERVER =TCP:<hostname/port>;
UID=<userid>;
PWD=<password>;
AutoCommit=ON;
..;..;..;..;
```

NOTE: For Windows driver, if `AutoCommit` property is set to `OFF` without using `SQLSetConnectAttrAPI`, then the application must execute the commit/rollback as a statement to end an active transaction. `SQLEndTrans` API cannot be used in this scenario.

Tracing

Unix

You can enable or disable tracing by modifying the `TraceStart` attribute in `MXODSN` or `ODBCDSN` file. For more information about tracing options, see MXODSN and ODBCDSN attributes.

NOTE: If tracing is enabled, the global trace flag is set and from then onwards, all the traceable ODBC data is logged in the trace file.

Perform the following steps to enable tracing:

1. Stop the application.
2. Set the `TraceStart` attribute to 1 in the `MXODSN` or `ODBCDSN` file, and save.
3. Start the application.

Perform the following steps to disable tracing:

1. Stop the application.
2. Set the `TraceStart` attribute to 0 in the `MXODSN` or `ODBCDSN` file, and save.
3. Start the application.

Windows

Perform the following steps to configure tracing for ANSI or Unicode drivers:

1. Select `Start -> Control Panel -> Administrative Tools -> Data Sources (ODBC).`
2. Click either the **System DSN** or the **User DSN** tab to get the list of data sources.
3. Select the data source name and click **Configure**.
4. Select the **Tracing** tab.
5. Choose **NonStop ODBCMX API** or **Network Layer**, or both.
6. Enter a log file name.
7. Click **Apply** and then click **OK**.

Perform the following steps to start tracing:

1. Select `Start -> Control Panel -> Administrative Tools -> Data Sources (ODBC).`
2. Select the **Tracing** tab.
3. Select the Custom Trace DLL. Use the **Select DLL** button to locate the correct file.

If you select the ODBC standard trace DLL (`odbctrac.dll` from the Windows System folder), only the driver manager is traced; additional trace options that you select are ignored.

If you select the ODBC/MX trace DLL (`tdm_odbctrace0300.dll` or `tdm_odbctracew0300.dll` from the Windows System folder), all trace options that you select for the client data source are considered.

4. Use the **Browse** button to locate the correct log file.
5. Click **Start Tracing Now**.
6. Click **Apply** and then click **OK**.

Perform the following steps to stop tracing:

1. Select `Start -> Control Panel -> Administrative Tools -> Data Sources (ODBC).`
2. Select the **Tracing** tab.
3. Click **Stop Tracing Now**.
4. Click **Apply** and then click **OK**.

4 Compatibility and considerations

The drivers are compatible with the following:

- ODBC 3.5
- SQL-92 features supported by SQL/MX

For more information about SQL/MX supported scalar and string functions, see the *SQL/MX Reference Manual for Release 3.x*.

These subsequent sections list the compatibility of the drivers for the following:

- "Considerations" (page 37)
- "ODBC data types" (page 38)
- "Unsigned data types" (page 39)
- "Partial DATE or TIME values" (page 40)
- "Microsoft escape clauses" (page 41)
- "Stored Procedures" (page 41)
- "Transaction and cursor behavior" (page 41)
- "Timestamp values with fraction" (page 42)

Considerations

Table 5 lists the considerations.

Table 5 Considerations

Item	Consideration
ODBC/MX client application	An ODBC client application must be installed on the client workstation.
Catalog and schema	An application can set the catalog and schema information in the `MXODSN` file on Linux or `ODBCDSN` file on OSS. If the application does not set the catalog and schema for the client data source, and no default catalog and schema are defined in MXCS for the server data source, then MXCS, by default, uses NONSTOP_SYSTEM_NSK and PUBLIC_ACCESS_SCHEMA. **NOTE:** By default, catalog and schema names are case insensitive and follow SQL/MX rules. Case sensitive catalog and schema names can be specified within quotes.
`SQL_QUERY_TIMEOUT` option	An application can call `SQLSetStmtOption` ODBC API with the `SQL_QUERY_TIMEOUT` option to specify the wait time for query execution. The `SQL_QUERY_TIMEOUT` option protects MXCS and the application from long running queries. If a query exceeds the specified time before the data source returns the result, the driver returns HYT00 (Timeout expired) to the application.
`SQLCancel` API	For an ODBC 2.0 application, when the query is not running, the driver implements the `SQLCancel` API as an `SQLFreeStmt` with the `SQL_CLOSE` option. If the query is running, MXCS tries to stop the ODBC server. If the attempt is successful, the application receives an error and must reconnect. For an ODBC 3.x application with a query (or queries) running, the behavior is the same as for a ODBC 2.0 application. For an ODBC 3.x application with no queries running, the driver does not implement the `SQLCancel` API. The application is responsible for closing the cursor.
`SQLPrimaryKeys` API	This API does not support pattern value arguments for table names, and returns keys information for a specified table name in the SQL/MX catalog.

Table 5 Considerations *(continued)*

Item	Consideration
`SQLSpecialColumns` API	This API does not support pattern value arguments for table names, and returns an optimal set of columns that the `SQLSpecialColumns` API uniquely identifies a row in the table information for a specified table name in the SQL/MX catalog.
`SQLStatistics` API	This API does not support pattern value arguments for table names, and returns index information for a specified table name in the SQL/MX catalog.
`SQLTables` API When `CatalogName` is SQL_ALL_CATALOGS	Returns all catalogs defined in the SQL/MX metadata.
`SQLTables` API When `SchemaName` is SQL_ALL_SCHEMAS	Returns all the schemas defined in the SQL/MX metadata.
`SQLTables` API When `TableType` is SQL_ALL_TABLE_TYPES	Returns all object types defined in the SQL/MX metadata.
SQL Table Names, SQL Catalog and Schema Names	The drivers use SQL_ATTR_METADATA_ID to determine whether to upshift a table name. This applies for catalog APIs `SQLTables`, `SQLPrimaryKey`, and `SQLStatistics`. This does not apply to `SQLColumns` API.
`SQLBindParameter` API	For this API, `SQL_DEFAULT_PARAM` and `SQL_LEN_DATA_AT_EXEC` (length) are not supported for the `pcbValue` argument.
`SQLPutData` API	This API can send data in parts only for the SQL_LONGVARCHAR data type. For all other data types, only the latest value in the `rgbValue` argument is sent.

For information on migrating SQL/MP ODBC client applications to SQL/MX environment, see *SQL/MX Comparison Guide for NonStop SQL/MP Users*.

ODBC data types

Table 6 lists the compatibility of drivers for ODBC data types.

Table 6 ODBC Data Types

ODBC data type	SQL/MX data Type	SQL/MP data Type	Supported by the driver?
CHAR(n)	CHAR(n)	CHAR(n)	Yes
VARCHAR(n)	VARCHAR(n)*	VARCHAR(n)*	Yes
LONGVARCHAR	VARCHAR(n)	VARCHAR(n)	Yes
DECIMAL(p,s)	DECIMAL(p,s)	DECIMAL(p,s)	Yes
NUMERIC(p,s)	NUMERIC(p,s)**	NUMERIC(p,s)	Yes
SMALLINT	SMALLINT	SMALLINT	Yes
INTEGER	INTEGER	INTEGER	Yes
REAL	REAL	REAL	Yes
FLOAT(p)	FLOAT(p)	FLOAT(p)	Yes
DOUBLE PRECISION	DOUBLE PRECISION	DOUBLE PRECISION	Yes
BIT	Not supported	Not supported	No
TINYINT	Not supported	Not supported	No

Table 6 ODBC Data Types *(continued)*

ODBC data type	SQL/MX data Type	SQL/MP data Type	Supported by the driver?
TINYINT UNSIGNED	Not supported	Not supported	No
BIGINT	LARGEINT	LARGEINT	Yes
BINARY(n)	Not supported	Not supported	No
VARBINARY(n)	Not supported	Not supported	No
LONG VARBINARY	Not supported	Not supported	No
DATE	DATE	DATE	Yes
TIME(p)	TIME	TIME	Yes
TIMESTAMP	TIMESTAMP	TIMESTAMP	Yes
INTERVAL MONTH(p)	INTERVAL MONTH(p)		Yes
INTERVAL YEAR(p)	INTERVAL YEAR(p)		Yes
INTERVAL YEAR(p) TO MONTH	INTERVAL YEAR(p) TO MONTH		Yes
INTERVAL DAY(p)	INTERVAL DAY(p)		Yes
INTERVAL HOUR(p)	INTERVAL HOUR(p)		Yes
INTERVAL MINUTE(p)	INTERVAL MINUTE(p)		Yes
INTERVAL SECOND(p)	INTERVAL SECOND(p)		Yes
INTERVAL DAY(p) TO HOUR	INTERVAL DAY(p) TO HOUR		Yes
INTERVAL DAY(p) TO MINUTE	INTERVAL DAY(p) TO MINUTE		Yes
INTERVAL DAY(p) TO SECOND	INTERVAL DAY(p) TO SECOND		Yes
INTERVAL HOUR(p) TO MINUTE	INTERVAL HOUR(p) TO MINUTE		Yes
INTERVAL HOUR(p) TO SECOND(q)	INTERVAL HOUR(p) TO SECOND(q)		Yes
INTERVAL MINUTE(p) TO SECOND(q)	INTERVAL MINUTE(p) TO SECOND(q)		Yes

* The default size allocated for VARCHAR length is 4K (4096). The VARCHAR length can be up to 4059 for key-sequenced files or 4070 for entry-sequenced files.

** Starting with SQL/MX Release 3.0, the precision of the NUMERIC data type is extended to 128 digits. If the precision is greater than 18 for signed and greater than 9 for the unsigned numeric data type, use the `SQLBindParameter` API to insert the data and the `SQLBindCol` API to select the data with the SQL_C_CHAR data type.

Unsigned data types

The behavior of unsigned data types is determined by the ODBC version and the `SQL_ATTR_MSACCESS_VERSION` flag setting. Table 7 explains the behavior of unsigned data types.

Table 7 Behavior of unsigned data types

ODBC version	Behavior
2.0	UNSIGNED SMALLINT is promoted to SIGNED INT and UNSIGNED INT is promoted to BIGINT.
3.x, when server data source is configured with `SQL_ATTR_MSACCESS_VERSION` flag	All unsigned data types are promoted to the next signed type.

Partial DATE or TIME values

To use partial DATE or TIME values as parameters, provide these values through the ODBC/MX data type returned.

To fetch partial DATE or TIME values, see the mappings in Table 8 (page 40). When you fetch partial DATE or TIME value character strings, the values are returned as partial even if the driver returns a full DATE or TIME or TIMESTAMP data type. Default values are returned as a structure (1 for YEAR, MONTH and DAY, and 0 for HOUR, MINUTE, SECOND and FRACTION).

Table 8 SQL/MP to ODBC/MX Date/Time Mappings

SQL/MP data type	ODBC/MX data type
DATETIME YEAR	SQL_DATE
DATETIME YEAR TO MONTH	SQL_DATE
DATETIME YEAR TO DAY	SQL_DATE
DATETIME YEAR TO HOUR	SQL_TIMESTAMP
DATETIME YEAR TO MINUTE	SQL_TIMESTAMP
DATETIME YEAR TO SECOND	SQL_TIMESTAMP
DATETIME YEAR TO FRACTION	SQL_TIMESTAMP
DATETIME MONTH	SQL_DATE
DATETIME MONTH TO DAY	SQL_DATE
DATETIME MONTH TO HOUR	SQL_TIMESTAMP
DATETIME MONTH TO MINUTE	SQL_TIMESTAMP
DATETIME MONTH TO SECOND	SQL_TIMESTAMP
DATETIME MONTH TO FRACTION	SQL_TIMESTAMP
DATETIME DAY	SQL_DATE
DATETIME DAY TO HOUR	SQL_TIMESTAMP
DATETIME DAY TO MINUTE	SQL_TIMESTAMP
DATETIME DAY TO SECOND	SQL_TIMESTAMP
DATETIME DAY TO FRACTION	SQL_TIMESTAMP
DATETIME HOUR	SQL_TIME
DATETIME HOUR TO MINUTE	SQL_TIME
DATETIME HOUR TO SECOND	SQL_TIME
DATETIME HOUR TO FRACTION	SQL_TIMESTAMP
DATETIME MINUTE	SQL_TIME
DATETIME MINUTE TO SECOND	SQL_TIME

Table 8 SQL/MP to ODBC/MX Date/Time Mappings *(continued)*

SQL/MP data type	ODBC/MX data type
DATETIME MINUTE TO FRACTION	SQL_TIMESTAMP
DATETIME SECOND	SQL_TIME
DATETIME SECOND TO FRACTION	SQL_TIMESTAMP
DATETIME VALUE RETRIEVAL	SQL DATETIME RETRIEVAL

Microsoft escape clauses

ODBC/MX accepts Microsoft escape clauses and translates them into equivalent SQL/MX clauses. Table 9 lists the SQL/MX equivalents.

Table 9 SQL/MX equivalents for Microsoft escape clauses

Microsoft escape clause	SQL/MX equivalent
`{ d 'date-literal' }`	`DATE 'date-literal'`
`{ t 'time-literal' }`	`TIME 'time-literal'`
`{ ts 'timestamp-literal' }`	`TIMESTAMP 'timestamp-literal'`
`{ oj join-expression }`	`join-expression` *
`{ INTERVAL sign interval-string interval-qualifier }`	`INTERVAL sign interval-string interval-qualifier`
`{ fn scalar-function }`	`scalar-function` **
`{ call procedure-name... }`	Supported
`{ escape 'escape-character' }`	Not supported
`{ [?=]call procedure-name... }`	Not supported

* ODBC syntax does not include nested joins, while SQL/MX does. ODBC/MX extends the Microsoft syntax for an outer join.

** Functions are controlled by `SQLGetInfo`. Only SQL/MX native functions are supported.

Stored Procedures

The drivers support NonStop SQL/MX stored procedures, with the following exceptions:

Pointers to cursors, and return codes are not supported

For more information about stored procedures, see the *SQL/MX Guide to Stored Procedures in Java*.

Transaction and cursor behavior

The transaction and cursor behavior is determined by the AUTOCOMMIT setting. Table 10 lists the transaction and cursor behavior.

Table 10 Transaction and cursor behavior

AUTOCOMMIT	SQL/MX behavior	Application action
ON	When any open statement reaches end of data or end of cursor, SQL/MX closes all other open statements.	Not applicable
OFF	Not applicable	The application must explicitly rollback or commit the transaction. All open statements are closed at that time.

Timestamp values with fraction

The SQLFractionValue attribute indicates whether the fraction value for SQL_C_TIMESTAMP data type must be in nanoseconds or microseconds. If set to 9, the fraction value is in nanoseconds. If set to SQL_DEFAULT, not specified, or set to a value other than 9, the fraction value is in microseconds.

Table 11 includes a set of sample values and expected values in the column of an SQL/MX table in microseconds.

Table 11 Sample values inserted from an application and expected values in microsecond

Fraction value in the application	Column type	Expected value in the column of SQL/MX table
987654	timestamp(6) or time(6)	0.987654
87654	timestamp(6) or time(6)	0.087654
7654	timestamp(6) or time(6)	0.007654
654	timestamp(6) or time(6)	0.000654
54	timestamp(6) or time(6)	0.000054
4	timestamp(6) or time(6)	0.000004
987654	timestamp(5) or time(5)	0.98765
87654	timestamp(5) or time(5)	0.08765
7654	timestamp(5) or time(5)	0.00765
654	timestamp(5) or time(5)	0.00065
54	timestamp(5) or time(5)	0.00005
4	timestamp(5) or time(5)	0.00000
987654	timestamp(4) or time(4)	0.9876
87654	timestamp(4) or time(4)	0.0876
7654	timestamp(4) or time(4)	0.0076
654	timestamp(4) or time(4)	0.0006
54	timestamp(4) or time(4)	0.0000
4	timestamp(4) or time(4)	0.0000
987654	timestamp(3) or time(3)	0.987
87654	timestamp(3) or time(3)	0.087
7654	timestamp(3) or time(3)	0.007
654	timestamp(3) or time(3)	0.000

Table 11 Sample values inserted from an application and expected values in microsecond *(continued)*

Fraction value in the application	Column type	Expected value in the column of SQL/MX table
54	timestamp(3) or time(3)	0.000
4	timestamp(3) or time(3)	0.000
987654	timestamp(2) or time(2)	0.98
87654	timestamp(2) or time(2)	0.08
7654	timestamp(2) or time(2)	0.00
654	timestamp(2) or time(2)	0.00
54	timestamp(2) or time(2)	0.00
4	timestamp(2) or time(2)	0.00
987654	timestamp(1) or time(1)	0.9
87654	timestamp(1) or time(1)	0.0
7654	timestamp(1) or time(1)	0.0
654	timestamp(1) or time(1)	0.0
54	timestamp(1) or time(1)	0.0
4	timestamp(1) or time(1)	0.0

Table 12 includes a set of sample values and expected values in the column of an SQL/MX table in nanoseconds.

Table 12 Sample values inserted from an application and expected values in nanoseconds

Fraction value in the application	Column type	Expected value in the column of SQL/MX table
987654321	timestamp(6) or time(6)	0.987654
87654321	timestamp(6) or time(6)	0.087654
7654321	timestamp(6) or time(6)	0.007654
654321	timestamp(6) or time(6)	0.000654
54321	timestamp(6) or time(6)	0.000054
4321	timestamp(6) or time(6)	0.000004
987654321	timestamp(5) or time(5)	0.98765
87654321	timestamp(5) or time(5)	0.08765
7654321	timestamp(5) or time(5)	0.00765
654321	timestamp(5) or time(5)	0.00065
54321	timestamp(5) or time(5)	0.00005
4321	timestamp(5) or time(5)	0.00000
987654321	timestamp(4) or time(4)	0.9876
87654321	timestamp(4) or time(4)	0.0876
7654321	timestamp(4) or time(4)	0.0076
654321	timestamp(4) or time(4)	0.0006
54321	timestamp(4) or time(4)	0.0000

Table 12 Sample values inserted from an application and expected values in nanoseconds *(continued)*

Fraction value in the application	Column type	Expected value in the column of SQL/MX table
4321	timestamp(4) or time(4)	0.0000
987654321	timestamp(3) or time(3)	0.987
87654321	timestamp(3) or time(3)	0.087
7654321	timestamp(3) or time(3)	0.007
654321	timestamp(3) or time(3)	0.000
54321	timestamp(3) or time(3)	0.000
4321	timestamp(3) or time(3)	0.000
987654321	timestamp(2) or time(2)	0.98
87654321	timestamp(2) or time(2)	0.08
7654321	timestamp(2) or time(2)	0.00
654321	timestamp(2) or time(2)	0.00
54321	timestamp(2) or time(2)	0.00
4321	timestamp(2) or time(2)	0.00
987654321	timestamp(1) or time(1)	0.9
87654321	timestamp(1) or time(1)	0.0
7654321	timestamp(1) or time(1)	0.0
654321	timestamp(1) or time(1)	0.0
54321	timestamp(1) or time(1)	0.0
4321	timestamp(1) or time(1)	0.0

Table 13 includes a set of sample values retrieved by the application when SQLFractionValue is set to SYSTEM_DEFAULT or to a value other than 9.

Table 13 Values retrieved by the application for SYSTEM_DEFAULT setting

Value in the column of the SQL/MX Table	Column type	Fraction value retrieved by the application
0.987654	timestamp(6) or time(6)	987654
0.087654	timestamp(6) or time(6)	87654
0.007654	timestamp(6) or time(6)	7654
0.000654	timestamp(6) or time(6)	654
0.000054	timestamp(6) or time(6)	54
0.000004	timestamp(6) or time(6)	4
0.98765	timestamp(5) or time(5)	98765
0.08765	timestamp(5) or time(5)	8765
0.00765	timestamp(5) or time(5)	765
0.00065	timestamp(5) or time(5)	65
0.00005	timestamp(5) or time(5)	5
0.00000	timestamp(5) or time(5)	0

Table 13 Values retrieved by the application for SYSTEM_DEFAULT setting *(continued)*

Value in the column of the SQL/MX Table	Column type	Fraction value retrieved by the application
0.9876	timestamp(4) or time(4)	9876
0.0876	timestamp(4) or time(4)	876
0.0076	timestamp(4) or time(4)	76
0.0007	timestamp(4) or time(4)	7
0.0000	timestamp(4) or time(4)	0
0.987	timestamp(3) or time(3)	987
0.087	timestamp(3) or time(3)	87
0.008	timestamp(3) or time(3)	8
0.000	timestamp(3) or time(3)	0
0.98	timestamp(2) or time(2)	98
0.09	timestamp(2) or time(2)	9
0.00	timestamp(2) or time(2)	0
0.9	timestamp(1) or time(1)	9
0.0	timestamp(1) or time(1)	0

Table 14 includes a set of sample values retrieved by the application when SQLFractionValue is 9. The values are in nanoseconds.

Table 14 Sample values retrieved by the application

Value in the column of the SQL/MX Table	Column type	Fraction value retrieved by the application
0.987654	timestamp(6) or time(6)	987654000
0.087654	timestamp(6) or time(6)	87654000
0.007654	timestamp(6) or time(6)	7654000
0.000654	timestamp(6) or time(6)	654000
0.000054	timestamp(6) or time(6)	54000
0.000004	timestamp(6) or time(6)	4000
0.98765	timestamp(5) or time(5)	987650000
0.08765	timestamp(5) or time(5)	87650000
0.00765	timestamp(5) or time(5)	7650000
0.00065	timestamp(5) or time(5)	650000
0.00006	timestamp(5) or time(5)	60000
0.00000	timestamp(5) or time(5)	0
0.9876	timestamp(4) or time(4)	987600000
0.0876	timestamp(4) or time(4)	87600000
0.0076	timestamp(4) or time(4)	7600000
0.0007	timestamp(4) or time(4)	700000
0.0000	timestamp(4) or time(4)	0

Table 14 Sample values retrieved by the application *(continued)*

Value in the column of the SQL/MX Table	Column type	Fraction value retrieved by the application
0.987	timestamp(3) or time(3)	987000000
0.087	timestamp(3) or time(3)	87000000
0.008	timestamp(3) or time(3)	8000000
0.000	timestamp(3) or time(3)	0
0.98	timestamp(2) or time(2)	980000000
0.09	timestamp(2) or time(2)	90000000
0.00	timestamp(2) or time(2)	0
0.9	timestamp(1) or time(1)	900000000
0.0	timestamp(1) or time(1)	0

NOTE: The applications can insert the time and timestamp values into the time and timestamp columns of SQL/MX table, using the `FRACTION_STRUCT` structure. The structure fields are:

```
struct FRACTION_STRUCT {
SQLUINTEGER day,
SQLUINTEGER  hour,
SQLUINTEGER minute,
SQLUINTEGER second,
SQLUINTEGER fraction
};
```

Table 15 includes a sample set of `FRACTION_STRUCT` field values when `SQL Fraction Precision Value:` is set to 6 or 9.

Table 15 `FRACTION_STRUCT` field values with `SQL Fraction Precision Value:` set to 6 or 9.

Table precision/ fraction	6	5	4	3	2	1
123456	123456	12345	1234	123	12	1
12345	012345	12345	1234	123	12	1
1234	001234	01234	1234	123	12	1
123	000123	00123	0123	123	12	1
12	000012	00012	0012	012	12	1
1	000001	00001	0001	001	01	1

SQL Interval Behaviour property

The `SQL Interval Behaviour` property includes zeroes at the beginning or end of an interval datatype for micro or nano seconds. The `SQL Interval Behaviour` has two values:

- `SYSTEM_DEFAULT` — Trailing zeroes are added to the fraction. For example, `0.1234` changes to `0.123400`.
- `MSDN_DEFAULT` — Leading zeroes are added to the fraction. For example, `0.1234` changes to `0.001234`.

If not specified, the default is SYSTEM_DEFAULT.

Set the SQL Interval Behaviour property using the MXODSN property.

- The following example describes when the value of SQL Interval Behaviour is set to SYSTEM_DEFAULT, the behavior of the drivers is same as in earlier releases of SQL/MX. In this example, zeroes are suffixed in the following way:

  For example,

```
create table interval_frac
(
     ID2S interval day to second,
   ID2S0 interval day to second (0),
   ID2S1 interval day to second (1),
   ID2S2 interval day to second (2),
   ID2S3 interval day to second (3),
   ID2S4 interval day to second (4)
)
```

  For the above DDL, when you insert values for all columns as 11 22:33:44.1 and 02 11:21:12.1234 binding to valid datatypes, the values are inserted as follows:

Table 16 SQL Interval Behaviour property set to SYSTEM_DEFAULT

ID2S	ID2S0	ID2S1	ID2S2	ID2S3	ID2S4
11 22:33:44.100000	11 22:33:44	11 22:33:44.1	11 22:33:44.10	11 22:33:44.100	11 22:33:44.1000
11 22:33:44.123400	11 22:33:44	11 22:33:44.1	11 22:33:44.12	11 22:33:44.123	11 22:33:44.1234

- The following example describes when the SQL Interval Behaviour property is set to MSDN_DEFAULT. In this example, zeroes are prefixed in the following way:

  For example,

```
create table interval_frac
(
     ID2S interval day to second,
   ID2S0 interval day to second (0),
   ID2S1 interval day to second (1),
   ID2S2 interval day to second (2),
   ID2S3 interval day to second (3),
   ID2S4 interval day to second (4)
)
```

  For the above DDL, when you insert values for all columns as 11 22:33:44.1 and 02 11:21:12.1234 binding to valid datatypes, the following values are inserted:

Table 17 SQL Interval Behaviour property set to MSDN_DEFAULT

ID2S	ID2S0	ID2S1	ID2S2	ID2S3	ID2S4
11 22:33:44.000001	11 22:33:44	11 22:33:44.1	11 22:33:44.01	11 22:33:44.001	11 22:33:44.0001
11 22:33:44.001234	11 22:33:44	11 22:33:44.1	11 22:33:44.12	11 22:33:44.123	11 22:33:44.1234

NOTE: In the above examples, the SQL Fraction Precision Value is set to SYSTEM_DEFAULT (microseconds).

SQL Datetime Retrieval property

The `SQL Datetime Retrieval` specifies how the SQL/MP Datetime data type is retrieved from the drivers. The `SQL Datetime Retrieval` has two values:

- `SYSTEM_DEFAULT` — Year is set to 0001.
- `MSDN_DEFAULT` — Year is set to 1900.

If not specified, the default is `SYSTEM_DEFAULT`. This property is applicable only when SQL/MP Datetime datatype is bound to SQL/MX datatypes such as DATE or TIMESTAMP.

Set the `SQL Datetime Retrieval` property using the `MXODSN` property.

- If the value of `SQL Datetime Retrieval` is set to `SYSTEM_DEFAULT`, the behavior of the drivers is the same as in earlier releases of SQL/MX.

  For example,

  1. Create a table in SQL/MP with Datetime column as `month to day`.
  2. Insert the data `12-31` into the table using SQL/MX or SQL/MP.
  3. Retrieve the column data from the driver by binding to DATE. The retrieved value is `0001-12-31`.

- If the value of `SQL Datetime Retrieval` is set to `MSDN_DEFAULT`, there is a change in the behavior of the driver when compared to earlier releases of SQL/MX.

  For example,

  1. Create a table in SQL/MP with Datetime column as `month to day`.
  2. Insert the data `12-31` into the created table using SQL/MX or SQL/MP.
  3. Retrieve the column data from the driver by binding to DATE. The retrieved value is `1900-12-31`.

5 Error messages

Table 17 lists the driver error codes and error messages.

Table 17 Error codes and error messages

Error codes	Error messages
01000	General Warning.
01000	General Warning. Connected to the default data source.
01002	Disconnect error. Transaction rolled back.
01004	Data truncated.
01006	Privilege not revoked.
01033	TRANSPORT LAYER ERROR.
01S02	Option value changed.
07001	Wrong number of parameters.
07003	Dynamic SQL error. Cursor specification cannot be executed.
07005	Dynamic SQL error. Prepared statement is not a cursor specification.
07006	Restricted data type attribute violation.
07008	Dynamic SQL error. Wrong number of bound columns.
07009	The value specified for the argument *ColumnNumber* was greater than the number of columns in the result set.
08001	No more MXCS servers available to connect.
08001	MXCS services not yet available.
08001	Data source not yet available or not found.
08001	No more ports available to start MXCS servers.
08001	Retry attempts to connect to the data source failed, May be MXCS server not able to register to the MXCS service process.
08001	No more MXCS servers available to connect.
08002	Connection in use.
08003	Connection not open.
08004	Data source rejected establishment of connection since the MXCS server is connected to a different client now.
08004	Data source rejected establishment of connection for implementation defined reasons.
08005	Communication failure.
08006	Transaction rolled back.
08007	Connection failure during transaction.
08S01	Communication link failure.
08S02	TRANSPORT LAYER ERROR.

Table 17 Error codes and error messages *(continued)*

Error codes	Error messages
21001	Cardinality violation; insert value list does not match column list.
21002	Cardinality violation; insertion value list does not match column list.
21S01	Cardinality violation; parameter list does not match column list.
21S02	String data right truncation.
22001	Numeric value out of range.
22003	Error in assignment.
22005	Precision or scale out of range.
22008	Datetime field overflow.
22012	Division by zero.
22015	Interval field overflow
22018	Invalid character value for cast specification.
23000	Integrity constraint violation.
24000	Invalid cursor state.
25000	Invalid transaction state.
26000	Invalid SQL statement identifier.
28000	Invalid authorization specification.
28000	Invalid authorization specification; access to selected database is denied.
34000	Invalid cursor name.
37000	Syntax error in SQL dynamic statement.
3C000	Duplicate cursor name.
40001	Attempt to initiate new SQL server operation with data pending.
42000	Syntax error or access rule violation.
70100	Operation aborted (server did not process cancel request).
S0001	Invalid table name; base table or view already exists.
S0002	Invalid table name; table or view not found.
S0011	Invalid index name; index already exists.
S0012	Invalid index name; index not found.
S0021	Invalid column name; column already exists.
S0022	Invalid column name; column not found.
S1000	General error.
S1000	General error: Ongoing transaction has been committed.
S1000	General error. Failed since resource governing policy is hit.

Table 17 Error codes and error messages *(continued)*

Error codes	Error messages
S1000	The stored procedure required to complete this operation could not be found on the server (they were supplied with the ODBC/MX setup disk for the SQL Server driver). Contact your service provider.
S1000	Unknown token received from SQL Server
S1000	Unable to load communication module. Driver has not been correctly installed.
S1000	Communication module is not valid. Driver has not been correctly installed.
S1000	Data type mismatch.
S1000	Program Error, Contact your service provider.
S1001	Memory allocation error.
S1002	Invalid column number.
S1003	Program type out of range.
S1004	SQL data type out of range.
S1005	Parameter number out of range.
S1006	Invalid conversion specified.
S1007	Row count not available from the data source.
S1008	Operation cancelled.
S1009	Invalid argument value.
S1010	Function sequence error
S1012	Invalid transaction operator code specified.
S1015	No cursor name available.
S1090	Invalid string or buffer length.
S1091	Descriptor type out of range.
S1092	Option type out of range.
S1093	Invalid parameter number.
S1094	Invalid scale value.
S1095	Function type out of range.
S1096	Information type out of range.
S1097	Column type out of range.
S1098	Scope type out of range.
S1099	Nullable type out of range.
S1100	Uniqueness option type out of range.
S1101	Accuracy option type out of range.
S1102	Table type out of range.
S1103	Direction option out of range.
S1105	Invalid parameter type or parameter type not supported.

Table 17 Error codes and error messages *(continued)*

Error codes	Error messages
S1106	Fetch type out of range.
S1107	Row value out of range.
S1108	Concurrency option out of range.
S1109	Invalid cursor position; no keyset defined.
S1C00	Driver not capable.
S1LD0	No long data values pending.
S1T00	Timeout expired.
6001	INVALID DLL HANDLE.
6002	CANNOT LOAD PROCADDRESS.
6003	WRONG WINSOCK VERSION.
6004	WRONG SIGNATURE.
6005	WRONG VERSION.
6006	ERROR FROM SERVER.
6007	INCORRECT LENGTH.
6008	MEMORY ALLOCATE.
6009	WRONG IP ADDRESS.
6010	Connection lost, invalid code path.

A Sample ODBC application

Compiling and linking the sample application

HP-UX

Use the g++ compiler to compile the application with the header files that are shipped along with the driver:

```
odbc]$ g++ -m64 -o < output object name > -I < location of the header
files > \
< input file for compilation > -l mxodbc_ia64
```

Linux 32

Use the g++ compiler to compile the application with the header files that are shipped along with the driver:

```
odbc]$ g++ -m32 -o < output object name > -I < location of the header
files > \
< input file for compilation > -l mxodbc
```

Linux 64

Use the g++ compiler to compile the application with the header files that are shipped along with the driver:

```
odbc]$ g++ -m64 -o < output object name > -I < location of the header files > \
< input file for compilation > -l mxodbc64
```

OSS

Use the c89 compiler to compile the application with the header files that are shipped along with the driver:

```
oss> c89 -o < output object name> -I/usr/include/odbc/ \
-c < input file for compilation >
```

By default, the header files are located in the `/usr/include/odbc` directory.

Example for compiling a threaded application

HP-UX

The following is a sample compilation command:

```
odbc]$ /usr/bin/g++ –m64 I < path for include files >
ThreadODBCTestAppl.c -o ThreadODBCTestAppl –l mxodbc_ia64
```

Linux 32

The following is a sample compilation command for a 32–bit application:

```
odbc]$ /usr/bin/g++ -m32  -I < path for include files >
ThreadODBCTestAppl.c -o ThreadODBCTestAppl –l mxodbc
```

Linux 64

The following is a sample compilation command for a 64–bit application:

```
odbc]$ /usr/bin/g++ -m64 -I < path for include files >
ThreadODBCTestAppl.c -o ThreadODBCTestAppl –l mxodbc64
```

OSS

The following is a sample compilation command for a 32–bit application:

```
OSS> c89 -g -Winspect -Wextensions -Ww -Woptimize=0 \
-Wsystype=oss -Wrefalign=8 -Wfieldalign=auto \
-Wallow_cplusplus_comments -Wcplusplus -Wversion3 \
-Wtarget=tns/e -DYOSEMITE -Wcall_shared \
-D__GNUC__ -DOSS_DRIVER -D_DEBUG -DNSK_PLATFORM \
-DTCL_MEM_DEBUG -D__TANDEM -Dset_fieldalign \
-D_TNS_R_TARGET -WIEEE_float -I/usr/include/odbc \
```

```
-c ThreadODBCTestAppl.c -o ThreadODBCTestAppl.o
```

The following is a sample compilation command for a 64–bit application:

```
OSS> c89 -g -Winspect -Wextensions -Ww -Woptimize=0 \
—W lp64 —Wsystype=oss -Wrefalign=8 -Wfieldalign=auto \
-Wallow_cplusplus_comments -Wcplusplus -Wversion3 \
-Wtarget=tns/e -DYOSEMITE -Wcall_shared —D_PUT_MODEL_\
-D__GNUC__ -DOSS_DRIVER -D_DEBUG -DNSK_PLATFORM \
-DTCL_MEM_DEBUG -D__TANDEM -Dset_fieldalign \
-D_TNS_R_TARGET -WIEEE_float -I/usr/include/odbc \
-c ThreadODBCTestAppl.c -o ThreadODBCTestAppl.o
```

NOTE: For C programming applications, add `-Wallow_cplusplus_comments` flag.

Use the eld linker to link the application with the driver (32–bit ZODBCDLL, 64–bit YODBCDLL). By default, the driver is located in $SYSTEM.ZMXODBC directory.

Testing the sample

HP-UX and Linux

To run the sample test application, issue any of the following commands on the bash shell:

```
odbc]$ ./<application name> <DS Name>
odbc]$ ./<application name> –help
```

Considerations:

- If the library is not present in the default location (`/usr/lib` or `/usr/lib64` in 64 Bit Linux), ensure that the location of the driver (`libmxodbc.so`, `libmxodbc64.so` in 64 Bit Linux) is set in the environment variable LD_LIBRARY_PATH (Linux 32 or Linux 64) or SHLIB_PATH(HP-UX). To set the LD_LIBRARY_PATH variable, enter the following command:

  ```
  export LD_LIBRARY_PATH = < driver directory >
  ```

- Ensure that the `MXODSN` file is present in the same location as that of the sample application. If not, ensure that the default `MXODSN` file is present in `/etc/hpodbc` (Linux) or `/etc/mxodbc`(HP-UX), or set DSN_PATH to MXODSN location.
- To enable tracing, set the `TraceStart` attribute in the `MXODSN` file to 1.

You must edit the `MXODSN` file to include your preferred DSN. Update all the mandatory fields for the DSN.

OSS

To run the sample test application, issue any of the following commands on the OSS shell:

```
OSS> ./<application name> <DS Name>
OSS> ./<application name> –help
```

Considerations:

- Ensure that the `ODBCDSN` file is present in the same location as that of the sample application. If not, ensure that the `ODBCDSN` file is present in the default location, $SYSTEM.SYSTEM or in the `DSN_PATH`.
- To enable tracing, set the `TraceStart` attribute in the `ODBCDSN` file to 1.

You must edit the `ODBCDSN` file to include your preferred DSN. Update all the mandatory fields for the DSN.

Sample application code

```
/**********************************************************
*****
//
// @@@ START COPYRIGHT @@@
```

```
//
// Copyright 2007
//
// HP CONFIDENTIAL: NEED TO KNOW ONLY
//
// Copyright Hewlett-Packard Development Company, L.P.
// Protected as an unpublished work.
//
// The computer program listings, specifications and
// documentation herein are the property of Hewlett-Packard
// Development Company, L.P., or a third party supplier and
// shall not be reproduced, copied, disclosed, or used in whole
// or in part for any reason without the prior express written
// permission of Hewlett-Packard
// Development Company, L.P.
//
// @@@ END COPYRIGHT @@@
**************************************************************/
/**************************************************************
@@@@ ODBC TEST APPLICATION @@@
==============================================================
This application utilizes ::
- Multiple connections
- Gives information about DBMS NAME
- ODBC API's
- Uses insertion through parameter marker,
- Setting the transaction isolation level as serializable and
fetches the data.
**************************************************************
******/


#include <stdio.h>
#include <stdlib.h>
#include <sqlext.h>
#include <string.h>

#define UnSignCharCast unsigned char *
#define TRUE 1
typedef struct
{
  SDWORD pfNativeError[1];
  SWORD pcbErrorMsg[1];
  SWORD cbErrorMsgMax;
  UCHAR *szErrorMsg;
  UCHAR *szSqlState;
} ERR_INFO;
RETCODE odbc_Error(SQLHENV hEnv, SQLHDBC hDbc, SQLHSTMT hStmt);
RETCODE odbc_Error(
   SQLHENV hEnv,
   SQLHDBC hDbc,
   SQLHSTMT hStmt
)
{
   char *szBuf;
   int pt_ch ='.' ;
   int brac_ch =']' ;
   ERR_INFO *Err;
   RETCODE st = 0;
   Err = (ERR_INFO *)malloc (sizeof(ERR_INFO));
   Err->szErrorMsg = (UnSignCharCast )malloc(200);
   Err->szSqlState = (UnSignCharCast )malloc(50);
   szBuf = (char *)malloc(600);
   memset (Err->szErrorMsg,'\0',200);
   memset (Err->szSqlState,'\0',50);
```

```
    memset (szBuf,'\0',600);
    if (hStmt)
      st = SQLGetDiagRec(SQL_HANDLE_STMT,hStmt,1,Err->szSqlState,
      Err->pfNativeError, Err->szErrorMsg,150,Err->pcbErrorMsg);
    else if (hDbc)
      st = SQLGetDiagRec(SQL_HANDLE_DBC,hDbc,1,Err->szSqlState,
      Err->pfNativeError, Err->szErrorMsg,150,Err->pcbErrorMsg);
    else if (hEnv)
      st = SQLGetDiagRec(SQL_HANDLE_ENV,hEnv,1,Err->szSqlState,
      Err->pfNativeError, Err->szErrorMsg,150,Err->pcbErrorMsg);
    if ((st == SQL_SUCCESS) || (st == SQL_SUCCESS_WITH_INFO))
    {
      sprintf(szBuf," %s - [%s]\n", (char *)Err->szErrorMsg,
      Err->szSqlState);
      printf("%s \n",szBuf);
    }
    free(Err->szErrorMsg);
    free(Err->szSqlState);
    free(Err);
    free(szBuf);
    return SQL_SUCCESS;
}
int main(int argc, char *argv[])
{
    RETCODE st = 0;
    SQLHENV henv = (SQLHENV) NULL;
    SQLHDBC hdbc = (SQLHDBC) NULL;
    SQLHDBC hdbc1 = (SQLHDBC) NULL;
    SQLHDBC hdbc2 = (SQLHDBC) NULL;
    SQLHSTMT hstmt = (SQLHSTMT) NULL;
    SQLHSTMT hstmt1 = (SQLHSTMT) NULL;
    CHAR CreateTable[100] = {'\0'};
    CHAR InsertTable[100] = {'\0'};
    CHAR SelectTable[100] = {'\0'};
    int value;
    SQLINTEGER ValInd;
    SQLINTEGER RowCount = 0;
    SQLINTEGER val;
    SDWORD pcbValue = 0;
    SQLCHAR infoValueBuf[100] = {'\0'};
    SQLCHAR*infoValuePtr=0;
    SQLSMALLINT StringLengthPtr = NULL;
    char Password[40]="\0";
    char UserID[40]="\0";
    char *serverName="TDM_Default_DataSource";
    short value1=10;
    short value2=20;
    bool defaultDS=false;
    printf("\n\tPURPOSE :: TO TEST ODBC API'S ON ODBC
     DRIVER\n\n");
    if(argc>1)
    {
      if(!strcmp(argv[1], "-help"))
      {
        printf("USAGE: odbcdemo <DataSourceName> \n");
        printf("Eg: odbcdemo DEMO\n");
        printf("Note: Default Data Source will be used in case no DS
         specified!\n");
        return 0;
      }
      serverName = argv[1];
    }
    printf("User Name: ");
    scanf("%s", UserID);
    printf("Password : ");
```

```
scanf("%s", Password);
strcpy(CreateTable,"create table GGTest(c1 int)");
sprintf(InsertTable, "%s %d %s", "insert into GGTest(c1) values
  (", value1, ");");
strcpy(SelectTable,"select * from GGTest" );
printf("\n\tUsing Data Source : %s \n\n", serverName);
if (!strncmp(serverName, "TDM_Default_DataSource",
 strlen(serverName)))
 defaultDS = true;
st = SQLAllocEnv(&henv);
if (henv == NULL)
{
  printf("Error in allocating Env Handle!\n");
  odbc_Error(SQL_NULL_HENV,SQL_NULL_HDBC,SQL_NULL_HSTMT);
}
st = SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION,
  (void*)SQL_OV_ODBC3, 0);
if (st != SQL_SUCCESS)
{
  printf("Error in SQLSetEnvAttr :: %d\n", st);
  odbc_Error(henv,SQL_NULL_HDBC,SQL_NULL_HSTMT);
}
st = SQLAllocConnect(henv, &hdbc);
if (hdbc == NULL)
{
  printf("Error in allocating connection Handle 1!\n");
  odbc_Error(henv,SQL_NULL_HDBC,SQL_NULL_HSTMT);
}
st = SQLAllocConnect(henv, &hdbc1);
if (hdbc1 == NULL)
{
  printf("Error in allocating connection Handle 2!\n");
  odbc_Error(henv,SQL_NULL_HDBC,SQL_NULL_HSTMT);
}
st = SQLAllocConnect(henv, &hdbc2);
if (hdbc2 == NULL)
{
  printf("Error in allocating connection Handle 3!\n");
  odbc_Error(henv,SQL_NULL_HDBC,SQL_NULL_HSTMT);
}
st =
 SQLConnect(hdbc,(SQLCHAR*)serverName,SQL_NTS,(SQLCHAR*)UserID,SQ
 L_NTS,(SQLCHAR*)Password,SQL_NTS);
if (st != SQL_SUCCESS)
{
  printf("Error in Connection 1\n");
  odbc_Error(henv,hdbc,SQL_NULL_HSTMT);
  exit(1);
}
else
  printf("Connection 1 Successful\n");
st = SQLSetConnectAttr(hdbc,
  SQL_ATTR_TXN_ISOLATION,(SQLPOINTER)8,0);
if (st != SQL_SUCCESS)
{
  printf("Error in SQLSetConnectAttr()\n");
  odbc_Error(henv,hdbc,hstmt);
}
st =
  SQLConnect(hdbc1,(SQLCHAR*)serverName,SQL_NTS,(SQLCHAR*)UserID,
  SQL_NTS,(SQLCHAR*)Password,SQL_NTS);
if (st != SQL_SUCCESS)
{
  printf("Error in Connection 2\n");
  odbc_Error(henv,hdbc,SQL_NULL_HSTMT);
```

```
}
else
  printf("Connection 2 Successful\n");
st =
  SQLConnect(hdbc2,(SQLCHAR*)serverName,SQL_NTS,(SQLCHAR*)UserID,
  SQL_NTS,(SQLCHAR*)Password,SQL_NTS);
if (st != SQL_SUCCESS)
{
  printf("Error in Connection 3\n");
  odbc_Error(henv,hdbc,SQL_NULL_HSTMT);
}
else
  printf("Connection 3 Successful\n");
st = SQLGetInfo(hdbc,SQL_DBMS_NAME, infoValueBuf,
  sizeof(infoValueBuf), &StringLengthPtr);
infoValuePtr = infoValueBuf;
if (st != SQL_SUCCESS)
{
  printf("Error in SQLGetInfo\n");
  odbc_Error(henv,hdbc,hstmt);
}
else
  printf("\n\tDBMS NAME : %s\n\n", infoValuePtr);
printf("\n\tUsing Connection 1\n\n");
st = SQLAllocHandle(SQL_HANDLE_STMT, (SQLHDBC)hdbc, &hstmt);
if (hstmt == NULL)
{
  printf("Error in allocating Statement Handle\n");
  odbc_Error(henv,hdbc,SQL_NULL_HSTMT);
}
st = SQLExecDirect(hstmt,(SQLCHAR *)
        CreateTable,strlen(CreateTable));
if ( st != SQL_SUCCESS)
{
  printf("Error in Create Table: GGTest\n");
  odbc_Error(henv,hdbc,hstmt);
}
else
  printf("Create table GGTest Successful\n");
st = SQLExecDirect(hstmt, (SQLCHAR *)
   InsertTable,strlen(InsertTable));
if (st != SQL_SUCCESS)
{
  printf("Error in Insert Table 1: GGTest\n");
  odbc_Error(henv,hdbc,hstmt);
}
else
  printf("Insert Successful: %d\n", value1);
memset(InsertTable, '\0', sizeof(InsertTable));
sprintf(InsertTable, "%s %d %s", "insert into GGTest(c1) values
  (", value2, ");");
st = SQLExecDirect(hstmt,(SQLCHAR*)InsertTable,SQL_NTS);
if (st != SQL_SUCCESS)
{
  printf("Error in Insert Table 2: GGTest\n");
  odbc_Error(henv,hdbc,hstmt);
}
else
  printf("Insert Successful: %d\n", value2);
st = SQLRowCount(hstmt,&RowCount);
if (st != SQL_SUCCESS)
{
  printf("Error in SQLRowCount()\n");
  odbc_Error(henv,hdbc,hstmt);
}
```

```
else
  printf("Row(s) affected %d\n", RowCount);
st = SQLExecDirect(hstmt,(SQLCHAR *)
  SelectTable,strlen(SelectTable));
if (st != SQL_SUCCESS)
{
  printf("Error in Select Table: GGTest\n");
  odbc_Error(henv,hdbc,hstmt);
}
st = SQLBindCol(hstmt, 1, SQL_C_SLONG, &value, 0, &ValInd);
if (st != SQL_SUCCESS)
{
  printf("Error in SQLBindCol\n");
  odbc_Error(henv,hdbc,hstmt);
}
while(TRUE)
{
  if ((st = SQLFetch(hstmt)) == SQL_NO_DATA_FOUND)
   break;
  if ( (st != SQL_SUCCESS) && (st != SQL_SUCCESS_WITH_INFO) )
  {
   printf("Error in SQLFetch!!! Returned Status: %d\n", st);
   break;
  }
  if (ValInd == SQL_NULL_DATA)
  {
   printf("No Data fetched!\n");
   break;
  }
  else
  {
   printf("c1 :: %d\n", value);
  }
}
// ***************** Done with connection 1 ********************
printf("\n\tUsing Connection 2\n\n");
st = SQLAllocHandle(SQL_HANDLE_STMT, (SQLHDBC)hdbc1, &hstmt1);
if (hstmt1 == NULL)
{
  printf("Error in allocating statement handle\n");
  odbc_Error(SQL_NULL_HENV,SQL_NULL_HDBC,hstmt);
}
st = SQLPrepare(hstmt1,(SQLCHAR *)"insert into GGTest(c1)
        values(?)",SQL_NTS);
if (st != SQL_SUCCESS)
{
  printf("Error in SQLPrepare\n");
  odbc_Error(henv,hdbc1,hstmt1);
}
st =
 SQLBindParameter(hstmt1,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_INTEGER
  ,0,0,&val, 0, &pcbValue);
val = 50;
if(st != SQL_SUCCESS)
{
  printf("Error in SQLBindParameter : %d\n", st);
  odbc_Error(henv,hdbc1,hstmt1);
}
st = SQLExecute(hstmt1);
if(st != SQL_SUCCESS)
{
  printf("Error in SQLExecute : %d\n", st);
  odbc_Error(henv,hdbc1,hstmt1);
}
else
```

```
      printf("Insert Successful : %d\n", val);
    strcpy(SelectTable,"select * from GGTest" );
    st = SQLExecDirect(hstmt1,(SQLCHAR *)
      SelectTable,strlen(SelectTable));
    if (st != SQL_SUCCESS)
    {
      printf("Error in select table : GGTest\n");
      odbc_Error(henv,hdbc1,hstmt1);
    }
    st = SQLBindCol(hstmt1, 1, SQL_C_SLONG, &value, 0, &ValInd);
    if(st != SQL_SUCCESS)
    {
      printf("Error in SQLBindCol\n");
      odbc_Error(henv,hdbc1,hstmt1);
    }
    while(TRUE)
    {
     if ((st = SQLFetch(hstmt1)) == SQL_NO_DATA_FOUND)
      break;
     if ( (st != SQL_SUCCESS) && (st != SQL_SUCCESS_WITH_INFO) )
     {
      printf("Error in SQLFetch!!! Returned Status: %d\n", st);
      break;
     }
     if (ValInd == SQL_NULL_DATA)
     {
      printf("No Data Fetched!\n");
      break;
     }
     else
     {
      printf("c1 :: %d\n", value);
     }
    }
    st = SQLExecDirect(hstmt1,(SQLCHAR*)"drop table
      GGTest",SQL_NTS);
    if (st != SQL_SUCCESS)
    {
     printf("Error in drop table : GGTest\n");
     odbc_Error(henv,hdbc1,hstmt1);
    }
    else
     printf("\nTable GGTEST dropped!\n");
    st = SQLEndTran(SQL_HANDLE_DBC,(SQLHANDLE)hdbc,SQL_ROLLBACK);
    if (hstmt)
     SQLFreeStmt(hstmt,SQL_CLOSE);
    if (hstmt1)
     SQLFreeStmt(hstmt1,SQL_CLOSE);
    SQLDisconnect(hdbc);
    SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
    SQLDisconnect(hdbc1);
    SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
    SQLDisconnect(hdbc2);
    SQLFreeHandle(SQL_HANDLE_DBC, hdbc2);
    SQLFreeHandle(SQL_HANDLE_ENV, henv);
    return 0;
}
```

Glossary

Driver Manager

The ODBC component that manages access to Database Management System (DBMS) drivers for ODBC applications. The driver manager loads and unloads drivers and passes the calls for ODBC functions to the correct driver. The HP-UX and Linux drivers support both the Nonstop driver manager and the unixODBC driver manager.

Open Database Connectivity (ODBC)

An interface specification for an API that defines a standard set of routines that an ODBC application can use to access data in a database.

ODBC/MX

The ODBC components and products to support the ODBC functionality for the SQL/MX database.

Index

A
AutoCommit, 42

B
BIGINT data type, 39
BINARY data type, 39
BIT data type, 38

C
Catalog and schema
  consideration, 37
CHAR data type, 38
Client data source
  configuring, 13

D
Data sources
  managing, 13
data sources
  configuring, 24
  managing, 33
data types
  unsigned, 39
data types supported, 39
DATE data type, 39
DATETIME, 41
DECIMAL data type, 38
default data source
  TDM_Default_DataSource, 13
Documents, related information, 7
DOUBLE PRECISION data type, 38
Driver considerations, 38

E
error codes, 52

F
FLOAT data type, 38

I
INTEGER data type, 38
INTERVAL MONTH data type, 39
INTERVAL YEAR data type, 39
IPV6, 26

L
Linux ODBC Client Driver
  overview, 10
LONG VARBINARY data type, 39
LONGVARCHAR data type, 38

N
NUMERIC data type, 38

R
REAL data type, 38

S
Server data source
  configuring, 13
Server-side components, 12
SMALLINT data type, 38
SQL Datetime Retrieval, 48
SQL Interval Behaviour, 46
SQL_QUERY_TIMEOUT
  SQLSetStmtOption, 37
SQLBindParameter
  consideration, 38
SQLCancel
  consideration, 37
SQLPrimaryKeys
  consideration, 37
SQLPutData
  consideration, 38
SQLSpecialColumns
  consideration, 38
SQLStatistics
  consideration, 38
SQLTables
  SQL_ALL_CATALOGS, 38
  SQL_ALL_SCHEMAS, 38
  SQL_ALL_TABLE_TYPES, 38

T
TIME data type, 39
TIMESTAMP data type, 39
TINYINT data type, 38
tracing
  data sources, 35

U
unixODBC
  driver manager, 12
Unsupported ODBC APIs, 14
Unsupported ODBC data types, 15

V
VARBINARY data type, 39
VARCHAR data type, 38