



**MISER ODBC BRIDGE**

Effective:  
08/24/16

Revision:  
C

# Installing and Configuring MISER ODBC Bridge Services

## CONTENTS

Overview..... 2

    ODBC..... 2

Introduction..... 3

Windows Setup..... 3

    MISER ODBC Service Installation Package ..... 3

    MISER Database (MS SQL Server) ..... 3

MISER ODBC Bridge Data Source Name (DSN) ..... 8

MISER ODBC Service Installation..... 14

Single MISER ODBC Bridge Data Setup under OpenVMS..... 17

Double MISER ODBC Bridge Data Setup under OpenVMS..... 18

    MISER ODBC Bridge Testing Under OpenVMS..... 20

User-Executed MISER Applications ..... 24

How to Populate the POINTS Table from OpenVMS ..... 24

How to Populate the HST Table from OpenVMS ..... 25

How to Add or Remove Points from OpenVMS MISER..... 25

    Checking Which Points are Sent to the MISER ODBC Bridge Service ..... 26

    Checking for Errors Communicating with the ODBC Bridge from OpenVMS ..... 26

    Checking for Errors Communicating with the ODBC Bridge from Windows ..... 27

MISER ODBC Bridge Service Table Description ..... 29

    MISER.POINTS Format..... 29

    Table MISER.HST ..... 33

    Table MISER.ALARM..... 33

    Table MISER.QUALITY..... 33

*All information contained in this document is the sole property of HSQ Technology. Any reproduction in part or whole without the written permission of HSQ Technology is prohibited.*

## Overview

The MISER system is a proprietary design that has been developed to provide high performance SCADA systems. MISER applications work directly from the MISER database.

There is a need to make data from the MISER database available to general commercial applications where data can be easily manipulated and reported on. The MISER ODBC Bridge provides a cost efficient method to get data from the MISER system to any application on a Windows PC that supports the ODBC standard. This ability is available as an option for MISER systems version 6.05 and up.

This Application Note is not intended to demonstrate the step by step procedure to implement the ODBC Bridge on a system. It is meant to show the ODBC Bridge's capabilities and give an appreciation of the issues involved in its deployment.

The ODBC Bridge capabilities described in this note are built from a toolkit of functions. Based on the unique requirements for each specific installation, HSQ uses this toolkit to implement the ODBC Bridge on that system.

## ODBC

Open DataBase Connectivity (ODBC) is a widely supported application program interface for database access. Although it is called a database connectivity standard, ODBC is supported by a variety of other PC applications.

In order to use ODBC, there must be an ODBC Application and an ODBC Data Source which both adhere to the ODBC standard. The Data Source is a logical name that is defined to identify a specific database. The ODBC Application issues commands and the Data Source responds to them.

The ODBC Application that needs to access the data communicates with an ODBC Driver Manager on the PC using the Data Source to identify the particular data repository. Based upon which Data Source the ODBC application selects, the Driver Manager uses an appropriate ODBC Driver to "translate" the applications commands to statements that the Data Source understands. The data flow between the ODBC Application and the Data Source may be in either direction (i.e., the ODBC Application may pass data to the Data Source or retrieve data from the Data Source).

Microsoft provides the ODBC Driver Manager (called the ODBC Data Source Administrator) with the Windows operating system and the ODBC Drivers are supplied by the individual companies that make the ODBC compliant programs that support the ODBC interface.

This approach is similar to how printing is handled in the Windows operating system. The individual application sends its print file to a printer driver. The printer driver, which had been written specifically to communicate with the particular printer, then sends the commands to the printer to print out the file. The individual application does not need to know how to communicate directly to the printer; the drivers deal with those details. The ODBC Drivers perform a similar role.

## Introduction

The MISER ODBC Bridge service receives TCP messages from a MISER system through the network and places the content of those messages into corresponding database tables running on a Microsoft® Windows Service.

The MISER ODBC Bridge service places MISER historical data into data tables of a relational database. The MISER ODBC Bridge can communicate with the following relational databases:

- Oracle 9i/10g
- Microsoft SQL Server 2000 or greater
- MySQL V5 or greater
- Wonderware Industrial SQL

One of these relational database software packages must be installed and configured before the MISER ODBC Bridge software can be installed and configured. For demonstration purpose this document will use Microsoft SQL Server 2014 and it is assumed that Windows Server 2008 with Microsoft SQL server 2014 is pre-installed and running in a vertical machine.











## Windows Setup

### MISER ODBC Service Installation Package

The MISER ODBC Bridge Service software is delivered in a ZIP file named:

*modbc\_45.zip*

When unzipped, this installation file contains the following directory:

 MSSQL DB SQL	File folder
 Oracle DB SQL	File folder
 MiserODBC_Bridge_x64.exe	Application
 MiserODBC_Bridge_x86.exe	Application
 vcredist_x64.exe	Application
 vcredist_x86.exe	Application
 MiserODBC_Bridge_mssql.REG	Registration Entries
 MiserODBC_Bridge_oracle.REG	Registration Entries
 INSTALL_MiserODBCBridge.CMD	Windows Command Script
 UNINSTALL_MiserODBCBridge.CMD	Windows Command Script

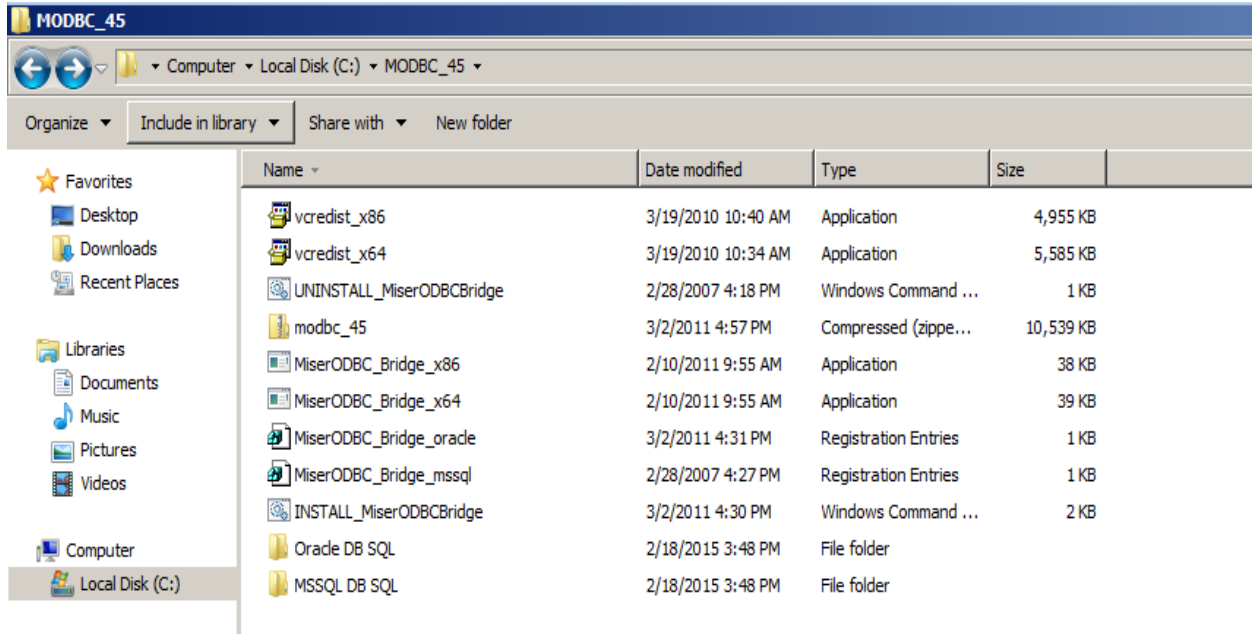
### MISER Database (MS SQL Server)

Within the installation directory is a directory named “MSSQL DB SQL” which contains the SQL scripts that create the MISER schema and populates it with the MISER tables holding MISER point, alarm, and history data.

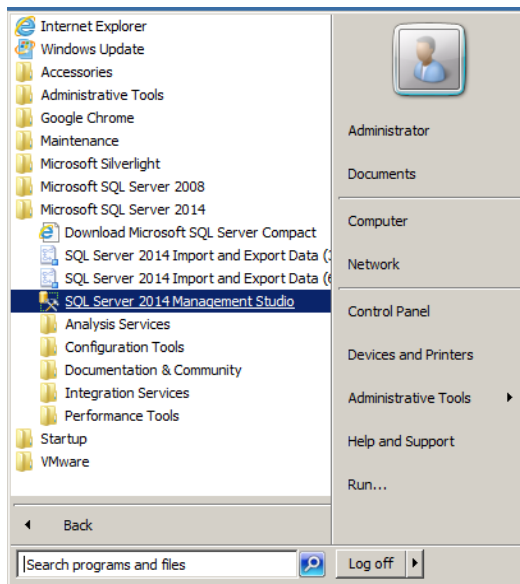
## MISER ODBC BRIDGE

The SQL scripts included with the MISER ODBC Bridge service software are run to create the MISER data tables under MS SQL Server using SQL server 2014 Management Studio as shown in the process below:

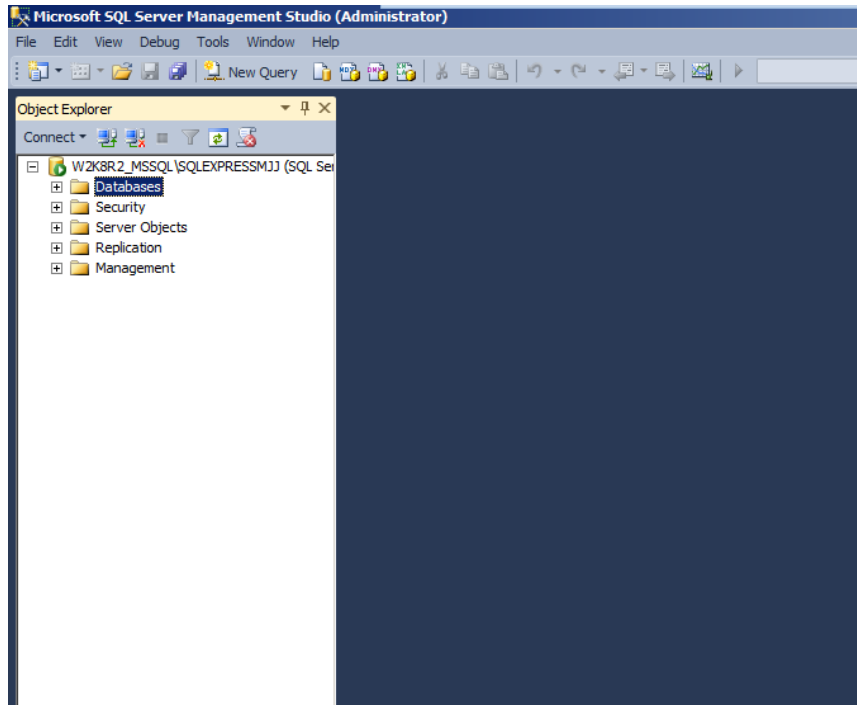
1. Extract the content of the zip file into a directory folder within the **C :** directory.



2. Go to **Start Menu > All Programs > Microsoft SQL Server 2014** and click on *SQL Server 2014 Management Studio*.

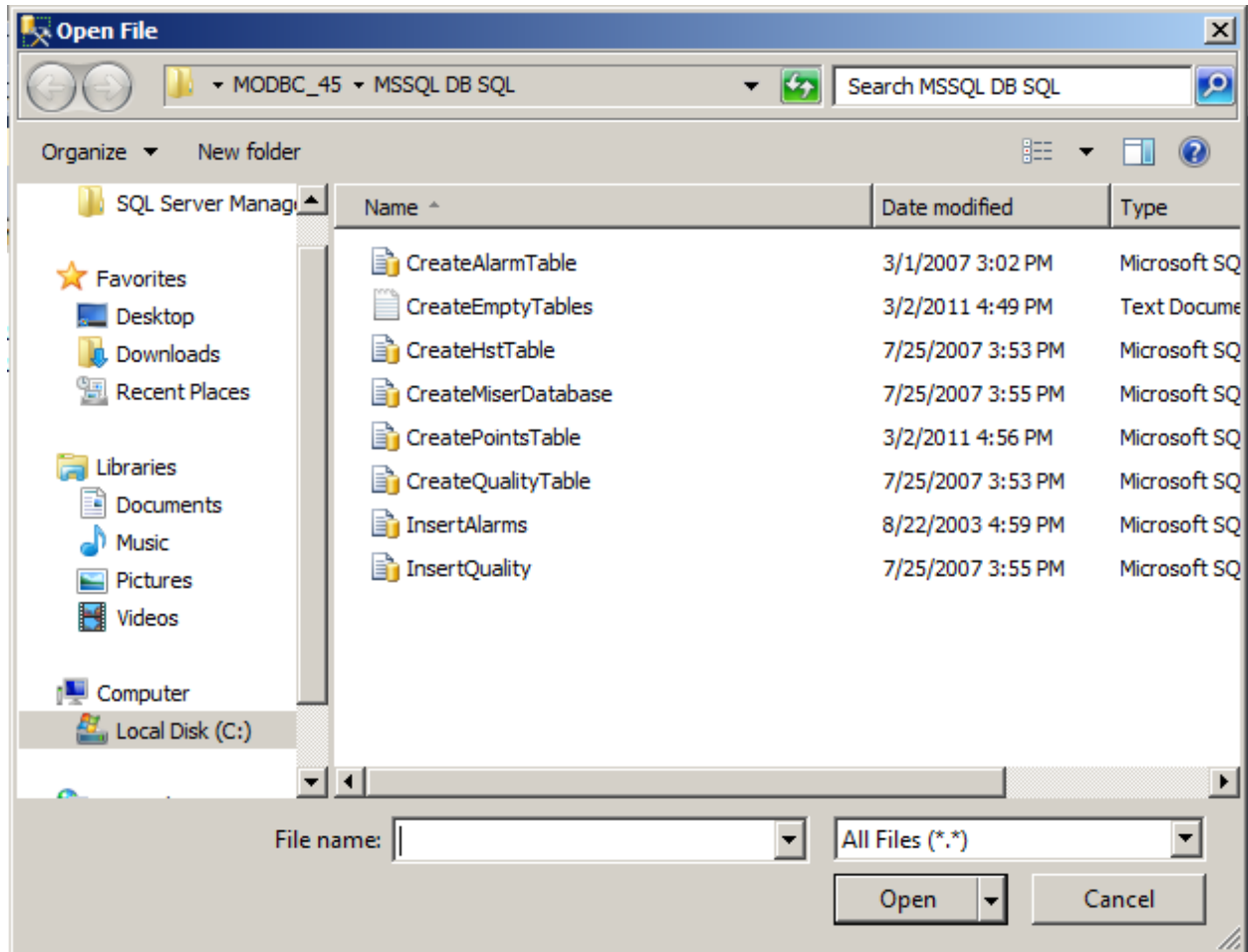


# MISER ODBC BRIDGE



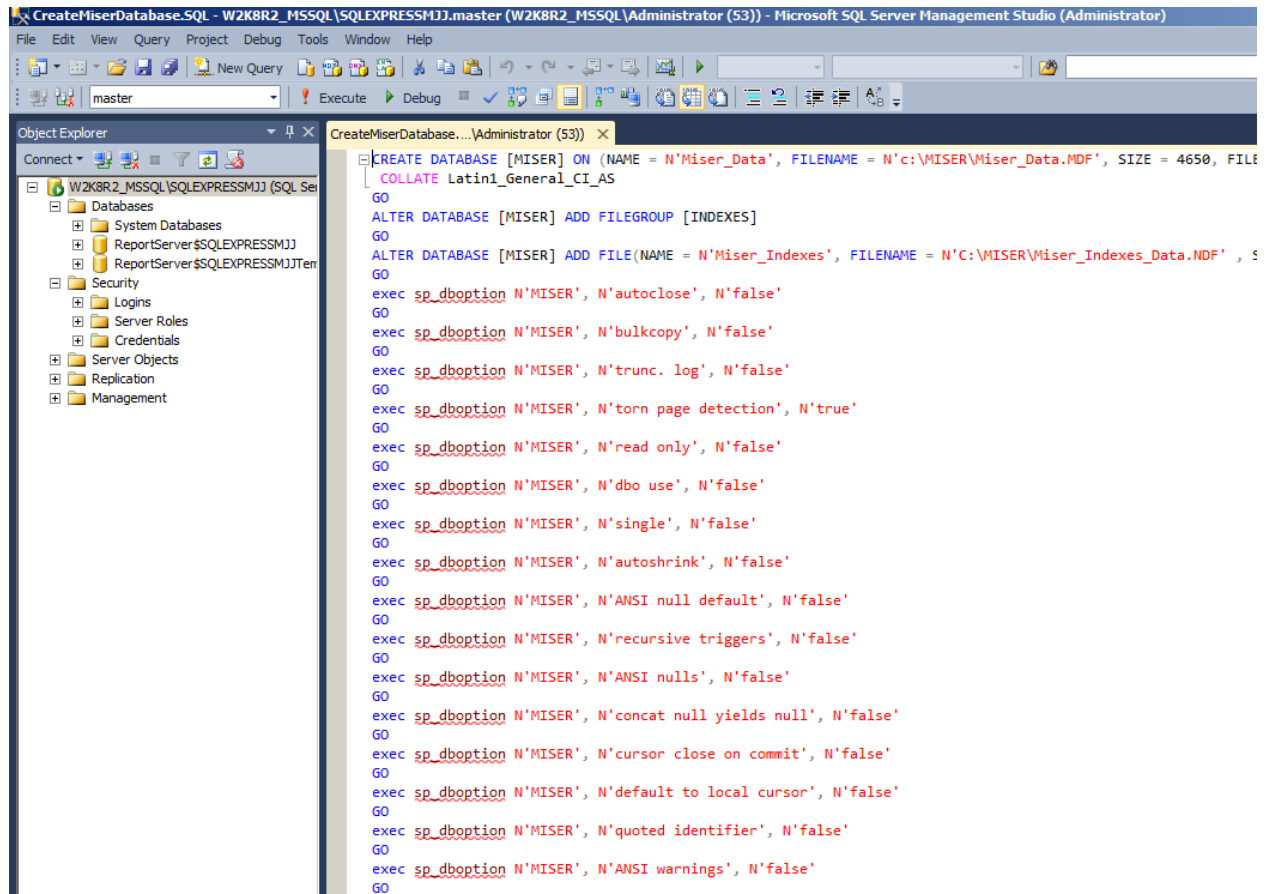
## MISER ODBC BRIDGE

3. Select *File* from the menu bar, click *Open*, and then select *File*. Direct the address to `C:\MODBC_45\MSSQL DB SQL`



4. Once the MISER database has been selected, the remaining SQL scripts can be executed to create the record structures and tables for the MISER database. The SQL scripts should be executed in the following order:
  - a. CreateMiserDatabase.sql
  - b. CreatePointsTable.sql
  - c. CreateHstTable.sql
  - d. CreateQualityTable.sql
  - e. CreateAlarmTable.sql
  - f. InsertAlarms.sql
  - g. InsertQuality.sql

# MISER ODBC BRIDGE



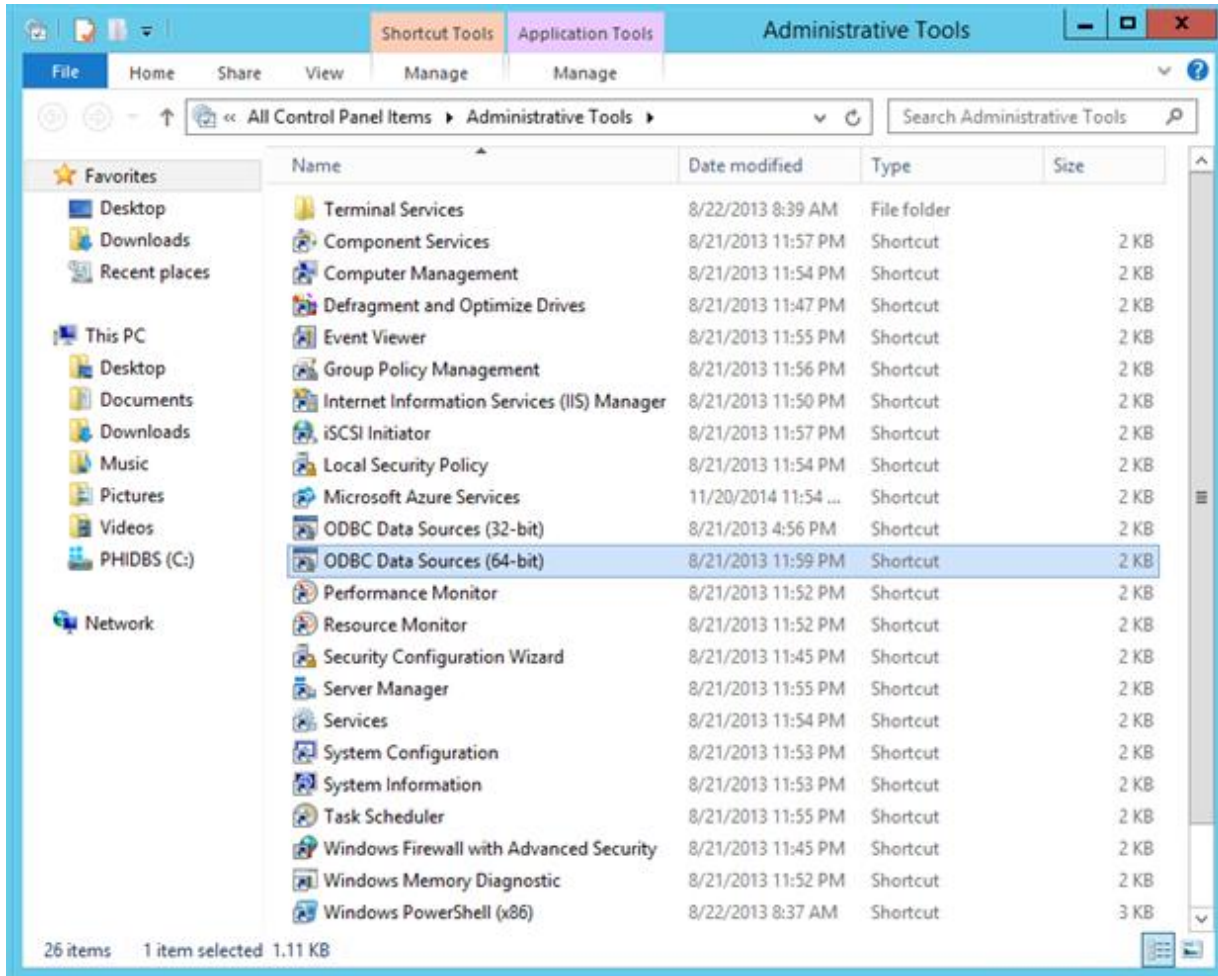
```
CREATE DATABASE [MISER] ON (NAME = N'Miser_Data', FILENAME = N'C:\MISER\Miser_Data.MDF', SIZE = 4650, FILE
COLLATE Latin1_General_CI_AS
GO
ALTER DATABASE [MISER] ADD FILEGROUP [INDEXES]
GO
ALTER DATABASE [MISER] ADD FILE(NAME = N'Miser_Indexes', FILENAME = N'C:\MISER\Miser_Indexes_Data.NDF' , S
GO
exec sp_dboption N'MISER', N'autoclose', N'false'
GO
exec sp_dboption N'MISER', N'bulkcopy', N'false'
GO
exec sp_dboption N'MISER', N'trunc. log', N'false'
GO
exec sp_dboption N'MISER', N'torn page detection', N'true'
GO
exec sp_dboption N'MISER', N'read only', N'false'
GO
exec sp_dboption N'MISER', N'dbo use', N'false'
GO
exec sp_dboption N'MISER', N'single', N'false'
GO
exec sp_dboption N'MISER', N'autoshrink', N'false'
GO
exec sp_dboption N'MISER', N'ANSI null default', N'false'
GO
exec sp_dboption N'MISER', N'recursive triggers', N'false'
GO
exec sp_dboption N'MISER', N'ANSI nulls', N'false'
GO
exec sp_dboption N'MISER', N'concat null yields null', N'false'
GO
exec sp_dboption N'MISER', N'cursor close on commit', N'false'
GO
exec sp_dboption N'MISER', N'default to local cursor', N'false'
GO
exec sp_dboption N'MISER', N'quoted identifier', N'false'
GO
exec sp_dboption N'MISER', N'ANSI warnings', N'false'
GO
```

## MISER ODBC BRIDGE

### MISER ODBC Bridge Data Source Name (DSN)

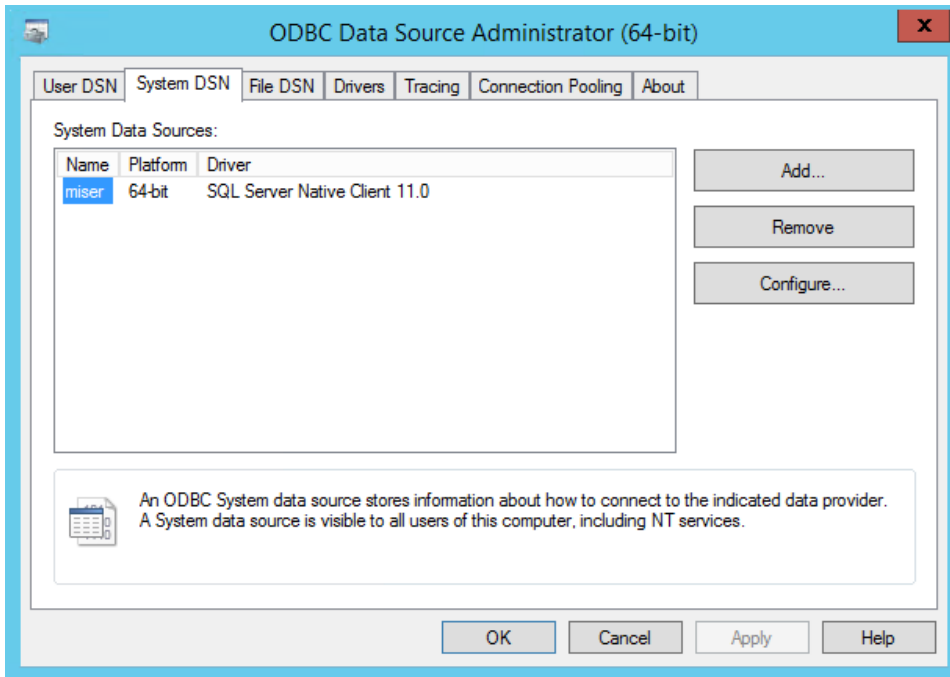
The MISER ODBC Bridge service establishes a connection with a relational database using the ODBC API. A connection to the relational database is made by creating a System Data Source Name (DSN) via the ODBC Data Source Administrator application.

Login as Administrator and open the “Administrative Tools” (within Control Panel) menu. Run “ODBC Data Sources (64-bit)”.

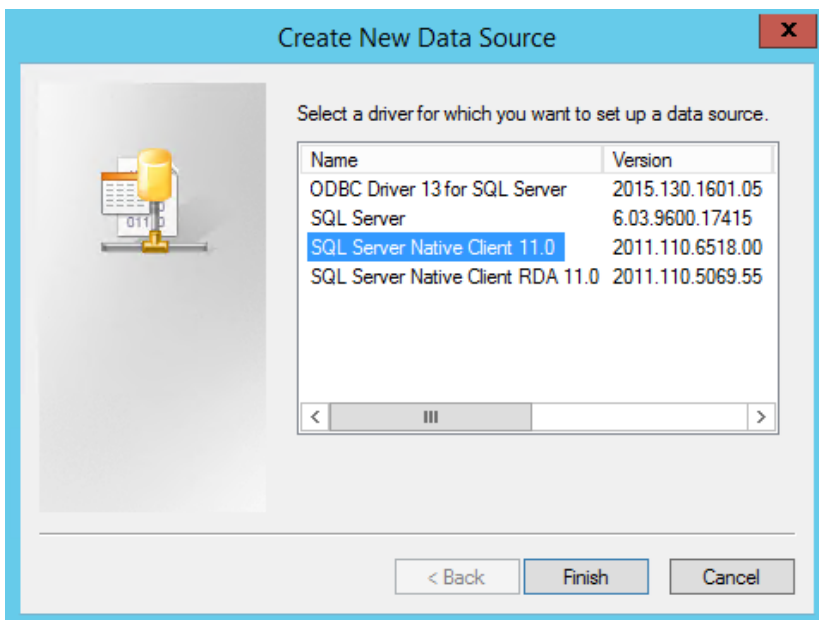


## MISER ODBC BRIDGE

When the **ODBC Data Source Administrator (64-bit)** application dialog box appears, select the *System DSN* tab.

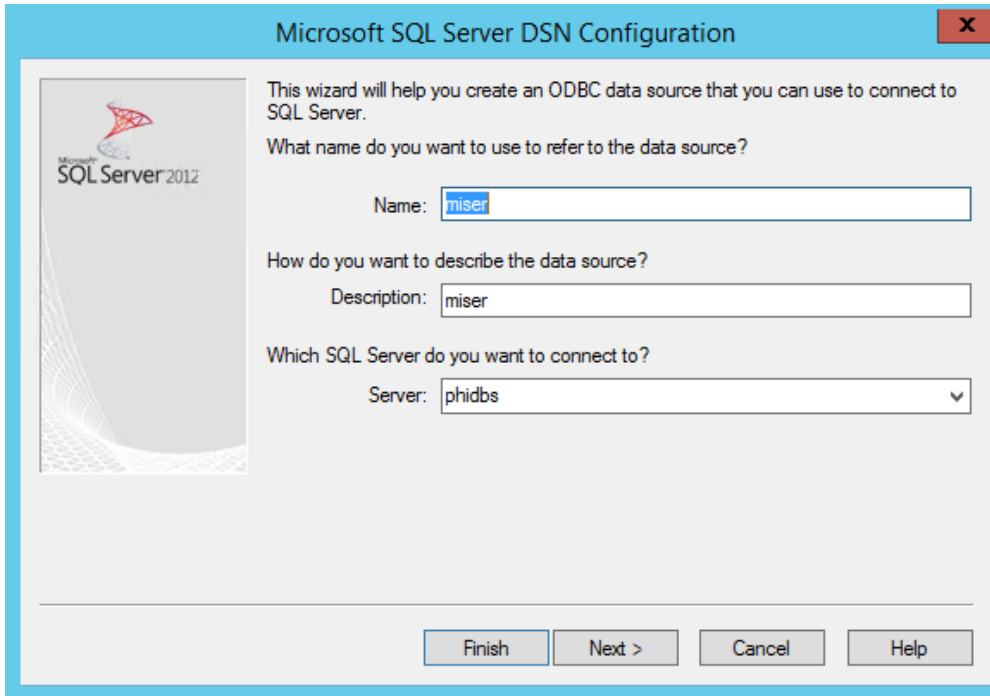


From the **Create New Data Source** dialog box, select the “SQL Server Native Client 11.0” driver (this represents the relational database that will receive the MISER data) and click **[Finish]**.



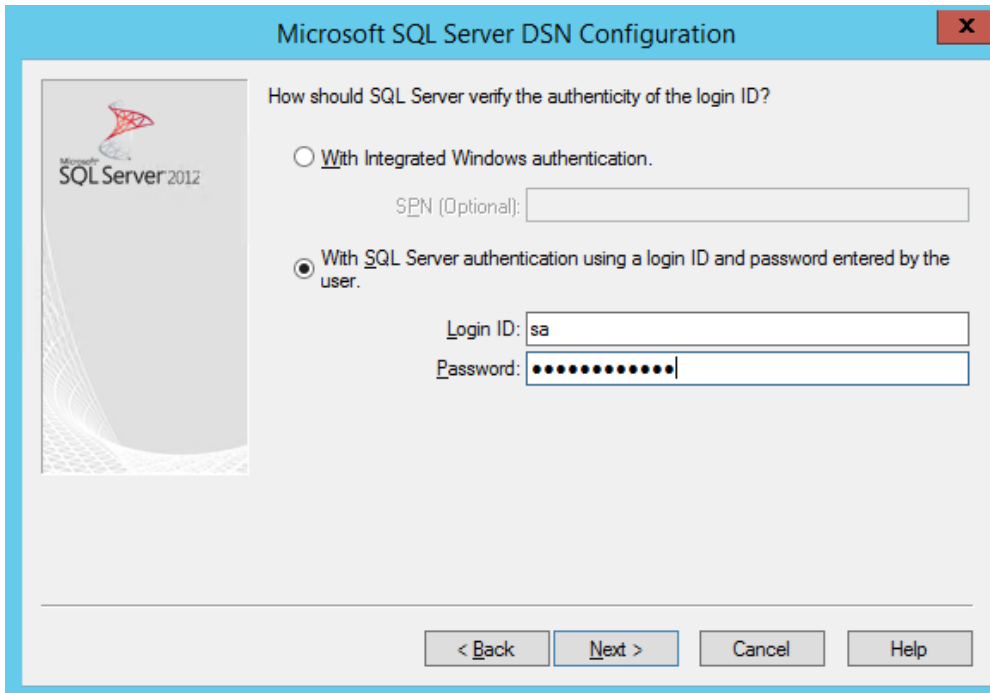
## MISER ODBC BRIDGE

Fill in the *Name* and *Description* fields with “miser”. Fill in the *Server* field with the name of the SQL Server name (in this case “phidbs”) and click **[Next>]**.



The screenshot shows the 'Microsoft SQL Server DSN Configuration' dialog box. The title bar includes a close button (X). On the left is a logo for Microsoft SQL Server 2012. The main text reads: 'This wizard will help you create an ODBC data source that you can use to connect to SQL Server. What name do you want to use to refer to the data source?'. Below this, the 'Name' field contains 'miser'. The next question is 'How do you want to describe the data source?', with the 'Description' field containing 'miser'. The final question is 'Which SQL Server do you want to connect to?', with a dropdown menu showing 'phidbs'. At the bottom are buttons for 'Finish', 'Next >', 'Cancel', and 'Help'.

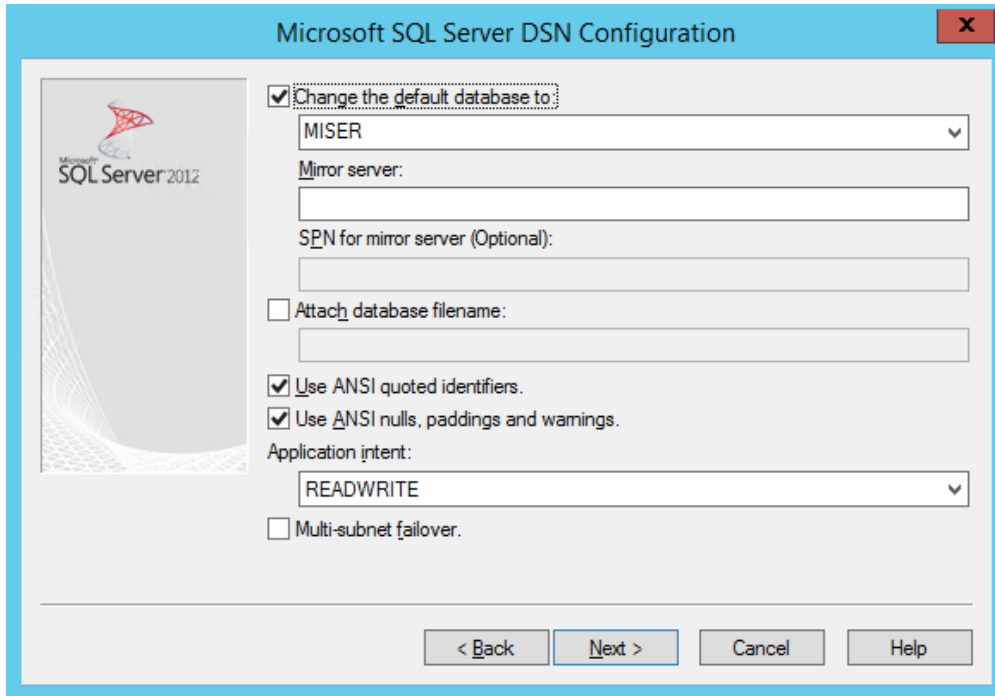
On the subsequent dialog box select “With SQL Server authentication ...” and fill in the *Login ID* and *Password* fields and click **[Next>]**.



The screenshot shows the second step of the 'Microsoft SQL Server DSN Configuration' dialog box. The title bar includes a close button (X). On the left is the same Microsoft SQL Server 2012 logo. The main text asks: 'How should SQL Server verify the authenticity of the login ID?'. There are two radio button options: 'With Integrated Windows authentication.' (unselected) and 'With SQL Server authentication using a login ID and password entered by the user.' (selected). Below the second option, there is an 'SPN (Optional):' field. The 'Login ID:' field contains 'sa' and the 'Password:' field contains a series of dots. At the bottom are buttons for '< Back', 'Next >', 'Cancel', and 'Help'.

## MISER ODBC BRIDGE

Checkmark “Change the default database to:” and then select “MISER” from the drop-down list and click [Next>].

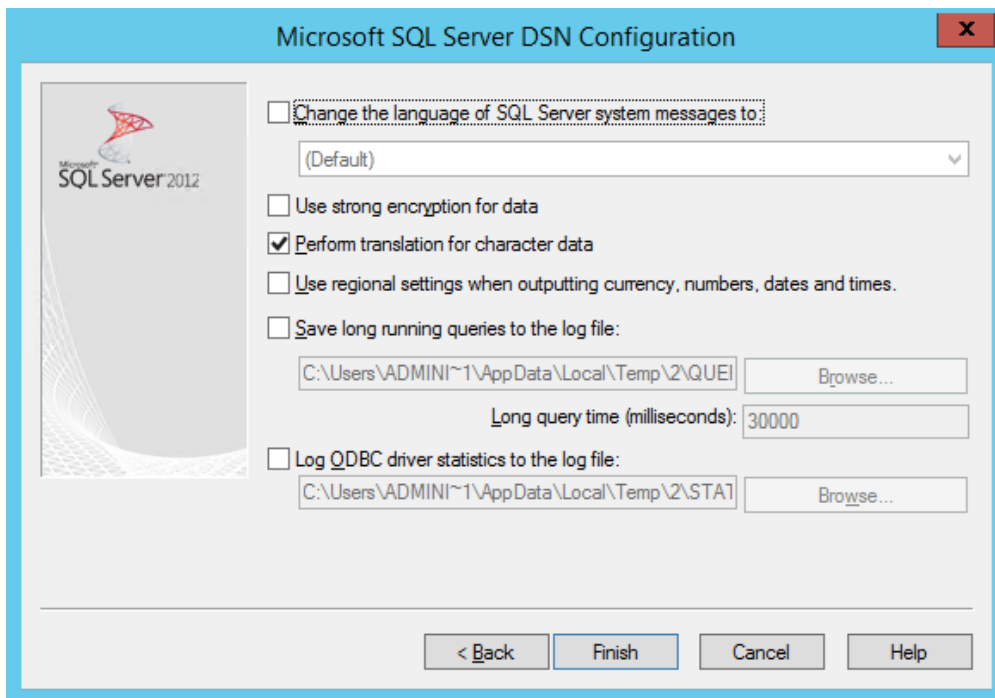


The screenshot shows the "Microsoft SQL Server DSN Configuration" dialog box. On the left is a logo for Microsoft SQL Server 2012. The main area contains the following settings:

- Change the default database to:
  - MISER
- Mirror server: [Empty text box]
- SPN for mirror server (Optional): [Empty text box]
- Attach database filename: [Empty text box]
- Use ANSI quoted identifiers.
- Use ANSI nulls, paddings and warnings.
- Application intent:
  - READWRITE
- Multi-subnet failover.

At the bottom are buttons: < Back, Next >, Cancel, and Help.

On the next dialog box, checkmark “Perform translation for character data” and click [Finish].



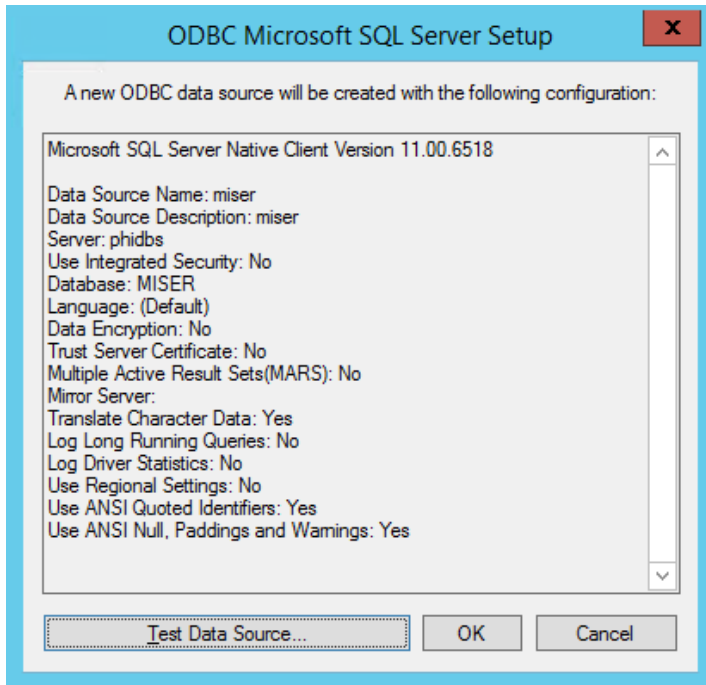
The screenshot shows the "Microsoft SQL Server DSN Configuration" dialog box. On the left is a logo for Microsoft SQL Server 2012. The main area contains the following settings:

- Change the language of SQL Server system messages to:
  - (Default)
- Use strong encryption for data
- Perform translation for character data
- Use regional settings when outputting currency, numbers, dates and times.
- Save long running queries to the log file:
  - C:\Users\ADMINI~1\AppData\Local\Temp\2\QUEI [Browse...]
  - Long query time (milliseconds): 30000
- Log ODBC driver statistics to the log file:
  - C:\Users\ADMINI~1\AppData\Local\Temp\2\STAT [Browse...]

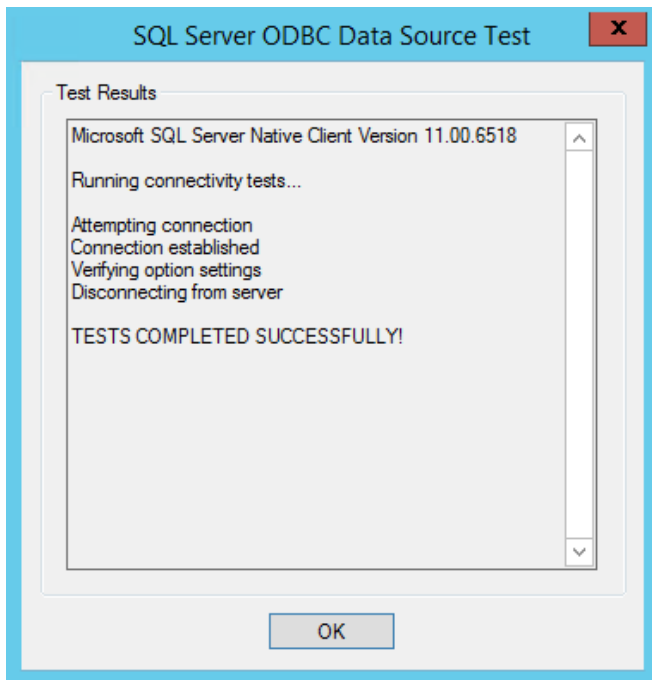
At the bottom are buttons: < Back, Finish, Cancel, and Help.

## MISER ODBC BRIDGE

On the configuration summary dialog box, click **[Test Data Source]**.



When the test results are completed successfully, click **[OK]** and then again on the next dialog box.



## MISER ODBC BRIDGE

**NOTES:** The *Name* field in the Microsoft SQL Server DSN Configuration dialog box represents the name that will be assigned to the resulting system DSN. In general, the name MISER is selected to distinguish the DSN from another customer DSN.

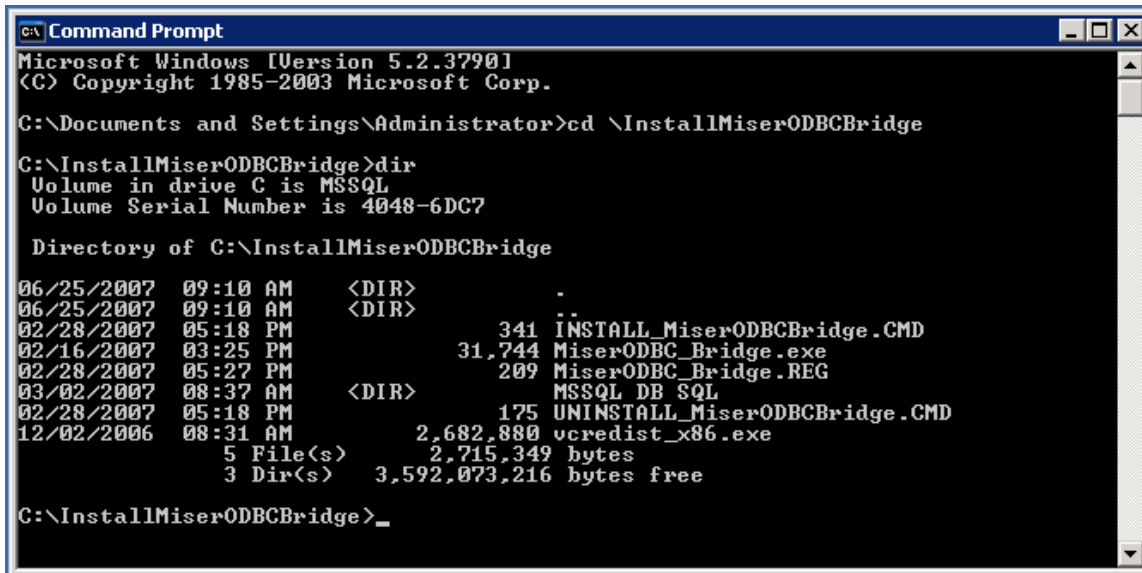
The *Description* field should contain a helpful text description of the function for this DSN. In general, a good description will keep system administrators from accidentally deleting a DSN when they do not recognize its function.

*phidbs* is the name of the server for this demonstration; your project server will have a different name specified by the system administrator.

## MISER ODBC BRIDGE

### MISER ODBC Service Installation

The MISER ODBC Service must be installed by executing the install script which is located in the install directory. Run a command window and change directory to the install directory as shown below:



```
c:\ Command Prompt
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>cd \InstallMiserODBCBridge

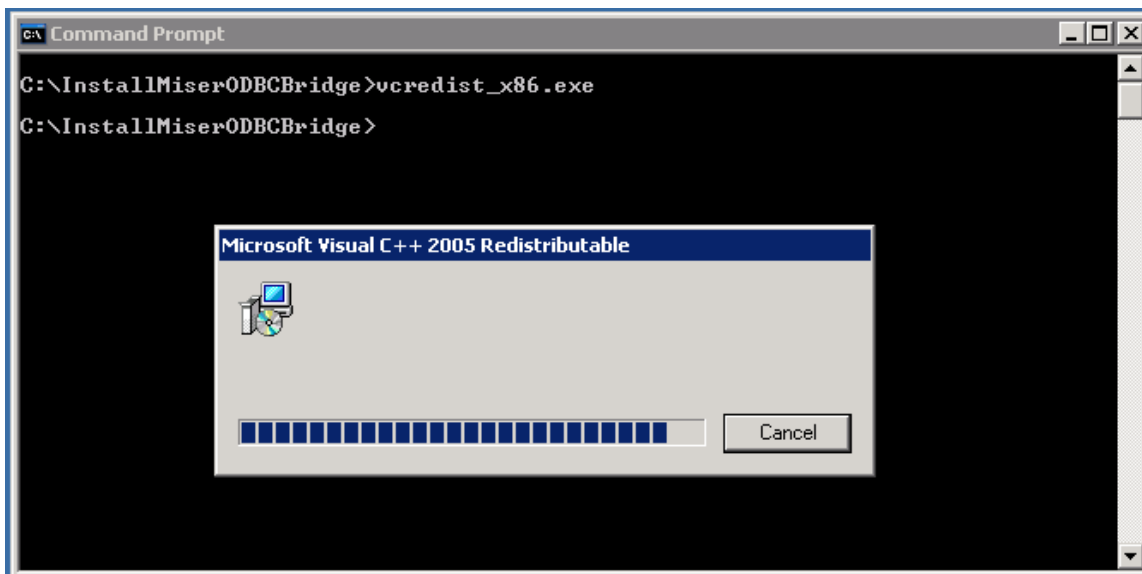
C:\InstallMiserODBCBridge>dir
Volume in drive C is MSSQL
Volume Serial Number is 4048-6DC7

Directory of C:\InstallMiserODBCBridge

06/25/2007  09:10 AM    <DIR>          .
06/25/2007  09:10 AM    <DIR>          ..
02/28/2007  05:18 PM              341  INSTALL_MiserODBCBridge.CMD
02/16/2007  03:25 PM          31,744  MiserODBC_Bridge.exe
02/28/2007  05:27 PM              209  MiserODBC_Bridge.REG
03/02/2007  08:37 AM    <DIR>          MSSQL DB SQL
02/28/2007  05:18 PM              175  UNINSTALL_MiserODBCBridge.CMD
12/02/2006  08:31 AM          2,682,880  vcredist_x86.exe
           5 File(s)          2,715,349 bytes
           3 Dir(s)          3,592,073,216 bytes free

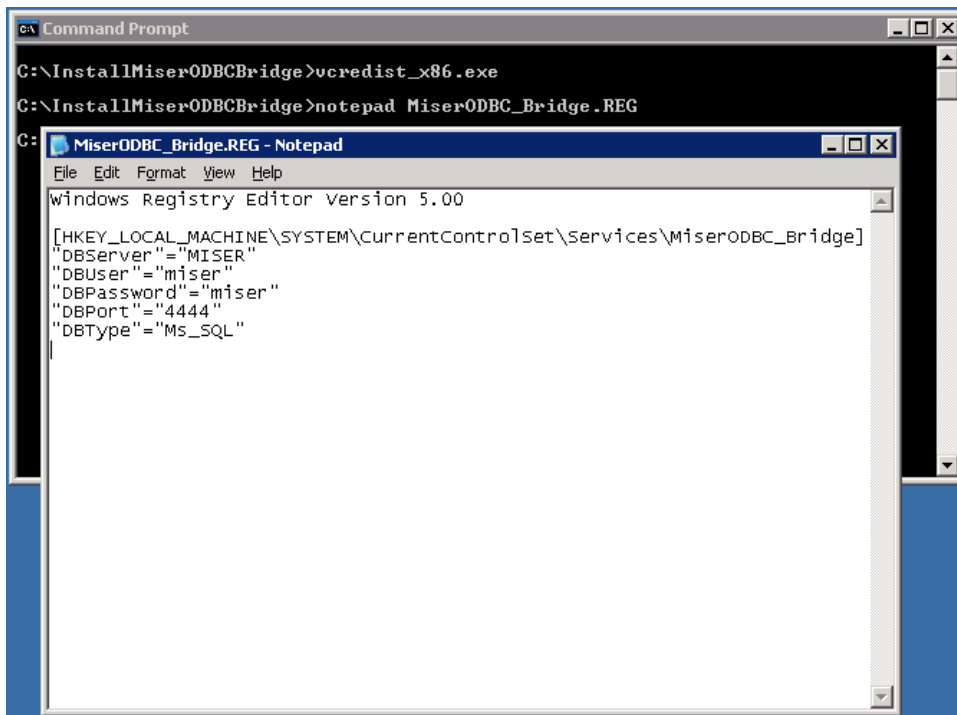
C:\InstallMiserODBCBridge>_
```

The first step is to install the VC runtime libraries used by the MISER ODBC Service. This is done by executing the “vcredist\_x86.exe” program as shown below:



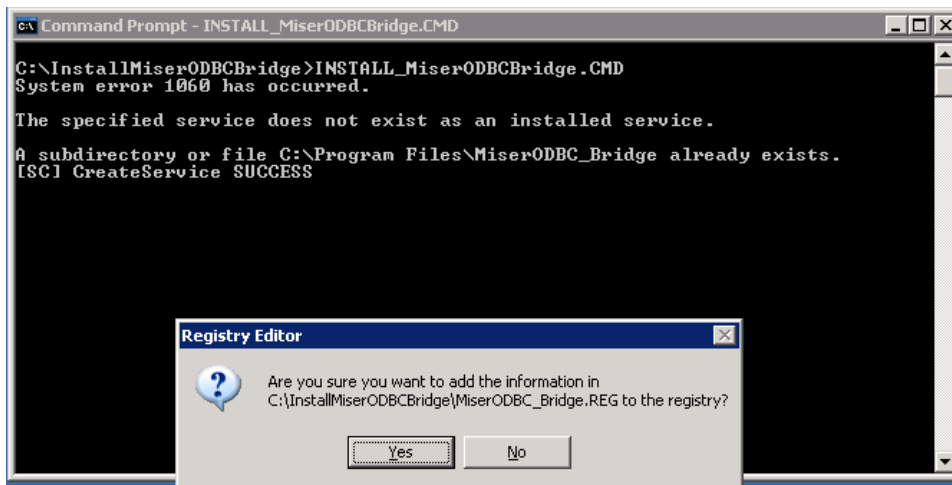
Before the **InstallMiserODBCBridge.CMD** install command procedure is run, the **MiserODBC\_Bridge.REG** file must be altered to match your configuration. This can be done using the “Notepad” program as shown below:

## MISER ODBC BRIDGE



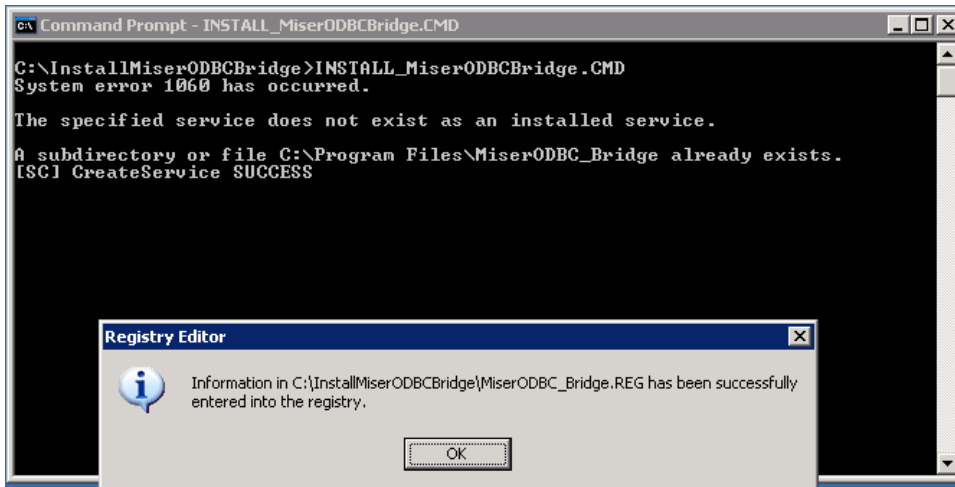
In general, this registry editor script will be correct for most configurations. The only entry that might need changing is the “DBType” entry. This entry must be set to exactly “MS\_SQL”.

To complete the installation, execute the `INSTALL_MiserODBCBridge.CMD` install command procedure as shown below:



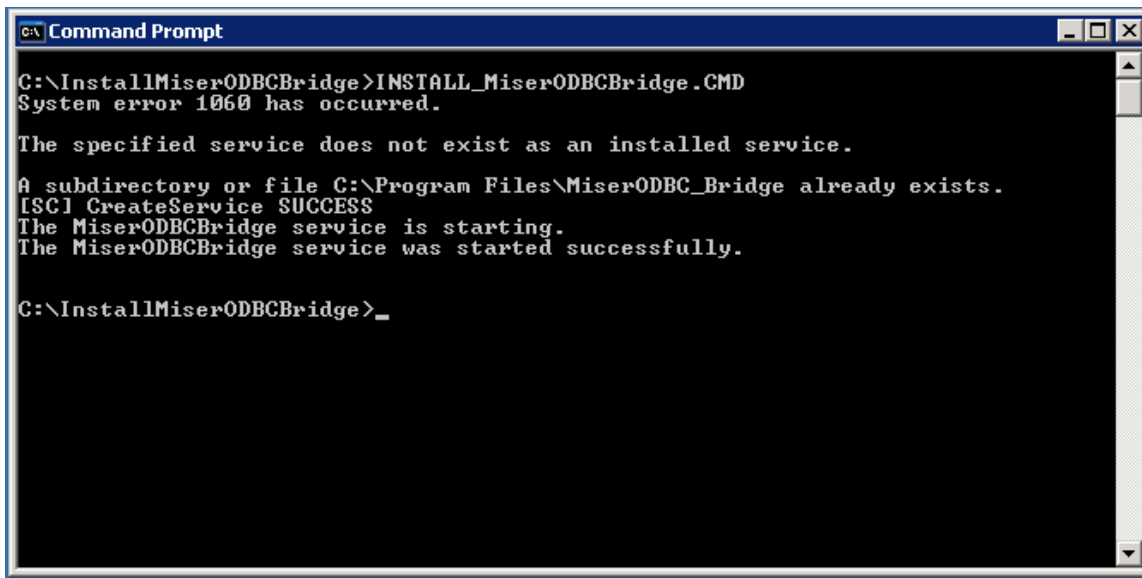
When the portion of the install procedure that alters the registry is executed, you will be asked “Are you sure ...?” Answer this question by clicking **[Yes]**. A **Registry Editor** dialog box will be displayed showing the success as shown below:

## MISER ODBC BRIDGE



After clicking [OK], the install script will continue execution.

The final part of the install will start the service as shown below:



At this point, the following has happened:

1. The Microsoft SQL Server has been installed.
2. The MISER database and tables have been created.
3. The MISER ODBC DSN has been created and tested.
4. The MISER ODBC Bridge Service has been installed and is running.

To complete the Installation of the MISER ODBC Bridge, the OpenVMS/MISER portion of the install has to be finished.

## Single MISER ODBC Bridge Data Setup under OpenVMS

There are two MISER OpenVMS processes that send information directly to the MISER ODBC Bridge Service running on a PC:

- **DBPROC** – Buffers and sends MISER history data and MISER APMSG messages.
- **UPDATE** – Sends data periodically about all changes in MISER point definitions or a point value.

Two other MISER OpenVMS processes send data to the remote MISER ODBC Service running on a PC through the DBPROC process:

- **MEMIO** – Sends history data for a selected set of MISER points to the relational database.
- **DSKIO** – Sends a complete point definition for any new point added to MISER point database.

The DBPROC process must be added to the MISER system by altering the start file for your host computers. The **RUN** command for DBPROC should be placed in the file just after the **RUN** command for HYPROC and is shown below:

```
RUN DBPROC -
!       /mailbox -
        /PROCESS_NAME=DBPROC -
        /PRIORITY=8
```

The start file is named **START\_NAME.DAT** where “NAME” is the MISER host name and this file is located in the **SITE\$DATA** directory (i.e., **START\_DEMVS.B.DAT**).

The CONFIG file must be altered to include the ODBC Bridge database for both host computers as shown below:

```
$DATABASE
! data base files to be included in MVA
!<node Id>, <database name>, <segment(s)>
!
DEMVSA, POINTS, ALL      ! (0-31)   Point Memory region
DEMVSA, HISTORY, ALL    ! (0-31)   HISTORY files
DEMVSA, ALARM, ALL      ! (0-31)   ALARM server
DEMVSA, VIEW, ALL       ! (0)      VIEW files
DEMVSA, EVENTS, ALL     ! (0)      EVENTS file
DEMVSA, CALC, ALL       ! (0)      CPD file
DEMVSA, REPORT, ALL     ! (0-31)   RDU files
DEMVSA, NCC, ALL        ! (0-31)   NCC Parameters
DEMVSA, BRIDGE, ALL     ! (0-31)   Send to DB server
DEMVSA, MTKMAN, ALL
DEMVSA, ALG, ALL
```

The **CONFIG** file is named **CONFIG.DAT** and is located in the **SITE\$DATA** directory.

The following MISER logical names are required to be defined near the end of the configuration file on both MISER host nodes.

## MISER ODBC BRIDGE

```
$ if nodename .eqs. "A" .or. nodename .eqs. "B"
$ then
$     defsys DB_SERVER      "CPCSV"
$     defsys DB_UPDATE     1
$     defsys DB_STANDBY   "NO"
$     defsys DB_APMSG     "NO"
$ endif
```

The configuration file is named **configure\_generic.com** and is located in the **SITE\$COM** directory.

The logical name **DB\_SERVER** should contain the host name or static IP address of the PC that is running the MISER ODBC Service.

The logical name **DB\_UPDATE** is used by the MISER UPDATE process. This defines the rate at which the UPDATE process sends data to the PC that is running the MISER ODBC Service. The update rate is computed by multiplying the value from **DB\_UPDATE** by ninety seconds.

The logical name **DB\_STANDBY** is used to indicate whether the MISER OpenVMS processes on both the online and standby MISER nodes both write to the MISER ODBC Service at the same time or just to the online host by itself. Setting the value of this logical to "NO" indicates that only the online host will write to the MISER ODBC Bridge Service.

The logical name **DB\_APMSG** is used by the MISER API routine **MNET\$APMSG** to determine whether calls to this routine should be written to the MISER ODBC Service and the **ALARM** table. Setting the value of this logical to "NO" indicates that these messages will not be written to the MISER ODBC Service.

After all of these files have been altered, the MISER system must be rebooted for all of these changes to take effect.

## Double MISER ODBC Bridge Data Setup under OpenVMS

There are two MISER OpenVMS processes that send information directly to the MISER ODBC Bridge Service running on a PC:

- **DBPROC** – Buffers and sends MISER history data and MISER APMSG messages.
- **UPDATE** – Sends data periodically about all changes in MISER point definitions or a point value.

Two other MISER OpenVMS processes send data to the remote MISER ODBC Service running on a PC through the DBPROC process:

- **MEMIO** – Sends history data for a selected set of MISER points to the relational database.
- **DSKIO** – Sends a complete point definition for any new point added to MISER point database.

## MISER ODBC BRIDGE

Two DBPROC processes must be added to the MISER system by altering the start file for your host computers. The **RUN** command for DBPROC should be placed in the file just after the **RUN** command for HYPROC and is shown below:

```
RUN DBPROC -
!       /mailbox -
        /PROCESS_NAME=DBPROC -
        /PRIORITY=9

RUN DBPRC2 -
!       /mailbox -
        /PROCESS_NAME=DBPRC2 -
        /PRIORITY=9
```

The start file is named **START\_NAME.DAT** where "NAME" is the MISER host name and this file is located in the **SITE\$DATA** directory (i.e., **START\_DEMVSB.DAT**).

The CONFIG file must be altered to include the ODBC Bridge database for both host computers as shown below:

```
$DATABASE
! data base files to be included in MVA
!<node Id>, <database name>, <segment(s)>
!
DEMVS, POINTS, ALL      ! (0-31)   Point Memory region
DEMVS, HISTORY, ALL    ! (0-31)   HISTORY files
DEMVS, ALARM, ALL      ! (0-31)   ALARM server
DEMVS, VIEW, ALL       ! (0)      VIEW files
DEMVS, EVENTS, ALL     ! (0)      EVENTS file
DEMVS, CALC, ALL       ! (0)      CPD file
DEMVS, REPORT, ALL     ! (0-31)   RDU files
DEMVS, NCC, ALL        ! (0-31)   NCC Parameters
DEMVS, BRIDGE, ALL     ! (0-31)   Send to DB server
DEMVS, MTKMAN, ALL
DEMVS, ALG, ALL
```

The **CONFIG** file is named **CONFIG.DAT** and is located in the **SITE\$DATA** directory.

The following MISER logical names are required to be defined near the end of the configuration file on both MISER host nodes.

```
$ if master
$ then
$     defsys DB_SERVER      "LALPCS"
$     defsys DB_SERVER_2   "LALPCB"
$     defsys DB_TWODB      "YES"
$     defsys DB_STANDBY    "NO"
$     defsys DB_UPDATE     1
$     defsys DB_APMMSG     "YES"
$     defsys DB_POINT      "YES"
$ endif
```

## MISER ODBC BRIDGE

Before running **XHST** or **XPOINTS** you must select which ODBC server you want to send the MISER data to by using this VMS command for the second ODBC server:

```
DEFINE/USER DB_SERVER DB_SERVER_2
```

immediately followed by the **XHST** or **XPOINTS** command.

Sending data to the first ODBC server does not require the “**DEFINE/USER DB\_SERVER DB\_SERVER\_2**” command. The **DB\_SERVER** logical name will return to the previous setting after the **XHST** or **XPOINTS** command is executed. (Be sure to use the **/USER** switch, otherwise the setting will remain permanent.)

The configuration file is named **configure\_generic.com** and is located in the **SITE\$COM** directory.

The logical name **DB\_SERVER** should contain the host name or static IP address of the PC that is running the MISER ODBC Service.

The logical name **DB\_UPDATE** is used by the MISER UPDATE process. This defines the rate at which the UPDATE process sends data to the PC that is running the MISER ODBC Service. The update rate is computed by multiplying the value from **DB\_UPDATE** by ninety seconds.

The logical name **DB\_STANDBY** is used to indicate whether the MISER OpenVMS processes on both the online and standby MISER nodes both write to the MISER ODBC Service at the same time or just to the online host by itself. Setting the value of this logical to “NO” indicates that only the online host will write to the MISER ODBC Bridge Service.

The logical name **DB\_APMSG** is used by the MISER API routine **MNET\$APMSG** to determine whether calls to this routine should be written to the MISER ODBC Service and the **ALARM** table. Setting the value of this logical to “NO” indicates that these messages will not be written to the MISER ODBC Service.

After all of these files have been altered, the MISER system must be rebooted for all of these changes to take effect.

### MISER ODBC Bridge Testing Under OpenVMS

After a reboot, MISER should be running the **DBPROC** process and should be talking with the MISER ODBC Service on a PC.

To check if the **DBPROC** process is running, execute the “WATCH SHOW ALL” command from the current on-line MISER host computer as shown below:

```
DEMVSA$ wat sh all
MISER 6.08 has been up since 30-MAY-07 7:48:36.230 D
  Pid   Process Name   State  Pri    I/O    CPU      Page flts Ph.Mem
00000444 ADPROC          LEF     8   41414  0 00:00:03.36    701 11408
00000465 ALARM           LEF    13  201419  0 00:00:06.87   1041 17456
00000431 APPROC          LEF    13 1595167  0 00:18:40.71 783551  2416
0000044B AXPROC          LEF     6 75438735  0 00:15:52.17   1102 18560
0000044E CPPROC          LEF     7  5124301  0 00:04:29.78   1057 17728
00000432 CSPROC          CEF    14  6662114  0 00:00:55.16    229  4432
```

*All information contained in this document is the sole property of HSQ Technology. Any reproduction in part or whole without the written permission of HSQ Technology is prohibited.*

## MISER ODBC BRIDGE

00000439	DBPROC	LEF	9	41356734	0 00:08:21.60	1412	23312
00000449	DIALER	LEF	14	1397237	0 00:01:01.05	191	4112
00000430	DSKIO	LEF	10	699212	0 00:00:29.81	4398	4048
00000451	EVPROC	LEF	6	77252	0 00:00:01.00	262	5168
00000442	FOPROC	HIB	12	11	0 00:00:00.03	150	3104
00000450	GPPROC	LEF	4	634928	0 00:05:38.05	1029	17312
000004C1	HOSTIP	LEF	28	1008315	0 00:00:00.66	1094	18080
0000044F	HSTRY	LEF	3	8042060	0 00:03:13.73	158	3216
00000438	HYPROC	LEF	6	*****	0 01:09:23.81	2960	33072
0000042E	MEMIO	LEF	20	*****	0 00:27:23.69	1074	18048
0000047A	MTKMAN	LEF	6	82149	0 00:02:52.63	1577	27232
00000445	NCCFA2	LEF	27	*****	0 02:04:10.26	1149	19440
00000448	NCCFA3	LEF	27	12595484	0 00:05:44.82	198	4288
00000446	NCCFA4	LEF	27	36117194	0 00:12:12.59	265	5376
00000447	NCCFA5	LEF	27	44465944	0 00:16:03.28	277	5632
0000044D	NCCFA6	LEF	27	2885956	0 00:00:56.83	193	4160
00000C98	NCCFAC	LEF	27	20687495	0 00:02:34.26	234	4784
0000043E	NCPROC	LEF	5	190	0 00:00:00.03	174	3616
0000043D	NETIO	HIB	31	*****	0 00:32:43.59	2607	42720
00000479	NTPROC	LEF	18	33	0 00:00:00.01	238	4208
0000042F	RAPROC	LEF	5	51823	0 00:00:00.49	1018	16960
00000435	RTUINI	LEF	4	58644	0 00:00:01.05	1032	17216
00000464	RUPROC	LEF	6	584689	0 00:03:47.07	2424	10944
000004C0	RUTASK	LEF	4	2605	0 00:00:00.12	143	2960
00000434	SSPROC	LEF	6	3777735	0 00:33:14.66	1027	17184
00000437	TIMER	HIB	29	7355934	0 00:00:28.17	361	3408
00000452	TRPROC	LEF	6	1184058	0 00:00:19.71	248	4656
00000466	UPDATE	LEF	3	4077531	0 00:04:57.31	1070	17856
00000433	USPROC	LEF	10	1434826	0 00:00:20.85	153	3232

To check to see if the MISER OpenVMS system is making connections with the MISER ODBC Service, the **GETELG** command can be run as shown below:

```
DEMVSA$ r mnet$exe:getelg
GETELG - Get Error Log
25-JUN-2007 15:08:21.17 GETELG Sending ELG
[8] INSERT ELG message, response required
25-JUN-2007 15:08:21.17 GETELG Rcvd ELG allocated events=6 active connections=5
```

The **GETELG** command shown above indicates five connections to the MISER ODBC Service. For each host computer there will be two connections, one from **DBPROC** and the other from **UPDATE**. The host computer where you run the **GETELG** command will temporarily create an additional connection. For two host computers that means  $2 \times 2 + 1 = 5$  connections.

Another method for checking if the MISER OpenVMS system is making connections with the MISER ODBC Service, the following command can be executed:

## MISER ODBC BRIDGE

```
DEMVSA$ ucx sh dev/port=4444
```

Device_socket	Type	Port		Service	Remote
		Local	Remote		Host
bg838	STREAM	49155	4444		10.5.70.20
bg839	STREAM	49156	4444		10.5.70.20

The above command shows two connections using TCP remote port 4444 to a host. The TCP/IP address should be the address on the node defined by the **DB\_SERVER** logical name defined in the configuration file.

## MISER ODBC BRIDGE

A more verbose version of the same command can be repeated to get I/O statistics as shown below:

```
DEMVSA$ ucx sh dev/port=4444 /fu
```

```
Device_socket: bg838      Type: STREAM
```

```
          LOCAL                      REMOTE
Port:      49155                      4444
Host:     10.5.69.129                10.5.70.20
Service:
```

			RECEIVE	SEND
		Queued I/O	0	0
Q0LEN	0	Socket buffer bytes	0	0
QLEN	0	Socket buffer quota	62780	62780
QLIMIT	0	Total buffer alloc	0	0
TIMEO	0	Total buffer limit	502240	502240
ERROR	0	Buffer or I/O waits	1	0
OOBMARK	0	Buffer or I/O drops	0	0
		I/O completed	30353	49584
		Bytes transferred	1815680	940471324

Options: None

State: ISCONNECTED PRIV

RCV Buff: ASYNC

SND Buff: ASYNC

```
Device_socket: bg839      Type: STREAM
```

```
          LOCAL                      REMOTE
Port:      49156                      4444
Host:     10.5.69.129                10.5.70.20
Service:
```

			RECEIVE	SEND
		Queued I/O	0	0
Q0LEN	0	Socket buffer bytes	0	0
QLEN	0	Socket buffer quota	62780	62780
QLIMIT	0	Total buffer alloc	0	0
TIMEO	0	Total buffer limit	502240	502240
ERROR	0	Buffer or I/O waits	1	0
OOBMARK	0	Buffer or I/O drops	0	0
		I/O completed	383	638
		Bytes transferred	3056	120796

Options: None

State: ISCONNECTED PRIV

RCV Buff: ASYNC

SND Buff: ASYNC

## User-Executed MISER Applications

The MISER **XHST** and **XPOINTS** commands are run by MISER users. Each command is used to transfer MISER data, which was lost because of Network or PC problems to the MISER ODBC Service running on a remote PC.

The **XHST** command is used to transfer MISER history data to the MISER database tables that are being populated by the MISER ODBC Service.

The **XPOINTS** command is used to transfer *Point* definition and *Last Value* information for any MISER point to the MISER database tables being populated by the MISER ODBC Service.

## How to Populate the POINTS Table from OpenVMS

The **XPOINTS** command can be used to populate the **POINTS** table in the MISER database. The **XPOINTS** command can take any acronym or wildcard acronym as input. The **XPOINTS** program will send the current copy of the complete point record for the specified acronyms as shown below:

```
DEMVSBS$ xpoints *
XPOINTS - Sending Points to a DB server
60TH-DIS-FLO           ANA      0.00 MGD
60TH-EL-TK-LVL   EL_TNK_LVL  ANA      11.8 FT
60TH-ELCUR-PER   TANK % FULL ANA      47.8 %
60TH-ELCUR-VOL   TANK VOL    ANA      0.47 MGAL
60TH-ELTNK-PER   TANK_VOL_PER ANA      47.7 %
60TH-ENG-P4-ON   P4_ENG_STAT  BIN OFF      (ON=1)
60TH-GDCUR-PER   TANK % FULL ANA      47.8 %
60TH-GDCUR-VOL   TANK VOL    ANA      1.60 MGAL
60TH-GS-SW-POS   GS_SW_POS    BIN REMOTE   (ON=1)
60TH-GS-V-CLO    GS_VLV_CLOSD BIN          (ON=1)

...

YOR-STA-FIRE      STATION_FIRE BIN NORMAL   (ON=1)
YOR-STA-FLOOD     STATION_FLD  BIN NORMAL   (ON=1)
YOR-STA-INTRU     STA_SECURITY BIN NORMAL   (ON=1)
YOR-STA-TROB      STAT_TROUBLE BIN NORMAL   (ON=0)
YOR-TEMP-HI       STATION_TEMP BIN NORMAL   (ON=1)
YOR-TEMP-LO       STATION_TEMP BIN NORMAL   (ON=1)
YOR-TK-HI-SPT     TK HIGH LVL  ANA      30.0 FT
YOR-TK-LO-SPT     TK LOW LVL   ANA      20.0 FT
Total points sent = 12895
```

## How to Populate the HST Table from OpenVMS

The **XHST** command can be used to populate the **HST** table in the MISER database. The **XHST** command accepts a file name that represents the history file that is to be read and transferred to the MISER database.

The **XHST** command accepts a filename as a command line argument that represent the history that will be sent to the MISER database.

MISER history files are stored in the **MNET\$HST** directory on the OpenVMS MISER host. Each history file has a name related to the date it was created and populated. The history file name is created as **yymmdda.hst** where:

- “yy” is last two digits from the year
- “mm” is two digits from the month
- “dd” is two digits from the day

For example, a history file for September 5, 2003 will be named as **030905a.hst**.

After the **XHST** program is run, the user must specify a data transmitting start and stop timestamp for the history file. All time intervals are specified using military time (24-hour clock).

The following is an example of using the **XHST** command to send the contents of a history file to the MISER database.

```
DEMVSA$ xhst 070520a.hst
  HST File names are: 070520A.HST
  Sending records from mnet$hst:070520A.HST
Enter Start Time [ 0:00:00] :
Enter Stop Time [ 0:00:00] :
      1406788 Records from existing      1486981 records was send
```

## How to Add or Remove Points from OpenVMS MISER

**MEMIO** and **DBPROC** will only send MISER history to the MISER ODBC Service if the data is also sent to OpenVMS MISER history by **HYPROC**. To place a point on OpenVMS MISER history the **HON** command must be executed. The **HON** command can take any acronym or wildcard acronym as input as shown below:

```
DEMVSA$ hon MTK-TEST*
  HON - Put point on history
Put MTK-TEST1      on history [Y/N/Q/G] ? G
Point MTK-TEST1   is on history
Point MTK-TEST2   is on history
Point MTK-TEST3   is on history
```

## MISER ODBC BRIDGE

After a point is "ON" OpenVMS MISER history, it can be added to the MISER ODBC database history using the **DBON** command. The **DBON** command can take any acronym or wildcard acronym as input as shown below:

```
DEMVSA$ dbon mtk-test*
  DBON - Put point on DataBase report
Put MTK-TEST1      on DataBase report  [Y/N/Q/G] ? g
Point MTK-TEST1   is on DataBase report
Point MTK-TEST2   is on DataBase report
Point MTK-TEST3   is on DataBase report
```

MISER points can be removed from MISER ODBC database history using the **DBOF** command. The **DBOF** command can take any acronym or wildcard acronym as input as shown below:

```
DEMVSA$ dbof mtk-test*
  DBOFF - Take point off DataBase report
Take MTK-TEST1    off DataBase report  [Y/N/Q/G] ? g
Point MTK-TEST1   is off DataBase report
Point MTK-TEST2   is off DataBase report
Point MTK-TEST3   is off DataBase report
```

### Checking Which Points are Sent to the MISER ODBC Bridge Service

The **DBRPT** command will list the OpenVMS MISER points that are being sent to MISER ODBC database history. The **DBRPT** command will report on all MISER points if an acronym is not specified as input. The **DBRPT** command can also take any acronym or wildcard acronym as input as shown below.

```
DEMVSA$ dbrpt mtk-test*
  DBRPT - Points on DataBase Report

Points currently on DataBase Report:
```

```
1300 MTK-TEST1      TEST POINT
1301 MTK-TEST2      TEST POINT
1302 MTK-TEST3      TEST POINT
```

### Checking for Errors Communicating with the ODBC Bridge from OpenVMS

As previously mentioned, the **GETELG** command can be used to test the link from OpenVMS MISER to the MISER ODBC Bridge Service running on a remote PC.

The MISER error logfile also contains messages that can indicate problems with the OpenVMS to ODBC Bridge link. To list the error log for OpenVMS MISER do the following:

```
DEMVSA$ tools
DEMVSA$ err
```

## MISER ODBC BRIDGE

An example of an informational message generated by OpenVMS MISER that concerns the link to a remote PC that is running the MISER ODBC Bridge Service is:

```
[2007-JUN-25 14:53:44.29] DEMVSB  DBPROC Connect to DBServer
```

– or –

```
[2007-JUN-25 14:55:41.20] DEMVSB  UPDATE Connect to DBServer
```

An example of an error message generated by OpenVMS MISER that concerns a link failure of a MISER process to the remote PC running the MISER ODBC Bridge Service is:

```
[2007-MAY-30 07:42:23.47] DEMVSA  DBPROC Failed to receive on port got <error 0>  
                                %MNET-E-TIMEOUT, The requested operation timed out
```

### Checking for Errors Communicating with the ODBC Bridge from Windows

To determine whether an OpenVMS MISER system is communicating with a MISER ODBC Bridge Service from a remote Microsoft Windows PC, the **NETSTAT** command can be used to list the network connection to the remote PC as shown below:

```
C:\Documents and Settings\Administrator>netstat
```

```
Active Connections
```

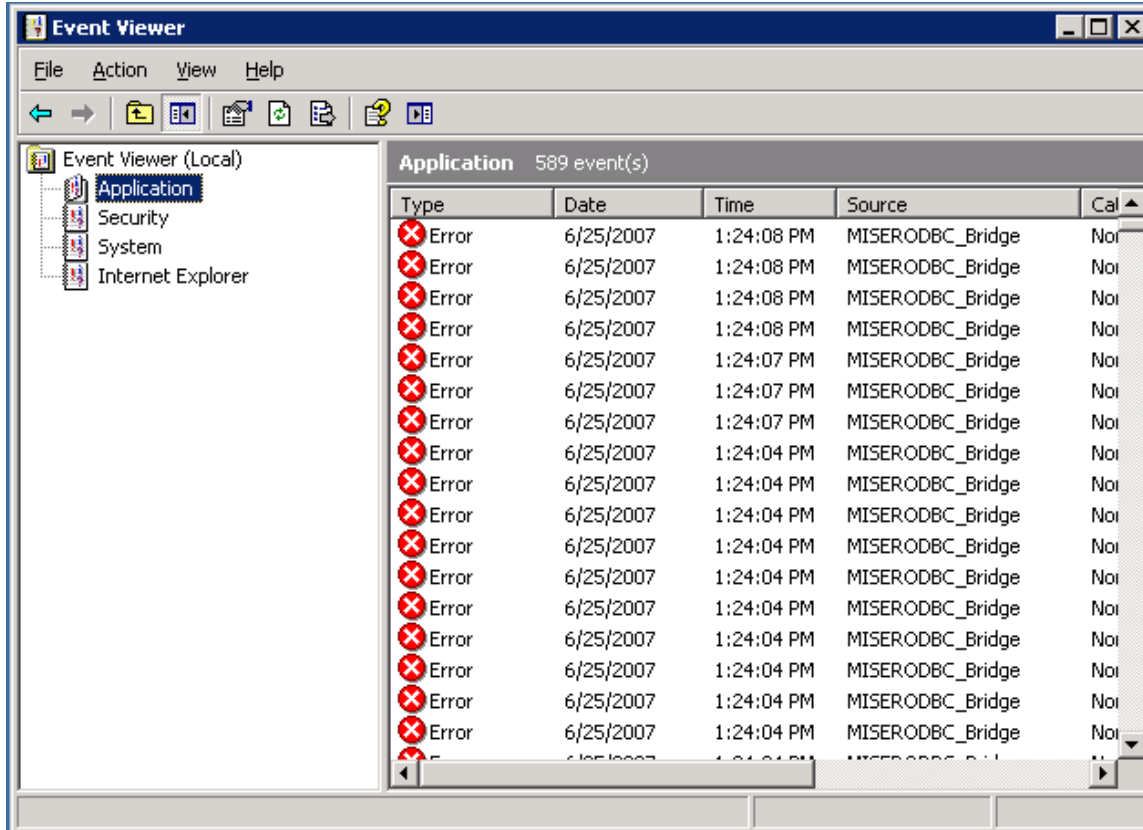
Proto	Local Address	Foreign Address	State
TCP	mssql:4444	10.5.64.133:50698	ESTABLISHED
TCP	mssql:4444	10.5.64.133:55511	ESTABLISHED

```
C:\Documents and Settings\Administrator>
```

This list shows two TCP connections to local port 4444 which is normally used by the MISER ODBC Bridge to accept connections from a remote OpenVMS MISER host. The IP address should be the address of one or more of the OpenVMS MISER hosts.

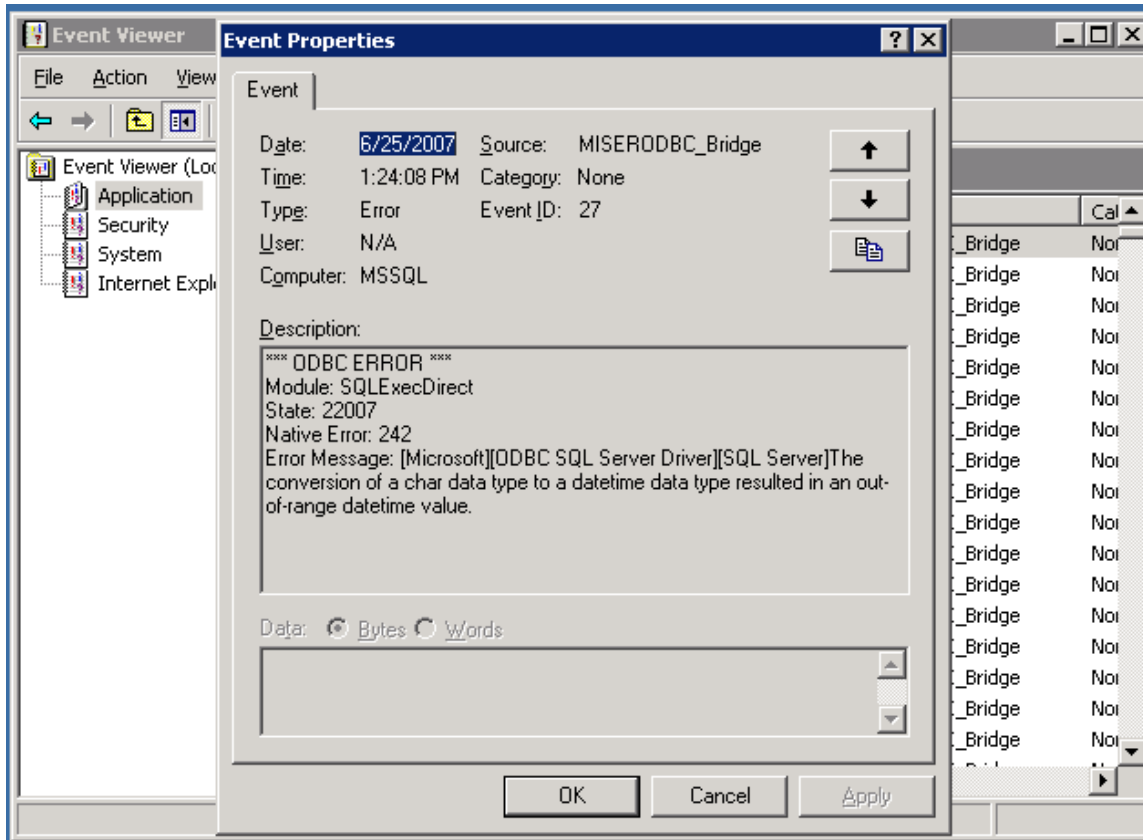
## MISER ODBC BRIDGE

The Windows event log can also be checked for errors from the MISER ODBC Bridge Service. Events from the MISER ODBC Service are recorded in the Application event logfile and can be displayed using the Event Viewer as shown below:



## MISER ODBC BRIDGE

To view a particular event, select it with a double click on the event line. The contents of that event will be displayed as shown below:



## MISER ODBC Bridge Service Table Description

This section describes each of the tables in the MISER database.

### MISER.POINTS Format

-- fixed area

acronym	CHAR (45) unique ,	-- character point acronym
recnm	smallint primary key ,	-- record int
relbin	CHAR (45),	-- Related point
pntnam	CHAR (12),	-- point name
area	CHAR (12),	-- area point is in or near
bldng	CHAR (12),	-- building point is in or near
unit	CHAR (12),	-- unit point is associated with
tsknam	CHAR (8),	-- related task name

-- binary point definition

## MISER ODBC BRIDGE

```
on_code      CHAR (8),
off_code     CHAR (8),
mid_code     CHAR (8),
strtv       CHAR (8),      -- start command verb
stopv       CHAR (8),      -- stop  command verb

-- analog point definition

code         CHAR (8),      -- engineering unit print code

-----

node         CHAR (6),
ncc         smallint,

-----

rtu         smallint,
mux         smallint,

-----

inptn       smallint,      -- input field address
otptn       smallint,      -- output field address

-----

group_id    int,

-----

seg_num     tinyint,      -- segment number
rectyp     tinyint,      -- point type code
subtyp     tinyint,      -- input subtype code
osbtp     tinyint,      -- output subtype code

-----

tirm       tinyint,      -- time interval reporting interval size
aclvl     tinyint,      -- level required to access point
cnlvl     tinyint,      -- level required to control point
relation  tinyint,      -- related point relationship type

-----

almprt     int,          -- bit map of Alarm Printer APn: (n=0,31)

-----

logprt     int,          -- login bit map Printer APn: (n=0,31)
```

*All information contained in this document is the sole property of HSQ Technology. Any reproduction in part or whole without the written permission of HSQ Technology is prohibited.*

## MISER ODBC BRIDGE

```
-- -----  
almdfn      int,          -- alarm handling definition bits  
pntdfn      int,          -- point definition bits  
asdsn       int,          -- ASD slide int  
  
-- -----  
delay       float,        -- alarm delay (seconds)  
  
-- -----  
dvspan      float,        -- Analog span (Eng. Units/count)  
dvbase      float,        -- Analog base (Eng. Units)  
filter      float,        -- Analog filtering time constant (seconds)  
hilim1      float,        -- Analog 1st level hi alarm limit (Eng. Units)  
lowlm1      float,        -- Analog 1st level low alarm limit (Eng. Units)  
hilim2      float,        -- Analog 2nd level hi alarm limit (Eng. Units)  
lowlm2      float,        -- Analog 2nd level low alarm limit (Eng. Units)  
dbandh      float,        -- Analog hi alarm deadband (Eng. Units)  
dbandl      float,        -- Analog low alarm deadband (Eng. Units)  
ratlim      float,        -- Analog rate of change limit (Eng. Units/hour)  
spllo      float,        -- Analog output lo limit (Eng. Units)  
splhi      float,        -- Analog output hi limit (Eng. Units)  
ptkw        float,        -- Binary point demand (KW)  
  
-- -----  
tolrnc      smallint,    -- Analog COS reporting tolerance (% * 100)  
shlcc       smallint,    -- Analog sensor high limits converter counts  
  
-- -----  
sllcc       smallint,    -- Analog sensor low limits converter counts  
fmtcd       smallint,    -- Analog display format code (# of dec.places)  
  
-- -----  
vfdly       smallint,    -- Binary verification delay (seconds)  
minon       smallint,    -- Binary minimum on time (seconds)  
minoff      smallint,    -- Binary minimum off time (seconds)  
  
-- -----  
bonst       smallint,    -- Binary point on status  
boffst      smallint,    -- Binary point off status  
bintst      smallint,    -- Binary point intermediate status  
  
-- -----
```

## MISER ODBC BRIDGE

```
Node_index  smallint,          -- unit index of Node in SYSCOM
NCC_index   smallint,          -- unit index of NCC in SYSCOM
RTU_index   smallint,          -- unit index of RTU in SYSCOM
MUX_index   smallint,          -- unit index of MUX in SYSCOM

-- -----

rel_num     smallint,          -- related point record int
msgnum      smallint,          -- related message int

-- -----

tskpnt      smallint,          -- history
alarm_pri_level  smallint,      -- alarm priority level
rtuport     smallint,          -- RTU communication port ID

-- variable area

pntsts      int,               -- point status
almsts      int,               -- alarm status

-- analog point

anarv       float,             -- analog current point input value (Eng u)
lanarv      float,             -- last value input
spval       float,             -- real output
cosct       int,               -- change of state count

-- binary point

blscm       smallint,          -- binary last command (output value)
cntrl       smallint,          -- control ownership bits
bcrvl       smallint,          -- binary current input value
blsvl       smallint,          -- binary last input
anaval      smallint,          -- Analog current unscaled reading

-- time stamps

tr_time     datetime,          -- most recent tr.
tscur       datetime,          -- most recent input value
alm_time    datetime,          -- alarm
tsout       datetime,          -- output value
lston       datetime,          -- last ON COS time
lstoff      datetime,          -- last OFF
text_value  CHAR (60)          -- text point

);
```

## MISER ODBC BRIDGE

### Table MISER.HST

RECNM	smallint	NOT NULL,
TIME	datetime	NOT NULL,
VALUE	float	NOT NULL,
ALARM	tinyint	NULL,
QUALITY	smallint	NULL,
CONSTRAINT	hst_PK	PRIMARY KEY (RECNM, TIME)

### Table MISER.ALARM

ALARM_ID	tinyint	NOT NULL PRIMARY KEY,
ALARM_TEXT	CHAR(20)	NOT NULL,

The MISER.ALARM table is initialized to a default record set. The default record set for the MISER.ALARM table is:

ALARM_ID	ALARM_TEXT
0	' '
15	'Limit Alarm'
14	'Rate Alarm'
13	'HIHI Alarm'
10	'LOLO Alarm'
12	'HI Alarm'
11	'LO Alarm'
5	'UNCOM COS Alarm'
16	'Verification Alarm'
4	'Undefined Alarm'
2	'ON Alarm'
1	'OFF Alarm'
3	'Intermediate Alarm'

### Table MISER.QUALITY

QUALITY_ID	smallint	NOT NULL PRIMARY KEY,
QUALITY_TEXT	CHAR(46)	NOT NULL,

The MISER.QUALITY table is initialized to a default record set. The default record set for the MISER.QUALITY table is:

QUALITY_ID	QUALITY_TEXT
0	' '
1	'Entered manually'
2	'Edited'
3	'Entered manually and then edited'
4	'Point was down'
5	'Entered manually, Down'
6	'Was edited, Down'
7	'Entered manually, then edited, Down'
8	'Point was disabled'

## MISER ODBC BRIDGE

9 'Entered manually, Disabled'  
10 'Was edited, Disabled'  
11 'Entered manually, then edited, Disabled'  
12 'Point was down and disabled'  
13 'Entered manually, Down, Disabled'  
14 'Edited, Down, Disabled'  
15 'Entered manually, then edited, Down, Disabled'