Minisoft® Middleware

ODBC/32® Driver
JDBC Driver
OLE DB Data Provider

Version 3.0.0

Minisoft, Inc.
1024 First Street
Snohomish, WA 98290
U.S.A.

1-800-682-0200
360-568-6602
Fax: 360-568-2923

Minisoft Marketing AG
Papiermühleweg 1
Postfach 107
Ch-6048 Horw
Switzerland

Phone: +41-41-340 23 20
Fax: +41-41-340 38 66
minisoftag@centralnet.ch

Internet access:
sales@minisoft.com
support@minisoft.com
http//www.minisoft.com

Disclaimer

The information contained in this document is subject to change without notice. Minisoft, Inc. makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Minisoft, Inc. or its agents shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishings, performance, or use of this material.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced, or translated to another programming language without the prior written consent of Minisoft, Inc.

All product names and services identified in this document are trademarks or registered trademarks of their respective companies and are used throughout this document in editorial fashion only and are not intended to convey an endorsement or other affiliation with Minisoft, Inc.

License Agreement

In return for payment of a onetime fee for this software product, the Customer receives from Minisoft, Inc. a license to use the product subject to the following terms and conditions:

- ♦ The product may be used on one computer system at a time: i.e., its use is not limited to a particular machine or user but to one machine at a time.
- ♦ The software may be copied for archive purposes, program error verification, or to replace defective media. All copies must bear copyright notices contained in the original copy.
- ♦ The software may not be installed on a network server for access by more than one personal computer without written permission from Minisoft, Inc.

Purchase of this license does not transfer any right, title, or interest in the software product to the Customer except as specifically set forth in the License Agreement, and Customer is on notice that the software product is protected under the copyright laws.

90-Day Limited Warranty

Minisoft, Inc. warrants that this product will execute its programming instructions when properly installed on a properly configured personal computer for which it is intended. Minisoft, Inc. does not warrant that the operation of the software will be uninterrupted or error free. In the event that this software product fails to execute its programming instructions, Customer's exclusive remedy shall be to return the product to Minisoft, Inc. to obtain replacement. Should Minisoft, Inc. be unable to replace the product within a reasonable amount of time, Customer shall be entitled to a refund of the purchase price upon the return of the product and all copies. Minisoft, Inc. warrants the medium upon which this product is recorded to be free from defects in materials and workmanship under normal use for a period of 90 days from the date of purchase. During the warranty period Minisoft, Inc. will replace media which prove to be defective. Customer's exclusive remedy for any media which proves to be defective shall be to return the media to Minisoft, Inc. for replacement.

ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS IS LIMITED TO THE 90-DAY DURATION OF THIS WRITTEN WARRANTY. Some states or provinces do not allow limitations on how long an implied warranty lasts, so the above limitation or exclusion may not apply to you. This warranty gives you specific rights, and you may also have other rights which vary from state to state or province to province.

LIMITATION OF WARRANTY: Minisoft, Inc. makes no other warranty expressed or implied with respect to this product. Minisoft, Inc. specifically disclaims the implied warranty of merchantability and fitness for a particular purpose.

EXCLUSIVE REMEDIES: The remedies herein are Customer's sole and exclusive remedies. In no event shall Minisoft, Inc. be liable for any direct, indirect, special, incidental, or consequential damages, whether based on contract, tort, or any other legal theory.

Contents

1. Introduction

Which product is right for you?

Minisoft provides a suite of products to allow cross platform access to information. No one size fits all or universal tool will permit the wide variety of applications and environments our customers need. The ODBC and JDBC drivers and the OLE DB data provider are standards based interfaces to access your data.

All three products allow you read and write to TurboImage and Eloquence as if they were like any other database. They also support advanced features such as access to multiple databases, KSAM, flat, and MPE files. The server can be installed on MPE as well as many UNIX and Windows systems supporting Eloquence®, flat files, or Robelle® SD files.

All Minisoft client server products can coexist on your server. This will allow you to provide all the interfaces needed by your developers or software suppliers without concern for compatibility.

What is ODBC?

Open Database Connectivity (ODBC) is a standard or open application programming interface (API) for accessing a database. SQL statements will allow you to access files in a number of different databases. This includes MS Access, Excel, Impromptu, Lotus Approach, Visual Basic, Crystal Reports, FrontPage, Cold Fusion, Speedware, Esperant, Brio, Delphi, PowerBuilder, Active Server Pages, BusinessObjects, .NET etc. In addition to the ODBC software that is built into each compliant program, a separate module or driver will be needed for each database you want to access.

ODBC is based on and closely associated with the Open Group standard Structured Query Language (SQL) Call-Level Interface. ODBC allows programs to use SQL requests that will access databases without having to know the proprietary interfaces to the databases. ODBC handles the SQL request and converts it into a request the individual database system understands.

What is JDBC?

Java Database Connectivity Driver (JDBC) is a standard or open application-programming interface (API) for accessing a database from JAVA programs. The Minisoft JDBC driver implements this API to give JAVA programs access to data in TurboImage and Eloquence® databases, KSAM files, MPE files, Self-describing files, and PowerHouse sub-files.

JDBC is based on and closely aligned with the Open Group standard Structured Query Language (SQL) Call-Level Interface. It allows programs to use SQL requests that will access databases without having to know the proprietary interfaces to the databases. The JDBC driver handles the SQL request and converts it into a request the individual database system understands.

What is OLE DB?

"OLE DB" opens the door to a rich set of development tools and platforms for Microsoft Windows Servers. Microsoft has announced that OLE DB will be the method by which all information is accessed. The roadmap for future Microsoft applications requires using OLE DB data sources to "provide uniform access to data stored in diverse information sources". The Minisoft OLE DB Provider provides that access to your existing TurboImage and Eloquence databases.

Use the Minisoft OLE DB Provider for Image/Eloquence to help solve the following problems:

- Transparent access to your Image/Eloquence data from the 32-bit or 64-bit editions of SQL Server 2005.

- Integrate Image or Eloquence database access smoothly into your .NET application development environment.
- Continue accessing Image or Eloquence data with Microsoft Visual Studio, Borland development tools, Microsoft Access (through VBA),Microsoft Excel, ActiveX Scripts, Crystal Reports, Windows Scripting, IIS web applications (ASP and ASP.NET) by using the Minisoft OLE DB Provider.

Other Products

Minisoft also provides Middleware that permits MPE calls from client systems to your host data using the familiar TurboImage and MPE intrinsic calls. Please contact your Minisoft sales office for details on MiddleMan if you need this option.

For those customers needing concurrent access to their existing host based applications, Minisoft provides a variety of terminal emulation packages on many different client platforms including; Windows, Pocket PC, Mac OSX, and java enabled browsers.

2. How it works

Environments

The choice of installation files and procedures will depend on your operating environment. The product is provided as a client/server application. Clients are the system used to present the data. Servers are the system used to store the data. A common configuration would be an MPE Server with an Image database being accessed by a Windows Client running MS Excel. You may also access data on a UNIX or Windows server with an Eloquence database from Windows, Mac, or UNIX clients.

With proper licensing, there are many combinations of clients and servers which will enable you to access your information. Please contact Minisoft for a current list of supported platforms or to request support for an additional platform.

Components

Minisoft's ODBC/32 product consists of three components: the server, client, and administrative tools along with documentation.

Server

The Server component is installed on the system where the database resides. The server will run in background or foreground processes and listen on one or more TCP ports. Licensing information for **both** the server and clients are stored on the server.

An instance of the server process is launched for each connection made by a client. The process runs its own security context.

On MPE/iX based systems, the server is run from a common Minisoft background process (MSJOB). On UNIX based platforms, the server is usually run from inetd. A

Windows service has been created to control the server as a set of background processes on Windows Servers. The server may also be run from a shell, command prompt, or script.

Client

The Client component is installed on the system where the ODBC/JDBC/OLDEB calls are made. This can be a workstation running Access, Excel or Crystal Reports. It could also be an application server running, for example, SQL Server, IIS, or Oracle. Clients for ODBC can be Windows, Mac OSX, Sun, AIX, HPUX, or Linux. OLE DB is currently available only on Windows. JDBC can run from any Java 5 or later system on your network.

Administrative Tools

The installation of Administrative Tools is optional. They are installed on any currently supported Windows desktop operating system.

These tools consist of the Schema Editor and the Catalog Editor. The Schema Editor (see page 51) provides for control of the data type and name mapping. The Catalog Editor (see page 74) provides very fine control of access to tables and items.

Documentation

The primary reference will be this manual. The Minisoft web site (http://www.minisoft.com) will contain updates, addenda, samples, and other pages of interest.

3. Installation

Sources

From our web site

After registering, you can download the required files based upon the client/server environment you will be using.

The current version of this manual is stored at our web site in PDF format.

Optional Media

Our products can be shipped on CD or other media for off-net installation.

Installing Servers

Installing the server application on MPE/iX

Installation only needs to be done from one PC – not each individual PC. No terminal emulator is needed for this installation. You must have "System Manager" permission to do this.

The installation is in two parts. The first is a common set of files used by all Minisoft Servers (ODBC/JDBC/OLEDB/MiddleMan). The other is an application specific server.

Installing the MSJOB files on MPE/iX

Run "install_msjob.exe" from a Windows based PC. Follow the on-screen prompts.

Installing the Servers on MPE/iX

Run "install_odbc####.exe from a Windows based PC. Follow the on-screen prompts.

Installing the server application on HPUX

Installing server files

1. Place all files in the /opt/minisoft directory. Be sure that the directory is readable and executable from public.

2. Create a symbolic link to your system specific Eloquence library. Choose the ONE best library that matches your HPUX system.

```
   /opt/eloquence6/lib/pa11_32/libimage3k.sl
   /opt/eloquence6/lib/pa20_32/libimage3k.sl
   /opt/eloquence6/lib/pa20_64/libimage3k.sl
   /opt/eloquence6/lib/hpux32/libimage3k.sl
   /opt/eloquence6/lib/hpux64/libimage3k.sl

   For example, if you have a newer PA-RISC system, use:
```

```
ln -s /opt/eloquence6/lib/pa20_32/libimage3k.sl \
 /usr/lib/libimage3k.sl
```

Configuring system files

1. Append the following to your /etc/inetd.conf file:

```
  #
  # odbcsrvr.exe
  #
  odbcsrvr stream tcp nowait root /opt/minisoft/odbcsrvr.exe odbcsrvr.exe
/S
  #
  # odbcsrvr.exe with trace enabled
  #
  #odbcsrvr stream tcp nowait root /opt/minisoft/odbcsrvr.exe odbcsrvr.exe
/S /T
```

2. Append the following to your /etc/services file:

```
  #
  # Minisoft odbcsrvr.exe
  #
  odbcsrvr 30006/tcp
```

3. Append the following to your /etc/pam.conf file:

```
#
# PAM configuration
#
odbcsrvr.exe auth required /usr/lib/security/libpam_unix.1
odbcsrvr.exe account optional /usr/lib/security/libpam_unix.1
odbcsrvr.exe password required /usr/lib/security/libpam_unix.1
#
```

4. For sam to work without error, append /opt/minisoft to the end of (/var/sam/ts/pam_dir.reg).

5. Report the System ID and System Name (see license.exe below) to your Minisoft sales office. Follow the directions returned to license the product.

6. Restart inetd using the command:

   ```
   #inetd -c
   ```

Getting information for licensing

Please report the value from this utility so that a License Number can be generated for your system.

This is a HP-UX (11) executable. Place it in the directory "/opt/minisoft". When run, it should display the System ID and System Name:

```
#/opt/minisoft/license.exe
System ID = [541750568]
System Name = [HP-UX B.11.00 9000/800]
```

```
Use one of the following parameters:
1 - View license information
2 - License a product
5 - Create license file
6 - View license for a product
#
```

Send this information to your Minisoft sales representative. When you receive your reply, follow the following directions:

1. Create an empty license file:

```
#cd /opt/minisoft
#touch MSLICFIL
#./license.exe 5 "your company name"
#
```

2. Execute /opt/minisoft/license.exe with the parameters as supplied in the reply.

```
#/opt/minisoft/license.exe 2
System ID = [1234567890]
System Name = [HP-UX B.11.00 9000/800]
Product ID to License ? 4
User Limit ? 0
Expiration Date ? 20071201
Extension code ? 1976153532
#
```

Installing the server application on Linux

Installing server files

Installing the server application on Windows

Installing server files

Installing Clients

Installing the Minisoft ODBC/32 driver

Windows

You must install the Minisoft ODBC/32 PC client on each network PC that will be running ODBC/32. To do this:

1. Execute odbcxxxc.exe from your ODBC/32 CD.
2. This must be executed on every Windows client that will be accessing data on the server.

Mac OSX

AIX

HPUX

Linux

Solaris

Installing the Minisoft JDBC driver

JDBC connections require two client components. The first is a jar file which contains the "classes" used by your Java applications. The second is a native application component. The native component can be used as a shared library on the same system as the jar file or as a Mid Tier Listener on the client, the server, or a third box. The primary need for a Mid Tier Listener or three tier approach will be when the java application is run inside a browser. The Mid Tier would in those cases be on the web server used. This configuration reduces "sandbox" issues with java network communications.

Jar file

Shared Library

Mid Tier Listener

Installing the Minisoft OLE DB data provider

Installing the Administrative Tools

Installing Minisoft ODBC/32 PC Administrator

Note that the administrative tools installation is not necessary to run any of the clients.

1. Execute odbcxxxa.exe.
2. Follow the on screen instructions.

ODBC/32 Deployment

The following instructions detail methods to distribute the Minisoft ODBC/32 driver to multiple workstations after completing your development. You should review the items listed below and choose the distribution scheme that best suites your environment:

- Web server deployment
- Development for each desktop
- Reporting tools (Crystal Reports, Access, Excel, etc.)

Note: Be certain to use the same driver version for all clients. The MSJOB background process has the capability to operate multiple versions of each driver to aid in managing future driver upgrades.

Web server deployment

Install ODBC/32 client on your web server(s) by executing odbcxxxc.exe from your CD or via the web.

If you are using DSNs to access the ODBC driver, create a System DSN. Web servers usually operate without a current user. The DSN must be available to system level processes. Do not use question marks (?) in DSN fields. User interaction is not possible from the browser to complete the connection request. Use care in protecting the server as passwords are stored (encoded) in the system registry.

If you have included the information needed for the ODBC driver to connect to your database as either File DSN or SQLDriverConnect data, you should be able to use the driver with no further steps.

Development for each desktop

Install the ODBC/32 client on each system where the application will be running by executing odbcxxxc.exe from your CD or via the web.

If you are using DSNs to access the ODBC driver, create a User or System DSN. Use question marks (?) in the DSN fields if needed. User interaction would be required to complete the connection and no passwords would need to be stored in the system registry.

If you have included the information needed for the ODBC driver to connect to your database as either File DSN or SQLDriverConnect data, you should be able to use the driver with no further steps.

Reporting tools

Install the ODBC/32 client on each system where the reporting tools will be used by executing odbcxxxc.exe from your CD or via the web.

If you are using MSOffice'97, you will need to create a File DSN. Most other products can use any type of DSN including User and System.

Upgrading ODBC/32

Obtain the files to upgrade your ODBC/32 from our web or FTP site. Navigate to www.minisoft.com. Under Middleware select ODBC. Select ODBC Updates. Select the updates that best fit your deployment.

4. Client Configuration

ODBC data sources are generally configured through an ODBC Data Source Administrator. This is part of the Windows operating system. For UNIX systems, you can use the Data Source Administrator that is part of unixODBC or iODBC. Application properties can also be used to configure data sources.

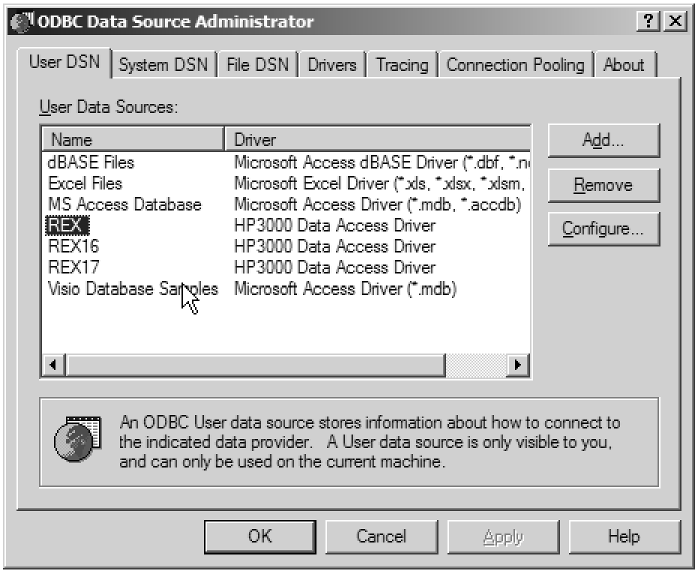
JDBC data sources are configured through either a URL or application properties. These two options are available when using most off the shelf applications or writing your own application.

An OLE DB Data Provider is configured through the application using it. Both the OLE DB data provider and ODBC driver use a common set of Windows dialog boxes.

Configuring a data source

To configure a data source with Windows ODBC Data Source Administrator:

1. Select your PC's Start Menu and click *Settings >Control Panel*. In the Control Panel dialog box, select *ODBC Data Source*.
2. The ODBC Data Source Administrator window appears as shown in Figure 1. Select the *User DSN* tab if it has not already been selected, then click the *Add* button.



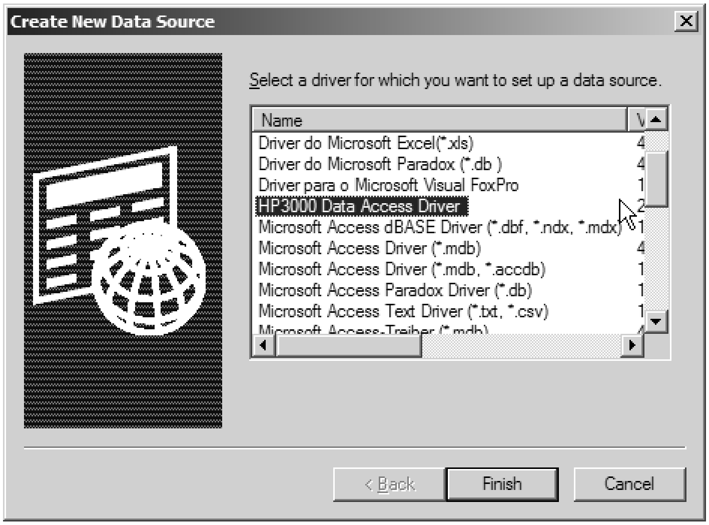
User Data Sources:

Name	Driver
dBASE Files	Microsoft Access dBASE Driver (*.dbf, *.n
Excel Files	Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm,
MS Access Database	Microsoft Access Driver (*.mdb, *.accdb)
REX	HP3000 Data Access Driver
REX16	HP3000 Data Access Driver
REX17	HP3000 Data Access Driver
Visio Database Samples	Microsoft Access Driver (*.mdb)

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

Figure 1

3. The *Create New Data Source* dialog box appears as shown in Figure 2. Select *HP3000 Data Access Driver* if it has not already been selected. Click the *Finish* button.

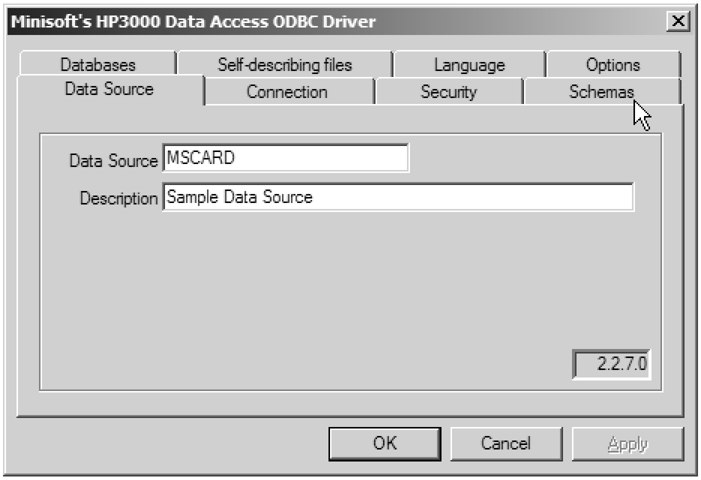


Select a driver for which you want to set up a data source.

Name	V
Driver do Microsoft Excel(*.xls)	4
Driver do Microsoft Paradox (*.db)	4
Driver para o Microsoft Visual FoxPro	1
HP3000 Data Access Driver	2
Microsoft Access dBASE Driver (*.dbf, *.ndx, *.mdx)	1
Microsoft Access Driver (*.mdb)	4
Microsoft Access Driver (*.mdb, *.accdb)	1
Microsoft Access Paradox Driver (*.db)	1
Microsoft Access Text Driver (*.txt, *.csv)	1
Microsoft Access-Treiber (*.mdb)	4

Figure 2

4. *Minisoft's HP3000 Data Access ODBC Driver* dialog box appears as shown in Figure 3. The ODBC Driver dialog box collects together all the information needed to configure ODBC/32 for a given application. You must fill in the Data Source tab, Connection tab, at least one database reference, schema reference or Self-describing file reference in order to create a Data Source Name (DSN).



Data Source MSCARD

Description Sample Data Source

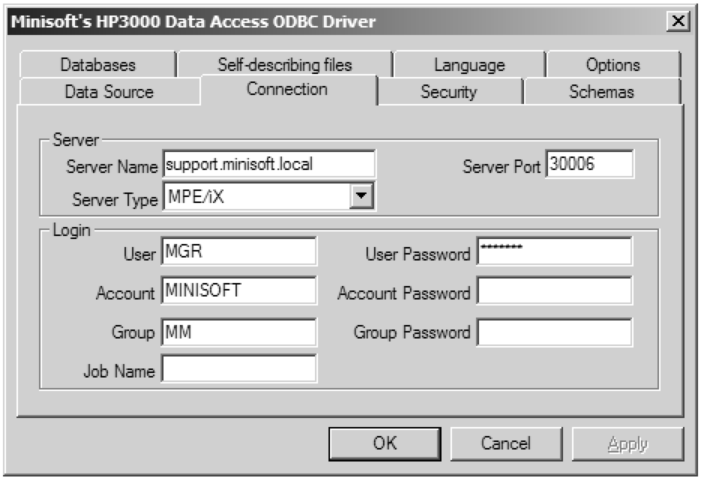
2.2.7.0

Figure 3

5. Under the Data Source tab, fill in the *Data Source* and *Description fields*. This information will appear in the list of available ODBC sources in your database application.

   Note: This information is valid for this computer. Choose a name for the Data Source that will help you remember it and find it later. Useful ideas are the name of the main database you wish to access or the MPE group the files resides in.

6. Click the *Connection* tab in the Minisoft's HP3000 Data Access ODBC Driver window. The display changes to as shown in Figure 4.



Server

Server Name: support.minisoft.local

Server Port: 30006

Server Type: MPE/iX

Login

User: MGR

User Password: ********

Account: MINISOFT

Account Password:

Group: MM

Group Password:

Job Name:

Figure 4

7. At the *Server Name* field, enter your HP e3000 *Server Name* or *IP* address. For *Server Port*, use 30006.

8. Fill in the Login fields with the appropriate HP e3000 user information for accessing your database (this information is required to verify your access rights to the requested database). Use a session name (Security/3000 user name) in the Job Name field to allow user identification.

9. Click the *Security* tab. The following screen appears as shown in Figure 5. If you are using VESOFT’s Security/3000, you should configure the session user password on the Security tab.

   If you are using a rotating password, the password will always be the first one entered when setting up the user. If a ‘?’ is entered in the password field, the user will be prompted at runtime for a password.

   A Catalog is another way to limit access to your data. Using the Catalog Editor (see Chapter 7), individual or groups of users can have access restricted down to the item level.

   The Catalog User is either the MPE/iX user and account or the Windows logon name. If security is to be based on the Windows logon name then ‘<WINUSER>’ is entered in the user field, otherwise the field is left blank.

The catalog password field is reserved for future use. Leave this blank.

If you do not use Security/3000 leave the Security/3000 field blank.

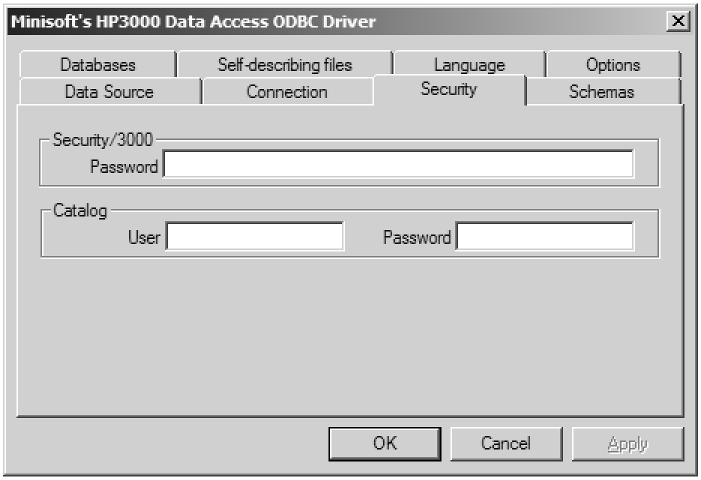
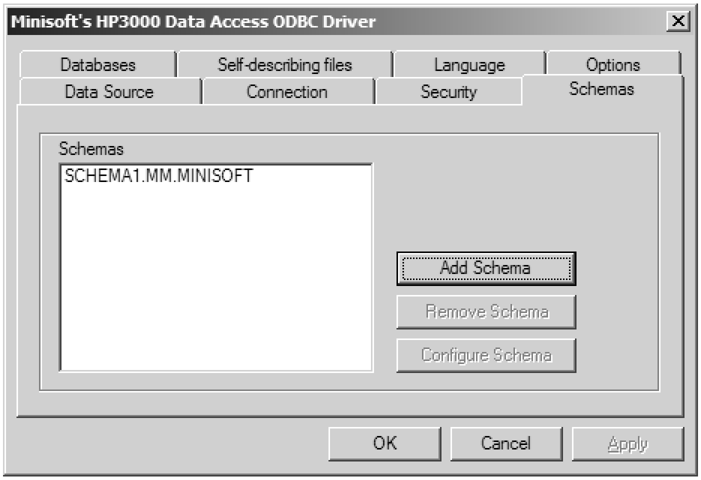


Figure 5

Using Schema Editor Files

Note: You must choose at least one Database reference, Schema reference or Self-describing file reference. You may choose more than one or all three. If you define the same database in both places for the same DSN, you will open the database twice. You should NOT add the same name for both a Schema and Database reference.

1. If you have or intend to define a Schema file using the Schema Editor, click the *Schemas* tab in the Minisoft's HP3000 Data Access ODBC Driver window as shown in Figure 6.

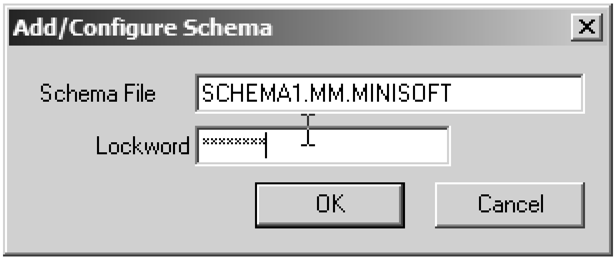


Schemas: SCHEMA1.MM.MINISOFT

Figure 6

2. Click *Add Schema* to enter the name of your schema file. You must have a Schema file to define the record layout for MPE and KSAM files.

3. The *Add/Configure Schema* will display as shown in Figure 7. Enter the fully qualified MPE file name and Lockword (if needed) for the Schema file created with the Schema Editor. Then click *OK*.



Schema File: SCHEMA1.MM.MINISOFT

Lockword: ********

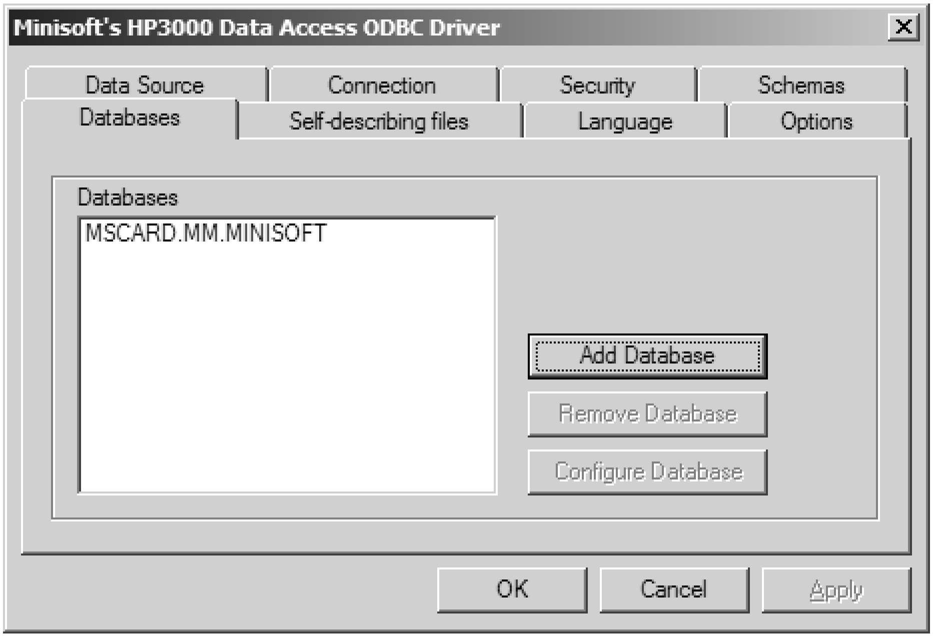
Figure 7

4. A Schema file maybe needed for a TurboImage database for redefining items in a database such as: item name, length, type, extracting substrings, or date format manipulation. See Chapter 6 *Schema Editor* for details on defining a file using the Schema Editor.

Using database files

Note: You must choose at least one Database reference, Schema reference or Self-describing file reference. You may choose more than one or all three. If you define the same database in both places for the same DSN, you will open the database twice. You should NOT add the same name as both a Schema and Database reference.

1. If you intend to define a database, click the *Databases* tab in the Minisoft's HP3000 Data Access ODBC Driver as shown in Figure 8.



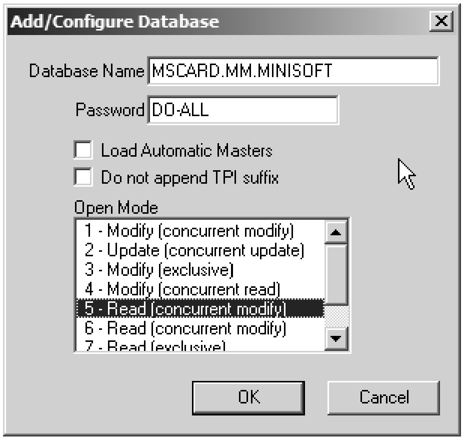
Databases

MSCARD.MM.MINISOFT

Figure 8

2. Then click *Add Database*.
3. The Add/Configure Database window will appear as shown in Figure 9
4. Enter your fully qualified MPE Database Name and Password and select the appropriate Type and Open mode as shown in Figure 9. Click the *OK* button.
   (If you are unaware of the database password, see *Using DBUTIL to find database passwords* on page 148.)

Note: Passwords for your TurboImage database are case sensitive.


Add/Configure Database

Database Name MSCARD.MM.MINISOFT

Password DO-ALL

□ Load Automatic Masters

□ Do not append TPI suffix

Open Mode

1 - Modify (concurrent modify)
2 - Update (concurrent update)
3 - Modify (exclusive)
4 - Modify (concurrent read)
5 - Read (concurrent modify)
6 - Read (concurrent modify)
7 - Read (exclusive)

Figure 9

Using Self-describing files

Note: You must choose at least one Database reference, Schema reference or Self-describing file reference. You may choose more than one or all three. If you define the same database in both places for the same DSN, you will open the database twice. You should NOT add the same name as both a Schema and Database reference.

1. If you intend to define a Self-describing file, select the *Self-describing files* tab from the Minisoft's HP3000 Data Access ODBC Driver. Then click *Add File* as shown in Figure 10.

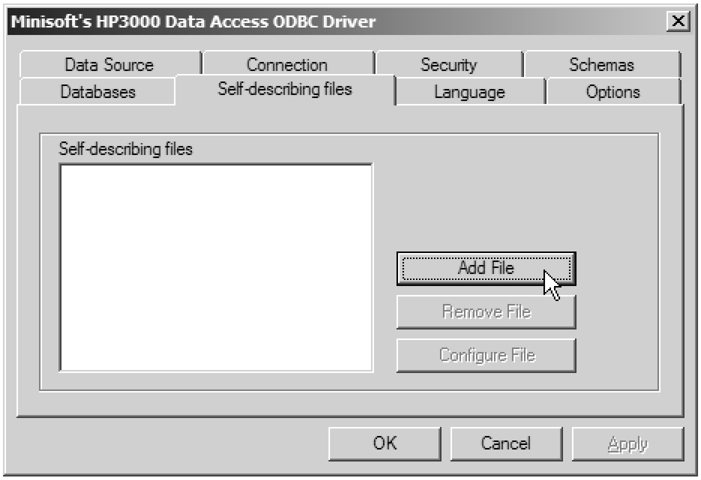
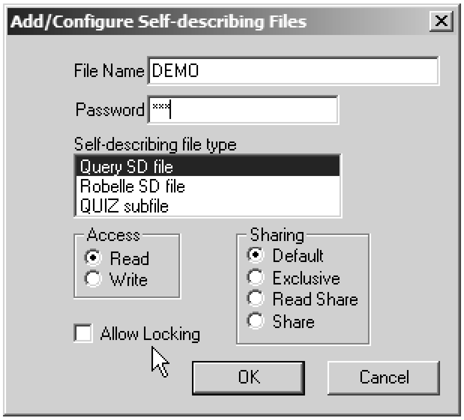

Minisoft's HP3000 Data Access ODBC Driver

Figure 10

2. The Add/Configure Self-describing File dialog box displays as shown in Figure 11.

3. Enter your file name and password and select the appropriate options for configuring your self-describing file.



Add/Configure Self-describing Files

File Name DEMO

Password ***

Self-describing file type

Query SD file
Robelle SD file
QUIZ subfile

Access
● Read
○ Write

Sharing
● Default
○ Exclusive
○ Read Share
○ Share

□ Allow Locking

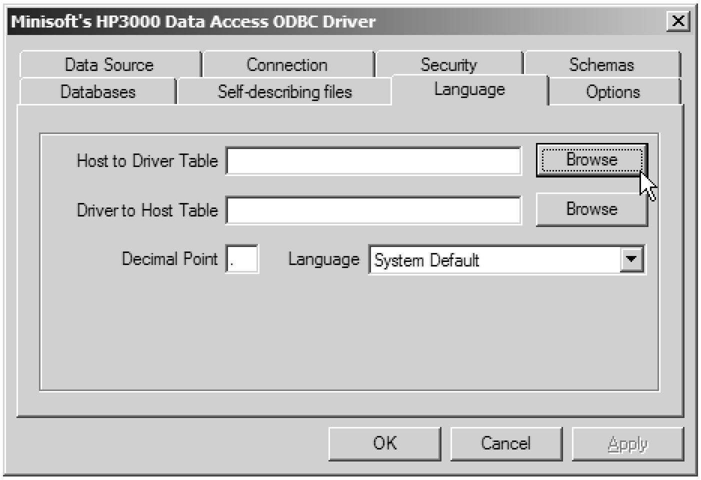
Figure 11

Using Translation Tables with ODBC/32

For conversion between different character sets on the HP e3000 server and the PC client, ODBC/32 uses character set translation tables. For more details on these settings and files, see Translation Tables on page 43.

To configure the respective conversion:

1. Select the Language tab in the ODBC Driver Configuration window as shown in Figure 12.
2. Enter the name of the existing .tbl translation table in the DSN configuration dialog box.
3. After filling out needed information in Minisoft HP3000 Data Access ODBC Driver dialog box, click *OK*. Open your database program and the ODBC data source should now be available.



Host to Driver Table

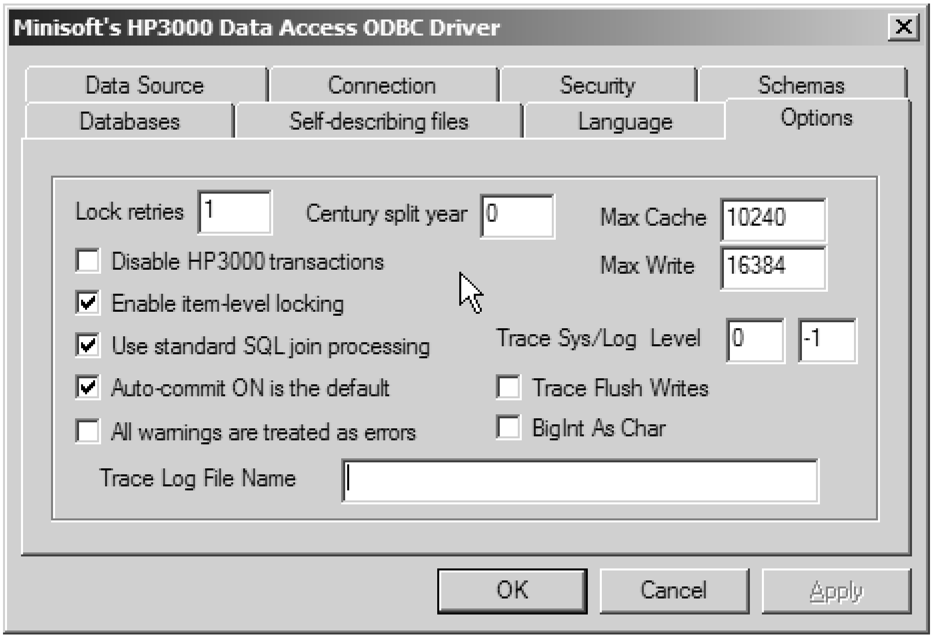
Driver to Host Table

Decimal Point .

Language System Default

Figure 12

Advanced Options



Lock retries 1

Century split year 0

Max Cache 10240

Max Write 16384

□ Disable HP3000 transactions

☑ Enable item-level locking

☑ Use standard SQL join processing

☑ Auto-commit ON is the default

□ All warnings are treated as errors

Trace Sys/Log Level 0 -1

□ Trace Flush Writes

□ BigInt As Char

Trace Log File Name

Figure 13

This tab (Figure 13) shows the advanced options available. The following table shows the mapping to the Connection Properties defined on page 37.

Lock retries – LockRetries

Century split year – CenturySplit

Disable HP3000 transactions – DisableTxns

Enable item-level locking – ItemLocking

Use standard SQL join processing – StandardJoins

Auto-commit ON is the default – AutoCommitDefault

All warnings are treated as errors – WarningsAreErrors

Max Cache – MaxCacheSize

Max Write – MaxWriteSize

BigInt as Char – BigIntAsChar

Trace Sys/Log Level – TraceSysLevel/ TraceLogLevel

Trace Flush Writes – TraceLogFlush

Trace Log File Name - TraceFileName

Creating a Standalone File DSN

Create a File DSN for users who will not need to learn the ODBC Administrator. The information entered in a File DSN is the same general information that would be entered when creating a User or System DSN in the ODBC/32 Administrator. You can specify the same parameters in a FILE DSN (See Connection Properties on page 37).

Standalone File DSN Example

The following is an example of a File DSN that includes connection, database and table information.

Note: All names are case sensitive.

```
[ODBC]
2DriverTable=
2HostTable=CUSTOMERS
DecimalPoint=.
Description=Sample VB created DSN
Language=-1
Server=000.000.000.000
Server Port=30006
User=MGR
User Password=
Account=MINISOFT
Account Password=
Group=MM
Group Password=
Jobname=ODBC
Security Password=
Catalog User=
Catalog Password=
Schema0=TESTSCH.MM.MINISOFT,
ImageDatabase0=MSCARD.MM.MINISOFT,DO-ALL,N,5,0
SDFile0=TESTSUB,LOCKIT,2,3,0
LockRetries=1
DisableTxns=NO
ItemLocking=ENABLED
StandardJoins=YES
WarningsAreErrors=NO
AutoCommitDefault=ON
CenturySplit=50
```

SQLDriverConnect

SQLRETURN **SQLDriverConnect**(
SQLHDBC *ConnectionHandle*,
SQLHWND *WindowHandle*,
SQLCHAR * *InConnectionString*,

SQLSMALLINT *StringLength1*,
SQLCHAR * *OutConnectionString*,
SQLSMALLINT *BufferLength*,
SQLSMALLINT * *StringLength2Ptr*,
SQLUSMALLINT *DriverCompletion*);

Where InConnectionString defined with properties separated with semicolons (See Connection Properties on page 37).

```
"DRIVER=HP3000 Data Access Driver;
Database0=MSDB.DEMO.MINISOFT,WRITER,N,1;
Server=000.000.000.000;Server Port=30006;
Jobname=MSJOB;User=MGR;User Password=PASSWORD;Account=MINISOFT"
```

Connection Properties

This is a list of the common connection properties used by ODBC, JDBC, and OLE DB.

2DriverTable

String Value

This string defines the read Translation Table file. See Translation Tables on page 43.

2HostTable

String Value

This string defines the write Translation Table file. See Translation Tables on page 43.

Account

String Value (Server Type 0 only)

Enter the HP e3000 logon account.

Account Password
AccountPassword

String Value (Server Type 0 only)

Enter the HP e3000 logon account password.

AutoCommitDefault

Enter ON or OFF.
Default state for the connection whether an INSERT, UPDATE or DELETE transaction should be committed.

BigIntAsChar

Contact Minisoft support before changing this value.
Some applications cannot correctly handle very large values as numbers. With this flag set, all columns that would normally be represented as SQL_BIGINT are converted to character and seen as SQL_CHAR.

Catalog Password
CatalogPassword

Deprecated

Catalog User
CatalogUser

Enter the windows user or leave this parameter blank.

CenturySplit

Enter a year for windowing dates.
Used for conversion between 2digit years and 4digit years. 2digit years less than the number will have 2000 added. 2digit years greater than or equal to the number will have 1900 added.

Database<n>

Deprecated – Use ImageDatabase<n>.

DecimalPoint

Insert a character for use with a decimal place.

Description

This is an optional descriptive text string.

DisableTxns

String Value (YES or NO)
Default “NO”

The use of the MPE Transaction Manager (XM) may be disabled with the DisableTxns registry entry. If set to No, ODBC/32 will use XM when executing UPDATES, DELETES, and INSERTS. If set to Yes, ODBC/32 will not use XM when executing UPDATES, DELETES, and INSERTS. This should only be used when an UPDATE, DELETES, or INSERT statement causes the XM’s log file to fill up.

Note: When DisableTxns is set to Yes, a rollback cannot be done if the statement encounters an error or the system goes down. All records modified up to that point will remain modified.

Domain

String Value (Server Type 2 only)

DSN

Data source name.

Group

String Value (Server Type 0 only)

Enter the HP e3000 logon group.

Group Password
GroupPassword

String Value (Server Type 0 only)

Enter the HP e3000 logon group password.

IgnoreUIDPWD

This setting is used with calls from applications which pass the unnecessary user name and password as API parameters.

ImageDatabase<n>

<database name>,<database password>,<load auto master flag Y/N>,<database open mode>,<suppress TPI suffix>

ItemLocking

String Value (ENABLED or DISABLED), case insensitive
Default “ENABLED”

The DSN registry variable ‘ItemLocking’ controls whether item-level locking will be used. The default enables item level locking when possible. Any value other than ‘ENABLED’ will cause all locking to be done at the set level. This is necessary when error -222 (Descriptor list exceeds 4094 bytes) is returned from TurboImage. This typically happens when an INSERT, UPDATE, or DELETE statement is executed many times, or many statements are executed within one transaction.

Examples:

```
ItemLocking=”ENABLED”
```

Enables item level locking.

```
ItemLocking=”DISABLED”
```

Forces set level locking.

If an UPDATE or DELETE (detail datasets only) statement has a where clause that specifies an IMAGE key with an ‘=’ relational operator or a BTREE key with ‘=’, ‘<=’, or ‘>=’ relational operators, the locking statement will be at the item level using the key item and the associated value and relational operator. For INSERT statements on detail datasets, the first item in the item list and its value will be used for the lock.

Jobname

String Value (Server Type 0 only)

Enter the Job name.

Language

This value is used for sort order processing on the server. If left at default, the server is queried for the appropriate language to use.

LockRetries

Enter a number or enter the default number 1.

Specify the number of times to retry a conditional lock. If set to 0, unconditional locking will be used. The order locks are applied, are determined by the order of the SQL statements in the transaction.

Note: Never use unconditional locking when locking multiple databases and/or files, unless all applications apply the locks in the exact same order.

MaxCacheSize

Contact Minisoft support before changing this value.
This can affect the size of the read-ahead buffer used when sequentially fetching records.

MaxWriteSize

Contact Minisoft support before changing this value.
This will affect the size of the network buffers used in communicating with the host. It may be changed if there are network latency or router difficulties.

Schema<n>

<schema file name>,<lockword>

SDFile<n>

<filename>,<lockword>,<type>,<foptions>,<aoptions>

For value type: 0 = Query SD file
1 = Robelle SD file
2 = QUIZ Subfile

Note: Refer to HP documentation for values for foptions and aoptions.

Security Password
SecurityPassword

Enter your security 3000 password.

Server

Enter the server IP Address or Hostname.

Server Port
ServerPort

Enter the server port, default is 30006.

Server Type
ServerType

0 = MPE

1 = UNIX (HPUX, Linux)

2 = Windows™

StandardJoins

Enter YES or NO.
StandardJoins=”YES”, cause inner joins to be done as the SQL standard requires. All new DSNs will have this set to “YES”. If the setting does not exist or is anything other than “YES” (case insensitive) the existing inner join logic will be used.

TraceFileName

Contact Minisoft support before changing this value.
See Tracing Facilities on page 141.

TraceLogFlush

Contact Minisoft support before changing this value.
See Tracing Facilities on page 141.

TraceLogLevel

Contact Minisoft support before changing this value.
See Tracing Facilities on page 141.

TraceSysLevel

Contact Minisoft support before changing this value.
See Tracing Facilities on page 141.

User
user
UID

Enter the HP e3000 user logon.

User Password
UserPassword
password
PWD

Enter the HP e3000 user password.

WarningsAreErrors

Enter YES or NO.
Some clients use SQLSetStmtAttr to set the maximum size for a result set. If this size has been exceeded an error message will be generated unless the registry entry "WarningsAreErrors" for the data source is set to "NO". Reaching the row limit set by the client with SQLSetStmtAttr will no longer cause this error. Reaching the row limit set in MSJOB with the MSQUERYRECLIMIT variable will still trigger the error as intended.

Translation Tables

Translation tables provide a means to substitute characters during ASCII data transfers. These tables are available from Minisoft, Inc. Contact Minisoft to receive the table(s) you need. You may also create you own tables.

To obtain proper conversion of data from the MPE server system to your Windows based software on the PC client, install the tables as directed on your system(s). From the table shown in Table 1, select the conversion table that corresponds to your environment.

Server Character Set	**Client Character Set**	**Host->Driver**	**Driver->Host**
ROMAN8	Windows ANSI	ROM8ANSI.TBL	ANSIROM8.TBL
LATIN2	Windows ANSI	LAT2ANSI.TBL	ANSILAT2.TBL
ISO 7bit DANISH/NORWEGIAN	Windows ANSI	DAN7ANSI.TBL	ANSIDAN7.TBL
ISO 7bit FINNISH/SWEDISH	Windows ANSI	FIN7ANSI.TBL	ANSIFIN7.TBL

Table 1

Note: The above files are the same as those used with other communication products.

Making XLAT Files

The size of these tables is always 256 bytes. The characters in this file are substituted for the equivalent (indexed) character requested. If the file contains the sequential binary values zero through 255, no change would effectively take place.

Example:

If the 33rd character (Decimal value 32, Hex 20) is replaced with the 66th character (Decimal value 65, Hex 41), all blank spaces will be replaced with the letter A.

5. Server Configuration

Common Options

How options are configured

You may configure the servers either through startup parameters or environment variables.

Startup Parameters

Startup parameters are characters or combinations of characters included on the command line or in a startup parameter field. While the format of these parameters is common for all servers, operating system specific details may vary.

- S – Secure Logon
- W - Write
- R – Row ID
- U - Allow CIUPDATE
- H – Host Commands
- C – Catalog Security
- F – Full Read
- T – Tracing
- P – Port Number
- N – Native Library
- V – Version Information
- E – Environment File

Configuring for Secure Logon access (S)

The servers are shipped with the “S” parameter in place. This option prevents access should the logon information be invalid. Do not remove this option without understanding that access will be granted to your data based upon the default user for the connection.

Configuring for Write access (W)

As shipped, the server will allow read-only access to databases and files. If you wish to permit writes and updates to files and databases, you must first modify the startup parameters to include the letter “W”.

For example:

```
SETVAR MSSERVER000004 “30006 0 ODBCSRVR.MM.MINISOFT S”
```

Becomes:

```
SETVAR MSSERVER000004 “30006 0 ODBCSRVR.MM.MINISOFT S W”
```

Configuring for ROWID (R)

This adds a virtual column of integers representing the current row number of the row in the image database. This is a read only value. The value returned may be changed by other applications that access the database at any time. Do not reply on this value to persist.

Configuring for CIUPDATE (U)

When set, all DBOPEN are followed by a DBCONTROL to enable Critical Item Update (CIUPDATE). This will succeed for databases that are configured to “allow” CIUPDATE.

Configuring for Host Commands (H)

Configuring for Catalog Security (C)

In order to use the extra security checking of a catalog file, the ‘C’ parameter must be added to the startup parameters.

A catalog file must then be created with the Catalog Editor and uploaded to the server. By default the server program looks for a file named ODBCCAT in the login group of the user. You can use multiple Catalog files if users are logging into different groups. Optionally, on MPE systems a file equation may be placed in MSJOB before the run line, to specify a single catalog file for all users.

Configuring for Full Read (F)

Enabling FullRead, by placing an “F” in the startup parameters, prevents the server from optimizing certain select SQL statements. Many client applications will issue “select * from table” statements with a series of sequential fetches. The servers can optimize this type of statement without affecting the outcome of the result set. Please contact Minisoft support if you feel this optimization should be disabled on your system.

Configuring for Tracing (T)

For more details, review “Tracing Facilities” on page 141.

Port Number (P)

Native Library (N)

When this is set, the native eloqdb library is used in place of the Image3k library. NATIVE_ELOQ is a synonym for the ‘N’ startup parameter.

Version Information (V)

Environment File (E)

Environment Variables

These values are used for more complex or universally applied configurations. On MPE systems they are set with the SETVAR command in the job or session which holds the listener process. UNIX and Windows servers read the shell variables. UNIX and Windows servers may also reference a port specific file of environment variables for added flexibility.

- MSQUERYRECLIMIT
- MSQUERYCPULIMIT

- MSQUERYCHECKINTERVAL
- MSDECADEDIGITS
- MSTRACESTATEMENT
- TraceSysLevel
- TraceLogLevel
- TraceLogFlush
- TraceFileName
- ROBELLE_SD_TYPE
- NATIVE_ELOQ

MSQUERYRECLIMIT
MSQUERYCPULIMIT

These variables are used to limit queries If you need to limit the time or disc space affected by ODBC, add one or both of the following statements to your MSJOB file (before the RUN command):

```
!SETVAR MSQUERYRECLIMIT 1024
```

```
!SETVAR MSQUERYCPULIMIT 20
```

Maximum records per query set with MSQUERYRECLIMIT is in records <records>. Maximum cpu time per query set with MSQUERYCPULIMIT is in seconds <seconds>. If either limit is reached, your action will be terminated with an error.

MSQUERYCHECKINTERVAL

This will set the minimum time in seconds between network client checks. Used during sort and select operations to verify the client is still attached. The default is 60 seconds.

MSDECADEDIGITS

Text array used to handle those server applications that choose to use ‘A0’ to represent the year 2000.

MSTRACESTATEMENT

Setting this will write the SQL statement being processed by the client to the current log file regardless of the TraceLogLevel.

TraceSysLevel

TraceLogLevel

TraceLogFlush

TraceFileName

ROBELLE_SD_TYPE

NATIVE_ELOQ

When this is set, the native eloqdb library is used in place of the Image3k library. NATIVE_ELOQ is a synonym for the 'N' startup parameter.

MPE Specific

Modifying SETVAR statements

Modify MSJOB by changing the lines as follows:

```
SETVAR MSSERVER000004 "30006 0 ODBCSRVR.MM.MINISOFT S"
```

To enable write / update access, modify the command line to include the "w" (write) option:

```
SETVAR MSSERVER000004 "30006 0 ODBCSRVR.MM.MINISOFT S W"
```

You can now stream MSJOB.MM.MINISOFT, the listener on the HP e3000.

Using Indirect files

Some customers use indirect file references to configure the servers. If you see lines that appear as follows in your MSJOB file, you will need to modify the referenced file.

```
SETVAR MSSERVER000004 "^ODBC0001"
```

In the case shown above, the indirect file is ODBC0001 in the current group.account for the current job. Changing the write access requires changing the line

```
30006 0 ODBCSRVR.EXE.MINISOFT;STDLIST=*LP;PRI=CS S
```

to

```
30006 0 ODBCSRVR.EXE.MINISOFT;STDLIST=*LP;PRI=CS S W
```

UNIX Specific

inetd

6. Schema Editor

Introduction

The Schema Editor is used to create and edit “schemas”. Schema files are saved in .XML format on the local PC and then uploaded to the HP e3000 for use by the end user. Schema files contain descriptions of a variety of files: TurboImage databases (derived from the databases themselves), KSAM files, MPE files, Cognos PDL files; and self-describing files generated by such programs as HP Query, Robelle SUPRTOOL and Powerhouse subfiles. Schemas can be included in a datasource on the Schema tab of the Data Source Configuration dialog box of the ODBC/32 Driver. Schemas can consist of a single TurboImage database or multiple KSAM, MPE, or self-describing files.

Note: Do not confuse schemas created and edited in Minisoft’s Schema Editor with the schema files created with a text editor on the HP e3000 and used by DBSCHEMA.PUB.SYS to create the TurboImage root file. While both schema files contain similar information, they are quite different in practice.

KSAM and MPE files must be described with the Schema Editor to be accessible through ODBC/32. TurboImage databases may be redefined with the Schema Editor to change the way they are presented through ODBC/32. Commonly, you may need to redefine a character or numeric type as SQL_DATE. TurboImage item names with any special character other than the hyphen (-) must be given an alias name to be accessed.

Adding a new item requires you to use the Schema Editor. Some items are composed of a number of sub-elements. By adding an item, you can access any portion of the Record Buffer as any data type.

The Schema Editor is executed by clicking Schema Editor in the Minisoft program group. Its main screen is shown in Figure 14:

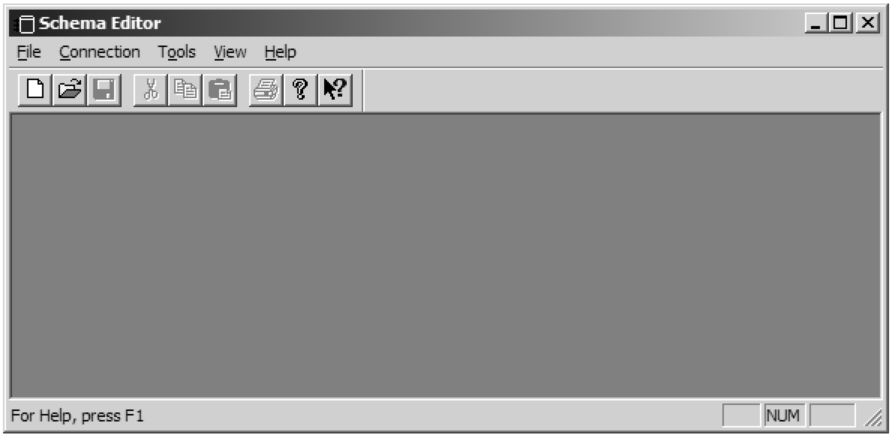


Figure 14

Schema Editor Menu Bar Definitions

File Menu

New -Allows you to create either a blank schema used for KSAM, MPE, or self-describing files or to import the database layout from a TurboImage database. If you choose the later you will be prompted to enter the Database name, Password, and open Mode. Normally the 'Load Automatic Masters' checkbox is not selected. Remember the database password is case sensitive.

Open - Allows you to open an existing schema file already saved on your PC. Schema files are saved in .xml format and by default have an .xml suffix.

Open Remote - Allows you to open a schema file saved on your HP e3000. You are prompted for the name of the schema file and the group and account, and the lockword if there is one.

Import PDL File - This option allows you to import a Powerhouse data dictionary source file. The source file can either reside on the HP e3000 or as a local file on your PC. If it is on the HP e3000, enter the name of the PDL file in the Dictionary field, the group and account, and lockword if there is one. If the file has been downloaded to your PC, check the 'Use local file' box then enter the full path name into the Dictionary field. Processing is faster with a local file but not significantly so.

Connection Menu

Load - Allows you to open a saved configuration file.

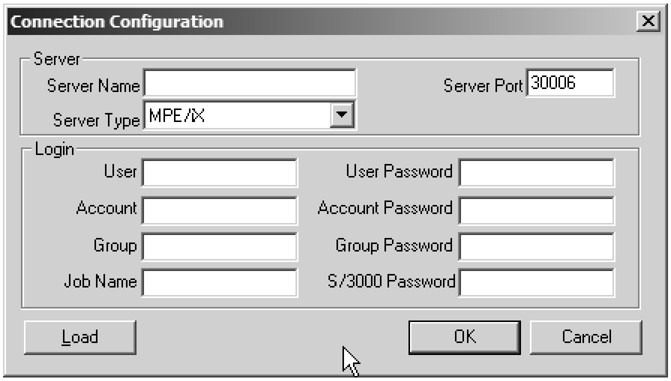
Save - Allows you to save a connection configuration file.

Edit - Allows you to edit a connection configuration file.

Connection to your HP e3000

The Schema Editor stores its “schemas” as .XML files on your local PC and optionally on the HP e3000. Upload your schemas to the HP e3000 before using.

The Schema Editor needs connection configuration information to make connections to the HP e3000. To add the connection configuration information select Connection - Edit from the main menu in the Schema Editor. The Connection Configuration dialog box displays as shown in Figure 15.



Connection Configuration

Server Port 30006

Server Type MPE/iX

Figure 15

1. Enter your HP e3000’s IP address or host name in the Server Name field.
2. The Server Port field default value is 30006 and is typically never changed.
3. Enter valid login information in the following fields.

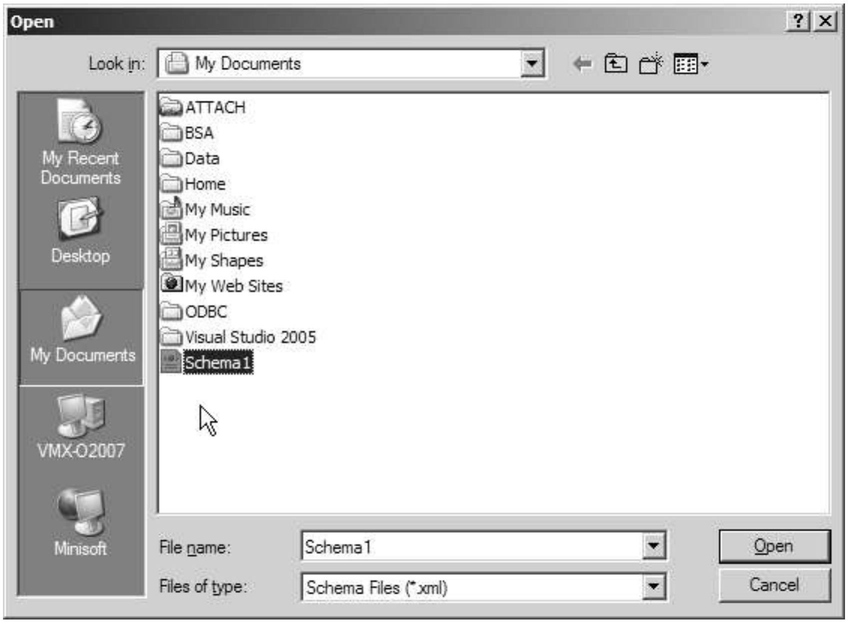
4. Optionally, click the ‘Load’ button to load a saved configuration file.
5. When finished press *OK*, and select *Connection* > *Save* from the Schema Editors main menu to save the connection configuration for later use.

Note: You can automatically load this connection configuration when the Schema Editor starts by including the file name of the connection configuration on the command line for the Schema Editor icon.

Opening a Schema Editor file

File > Open

This menu option allows you to open an existing schema file saved on your PC. Schema files are saved in .xml format and by default have an .xml suffix. Select *File* > *Open*. The Open Schema dialog box appears as shown in Figure 16.



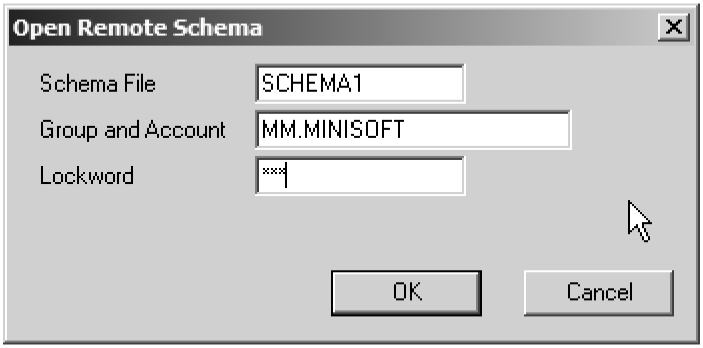
File name: Schema1

Files of type: Schema Files (*.xml)

Figure 16

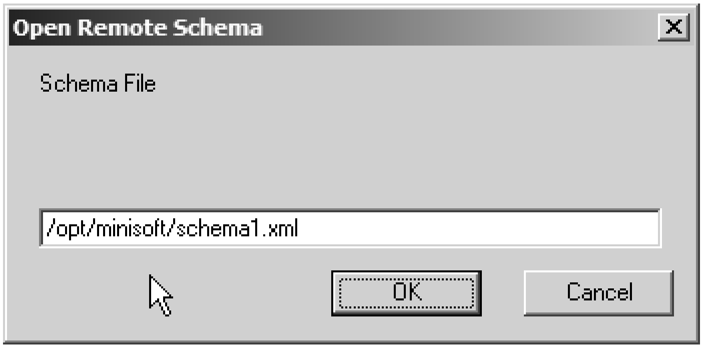
File > Open Remote

With MPE systems, you are prompted for the name of the schema file and the group and account, and the lockword if there is one. See Figure 17. For UNIX and Windows based systems, you will enter the path and file name where the Schema Editor file is located. See Figure 18.



Schema File: SCHEMA1
Group and Account: MM.MINISOFT
Lockword: ***

Figure 17



Schema File: /opt/minisoft/schema1.xml

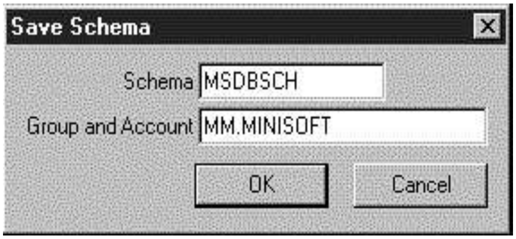
Figure 18

Saving a schema

3. From the file menu select *Save* or *Save As* to save your schema to your PC.

4. To save and then upload your schema file to the HP e3000, select *File > Save and Upload*. When selected, the Save Schema dialog box appears as shown in Figure 5-15.

5. Name the file to store the Schema in by filling in the *Schema* and *Group and Account fields*. Note: You must be logged into the account specified and have write and save access to the group and account in order to save the Schema file to the HP e3000.



Save Schema

Schema MSDBSCH

Group and Account MM.MINISOFT

Schema - Name of the schema file. Only valid HP e3000 names are allowed.

Group and Account - The group and account to save the schema file. For fewer problems, try to save the schema file in the account where data resides.

Lockword - Optional. Enter a lockword to prevent unauthorized access to the schema file.

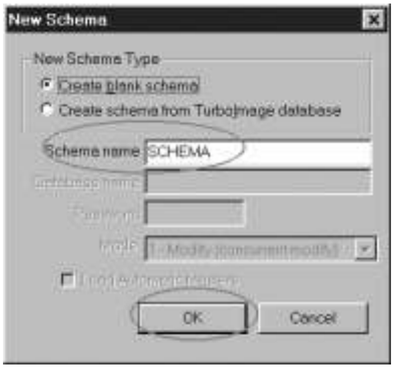
Creating a Schema for KSAM and MPE files

To create a new schema, select *File* > *New*. The New Schema dialog box appears as shown in Figure 5-3.

1. Select one of two choices when creating a New Schema:

- ♦ Create blank schema

2. In the *Schema name* field, enter the name of the file on the HP e3000 that the schema will be stored in. Do not include the group and account.



New Schema

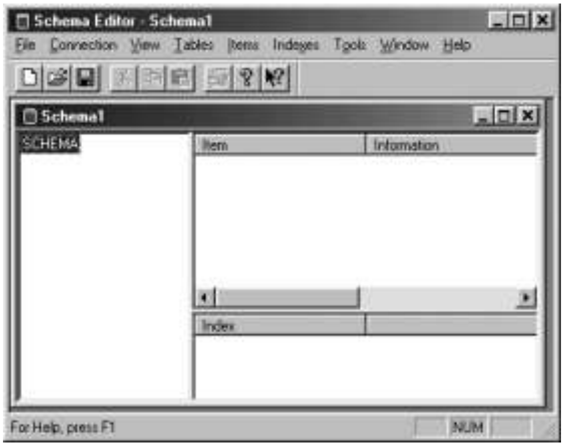
New Schema Type

● Create blank schema

○ Create schema from TurboImage database

Schema name: SCHEMA

3. Once you press *OK*, the schema appears in the main window as shown in Figure 5-4:



Creating a table

To create and add table definitions to your schema, select *Tables* > *Add* from the main menu.

To view and edit table properties, select *View* > *Properties* from the main menu. The “File Table Properties” dialog box will then be shown, see Figure 5-5 and Figure 5-6:

♦ The *Definition* tab shows table name, alias, and type. For KSAM and MPE files all the above fields can be modified. The alias name, if entered, is the name that will be presented through the ODBC/32 driver:

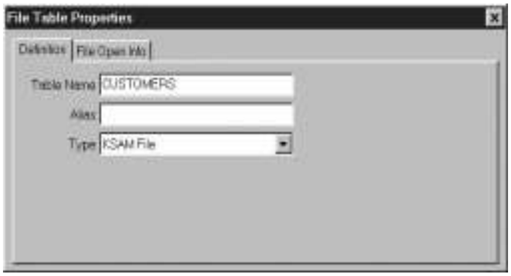
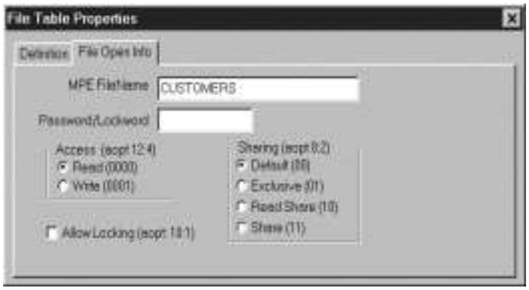


Table Name CUSTOMERS

Alias

Type KSAM File

- The *File Open Info* tab contains the open information for the file. All the following fields maybe changed:



MPE FileName CUSTOMERS

Password/Lockword

Access (aopt 12:4)
● Read (0000)
○ Write (0001)

Sharing (aopt 8:2)
● Default (00)
○ Exclusive (01)
○ Read Share (10)
○ Share (11)

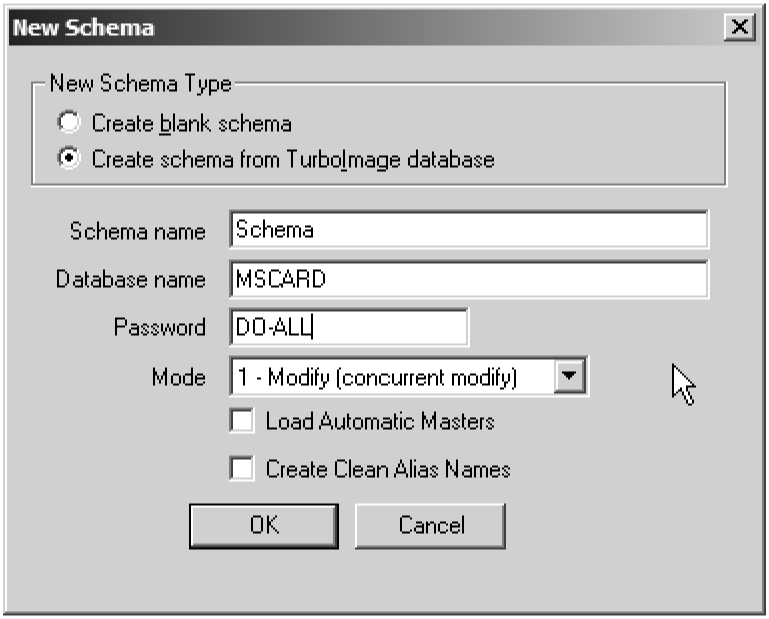
□ Allow Locking (aopt 10:1)

Note: The Table Properties dialog box is modeless, so it will remain on display until you explicitly close it with the close button.

Creating a schema for a TurboImage database

To create a new schema:

1. Select the *File* - *New* menu.
2. The New Schema dialog box appears as shown in Figure 19.
3. Select *Create Schema From TurboImage Database* and enter the database name and password. Then click *OK*.


New Schema

New Schema Type

○ Create blank schema

● Create schema from TurboImage database

Schema name: Schema

Database name: MSCARD

Password: DO-ALL

Mode: 1 - Modify (concurrent modify)

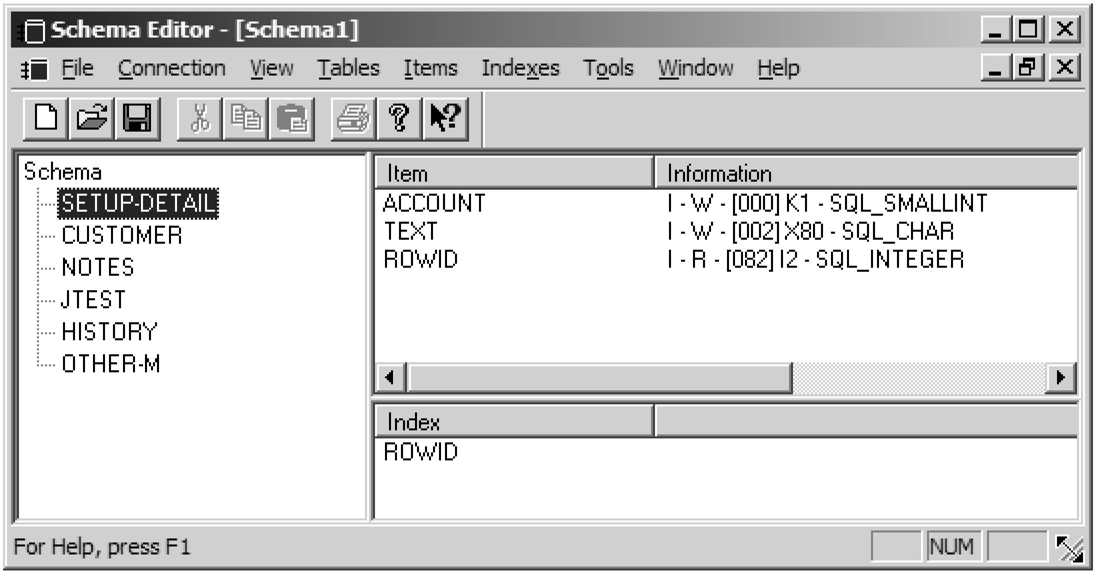
□ Load Automatic Masters

□ Create Clean Alias Names

Figure 19

4. The Connection Configuration dialog box appear as shown in Figure 5-2. Complete the dialog as explained under the heading *Connection to your HP e3000* and press *OK*.

5. A new schema will be initialized with a description of the database as shown in Figure 5-12.



Schema
- SETUP-DETAIL
- CUSTOMER
- NOTES
- JTEST
- HISTORY
- OTHER-M

Item	Information
ACCOUNT	I - W - [000] K1 - SQL_SMALLINT
TEXT	I - W - [002] X80 - SQL_CHAR
ROWID	I - R - [082] I2 - SQL_INTEGER

Index	
ROWID	

For Help, press F1

Figure 20

Database Dataset Properties

The Definition tab shows the Table Name, Alias, Dataset Type, and Lock Item for the Image table. The Alias name, if entered, will be the name presented through the ODBC/32 driver. If a Lock Item is selected, it will be used as the default lock item when doing Item Level Locking but only if it is part of the WHERE clause.

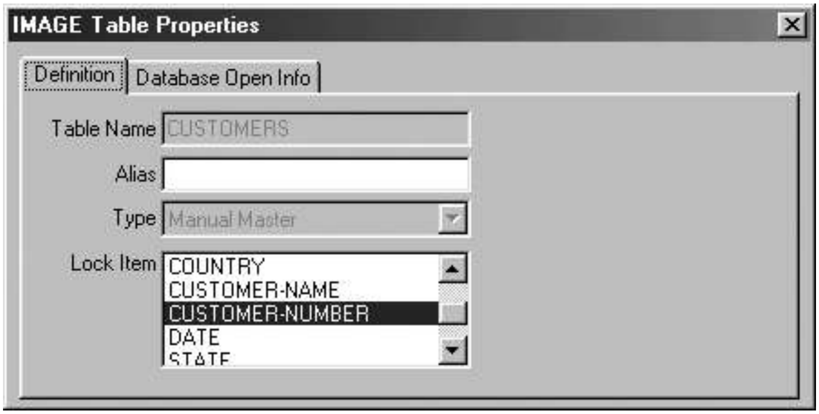


Table Name CUSTOMERS

Alias

Type Manual Master

Lock Item
- COUNTRY
- CUSTOMER-NAME
- CUSTOMER-NUMBER
- DATE
- STATE

The Database Open Info tab contains the Name of the database to open, the database password (which is case sensitive), and the database Open Mode. The most common

Open modes are Mode 1 for write access and Mode 5 for read only access.

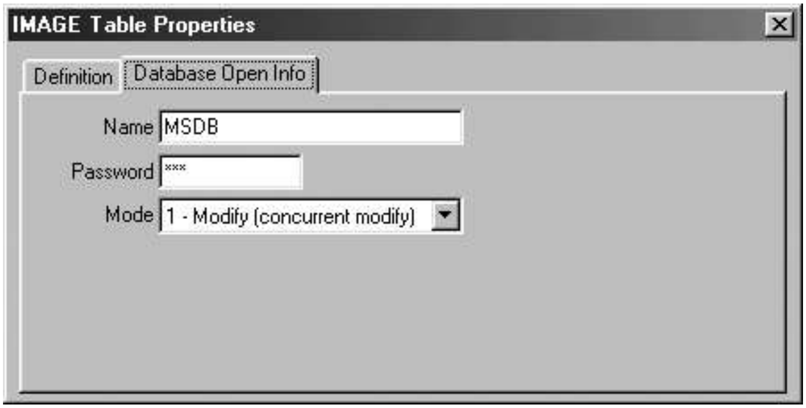


IMAGE Table Properties

Name MSDB

Password ***

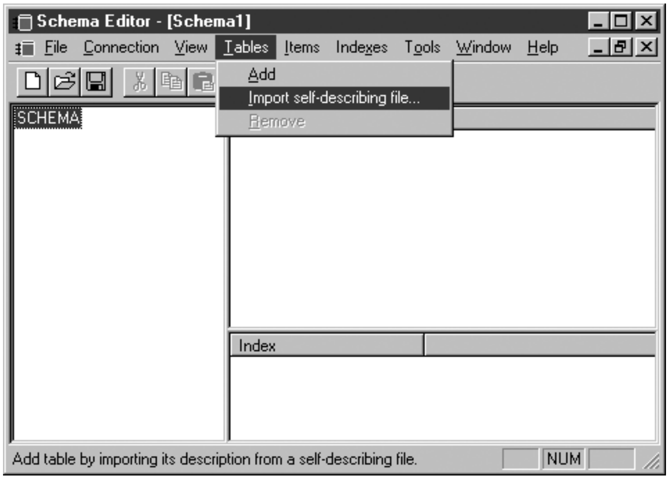
Mode 1 - Modify (concurrent modify)

Note: The Image Table Properties dialog box is modeless so it will remain on display until you explicitly close it. Saving the Schema file saves all changes that have been made.

Creating a schema for Self-describing files

To reference a DSN:

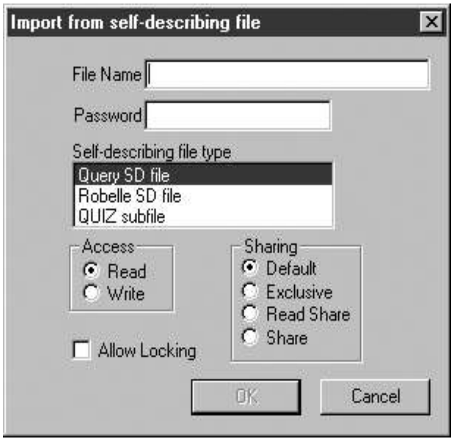
1. Start the Schema Editor and create a new blank schema.
2. From the menu bar select *Table > Import self-describing file* to import and modify the definition. (see Figure 5-13)



Add table by importing its description from a self-describing file.

Figure 5-13

3. The dialog for self-describing files appears (Figure 5-14), allowing you to specify the type of file and how it should be opened.



File Name

Password

Self-describing file type
- Query SD file
- Robelle SD file
- QUIZ subfile

Access
- ● Read
- ○ Write

Sharing
- ● Default
- ○ Exclusive
- ○ Read Share
- ○ Share

□ Allow Locking

Figure 5-14

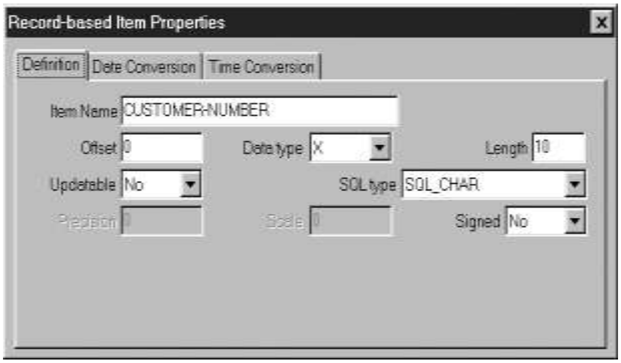
Adding and editing items

Note: Item Names for Record Based or Calculated items are Image syntax. Names with hyphens must be entered with hyphens as they will appear as underscores in your client application.

The expression of Calculated items are in SQL syntax. Names of the existing items must be entered in converted form where hyphens become underscores.

Record based

To add items to a table, select the table from the Schema Editor and select *Item>Add* from the main menu. The Item Properties dialog box will then appear as shown in Figure 5-7. Use the following dialog box to view and edit the item properties:



Item Name CUSTOMER-NUMBER

Offset 0 Data type X Length 10

Updatable No SQL type SQL_CHAR

Precision 0 Scale 0 Signed No

Figure 5-7

Definition tab:

Item Name and *Alias* fields: Name of the record item.

Offset, *Data Type* and *Length* fields: Describes type, size, and item location in the record. Data Type and Length must be valid TurboImage type and length specifications. The size may specify storage that is not an even number of bytes.

Note: The byte size of the item depends on the Data Type and Length according to the rules for TurboImage data items.

SQL Type field: Specifies the data type that is presented through the ODBC/32 driver. See Chapter 5 *Data conversion* for further information on using this field.

Precision field: Specifies the total number of digits contained in a fixed numeric data item. Fixed numeric data items have SQL types of SQL_SMALLINT, SQL_INTEGER, SQL_BIGINT, SQL_NUMERIC, and SQL_PACKED.

Scale field: Specifies how to scale numeric data items of SQL_NUMERIC and SQL_PACKED types. The number represents how many places the decimal point is moved to the left.

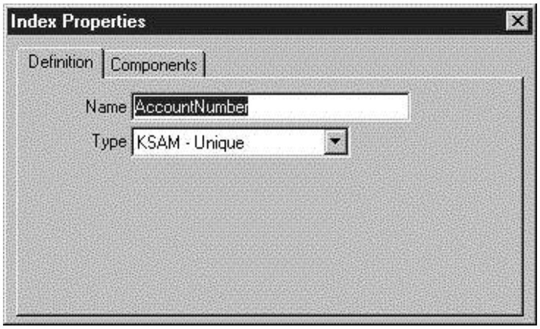
Keep in mind that the Item Properties dialog box is modeless, so it will remain displayed until you explicitly close it, using the close button.

Adding and editing keys (indexes) - KSAM only

To add indexes to a table:

1. Select the table from the main window, and select *Indexes* > *Add* from the main menu.
2. Use the *View* > *Index Properties* menu to view and edit the index properties. Index properties are contained on two tabs on the Index Properties dialog box, as shown in Figure 5-9 and Figure 5-10:

♦ ♦ The *Definition* tab names the index and specifies a type:



Name AccountNumber

Type KSAM - Unique

Figure 5-9

♦ ♦ The *Components* tab specifies which data items are available to be indexed. Select a data item from the *Available Items* listbox and click the *Add* button to add it to the Component Items listbox:

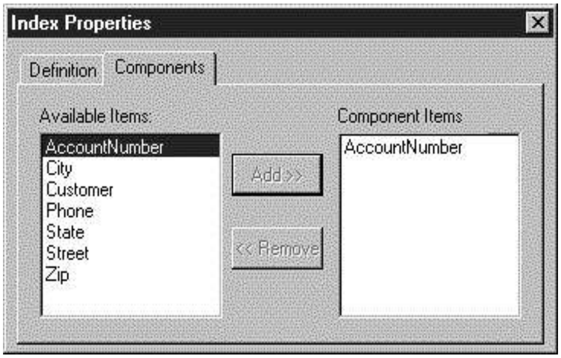


Figure 5-10

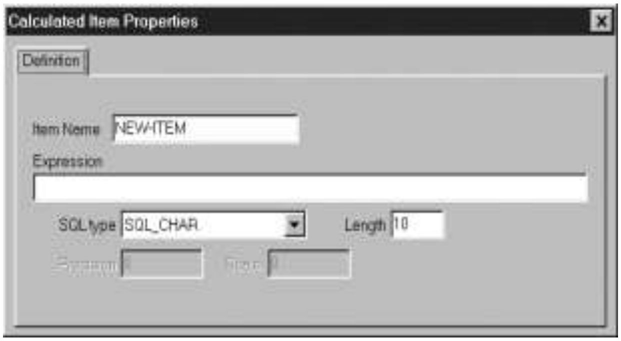
Adding calculated items

To create a calculated item:

1. Select *Add Calculated Item* from the Items menu.
2. Highlight the new item you created and select the *Properties* option from the View menu. The calculated Item Properties dialog box then appears, as shown in Figure 5-8.
3. Enter an appropriate name in the Item Name field.
4. Next, define your item with a valid SQL expression. For example:

   To create a NEW calculated item called TOTAL-DUE that would be the difference of items TOTAL-OWED and TOTAL-PAID, the expression would be “TOTAL_OWED -TOTAL_PAID”.

   Note: Do not use any special characters in the name of the calculated item. The expression must contain valid SQL expressions.



Calculated Item Properties

Definition

Item Name NEWITEM

Expression

SQL type SQL_CHAR

Length 10

Figure 5-8

SQL Examples

Note: For lists of supported functions see Appendix D on *ODBC/SQL Functions*.

Below are function examples using valid SQL expressions as follows:

♦ Enter valid SQL functions:

```
{fn CONCAT(CUSTOMER_NUMBER, CUSTOMER_NAME)}
```

♦ The following function concatenates a name:

```
({fn rtrim(name_first)} + ‵ ‵ + name_last)
```

♦ An 8 character field in the format of yyyymmdd is put into a date format using the substring function:

```
({fn substring(date_opened,5,2)} + ‵/′ +
{fn substring(date_opened,7,2)} + ‵/′ +
{fn substring(date_opened,1,4)})
```

♦ In the following three examples, the field address4 has data stored in the format of ‘City Name, WA 98275’.

♦ To return the city:

```
{fn substring(address4, 1, ({fn locate(‵,′, address4)} - 1))}
```

♦ To return the state:

```
{fn substring(address4, ({fn locate(‵,′, address4)} + 2), 2)}
```

♦ To return the zip code:

```
{fn substring(address4, ({fn locate(‵,′, address4)} + 5),
          ({fn char_length(problem_desc)} - ({fn locate(‘,’,
problem_desc)} + 5)))}
```

- Note: The zip code starting point is determined by locating the ‘,’ then adding 5. How many characters that are returned is determined by subtracting the starting point from the total length of the field.

MPE Data Types

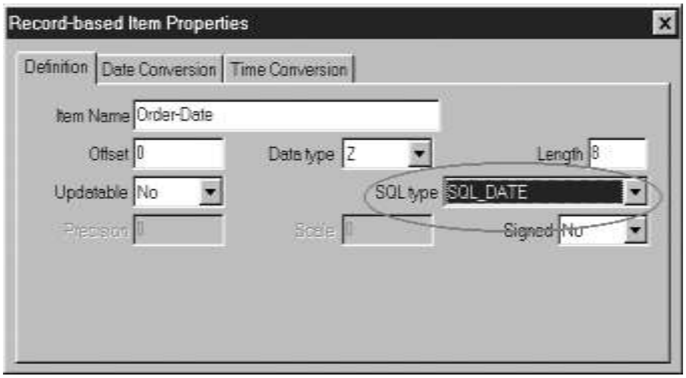
Use the Image Data Type and Length when defining the type used in Image, KSAM, or MPE fields. This reference chart can be used to determine the appropriate type from your existing source code. The bytes reference will guide you in determining the width of each item so that the offset for the nextiem can be found.

Using Dates

When changing the SQL type of a data item from its default to SQL_DATE, the conversion options will be masked based, value based, or both depending on the original Image Data Type.

Note: Your original Image Data Type will not be changed.

1. Select an Item to change its default SQL Type to SQL_DATE.
2. From the Record-based Item Properties dialog box change the SQL type to SQL_DATE as shown in Figure 5-18.



Item Name Order-Date

Offset 0

Data type Z

Length 8

Updatable No

SQL type SQL_DATE

Precision 0

Scale 0

Signed No

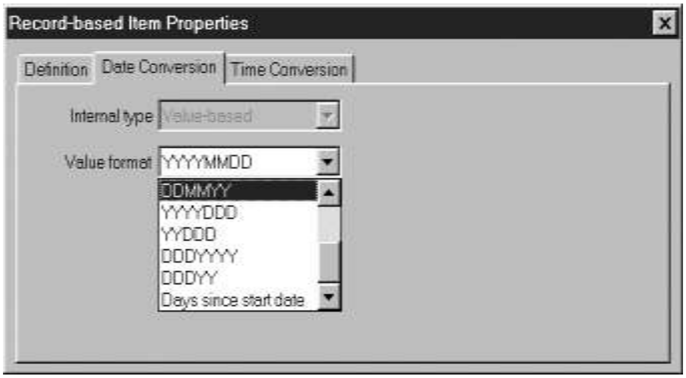
Figure 5-18.

Note: If you do not change the SQL type to SQL_DATE, the Date Conversion tab options will not be available.

3. After selecting SQL_DATE from the SQL type drop down list, click the Date Conversion tab as shown in Figure 5-19.

4. Depending on the original Image Type, you will see the options value-based, mask-based, or both in the *Internal type* field.

   For *value-based* type, select an appropriate value format from the options listed, as shown in Figure 5-19.



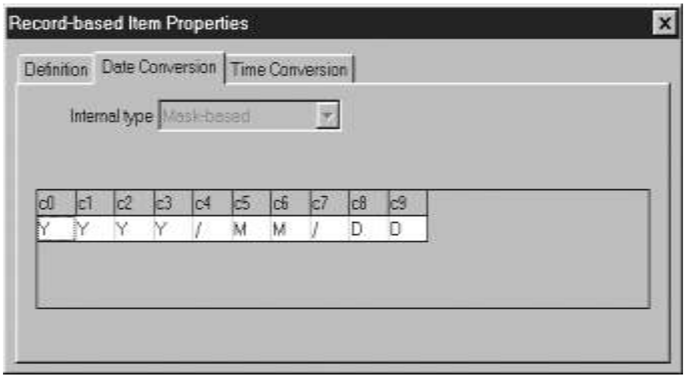
Internal type Value-based

Value format YYYYMMDD

Figure 5-19

5. For mask-based Internal types, enter year (Y), month (M), and day (D) in the appropriate column. For character Image types, you have the option to add

separators such as /, -, etc. for placeholders, as shown in Figure 5-20. This becomes very important if you are updating or adding records.

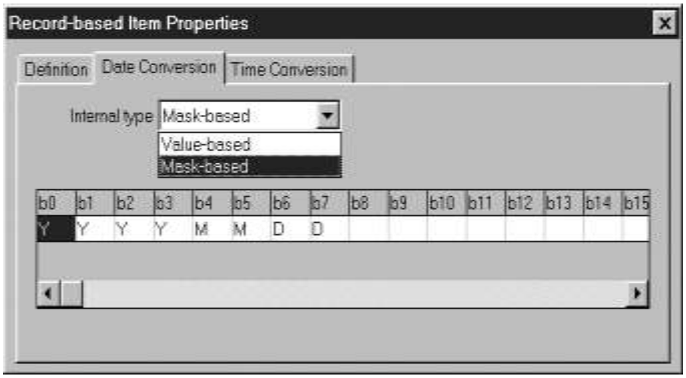

Record-based Item Properties

Internal type: Mask-based

c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
Y	Y	Y	Y	/	M	M	/	D	D

Figure 5-20

6. Integer Image Types have the option of being mask-based or value-based. If you choose mask-based, assign a year (Y), month (M), and day (D) to the appropriate bits as shown in Figure 5-22. If value-based, select the appropriate value format from the options listed.


Record-based Item Properties

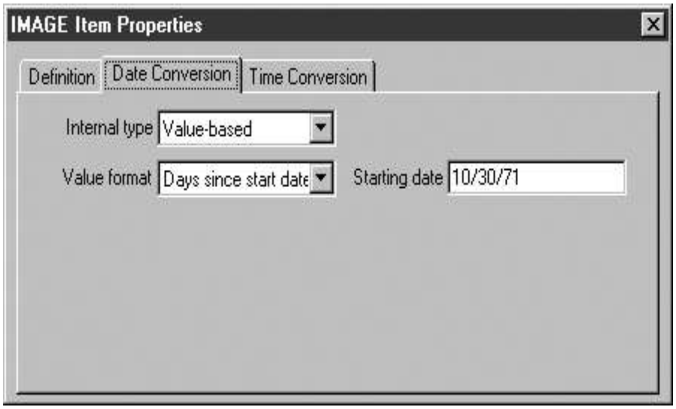
Internal type: Mask-based

b0	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12	b13	b14	b15
Y	Y	Y	Y	M	M	D	D								

Figure 5-22

ManMan Dates

ManMan dates can be used by selecting the appropriate field and setting the Internal type to *Value-based*, Value-format to *Days since start date*, and Starting date to *10/30/71* (See Figure 5-23).



Internal type: Value-based

Value format: Days since start date

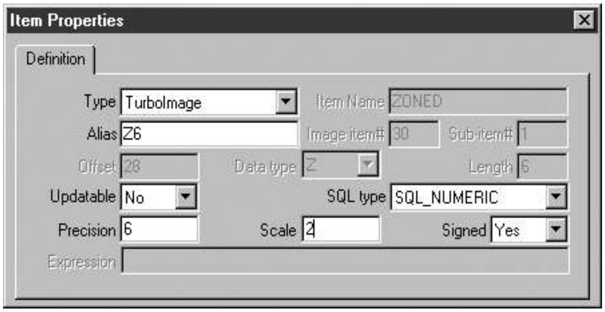
Starting date: 10/30/71

Figure 5-23

Data conversion

To change the way a client application sees a data item, change the *SQL Type* field in the Item Properties dialog box.

A common situation in a database or file is items of type such as J, I, or Z representing money amounts in cents. Typically client applications see an item of this type as an Integer or Long Integer, but the user would like to use it as a number with two decimal places. To change the way a client application sees a data item, change the SQL Type field in the Item Properties dialog box by specifying an SQL type of SQL_NUMERIC and a scale of 2 (as shown in Figure 5-17). The client application will then see the item as a dollar amount.



Type: TurboImage

Item Name: ZONED

Alias: Z6

Image item#: 30

Sub-item#: 1

Offset: 28

Data type: Z

Length: 6

Updatable: No

SQL type: SQL_NUMERIC

Precision: 6

Scale: 2

Signed: Yes

Expression:

Figure 5-17

Importing PowerHouse Definition Language

Minisoft has added support to the Schema Editor to read existing PowerHouse® Dictionary Files. The Schema Editor will allow you to create new or modify existing Schema Editor Files based upon the definitions of the tables and items. To do this:

1. *Optionally*, open an existing Schema to be modified. You can add File and KSAM file definitions from your PDL file to an existing Schema Editor file.
2. Select *Import PDL File* from the File menu, as shown in figure 5-21.

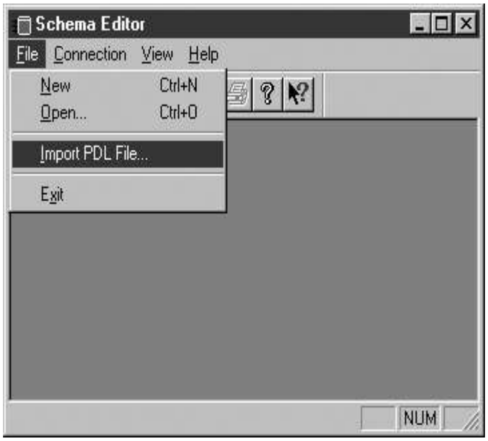
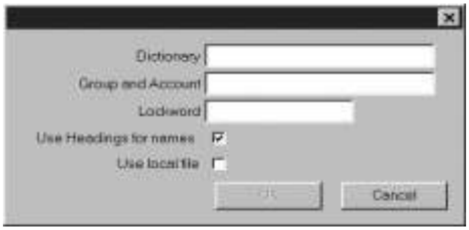


Figure 5-21

3. In Figure 5-22 enter the PDL file name. The file name should be the text source file for the PowerHouse Dictionary. Do not enter the name of the compiled (binary) file.



Dictionary

Group and Account

Lockword

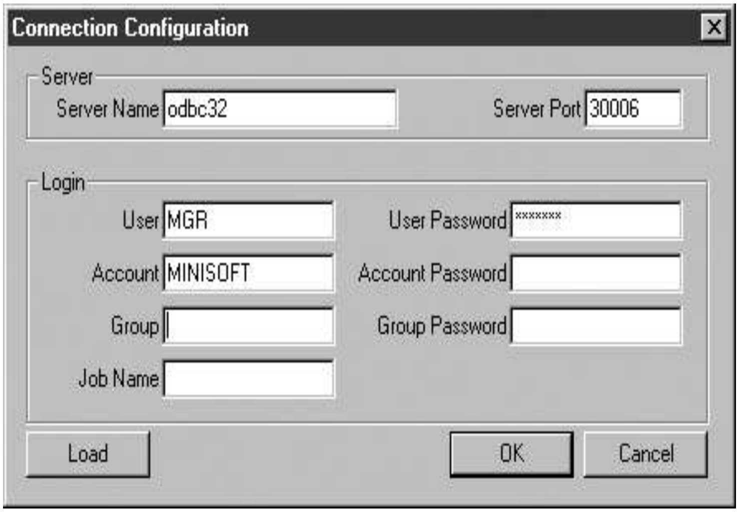
Use Headings for names ☑

Use local file □

OK Cancel

Figure 5-22

4. Enter connection information as shown in Figure 5-22.



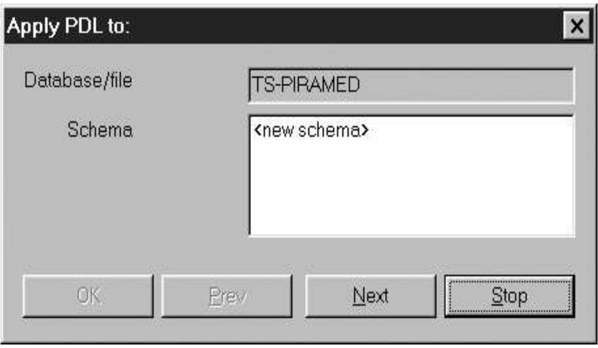
Connection Configuration

Server
Server Name: odbc32
Server Port: 30006

Login
User: MGR
User Password: *******
Account: MINISOFT
Account Password:
Group:
Group Password:
Job Name:

Figure 5-22

5. Review each Database/MPE/KSAM file. Select either **<new schema>** file or an existing (open) Schema Editor file. (See Figure 5-23)



Apply PDL to:

Database/file: TS-PIRAMED
Schema: <new schema>

Figure 5-23

Using QSHOW to generate a new PDL file

The documented file equation for QSHOW is:

```
FILE QSHOGEN;REC=-1276,1,V,ASCII;DISC=50000
```

Change to:

```
FILE QSHOGEN;REC=-80,,F,ASCII;DISC=50000
```

7. Catalog Editor

Introduction to the Catalog Editor

The Catalog Editor is a Windows-based program used to create and maintain a catalog. It is installed with the Administrator Setup program (Admin.exe).

Data security is controlled by the Catalog User name. There are two options for the Catalog User name. The first is to have it based on the MPE/iX user and account, for example MGR.MINISOFT. The user and account are obtained from the Connection tab of the User/System DSN or the ConnectionString information if using a DNS-less connection. The Catalog User name can be wildcarded thus @.MINISOFT, MGR.@ and M@.MINISOFT are all valid Catalog User names.

The second option uses the Windows logon name. To use this option a value of ‘<WINUSER>’ needs to be entered into the Catalog User field on the Security tab when setting up the DSN.

Note: The catalog file is stored as an ASCII file on the HP e3000. You must always use the “Catalog Editor” application to modify this file. Modifications with other tools could result in unpredictable behavior.

The Catalog Editor represents the catalog file in a graphical manner. The *Users* listbox as shown in Figure 7-1, contains a list of MPE/iX and Catalog user names. Groups of users can be created in the *Groups* window. Groups represent one or more users. The rightmost window contains the security list. It lists schemas (or databases), tables, and columns, along with lists of users and/or groups for various types of access.

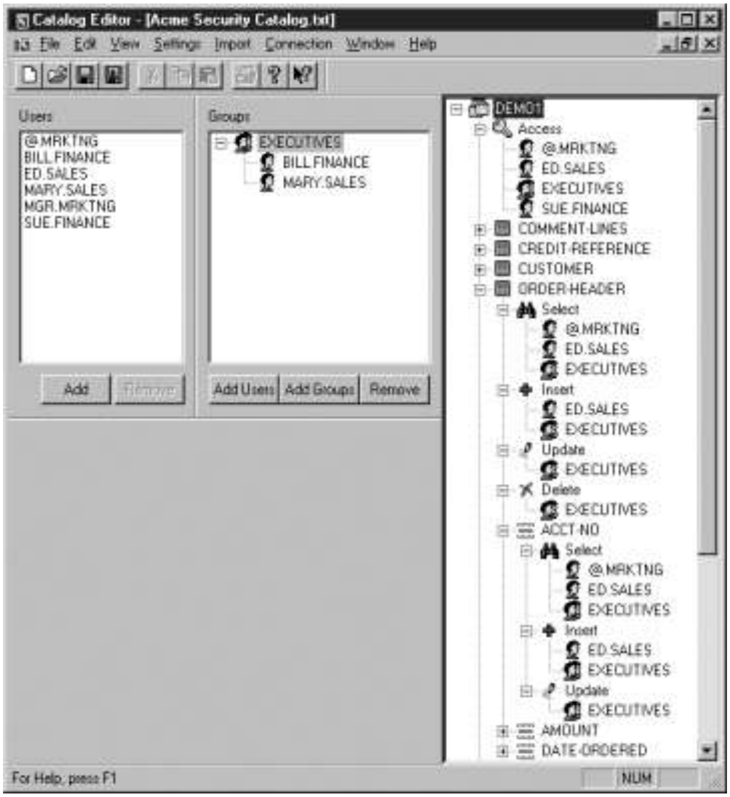


Figure 7-1

Note: It is not necessary to have any users in the *Users* window. These names are not checked by ODBC/32, but are used to drag and drop in the *Groups* window and the *Security List* window. It is also not necessary to have any groups in the *Groups* window. Groups, however, are very useful for creating categories of users and minimizing the changes needed when ODBC/32 users are added or deleted. The term user also refers to a group that the user is a member of. The *Security* list is checked by ODBC/32 when a client application wants to access data. The user must be in one of the data identifier's access list for the user to have any type of access to the data.

Schemas/Databases only have one access list named *Access.* Access lists the names of the users that have access to the named schema (Figure 7-2).

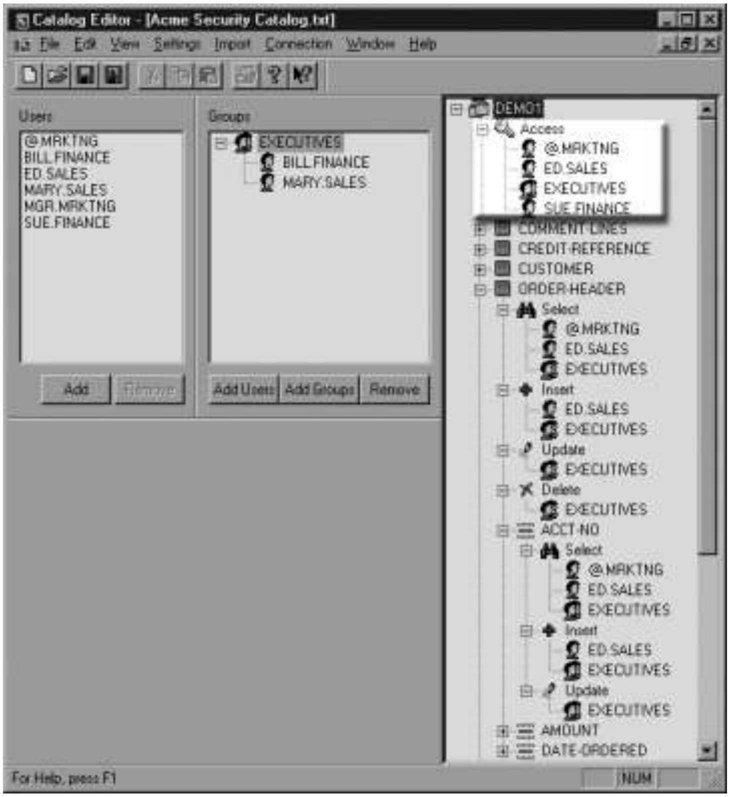


Figure 7-2

Tables have four access lists described as follows (See Figure 7-3):

Select: Users in this list may read records from the table.

Insert: Users in this list may add records to the table.

Update: Users in this list may update records in the table. This access also implies *Select* access.

Delete: Users in this list may delete records from the table.

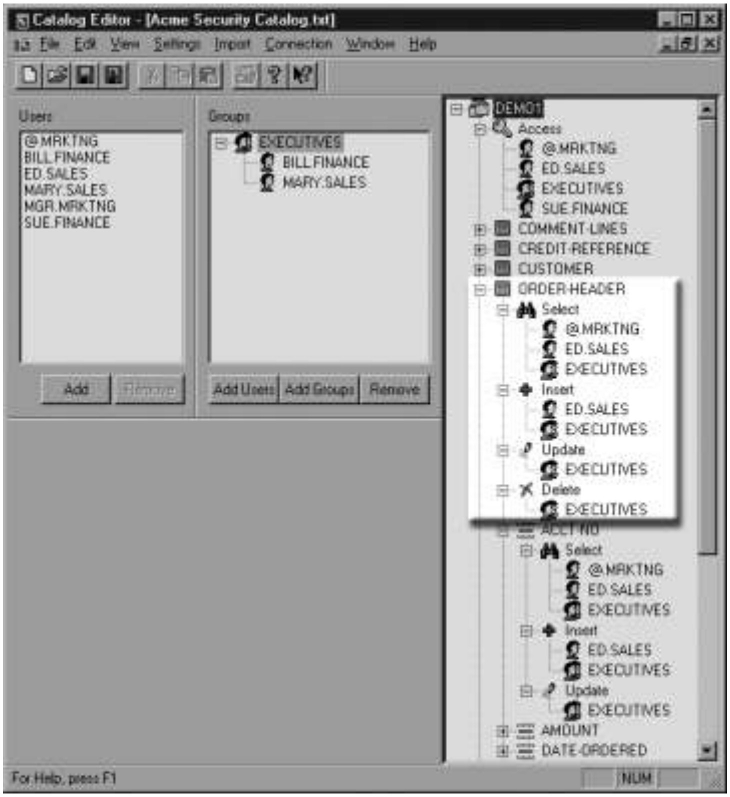


Figure 7-3

Columns have three access lists, described as follows (See Figure 7-4):

Select: Users in this list may read the column.

Insert: Users in this list may provide a value for this column for new records.

Update: Users in this list may update the values of this column. This access also implies *Select* access.

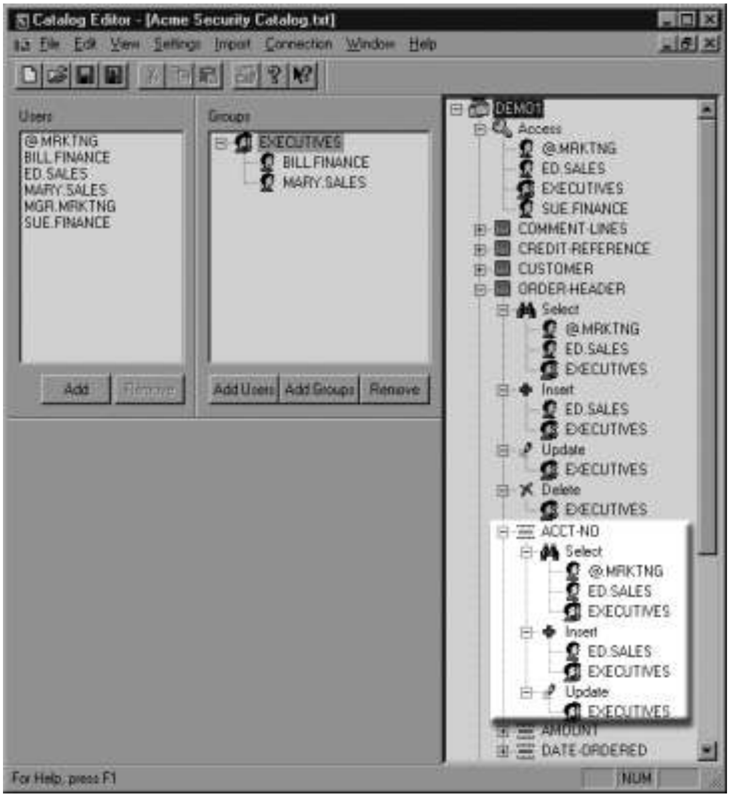


Figure 7-4

It is important to note that a user must be in the appropriate access list of any data identifier it wishes to access. Being in an access list for one data identifier never implies access to another data identifier. For example, to read data in a column, the user must at least be in the *Select* list of the column, the *Select* list of the table the column resides in, and the *Access* list of the schema the table resides in.

Catalog Editor user interface

File menu

New: Creates a new catalog file.

Open: Opens an existing catalog file.

Close: Closes the catalog file and its window.

Save: Saves the catalog to a file.

Save As: Saves the catalog to a different file name.

Save and Upload: Saves the catalog to a file, and then uploads it to the server.

Recent files: Opens the selected catalog file.

Exit: Exits the Catalog Editor.

Edit menu

Add User(s): Adds users to the *Users* window. This can also be accomplished with the Add button in the *Users* window.

Remove User(s): Removes the selected user(s) from the *Users* window. This can also be accomplished with the Remove button in the *Users* window.

Add User(s) to Group: Adds user(s) to the selected group in the *Groups* window. This can also be accomplished with the Add Users button in the *Groups* window.

Add Group(s): Adds group(s) to the *Groups* window. This can also be accomplished with the Add Groups button in the *Groups Window*.

Remove Group: Removes the selected group or users from the *Groups* window. This can also be accomplished with the Remove button in the *Groups* window.

Add Schema(s) to security list: Adds empty schema(s) to the security list.

Add Table(s) to security list: Adds empty table(s) to the security list.

Add Column(s) to security list: Adds empty column(s) to the security list.

Add User(s) to security list: Adds user(s) and/or group(s) to the selected item in the security list. This can also be accomplished by dragging and dropping a user, users, or group on an item in the security list. If the selected item in the security list is a schema, table, or column, a dialog will prompt for a subordinate access list to add the users to. If the selected item in the security list is a schema or table and it is compressed (shown preceded with a '+'), then the users are added to the selected item access list and all of its subordinate items access lists.

Remove selection from security list: Removes the selected item from the security list.

View menu

ToolBar: Controls the display of the toolbar.

Status Bar: Controls the display of the status bar.

Users: Controls the display of the *Users* window.

Groups: Controls the display of the *Groups* window.

Expand all: Expands all the items in the security list.

Settings menu

Recursive Add Types: Controls which access lists are added when adding users to a schema, table, or column in the security list.

Prompt when adding: Controls whether a prompt dialog for the access list is displayed when adding to a schema, table, or column in the security list.

Import menu

From Database: Imports the data-identifier names from the root file of a TurboImage database.

From Schema file: Imports the data-identifier names from a Schema Editor file.

From Self-describing file: Imports the data identifier names from Query Self-describing files, Robelle Self-describing files and Quiz subfiles.

Connection menu

Load: Loads the connection configuration from a file.

Save: Saves the connection configuration to a file.

Edit: Edits the connection configuration.

Catalog Editor command line options

<*catalog file*> *:* File to open when the Catalog Editor is started.

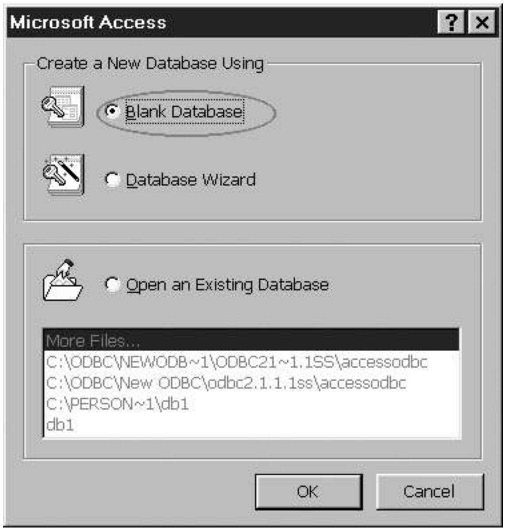
/C<*configuration file*> : File to load the connection configuration when the Catalog Editor is started.

Note: There are no spaces between the ‘/C’ and the configuration file name.

8. Using ODBC/32

MS Access

1. Start Access and click *Blank Database* to create a new blank database, as shown in Figure 8-1:



Microsoft Access

Create a New Database Using

● Blank Database

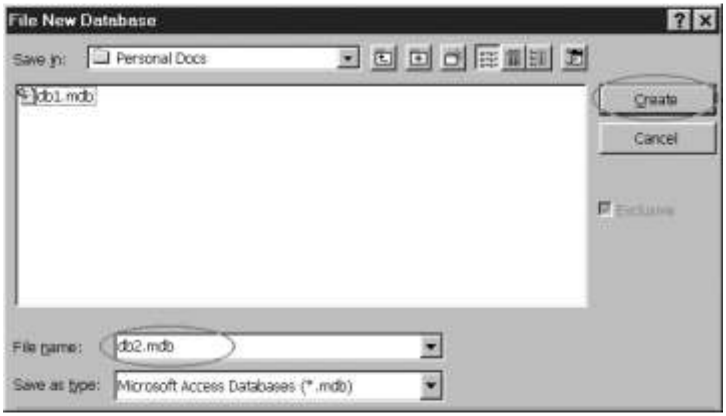
○ Database Wizard

○ Open an Existing Database

More Files...
C:\ODBC\NEWODB~1\ODBC21~1.1SS\accessodbc
C:\ODBC\New ODBC\odbc2.1.1.1ss\accessodbc
C:\PERSON~1\db1
db1

Figure 8-1

2. From the File New Database dialog box (Figure 8-2), enter a name for your database and select *Create*:



Save in: Personal Docs

db1.mdb

File name: db2.mdb

Save as type: Microsoft Access Databases (*.mdb)

Figure 8-2

3. Under the Tables tab select *New* as shown in Figure 8-3:

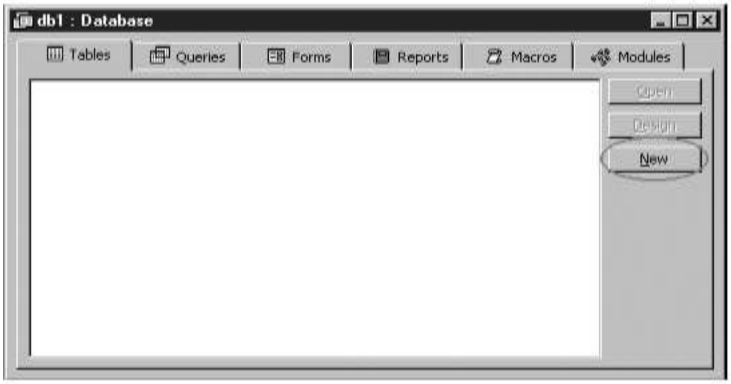
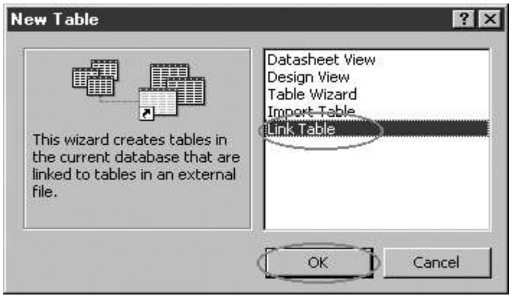


Figure 8-3

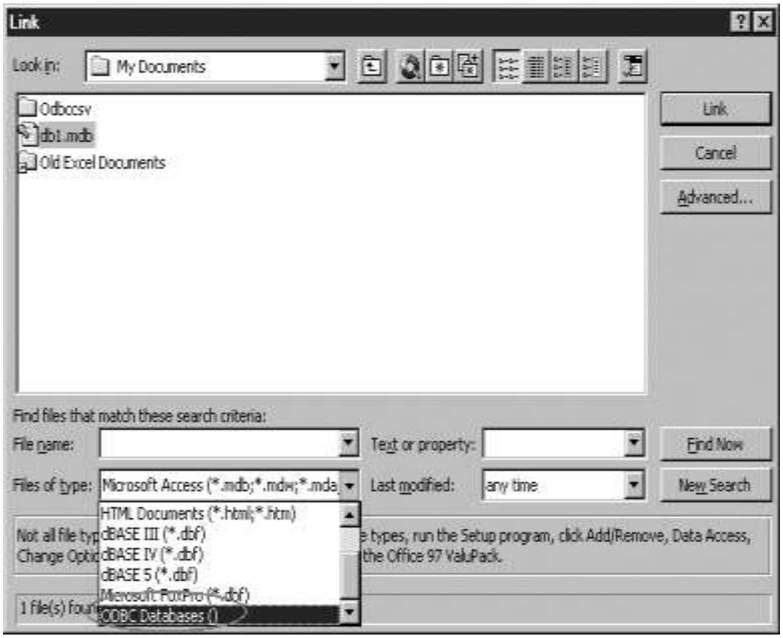
4. In the New Table dialog box (Figure 8-4), select *Link Table* and click *OK*:



This wizard creates tables in the current database that are linked to tables in an external file.

Figure 8-4

5. The Link dialog box will appear as shown in Figure 8-5. Pull down the *Files of Type* list box and select *ODBC Databases()*:



Look in: My Documents

File name:

Text or property:

Files of type: Microsoft Access (*.mdb;*.mdw;*.mda

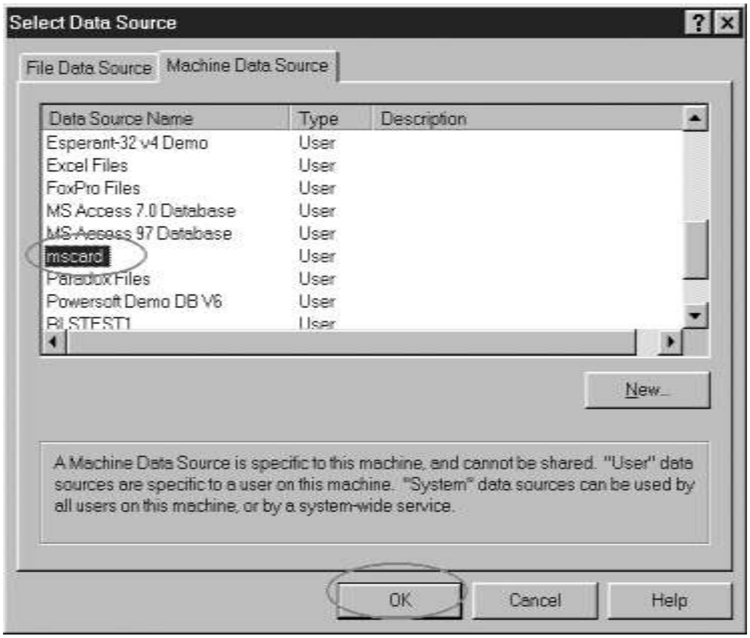
Last modified: any time

Not all file typ e types, run the Setup program, click Add/Remove, Data Access, Change Optio the Office 97 ValuPack.

1 file(s) foun

Figure 8-5

6. Once you select *ODBC Databases()* the Select Data Source dialog box appears (Figure 8-6), switch to the Machine Data Source tab. Select the Data Source Name previously created in the ODBC Administrator, then press *OK*:

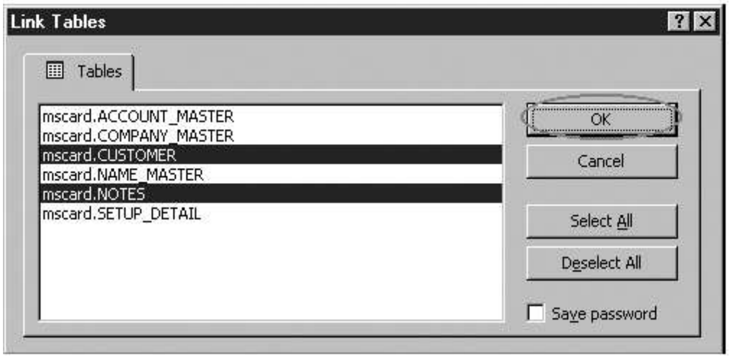


Data Source Name	Type	Description
Esperant-32 v4 Demo	User	
Excel Files	User	
FoxPro Files	User	
MS Access 7.0 Database	User	
MS Access 97 Database	User	
mscard	User	
Paradox Files	User	
Powersoft Demo DB V6	User	
RLSTEST1	User	

A Machine Data Source is specific to this machine, and cannot be shared. "User" data sources are specific to a user on this machine. "System" data sources can be used by all users on this machine, or by a system-wide service.

Figure 8-6

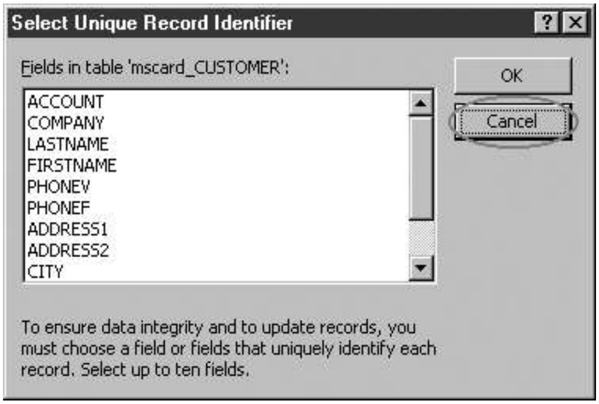
7. In the Link Tables dialog box (Figure 8-7) you now have the option of which Image datasets to include in this link. After you have chosen the necessary datasets press *OK*:



mscard.ACCOUNT_MASTER
mscard.COMPANY_MASTER
mscard.CUSTOMER
mscard.NAME_MASTER
mscard.NOTES
mscard.SETUP_DETAIL

Figure 8-7

8. If you see the Select Unique Record Identifier dialog box (Figure 8-8), press *Cancel* (selecting a field would cause delays in your queries):



Select Unique Record Identifier

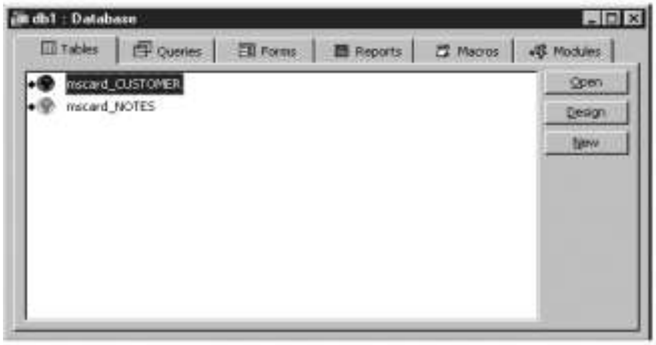
Fields in table 'mscard_CUSTOMER':

ACCOUNT
COMPANY
LASTNAME
FIRSTNAME
PHONEV
PHONEF
ADDRESS1
ADDRESS2
CITY

To ensure data integrity and to update records, you must choose a field or fields that uniquely identify each record. Select up to ten fields.

Figure 8-8

9. Your Database - Tables display should now include the datasets you selected earlier (Figure 8-9):



db1 : Database

mscard_CUSTOMER
mscard_NOTES

Figure 8-9

10. Now create your queries and reports that use the data currently stored in your Image Database.

    Note: If you select Open or double click on the dataset listed in the Tables display, MS Access will attempt to copy all of the data from the dataset to your PC. The intermediate file size is currently limited and may not store all the dataset. If you receive a warning about the temp file size, please use a query to select a subset of your data.

Using MSAccess to change or add data with Minisoft's 32-bit ODBC driver

Using MSAccess

1. Create an updateable DSN in the 32bit ODBC Administrator as explained in Chapter 3.
2. Use an Open Mode that allows modification of the database.
3. Use a Password that grants write access.
4. Follow the instructions to create links in MSAccess.

When you see the *Select Unique Record Identifier* dialog box, you MUST select a field or combination of fields that contain data that makes each record unique in the dataset.

You can now update, add, or delete information in your Image Database.

Notes:

1. If you select *Open* or double click on the dataset listed in the Table, MSAccess will attempt to copy all data from the dataset to your PC. The intermediate file size is currently limited and may not store all datasets. If you receive a warning about the temp file size, please use a query to select a subset of your data.
2. Your changes in datasheet mode take effect when you leave the current record or when you Save Record (shift + enter).
3. MSAccess allows one level of Undo.

Using MSAccess to insert an address from an Image database with ODBC/32

The following is an example of using Minisoft's ODBC/32 driver to insert an address from an Image database:

```
Sub InsertAddress()

ActiveDocument.Save

Dim CustomerNr, SQLString, SQLString1

CustomerNr = InputBox("Enter Customer Number", "Insert Address")

If CustomerNr <> "" Then
```

```
Set AdrDoc = Documents.Add

SQLString = “SELECT CUSTOMERS.CUSTOMER_NAME, “&_
”CUSTOMERS.ADDRESS1, CUSTOMERS.ADDRESS2,” & _
”CUSTOMERS.CITY, CUSTOMERS.STATE, CUSTOMERS.COUNTRY “

SQLString1 = “FROM TESTSAV3.CUSTOMERS CUSTOMERS “&_

“WHERE (CUSTOMERS.CUSTOMER_NUMBER=’” & CustomerNr & “‘)”

AdrDoc.Range.InsertDatabase Format:=0, Style:=0, _

LinkToSource:=False, _

Connection:=”DSN=MSDB”, _

SQLStatement:=SQLString, SQLStatement1:=SQLString1, _

PasswordDocument:=””, PasswordTemplate:=””, _
WritePasswordDocument:=””, WritePasswordTemplate:=””, _
DataSource:=””, From:=-1, To:=-1, _
IncludeFields:=False

Set Table1 = AdrDoc.Tables(1)

ReDim TabCells(Table1.Range.Cells.Count)
i = 1
For Each TabCell In Table1.Range.Cells
Set CellRange = TabCell.Range
CellRange.MoveEnd Unit:=wdCharacter, Count:=-1
TabCells(i) = CellRange.Text
i = i + 1
Next TabCell

AdrDoc.Close (wdDoNotSaveChanges)

Insert CUSTOMER_NAME
Selection.TypeParagraph
Selection.MoveUp Unit:=wdLine, Count:=1
Selection.Range.InsertBefore TabCells(1)
Selection.MoveDown Unit:=wdLine, Count:=1

Insert ADDRESS1
Selection.TypeParagraph
Selection.MoveUp Unit:=wdLine, Count:=1
Selection.Range.InsertBefore TabCells(2)
Selection.MoveDown Unit:=wdLine, Count:=1
```

```
Insert ADDRESS2
If TabCells(3) <> "" Then
Selection.TypeParagraph
Selection.MoveUp Unit:=wdLine, Count:=1
Selection.Range.InsertBefore TabCells(3)
Selection.MoveDown Unit:=wdLine, Count:=1
End If

Insert CITY
Selection.TypeParagraph
Selection.MoveUp Unit:=wdLine, Count:=1
Selection.Range.InsertBefore TabCells(4)
Selection.MoveDown Unit:=wdLine, Count:=1

Insert STATE
Selection.TypeParagraph
Selection.MoveUp Unit:=wdLine, Count:=1
Selection.Range.InsertBefore TabCells(5)
Selection.MoveDown Unit:=wdLine, Count:=1

Insert COUNTRY
Selection.TypeParagraph
Selection.MoveUp Unit:=wdLine, Count:=1
Selection.Range.InsertBefore TabCells(6)
Selection.MoveDown Unit:=wdLine, Count:=1

End If

End Sub
```

Sample Visual Basic Applications Source Code

The following sample was created and used in Microsoft Access

```
Option Compare Database
Option Explicit
Dim Db As Database
Dim Cust As Recordset

Sub LoadFields()

On Error GoTo Err_LoadFields
```

```
Text1.Value = “”
Text2.Value = “”
Text3.Value = “”
Text4.Value = “”

Text1.Value = Cust(0).Value
Text2.Value = Cust(1).Value
Text3.Value = Cust(“LAST_NAME”).Value
Text4.Value = Cust(“FIRST_NAME”).Value

Exit_LoadFields:
Exit Sub

Err_LoadFields:
If Err.Number = 3021 Then
Cust.MoveFirst
Resume
End If
MsgBox Err.Description
MsgBox Err.Number
Resume Exit_LoadFields

End Sub

Private Sub Form_Open(Cancel As Integer)

Dim Connect As String
Dim SQL As String

On Error GoTo Err_Form_Open

Connect$ = “ODBC;DSN=MSDB;”
Set Db = OpenDatabase(“”, dbDriverNoPrompt, True, Connect$)

SQL = “SELECT * FROM CONTACTS “
Set Cust = Db.OpenRecordset(SQL, dbOpenDynaset)

Cust.MoveFirst
LoadFields
Exit Sub

Err_Form_Open:
MsgBox Err.Description, vbCritical
```

```
End

End Sub

Private Sub Command1_Click()
On Error GoTo Err_Command1_Click

Cust.MoveNext
If Cust.EOF Then
MsgBox “EOF”
End If
LoadFields

Exit_Command1_Click:
Exit Sub

Err_Command1_Click:
MsgBox Err.Description
Resume Exit_Command1_Click

End Sub

Private Sub Command2_Click()
On Error GoTo Err_Command2_Click


Cust.MovePrevious
If Cust.BOF Then
MsgBox “BOF”
End If
LoadFields

Exit_Command2_Click:
Exit Sub

Err_Command2_Click:
MsgBox Err.Description
Resume Exit_Command2_Click

End Sub

Private Sub Command3_Click()
```

```
Dim SQL As String

On Error GoTo Err_Command3_Click

If (Len(Text2.Value) < 1) Then
SQL = “SELECT * FROM CONTACTS”
Else
SQL = “SELECT * FROM CONTACTS WHERE CONTACT_NUMBER = ‘“ & _
Format(Text2.Value, “000000”) & “‘“
End If

Set Cust = Db.OpenRecordset(SQL, dbOpenDynaset)
MsgBox Cust.RecordCount
LoadFields

Exit_Command3_Click:
Exit Sub

Err_Command3_Click:
MsgBox Err.Description
Resume Exit_Command3_Click

End Sub
```

Using SQL Server

The NT environment and SQL Server 7 offer a powerful set of tools for developing new client/server and Web based applications. Most new applications, however, are not completely isolated from existing applications. They must use and generate data that is compatible with existing applications, which in some cases resides on an HP e3000.

SQL Server Linked Servers

SQL Server has support for linked servers. Link servers represent connections to other databases. The databases can be non-SQL Server databases and can reside on any server reachable through the network. The connection is accomplished through an OLE DB provider. An OLE DB provider is a DLL that provides access to data from the supported database in a common way.

There are OLE DB providers for a number of databases, including Oracle and JET. The most interesting OLE DB provider, however, is the one for access to data through ODBC. Using this OLE DB provider, any database accessible through ODBC can be accessed as a SQL Server linked server. Once a database is configured as a linked server any SQL Server application can use the linked server's tables just like any other table in the SQL Server database. The following diagram illustrates how a linked server is used to access a TurboIMAGE database on an HP e3000.

SQL Server linked servers are especially useful for new client/server applications that will use SQL Server as the main database, but need some access to legacy data on an HP e3000. The application could make use of two connections, one to the SQL Server database and another through ODBC to the TurboIMAGE database. The main disadvantages of this scenario are:

1. The connection information for both databases would have to be configured on each client machine.
2. Each client machine would have to have an ODBC driver that can access HP e3000 database(s) installed on it. Additionally, an ODBC datasource for access to the HP e3000 database(s) would have to be duplicated on each client.
3. Each client machine would need network access to the HP e3000.

Using a linked server in the SQL Server database to access the HP e3000 database(s) resolves these problems as follows:

1. The connection information for the HP e3000 database(s) is part of the machine. The application only needs to know how to connect to the SQL Server machine.
2. The ODBC driver that accesses HP e3000 database(s) only needs to be installed on the SQL Server machine. The datasource for accessing the HP e3000 database(s) would be kept and maintained on the SQL Server machine.
3. Each client only needs network access to the SQL Server machine. The SQL Server machine is the only one that needs network access to the HP e3000.

Sample Application

Anwar, Inc. is a provider of telephony equipment. Each new customer gets assigned to a team at Anwar, Inc. which handles all of the customer's needs. A typical team would consist of personnel from the sales, technical, and administrative departments at Anwar, Inc. At the present time, it is a time consuming and confusing task to introduce all the members of the team to the customer. Management at Anwar, Inc. would like IT to develop a Web site for the customer to use that would contain all of the team members, their pictures, and their contact information.

At this point the personnel system and Anwar, Inc., which is on the HP e3000, has all of the employees with their contact information, but does not contain their pictures. Also the basic customer information is on the HP e3000, but team assignments for each customer are kept manually.

The proposal is to create a SQL Server database to store the team assignments and personnel pictures and use linked servers in the SQL Server database to access personnel contact information and customer information from the HP e3000. Since the application will be web-based, the NT machine that has the SQL server database, will also be a web server.

Two tables will go in the SQL Server database *CustomerAccess*. They are

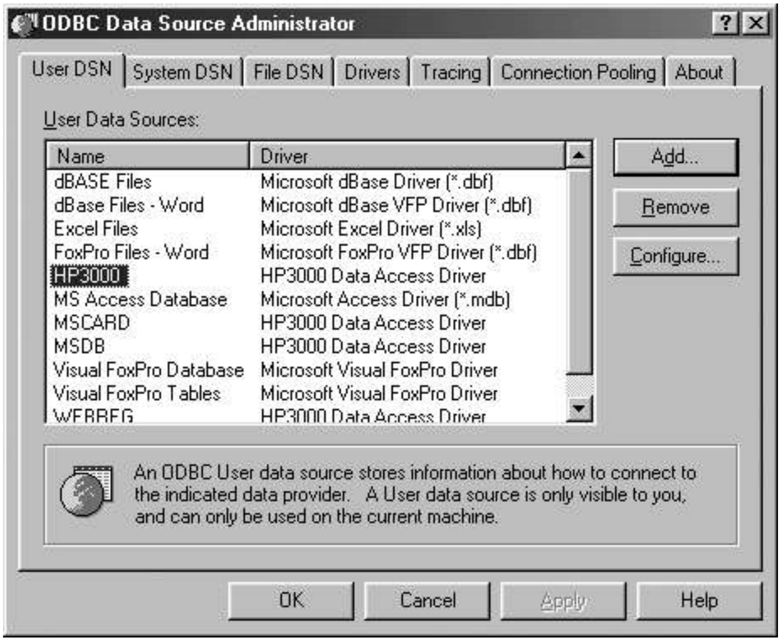
The tables on the HP e3000 that will be accessed are

In order to access the HP e3000 from the SQL Server machine, an ODBC datasource called HP e3000 will be setup. The configuration of the datasource will vary depending on the ODBC driver used. The following section shows the configuration in ODBC/32 with ODBC/32 Administrator.

Creating the ODBC Data Source

1. From the *ODBC DataSource Administrator* (Figure 21) create a New User

DSN called *HP3000*. If you need help creating the DSN refer to Chapter 3 on *Configuring a Data Source with ODBC/32 Administrator*.



User Data Sources:

Name	Driver
dBASE Files	Microsoft dBase Driver (*.dbf)
dBase Files - Word	Microsoft dBase VFP Driver (*.dbf)
Excel Files	Microsoft Excel Driver (*.xls)
FoxPro Files - Word	Microsoft FoxPro VFP Driver (*.dbf)
HP3000	HP3000 Data Access Driver
MS Access Database	Microsoft Access Driver (*.mdb)
MSCARD	HP3000 Data Access Driver
MSDB	HP3000 Data Access Driver
Visual FoxPro Database	Microsoft Visual FoxPro Driver
Visual FoxPro Tables	Microsoft Visual FoxPro Driver
WEBREG	HP3000 Data Access Driver

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

Figure 21

2. Once the DSN for accessing the HP e3000 database(s) is configured, the SQL Server Enterprise Manager can be used to configure a linked server.

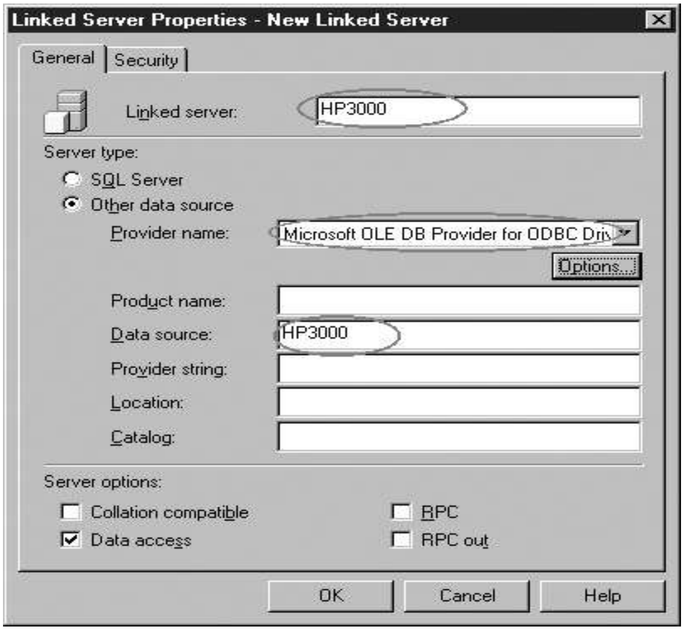
Configuring a linked server

Using SQL Server Enterprise Manager:

1. Expand the *Security* tree and right mouse click on the *Linked Servers* label. Select *New Linked Server* and configure the linked server as shown in Figure 8-11.

2. As shown in Figure 8-11, the linked server is given a name, OLE DB provider is selected, and the datasource for the OLE DB provider is named. In this case the OLE DB provider is *Microsoft OLE DB Provider for ODBC Drivers* and the datasource is *HP3000*.

   Note: If needed, there are additional options for security that may be configured in Figure 22.



Linked server: HP3000

Server type:
○ SQL Server
● Other data source

Provider name: Microsoft OLE DB Provider for ODBC Driv

Product name:

Data source: HP3000

Provider string:

Location:

Catalog:

Server options:
□ Collation compatible
☑ Data access
□ RPC
□ RPC out

Figure 22

3. At this time the linked server is ready to be accessed.

Creating sample application using the development tools

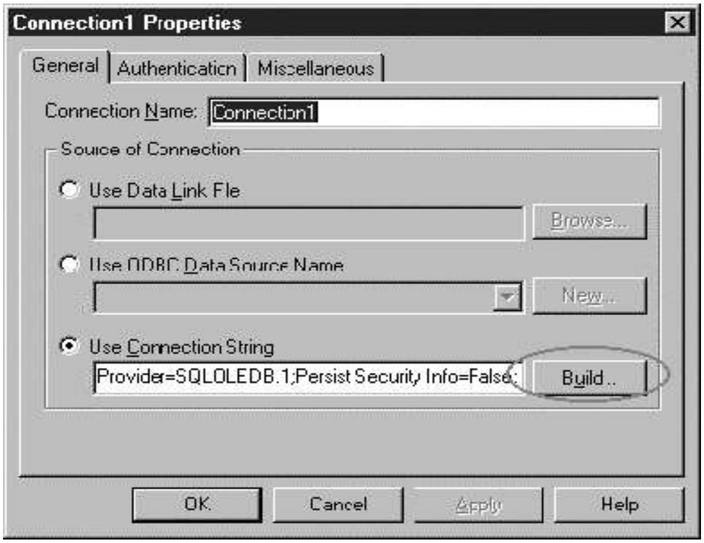
Any development tool that can access SQL Server can use linked servers. This would include Visual Basic, PowerBuilder, Delphi, FrontPage and many others. Linked servers also are very useful in developing queries that retrieve data from many different databases on many different machines. There are new applications available on the NT platform that will use SQL Server. Using linked servers is a powerful way to give these new applications access to legacy data.

In the following example we will use Visual InterDev to set up an Active Server Page to access data through a SQL Server.

After a new Active Server Page and Data Connection have been created in a new project, the configuration for the Data Connection is as follows:

1. In the *Connection1 Properties* dialog box (Figure 23), select Build under the

General tab to build your connection string.

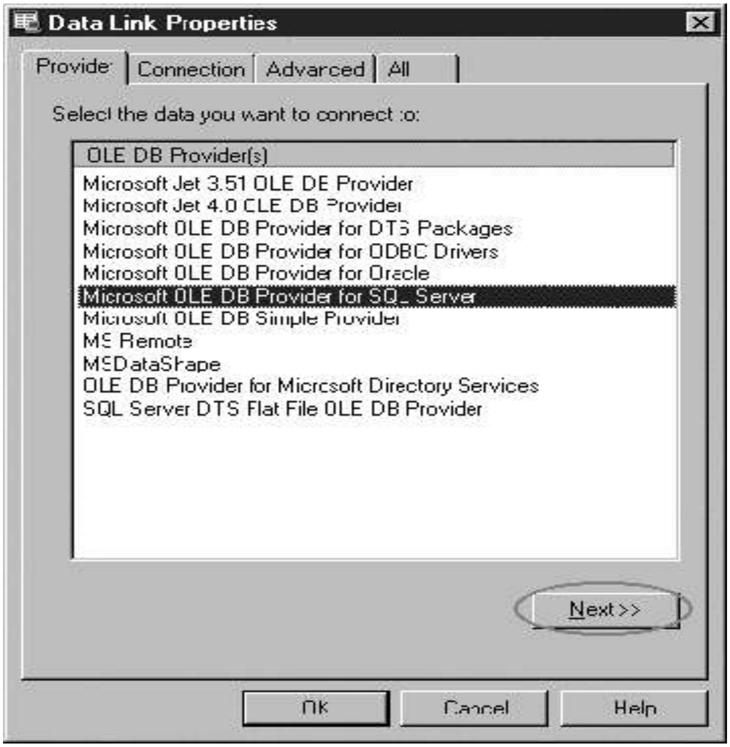

Connection Name: Connection1

● Use Connection String

Provider=SQLOLEDB.1;Persist Security Info=False;

Figure 23

2. Under the *Provider* tab select *Microsoft OLE DB Provider for SQL Server* and select *Next*, as shown in Figure 24:



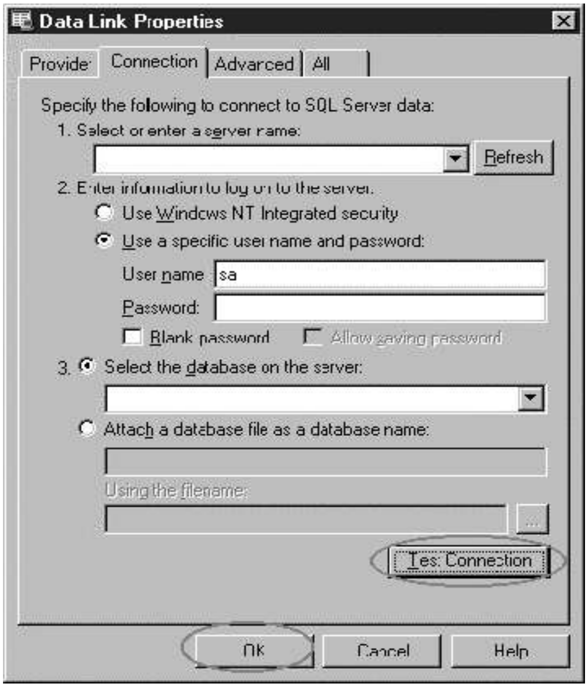
Select the data you want to connect to:

OLE DB Provider(s)

Microsoft Jet 3.51 OLE DB Provider
Microsoft Jet 4.0 OLE DB Provider
Microsoft OLE DB Provider for DTS Packages
Microsoft OLE DB Provider for ODBC Drivers
Microsoft OLE DB Provider for Oracle
Microsoft OLE DB Provider for SQL Server
Microsoft OLE DB Simple Provider
MS Remote
MSDataShape
OLE DB Provider for Microsoft Directory Services
SQL Server DTS Flat File OLE DB Provider

Figure 24

3. Select the *Connection* tab as shown in Figure 25 and enter your valid user login information and select *Test Connection*. If connection proves successful, select *OK*.



Specify the following to connect to SQL Server data:

1. Select or enter a server name:

2. Enter information to log on to the server:

○ Use Windows NT Integrated security

● Use a specific user name and password:

User name: sa

Password:

□ Blank password □ Allow saving password

3. ● Select the database on the server:

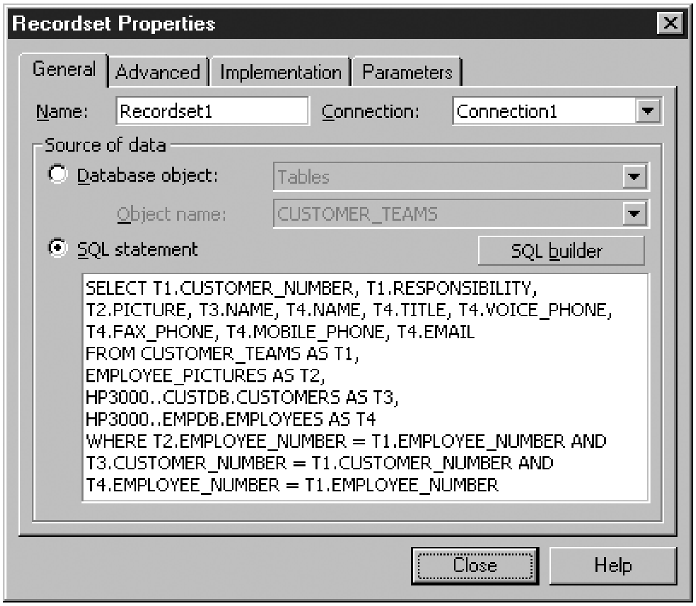
○ Attach a database file as a database name:

Using the filename:

Figure 25

4. Selecting *OK* from the *Data Link Properties* dialog box will bring you back to the *Connection1 Properties* dialog box (Figure 8-12), select *OK*.

5. After the data connection is created and tested a RecordSet control is dropped on the Active Server Page and configured. This configuration will control how the data is retrieved from the databases. In this case four data tables will need to be joined by an SQL SELECT statement, to execute a distributed query against the SQL Server (See Figure 26).



Name: Recordset1

Connection: Connection1

○ Database object: Tables

Object name: CUSTOMER_TEAMS

● SQL statement

```
SELECT T1.CUSTOMER_NUMBER, T1.RESPONSIBILITY,
T2.PICTURE, T3.NAME, T4.NAME, T4.TITLE, T4.VOICE_PHONE,
T4.FAX_PHONE, T4.MOBILE_PHONE, T4.EMAIL
FROM CUSTOMER_TEAMS AS T1,
EMPLOYEE_PICTURES AS T2,
HP3000..CUSTDB.CUSTOMERS AS T3,
HP3000..EMPDB.EMPLOYEES AS T4
WHERE T2.EMPLOYEE_NUMBER = T1.EMPLOYEE_NUMBER AND
T3.CUSTOMER_NUMBER = T1.CUSTOMER_NUMBER AND
T4.EMPLOYEE_NUMBER = T1.EMPLOYEE_NUMBER
```

Figure 26

The data access of the Active Server Page is complete. Additional work would have to be done to format the page, and add controls to display the data. Most likely, an additional start-up page, querying the user for their customer number and password would be added, this would be linked to the Active Server Page when the customer was validated.

MS Excel

1. When Microsoft Excel opens, click *Data* > *Get External Data* > *Create New Query* as shown in Figure 27.

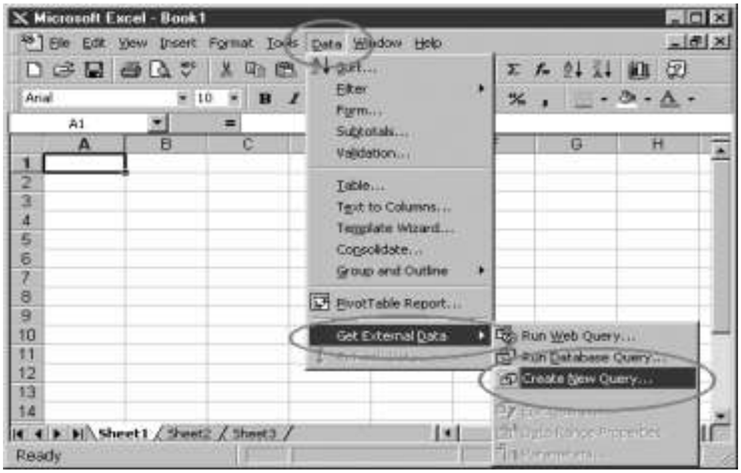
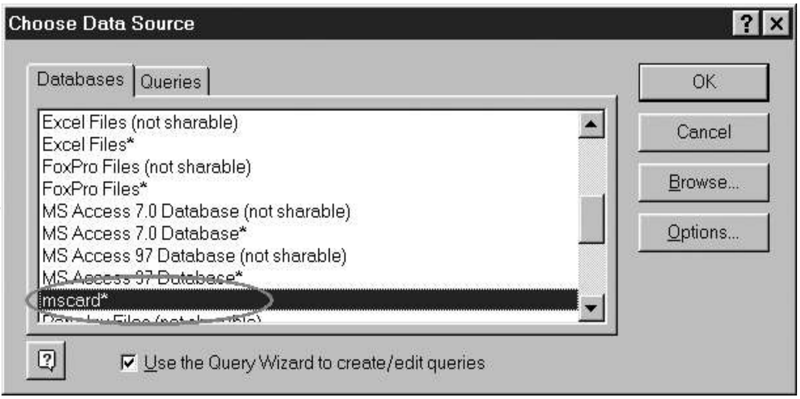


Figure 27

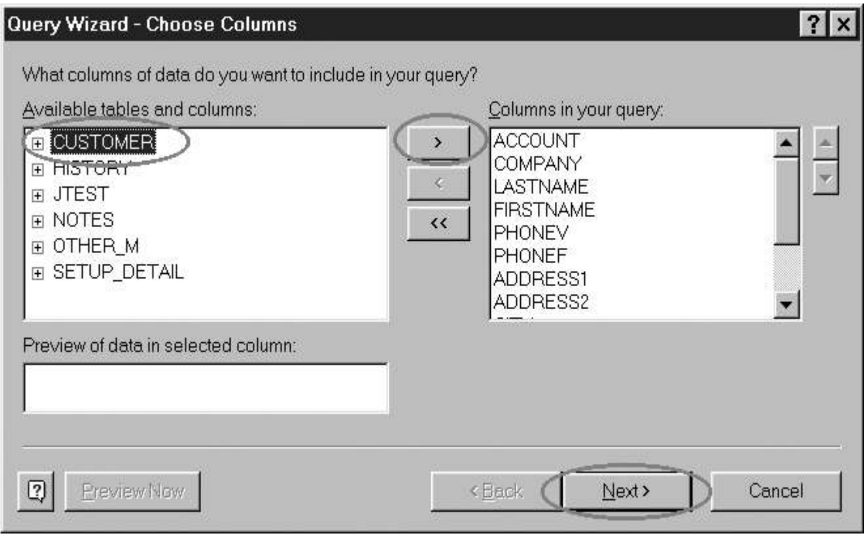
2. The Choose Data Source dialog box will appear (Figure 9-2). Click *mscard** from the Databases tab, then click *OK*.

Figure 9-2



☑ Use the Query Wizard to create/edit queries

3. From the dialog box shown in Figure 9-3 choose available tables and columns to add to your query, click the right arrow to add, then click *Next*.



Query Wizard - Choose Columns

What columns of data do you want to include in your query?

Available tables and columns:

- CUSTOMER
- HISTORY
- JTEST
- NOTES
- OTHER_M
- SETUP_DETAIL

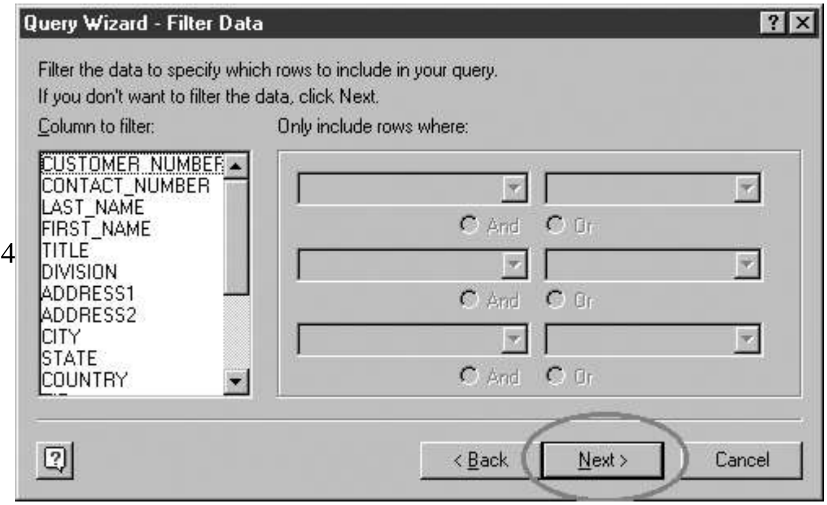
Columns in your query:

- ACCOUNT
- COMPANY
- LASTNAME
- FIRSTNAME
- PHONEV
- PHONEF
- ADDRESS1
- ADDRESS2

Preview of data in selected column:

Figure 9-3

4. As shown in Figure 9-4, click *Next*.



Query Wizard - Filter Data

Filter the data to specify which rows to include in your query.
If you don't want to filter the data, click Next.

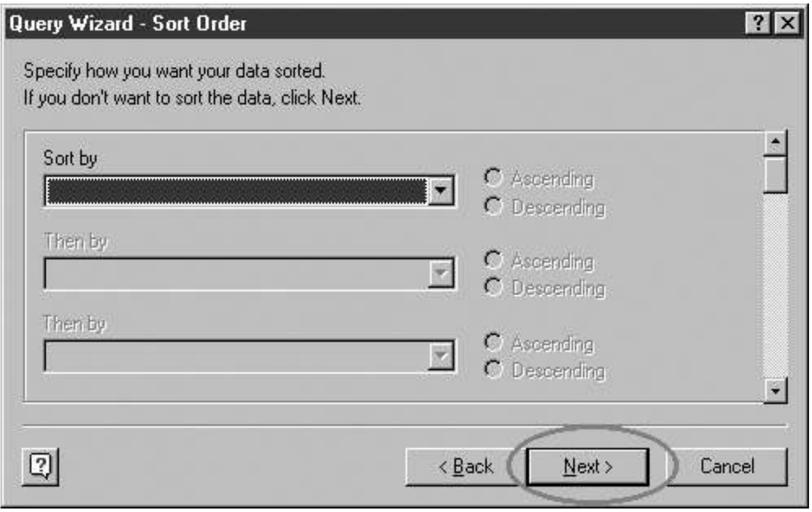
Column to filter:

- CUSTOMER_NUMBER
- CONTACT_NUMBER
- LAST_NAME
- FIRST_NAME
- TITLE
- DIVISION
- ADDRESS1
- ADDRESS2
- CITY
- STATE
- COUNTRY

Only include rows where:

Figure 9-4

5. As shown in Figure 9-5, click *Next*.



Query Wizard - Sort Order

Specify how you want your data sorted.
If you don't want to sort the data, click Next.

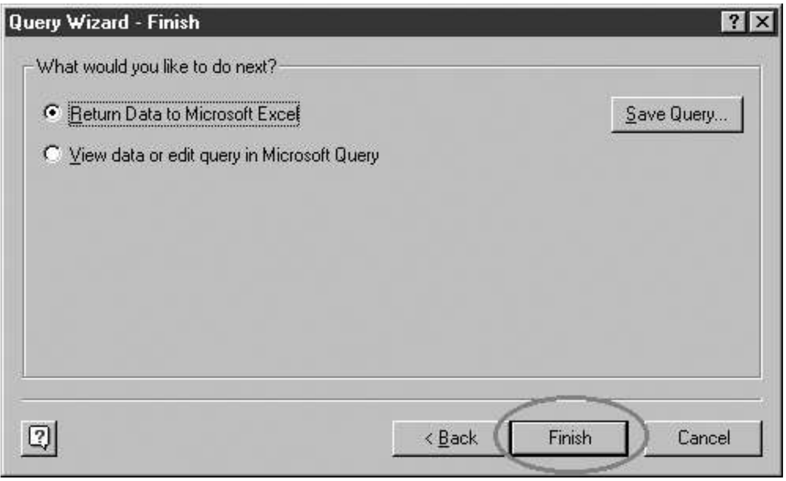
Sort by
○ Ascending
○ Descending

Then by
○ Ascending
○ Descending

Then by
○ Ascending
○ Descending

Figure 9-5

6. Under What Would You Like To Do Next, click Return Data to Microsoft Excel, then click Finish as shown in Figure 9-6.



Query Wizard - Finish

What would you like to do next?

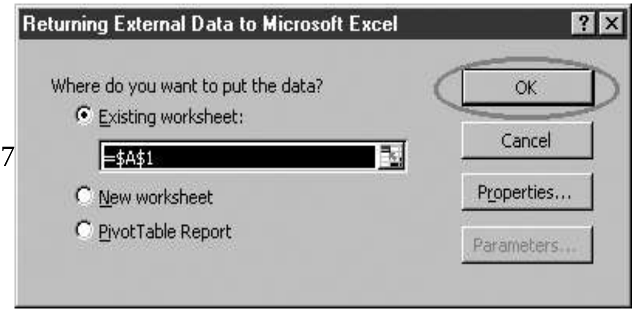
● Return Data to Microsoft Excel

○ View data or edit query in Microsoft Query

Figure 9-6

7. Under *Where Do You Want To Put The Data*, click *Existing Worksheet*, and then click *OK* as shown in Figure 9-7.

Figure 9-7



Returning External Data to Microsoft Excel

Where do you want to put the data?

● Existing worksheet:

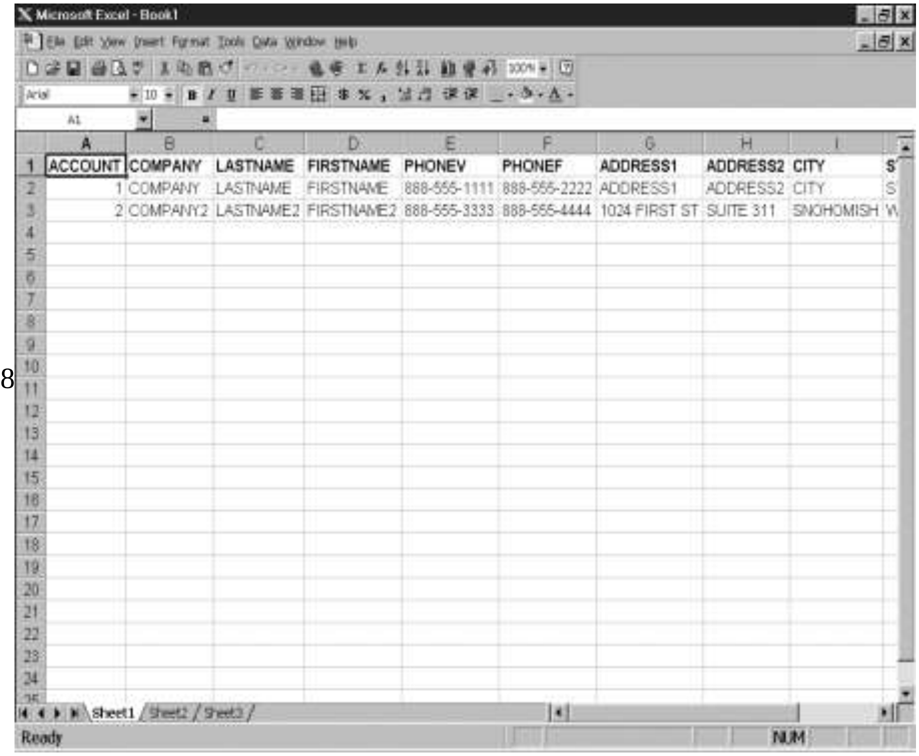
=A1

○ New worksheet

○ PivotTable Report

8. Microsoft Excel - Notebook will appear with your query information, as shown in Figure 9-8.

Figure 9-8

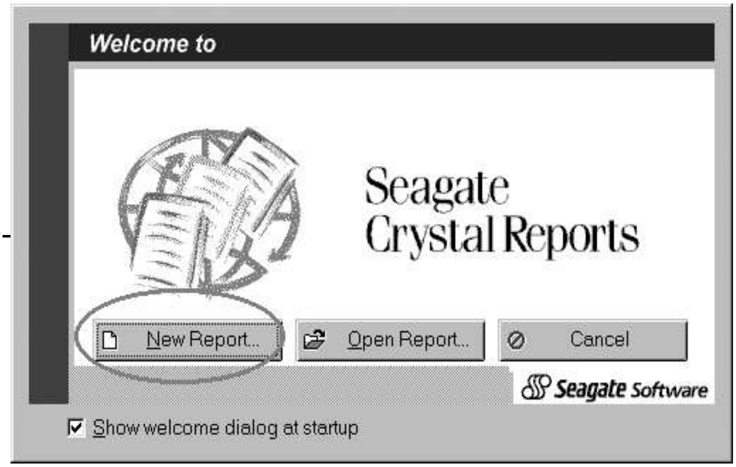


A	B	C	D	E	F	G	H	I
ACCOUNT	COMPANY	LASTNAME	FIRSTNAME	PHONEV	PHONEF	ADDRESS1	ADDRESS2	CITY
1	COMPANY	LASTNAME	FIRSTNAME	888-555-1111	888-555-2222	ADDRESS1	ADDRESS2	CITY
2	COMPANY2	LASTNAME2	FIRSTNAME2	888-555-3333	888-555-4444	1024 FIRST ST	SUITE 311	SNOHOMISH

Seagate's Crystal Reports

1. Select *New Report*, as shown in Figure 9-9:

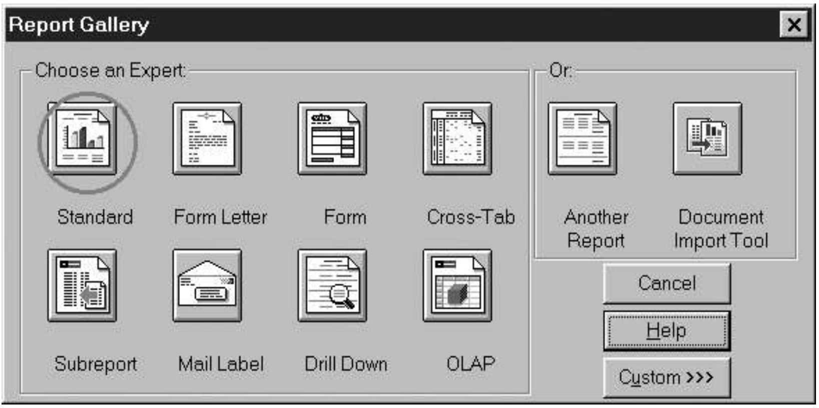
Figure 9-



☑ Show welcome dialog at startup

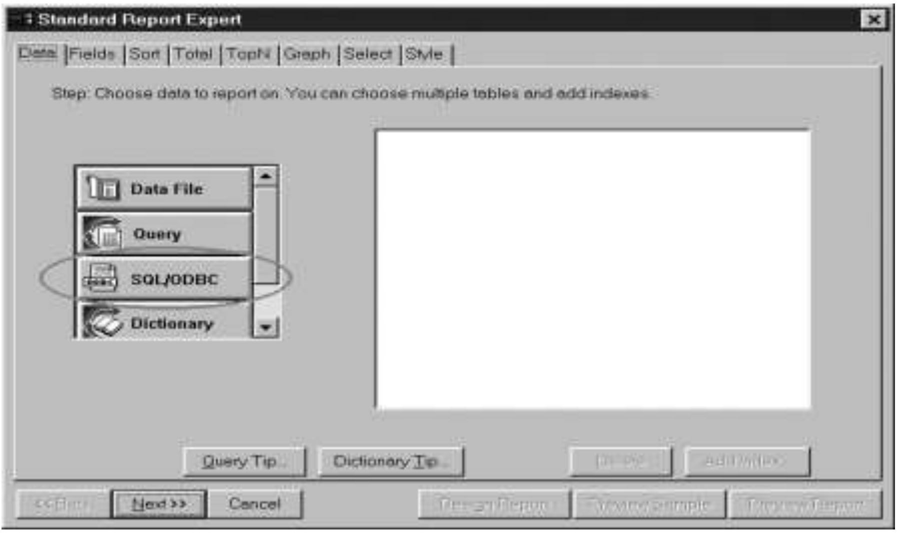
2. As shown in Figure 9-10, choose Standard from the Report Gallery:

Figure 9-1



3. Under the Data tab (Figure 9-11) select *SQL/ODBC* from the list of data to report on:

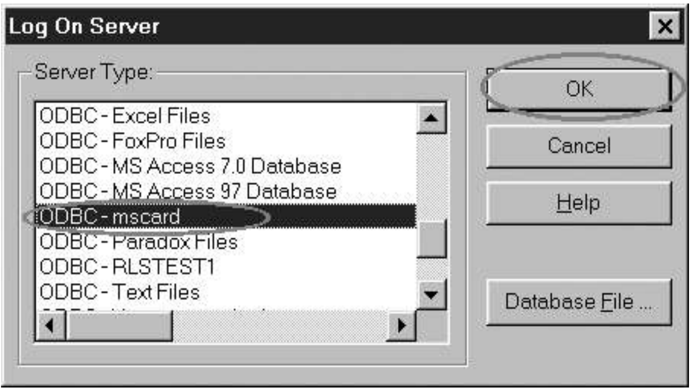
Figure 9-1



Step: Choose data to report on. You can choose multiple tables and add indexes.

4. The Log On Server dialog box appears (Figure 9-12) select the ODBC data source you created earlier from the list, then press *OK*:

Figure 9-1



5. From the Choose SQL Table dialog box (Figure 9-13) add your dataset, then click *Add*:

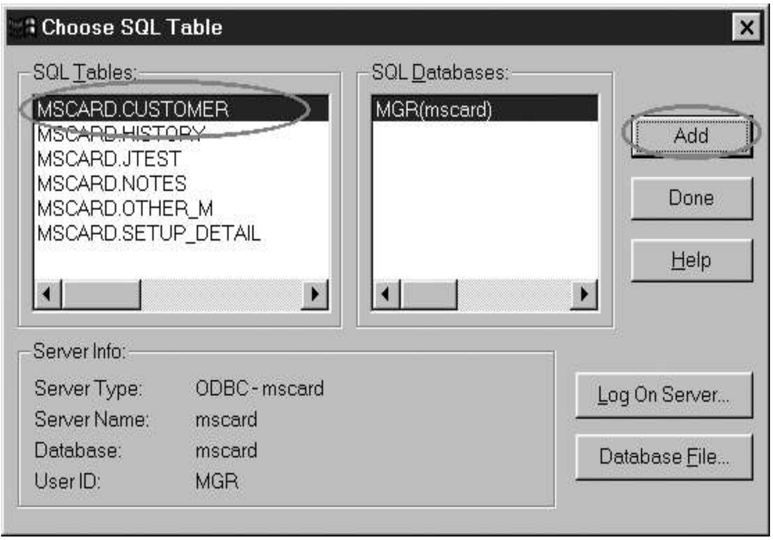


Figure 9-1

Server Type: ODBC - mscard
Server Name: mscard
Database: mscard
User ID: MGR

6. Then select *Done* as shown in Figure 9-14:

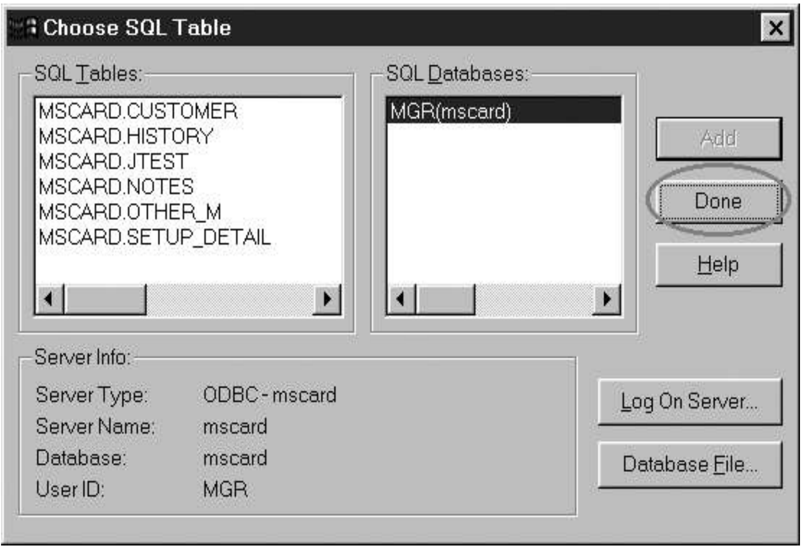
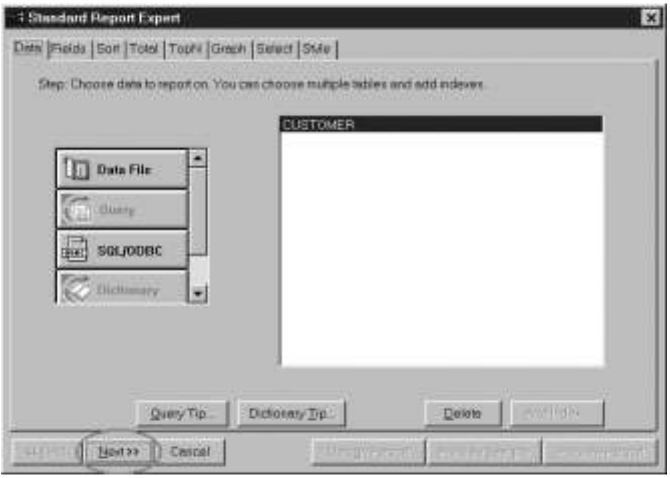


Figure 9-1

Server Type: ODBC - mscard
Server Name: mscard
Database: mscard
User ID: MGR

7. You will then be brought back to the Data tab in the Standard Report Expert dialog box, as shown in Figure 9-15. Select *Next*:

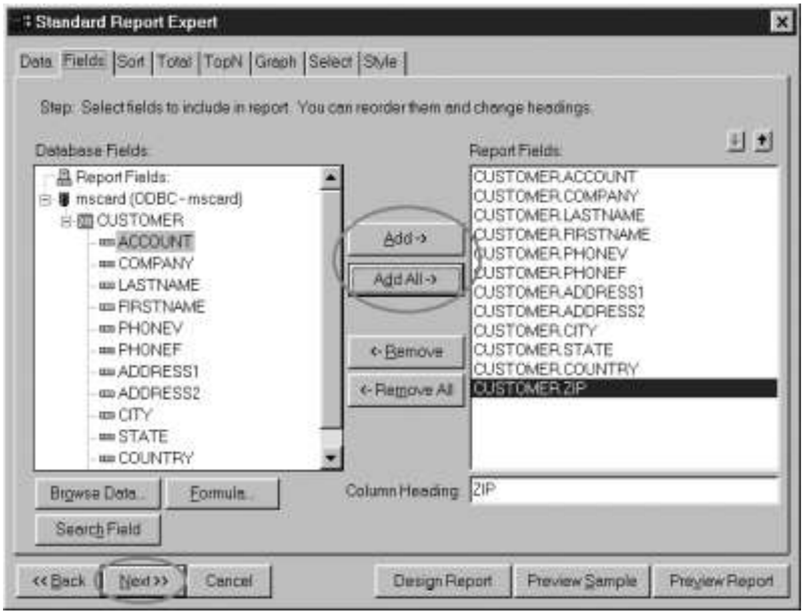
Figure 9-1



Step: Choose data to report on. You can choose multiple tables and add indexes.

CUSTOMER

8. Under the Fields tab (Figure 9-16) add some or all fields, then select *Next*:



Step: Select fields to include in report. You can reorder them and change headings.

Database Fields:
- Report Fields:
- mscard (ODBC - mscard)
  - CUSTOMER
    - ACCOUNT
    - COMPANY
    - LASTNAME
    - FIRSTNAME
    - PHONEV
    - PHONEF
    - ADDRESS1
    - ADDRESS2
    - CITY
    - STATE
    - COUNTRY

Report Fields:
CUSTOMER.ACCOUNT
CUSTOMER.COMPANY
CUSTOMER.LASTNAME
CUSTOMER.FIRSTNAME
CUSTOMER.PHONEV
CUSTOMER.PHONEF
CUSTOMER.ADDRESS1
CUSTOMER.ADDRESS2
CUSTOMER.CITY
CUSTOMER.STATE
CUSTOMER.COUNTRY
CUSTOMER.ZIP

Column Heading: ZIP

9. Under the Sort tab (Figure 9-17) add a sort field, then select *Preview Report*:

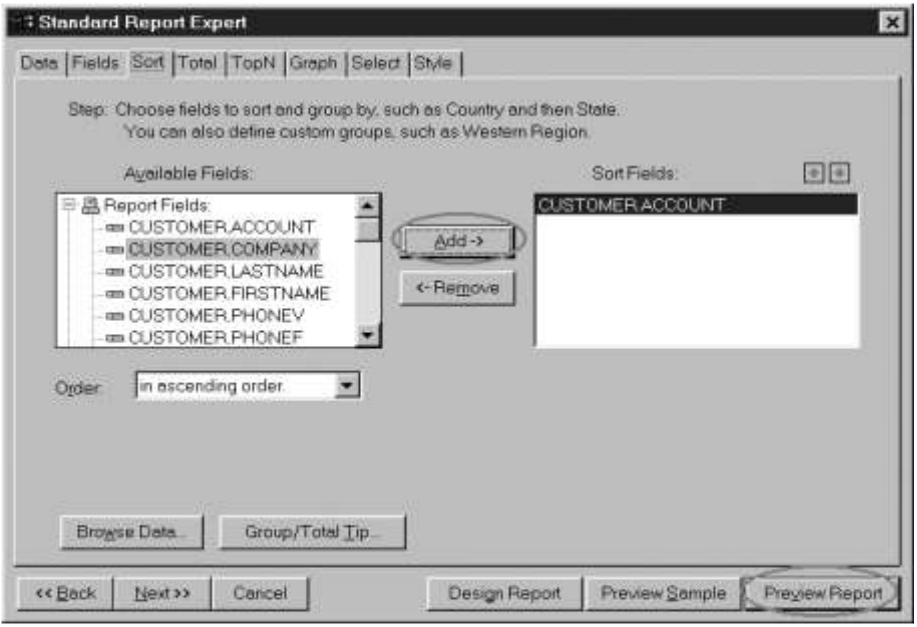


Figure 9-1

Step: Choose fields to sort and group by, such as Country and then State.
You can also define custom groups, such as Western Region.

Available Fields:

Report Fields:
- CUSTOMER.ACCOUNT
- CUSTOMER.COMPANY
- CUSTOMER.LASTNAME
- CUSTOMER.FIRSTNAME
- CUSTOMER.PHONEV
- CUSTOMER.PHONEF

Sort Fields:

CUSTOMER.ACCOUNT

Order: in ascending order

10. Data from your dataset will be displayed in a default format. You can now adjust the report layout.

Crystal Reports three table join

To create a three table inner join in Crystal Reports, you will need to add an OuterJoin key to your registry. This will allow Crystal Reports to generate the correct syntax for your ODBC driver. The following instructions will show you how:

1. Edit your registry. From the taskbar click *Start* > *Run* and enter “regedit” then click *OK*.

2. Find the following key:

HKEY_CURRENT_USER/Software/Seagate Software/Crystal Reports/DatabaseOptions/OuterJoin

3. If there is no OuterJoin key, click *DatabaseOptions* on the menu bar. Click *Edit* > *New* > *Key*. Enter *OuterJoin* for the value.

4. Click *OuterJoin*. On the menu bar, click *Edit* > *New* > *String Value*. Type

SQL2outerjoin for the key value. Press *Enter*.

5. With *SQL2outerjoin* highlighted on the menu bar click *Edit* > *Modify*. In the Value data field enter *3kodbc*, then click *OK*.
6. You will now be able to do a three table (or more) inner join in Crystal Reports. Unfortunately this fix breaks the outerjoin functionality. In order to perform an outerjoin you will need to edit the SQL code and add the following highlighted code ({oj }) to your Select statement:

```
SELECT

TRACK_DETAIL.”PROBLEM_ID”, TRACK_DETAIL.”COMPANY”,
TRACK_DETAIL.”MASTER_ID”,

MASTER_DETAIL.”MASTER_ID”, MASTER_DETAIL.”MASTER_DESC”

FROM

    {oj “TRACX”.”TRACK_DETAIL” TRACK_DETAIL LEFT OUTER JOIN
    “TRACX”.”MASTER_DETAIL”

    MASTER_DETAIL ON TRACK_DETAIL.”MASTER_ID” =
    MASTER_DETAIL.”MASTER_ID”}
```

9. Using JDBC

JDBC Classes and Interfaces

The JDBC classes and interfaces are declared in *java.sql*. The classes are the common components of JDBC. The main class is the ***DriverManager***. Use the ***DriverManager*** class to make a connection to a database. Other classes in *java.sql* are used for date handling, exception handling and common constants. The interfaces in *java.sql* are the blueprints used by JDBC driver developers to create a JDBC driver. Each JDBC driver must contain classes that implement the interfaces in *java.sql*. A typical JDBC driver is a set of these classes, plus some support classes, contained in an archive.

The Big Four

Basic data access through JDBC can be accomplished using only four JDBC classes. They are:

DriverManager
Connection
Statement
ResultSet

You can access these classes through the interfaces since each of the driver's classes are implementations of the interfaces described in *java.sql*. This means that you only need to specify which driver to use during the connection request. After that the code is driver independent.

Registering the driver

The first step in using a JDBC driver is to register it with the ***DriverManager***. A common way to do this is shown below:

Class.forName (“com.minisoft.jdbc.MSJDBCDriver”);

The *forName* method of the ***Class*** class returns a ***Class*** object associated with the class with the given string name. This is not needed but the code will force the class named by the string to be loaded. The class name of the JDBC driver is “com.minisoft.jdbc.MSJDBCDriver”. When the JDBC driver is loaded it will register itself with the ***DriverManager***.

Connecting to a database

Now that the driver is registered with the ***DriverManager***, you can request a connection to a database. The *getConnection* method of the ***DriverManager*** class will do this. This is illustrated below:

Connection con=DriverManager.getConnection
(“jdbc:MSJDBC//127.0.0.1:30504/MSDB”);

The string parameter is a url which describes the connection. The first part “jdbc” is the main protocol. The second part “MSJDBC” is a sub-protocol that identifies Minisoft’s JDBC driver. The ***DriverManager*** will use the sub-protocol to query each registered driver to see if the driver handles this sub-protocol. The third and subsequent parts identify how to connect to the database.

Note: This is driver specific. In this case, using Minisoft’s JDBC driver, the third part is specifying an IP address and port number for the mid-tier server, while the fourth part is specifying a datasource. The datasource contains the HP3000 specific information to connect to the database(s).

Connecting using connection properties

MiniSoft’s JDBC driver also supports another form of *getConnection*, in which connection properties can be passed. This form of *getConnection* is shown in Figure 28.

```
String url = “jdbc:MSJDBC://127.0.0.1:30504/”;
Properties p = new Properties () ;
p.put ( “Server”, “data.minisoft.com” );
p.put ( “Server Port”, “31100” );
p.put ( “User”, “MGR” );
p.put ( “User Password”, “HEREYAGO” );
p.put ( “Account”, “MINISOFT” );
p.put ( “Group”, “MM” ) ;
p.put ( “Database0”, “MSDB,DBPSWD,1,5”);
Connection con = DriverManager.getConnection ( url, p );
```

Figure 28

For a list of all properties supported see chapter 4 on *JDBC API Reference*. By using this form of *getConnection* you do not need a datasource set up on the mid-tier machine. You can specify everything that is needed to connect to the database as a property.

The *getConnection* method returns an object from the selected driver that conforms to the *Connection* interface. From now on, objects created by the driver that conform to an interface in *java.sql*, will be referred to by their interface name, not the internal name of the class in the driver. This is because you never need to know what the internal name is, you always access the object through its interface. The ***Connection*** object can now be used to create a ***Statement*** object.

Creating a Statement Object

To create a Statement Object use the example shown below:

```
Statement stmt = con.createStatement ();
```

The ***createStatement*** method of a ***Connection*** object returns a ***Statement*** object. The *Statement* object can now be used to execute a SQL query with the *executeQuery* method as shown below:

```
ResultSet rs = stmt.executeQuery ( “SELECT * FROM CUSTOMERS” );
```

This will execute the SQL statement passed as a parameter and create a ***ResultSet*** object. The ***ResultSet*** object can now be used to retrieve the results of the SQL statement as shown below:

```
while ( rs.next () ) {
s = rs.getString ( “CUSTOMER_NUMBER” );
System.out.println ( s );
s = rs.getString ( “CUSTOMER_NAME” );
System.out.println ( s );
s = rs.getString ( “ADDRESS1” );
System.out.println ( s );
s = rs.getString ( “ADDRESS2” );
System.out.println ( s );
s = rs.getString ( “CITY” );
System.out.println ( s );
s = rs.getString ( “STATE” );
System.out.println ( s );
s = rs.getString ( “COUNTRY” );
System.out.println ( s );
s = rs.getString ( “ZIP” );
System.out.println ( s );
s = rs.getString ( “DATE” );
System.out.println ( s );
}
```

The *Next* method obtains the next record from the result set. The *getString* method is used to return data from a column. The parameter can be either a column name as a string or a column number (1 based). The *getString* method is normally used to return alphanumeric columns, although some drivers will return any data type as a string. If you need to retrieve the data in its normal data type, there are get... methods for all the various data types.

Exiting JDBC

It is good practice to exit a program gracefully, although the JDBC driver will take care of things if you do not. To exit the JDBC program use the code shown below:

```
rs.close ( );
stmt.close ( );
con.close ( );
```

When the *close* method of a ***Connection*** object is called, it will request the close method of any ***Statement*** objects it created. Similarly, when the close method of a ***Statement*** object is called it will request the *close* method of any ***ResultSet*** objects it created. Given this scenario, you could just call *con.close*.

A complete example that puts together all the above code pieces with added exception handling is shown in Figure 29.

The exception handling catches any exceptions thrown by the JDBC methods. Many of the methods in JDBC can throw a ***SQLException***. Printing out the exception will typically show the error message.

```
import java.sql.*;
public class FirstDBAccess
{
   public static void main (String [ ] args)
   {
      try {
         Class.forName ( “com.minisoft.jdbc.MSJDBCDriver” );
         String url = “jdbc:MSJDBC://127.0.0.1:30504/MSDB”;
         Connection con = DriverManager.getConnection ( url );
         try {
            Statement stmt = con.createStatement ( );
            String query = “SELECT * FROM CUSTOMERS”;
            ResultSet rs = stmt.executeQuery ( query );
            while ( rs.next ( ) ) {
               s = rs.getString ( “CUSTOMER_NUMBER” );
               System.out.println ( s );
            }
            stmt.close ( );
         }
         catch ( SQLException e2 )
         {
            System.out.println ( e2 );
         }
         con.close ( );
      }
      catch ( SQLException e0 )
      {
         System.out.println ( e0 );
      }
      catch ( ClassNotFoundException e1 )
      {
         System.out.println ( e1 );
      }
   }
}
```

Figure 29: A complete application that retrieves data from a TurboImage database.

Prepared Statements and Parameters

In many situations you will want to execute essentially the same statement a number of times. The only difference between each execution of the statement might be some selection criteria or update values.

For example, if you needed to retrieve customer information by customer number based upon a customer number entered by the user. When the user requested customer number ‘000001’, you could build a string that contained “SELECT * FROM CUSTOMERS WHERE CUSTOMER_NUMBER = ‘000001’” and execute it. Then when the user requested customer number ‘000002’, you would build a string that contained “SELECT * FROM CUSTOMERS WHERE CUSTOMER_NUMBER = ‘000001’” and execute it. This method is very inefficient because the driver will have to compile essentially the same statement many times.

The preferred way to accomplish this task is to use a statement that contains parameters and compile it. Once prepared, you can supply values for the parameters and execute the statement as many times as needed without compiling it again. Instead of using the *createStatement* method of a ***Connection*** object, you can use the *prepareStatement* method. Its one parameter is a string that contains the SQL statement to prepare. The *prepareStatement* method returns a ***PreparedStatement*** object, which is a subclass of the ***Statement*** class. The ***PreparedStatement*** class has *set…* methods to set the values of parameters. Figure 30 illustrates preparing a statement, setting its parameters and executing it many times.

SQL statements use the question mark (‘?’) to mark the position of the parameters. The parameters are numbered, starting at 1, in the order they appear in the statement. The *setString* method’s first parameter is the parameter number, its second is the value for the parameter. All parameter values can be cleared with the *clearParameters* method.

```
void DoneOneTime ( )
{
   ...
   Connection con = DriverManager.getConnection ( url );
   String query = “SELECT * FROM CUSTOMERS WHERE
   CUSTOMER_NUMBER = ?”;
   PreparedStatement stmt = con.prepareStatement ( query );
   ...
}

void DoneManyTimes ( String CustomerNumber )
{
   stmt.setString ( 1, CustomerNumber );
   ResultSet rs = stmt.executeQuery ( );
   ...
}
```

Figure 30: A prepared statement with parameters.

Metadata

Metadata is data which describes data. There are two types of metadata objects that a JDBC driver can provide. One is based upon the ***DatabaseMetaData*** interface and the other is based upon the ***ResultSetMetaData*** interface.

The ***DatabaseMetaData*** class mainly contains methods to gather information concerning a database. The most common information is the names of the tables in the database and the layout of those tables. Other less frequently used information is access privileges and the relationships between tables. A ***DatabaseMetaData*** object is created with the *getMetaData* method of a ***Connection*** object. Figure 31 illustrates using a ***DatabaseMetaData*** object to retrieve the names of all the tables in a database.

The *getTables* method returns a ***ResultSet*** object that is used to retrieve the information about the tables. Each row in the result set has 5 columns as follows:

1. **TABLE_CAT** String => table catalog (may be null).
2. **TABLE_SCHEM** String => table schema (may be null).
3. **TABLE_NAME** String => table name.
4. **TABLE_TYPE** String => table type. Typical types are “TABLE”, “VIEW”, “SYSTEM TABLE”, “GLOBAL TEMPORARY”, “LOCAL TEMPORARY”, “ALIAS” and “SYNONYM”.
5. **REMARKS** String => explanatory comment on the table.

Depending to the driver, the TABLE_CAT and/or the TABLE_SCHEM columns may be null indicating that these are not attributes of the database the driver supports. Figure 4 illustrates loading table names into a list box.

```
...

   Connection con = DriverManager.getConnection ( url );
   Choice tableNames = new Choice ( );
   DatabaseMetaData md = con.getMetaData ( );
   String [ ] types = { “TABLE” };
   ResultSet mrs = md.getTables ( null, “”, “”, types );
   while ( mrs.next ( ) ) {
      tableNames.addItem ( mrs.getString ( 3 ) );
   }
}
...
```

Figure 31: Loading table names into a list box.

The ***ResultSetMetaData*** class contains methods to gather information about a result set. This information contains the number of columns in each row of the result set and the layout of each column. This information is very useful to programs that need to dynamically create the layout for displaying data from a database. A ***ResultSetMetaData*** object is created with the *getMetaData* method of a ***ResultSet*** object. Figure 32 shows how to gather and use the data type of a result set column.

```
String s;
int i;

...

Connection con = DriverManager.getConnection ( url );
Statement stmt = con.createStatement ( );
ResultSet rs = stmt.executeQuery ( “SELECT * FROM CUSTOMERS” );
ResultSetMetaData md = rs.getMetaData ( );
while ( rs.next ( ) ) {
   int col;
   for ( col = 1; col <= md.getColunmCount; ++col ) {
      switch ( md.getColumnType ( col ) ) {
      case Types.CHAR :
         s = rs.getString ( col );

...

         break;
      case Types.INTEGER:
         i = rs.getInt ( col );

...

         break;

...

      }
   }
}
```

Figure 32: Using ResultSetMetaData to get a column’s data type.

Adding, Updating and Deleting Data

JDBC can be used to add, update and/or delete records in a table. The *executeUpdate* method of a ***Statement*** (or ***PreparedStatement***) object is used to execute SQL INSERT, UPDATE and DELETE statements. The return of the *executeUpdate* method indicates the number of records affected by the SQL statement. The setting of auto-commit for the ***Connection*** object determines if each statement is committed automatically, or if an explicit commit or rollback must be done. If auto-commit is on, then each statement executed with *executeUpdate* method will be committed immediately. If auto-commit is off, a commit or rollback will only be done when a *commit* or *rollback* method of the ***Connection*** object is called.

By default, new connections start with auto-commit on. Different drivers handle locking, transaction isolation and concurrency differently. The driver’s documentation will need to be consulted to determine how the driver behaves, and how compatible it will be with other applications that are accessing the database. Figure 33 demonstrates adding a customer record to the customers table, while querying and updating the next value for customer numbers.

```
Connection con = DriverManager.getConnection ( url );
con.setAutoCommit ( false );

...

Statement stmt1 = con.createStatement ( );
ResultSet rs = stm1.executeQuery ( “SELECT NEXT_NUMBER FROM
NEXT_NUMBERS WHERE CATAGORY = ‘CU’” );
rs.next ( );
int nextCust = getInt ( 1 );
PreparedStatement stmt2 = con.prepareStatement ( “INSERT INTO
CUSTOMERS (CUSTOMER_NUMBER, ...) VALUES (?,...)”);
stmt2.setInt ( 1, nextCust );

...

stmt2.executeUpdate ( );
PreparedStatement stmt3 = con.prepareStatement ( “UPDATE
NEXT_NUMBERS SET NEXT_NUMBER = ? WHERE CATAGORY = ‘CU’”);
stmt3.setInt (1, ++nextCust );
stmt3.executeUpdate ( );
con.commit ( );

...
```

Figure 33: INSERT and UPDATE in a single transaction.

JDBC API Reference

The following is a reference to the most commonly used methods of the JDBC API. A reference of the complete JDBC API can be found on the JavaSoft web site at www.javasoft.com.

DriverManager Methods

getConnection

This method requests a connection to a database. If a connection is made, a ***Connection*** object is returned.

Example:

public static synchronized Connection getConnection(String url) throws SQLException

public static synchronized Connection getConnection(String url, Properties info) throws SQLException

Parameters:

- url - A string which describes the connection in the form “jdbc:MSJDBC://<mid-tier server host name>[:<mid-tier server port>][/<datasource>].
- The main protocol. <mid-tier server host name> and < mid-tier server port> identify the machine the mid-tier server is running on and the TCP port it is listening on.
- <datasource> specifies datasource on the mid-tier server machine that contains the HP3000 specific information to connect to the database(s).
- info – A Properties collection that contains the HP3000 specific information to connect to the database(s). Information in the info parameter will override information contained in a datasource. The available properties are:

2DriverTable - <name of a translation table file for translating data coming to the client>
2HostTable - <name of a translation table file for translating data going from the client>
Account - <HP3000 logon account>
Account Password - <HP3000 logon account password>
Database<n> - <database name to access on the HP3000>,<database password>,<load auto master flag>,<database open mode>
DecimalPoint - <decimal point character>
Group - <HP3000 logon group>
Group Password - <HP3000 logon group password>
Jobname - <jobname>
Langauge - <NLS language>
Schema<n> - <name of the schema file>,<lockword>
Server - <HP3000 IP address or hostname>
Server Port - <server TCP port>
User - <HP3000 user logon>
User Password=<HP3000 user password>

Note: All property names are case sensitive.

For the properties Database<n>, and Schema<n>, <n> indicates a number, starting with 0. The first database defined would be with property “Database0”, the second would be with “Database1”, and so on.

The values for <load auto master flag> is 1 to load automatic masters and 0 not to load automatic masters.

The values for <database open mode> are the same as on the DBOPEN intrinsic, i.e. 1 – 8.

Connection Methods

createStatement

This method creates a ***Statement*** object. This is typically used when the statement has no parameters and is only going to be executed once.

Example:

```
public abstract Statement createStatement() throws SQLException
```

PrepareStatement

Creates a ***PreparedStatement*** object that represents a compiled SQL statement. This is typically used when the statement contains parameters, or when the statement will be executed more than once.

Example:

```
public abstract PreparedStatement prepareStatement(String sql) throws
SQLException
```

Parameters:

- sql - A string which contains the SQL statement to compile.

setAutoCommit

Sets the state of the connection’s auto-commit mode. If a connection’s auto-commit mode is true, then each statement that modifies the database will be committed upon completion. If a connection’s auto-commit mode is false, then data will only be committed to the database when the commit method is called.

Example:

public abstract void setAutoCommit(boolean autoCommit) throws SQLException

Parameters:

- autoCommit - The state to set the auto-commit mode.

commit

Commits any changes made by SQL statements since the transaction started to the database. Transactions are automatically started when the first SQL statement that modifies the database is executed.

Example:

public abstract void commit() throws SQLException

rollback

Cancels and changes made by SQL statements since the transaction started.

Example:

public abstract void rollback() throws SQLException

close

Closes the connection and all open databases, statements and result sets.

Example:

public abstract void close() throws SQLException

getMetaData

Creates a ***DatabaseMetaData*** object. The ***DatabaseMetaData*** object is used to provide information about the connection and the open database(s).

Example:

public abstract DatabaseMetaData getMetaData() throws SQLException

Statement Methods

executeQuery

Compiles and executes a SQL statement and returns a ***ResultSet*** object.

Example:

public abstract ResultSet executeQuery(String sql) throws SQLException

Parameters:

- sql - A string containing the statement to be executed.

executeUpdate

Executes a DELETE, INSERT or UPDATE SQL statement. The return is the number of records effected by the SQL statement.

Example:

public abstract int executeUpdate(String sql) throws SQLException

Parameters:

- sql - A string containing the statement to be executed.

close

Closes the ***Statement*** and its current ***ResultSet*** if one exists.

Example:

public abstract void close() throws SQLException

PreparedStatement Methods

executeQuery

Executes a previously compiled SQL statement and returns a ***ResultSet*** object.

Example:

public abstract ResultSet executeQuery() throws SQLException

executeUpdate

Executes a previously compiled DELETE, INSERT or UPDATE SQL statement.

Example:

```
public abstract int executeUpdate() throws SQLException
```

Set...

```
public abstract void setShort(int parameterIndex, short x) throws
SQLException
public abstract void setInt(int parameterIndex, int x) throws
SQLException
public abstract void setLong(int parameterIndex, long x) throws
SQLException
public abstract void setFloat(int parameterIndex, float x) throws
SQLException
public abstract void setDouble(int parameterIndex, double x) throws
SQLException
public abstract void setBigDecimal(int parameterIndex, BigDecimal x)
throws SQLException
public abstract void setString(int parameterIndex, String x) throws
SQLException
public abstract void setDate(int parameterIndex, Date x) throws
SQLException
```

Set the value of a parameter in a SQL statement.

Parameters:

- parameterIndex - The number of the parameter in the SQL statement, starting at 1.
- x - The value for the parameter.

clear Parameters

Clears all SQL statement parameters.

Example:

```
public abstract void clearParameters() throws SQLException
```

ResultSet Methods

next

Fetches the next available row of a result set.

Example:

public abstract boolean next() throws SQLException

close

Closes a ***ResultSet***.

Example:

public abstract void close() throws SQLException

Get...

public abstract String getString(int columnIndex) throws SQLException
public abstract short getShort(int columnIndex) throws SQLException
public abstract int getInt(int columnIndex) throws SQLException
public abstract long getLong(int columnIndex) throws SQLException
public abstract float getFloat(int columnIndex) throws SQLException
public abstract double getDouble(int columnIndex) throws SQLException
public abstract BigDecimal getBigDecimal(int columnIndex, int scale) throws SQLException
public abstract Date getDate(int columnIndex) throws SQLException
public abstract String getString(String columnName) throws SQLException
public abstract short getShort(String columnName) throws SQLException
public abstract int getInt(String columnName) throws SQLException
public abstract long getLong(String columnName) throws SQLException
public abstract float getFloat(String columnName) throws SQLException
public abstract double getDouble(String columnName) throws SQLException
public abstract BigDecimal getBigDecimal(String columnName, int scale) throws SQLException
public abstract Date getDate(String columnName) throws SQLException

Returns the value of a column from the current row within the result set.

Parameters:

- columnIndex - The number of the columns together starting at 1.
- columnName - The name of the column to get.

getMetaData

Returns a ***ResultSetMetaData*** object that has information about this ***ResultSet***.

Example:

public abstract ResultSetMetaData getMetaData() throws SQLException

findColumn

Obtains a column number that corresponds to a column name.

Example:

public abstract int findColumn(String columnName) throws SQLException

Parameters:

- columnName - The name of the column.

ResultSetMetaData Methods

getColumnCount

Finds the number of columns in the ***ResultSet***.

Example:

public abstract int getColumnCount() throws SQLException

getColumnDisplaySize

Obtains the number of characters necessary to display the data from a column.

Example:

public abstract int getColumnDisplaySize(int column) throws SQLException

Parameters:

- column - Column number.

getColumnName

Obtains the name of a column in the ***ResultSet***.

Example:

public abstract String getColumnName(int column) throws SQLException

Parameters:

- column - Column number.

getColumnType

Obtains the data type of a column in the ***ResultSet***. Data type constants are defined in java.sql.Types.

Example:

public abstract int getColumnType(int column) throws SQLException

Parameters:

- columns - Column number.

DatabaseMetaData Methods

getTables

Obtains a ***ResultSet*** object containing information about the tables within the database.

Example:

public abstract ResultSet getTables(String catalog, String schema, String tableName, String types[]) throws SQLException

Parameters:

- catalog - Ignored. Catalog is not supported by MiniSoft's JDBC driver.
- schema - The name of schema to search for tables. A schema name is either a database name or a schema editor file name. If empty all

schemas are searched.

- tableName - The name of a table to gather information about. If empty information about all the tables is returned.
- types - An array of strings designating the types of tables to be included. If empty all types are included. The only type supported is “TABLE”.

The columns in the ***ResultSet*** are:

1. **TABLE_CAT** string => table catalog (may be null).
2. **TABLE_SCHEM** String => table schema (may be null).
3. **TABLE_NAME** String => table name.
4. **TABLE_TYPE** String => table. Typical types are “TABLE”, “VIEW”, “SYSTEM TABLE”, “GLOBAL TEMPORARY”, “LOCAL TEMPORARY”, “ALIAS” and “SYNONYM”.
5. **REMARKS** String => explanatory comment on the table.

getColumns

Obtains a ***ResultSet*** object containing information about the columns in the table.

Example:

```
public abstract ResultSet getColumns(String catalog, String schema,
String tableName, String columnName) throws SQLException
```

Parameters:

- catalog - ignored. Catalog is not supported by MiniSoft’s JDBC driver.
- schema - The name of a schema. A schema name is either a database name or a schema editor file name. If empty all schemas are searched.
- tableName - The name of a table to gather the column information for.
- columnName - The name of a column. If empty includes all columns.

The columns in the ***ResultSet*** are:

1. **TABLE_CAT** String => table catalog (may be null)
2. **TABLE_SCHEM** String => table schema (may be null)

3. **TABLE_NAME** String => table name
4. **COLUMN_NAME** String => column name
5. **DATA_TYPE** short => SQL type from java.sql.Types
6. **TYPE_NAME** String => Data source dependent type name
7. **COLUMN_SIZE** int => column size. For char or date types this is the maximum number of characters, for numeric or decimal types this is precision.
8. **BUFFER_LENGTH** is not used
9. **DECIMAL_DIGITS** int => the number of fractional digits
10. **NUM_PREC_RADIX** int => Radix (typically either 10 or 2)
11. **NULLABLE** int => is NULL allowed?
    - columnNoNulls - might not allow NULL values
    - columnNullable - allows NULL values
    - columnNullableUnknown - nullability unknown
12. **REMARKS** String => comment describing column (may be null)
13. **COLUMN_DEF** String => default value (may be null)
14. **SQL_DATA_TYPE** int => unused
15. **SQL_DATETIME_SUB** int => unused
16. **CHAR_OCTET_LENGTH** int => for char types the maximum number of bytes in the column
17. **ORDINAL_POSITION** int => index of column in table (starting at 1)
18. **IS_NULLABLE** String => “NO” means column definitely does not allow NULL values; “YES” means the column might allow NULL values. An empty string means no one knows.

getIndexInfo

Get a ***ResultSet*** object containing information about the indexes or keys for the table.

Example:

public abstract ResultSet getIndexInfo(String catalog, String schema, String table, boolean unique, boolean approximate) throws SQLException

Parameters:

- catalog - Ignored. Catalog is not supported by MiniSoft's JDBC driver.
- schema - The name of schema. A schema name is either a database name or a schema editor file name. If empty all schemas are searched.
- table - The name of a table to gather the column information for.
- approximate - Ignored. All statistics are exact.

The columns in the ***ResultSet*** are:

1. **TABLE_CAT** String => table catalog (may be null)
2. **TABLE_SCHEM** String => table schema (may be null)
3. **TABLE_NAME** String => table name
4. **NON_UNIQUE** boolean => Can index values be non-unique? False when TYPE is tableIndexStatistic.
5. **INDEX_QUALIFIER** String => index catalog (may be null); null when TYPE is tableIndexStatistic
6. **INDEX_NAME** String => index name; null when TYPE is tableIndexStatistic
7. **TYPE** short => index type:
    - tableIndexStatistic - identifies table statistics that are returned in conjugation with a table's index descriptions.
    - tableIndexClustered - is a clustered index.
    - tableIndexHashed - is a hashed index.
    - tableIndexOther - is some other style of index.
8. **ORDINAL_POSITION** short => column sequence number within index; zero when TYPE is tableIndexStatistic

9. **COLUMN_NAME** String => column name; null when TYPE is tableIndexStatistic
10. **ASC_OR_DESC** String => column sort sequence, “A” => ascending, “D” => descending, may be null if sort sequence is not supported; null when TYPE is tableIndexStatistic
11. **CARDINALITY** int => When TYPE is tableIndexStatistic, then this is the number of rows in the table; otherwise, it is the number of unique values in the index.
12. **PAGES** int => When TYPE is tableIndexStatisic then this is the number of pages used for the table, otherwise it is the number of pages used for the current index.
13. **FILTER_CONDITION** String => Filter condition, if any (may be null).

10. Using OLE DB

Building Client/Server Applications

"OLE DB" opens the door to a rich set of development tools and platforms for Microsoft Windows Servers. Microsoft has announced that OLE DB will be the method by which all information is accessed. The Microsoft OLE DB to ODBC Bridge known as MSDASQL is NOT available for the 64-bit environment. The roadmap for future Microsoft applications requires using OLE DB data sources to "provide uniform access to data stored in diverse information sources". The Minisoft OLE DB Provider provides that access to your existing Image and Eloquence databases.

Use the Minisoft OLE DB Provider for Image/Eloquence to help solve the following problems:

- Transparent access to your Image/Eloquence data from the 32-bit or 64-bit editions of SQL Server 2005.
- Integrate Image or Eloquence database access smoothly into your .NET application development environment.
- Continue accessing Image or Eloquence data with Microsoft Visual Studio, Borland development tools, Microsoft Access (through VBA), ActiveX Scripts, Crystal Reports, Windows Scripting, IIS web applications (ASP and ASP.NET) by using the Minisoft OLE DB Provider

FAQ

How do I create a Connection String?

A Connection String (or Provider String, or Extended Properties - depending on what application you are running) consists of the driver specific properties separated by semicolons. At the very least it needs to consist of ServerPort, logon information, and database and/or schema file references. For example:

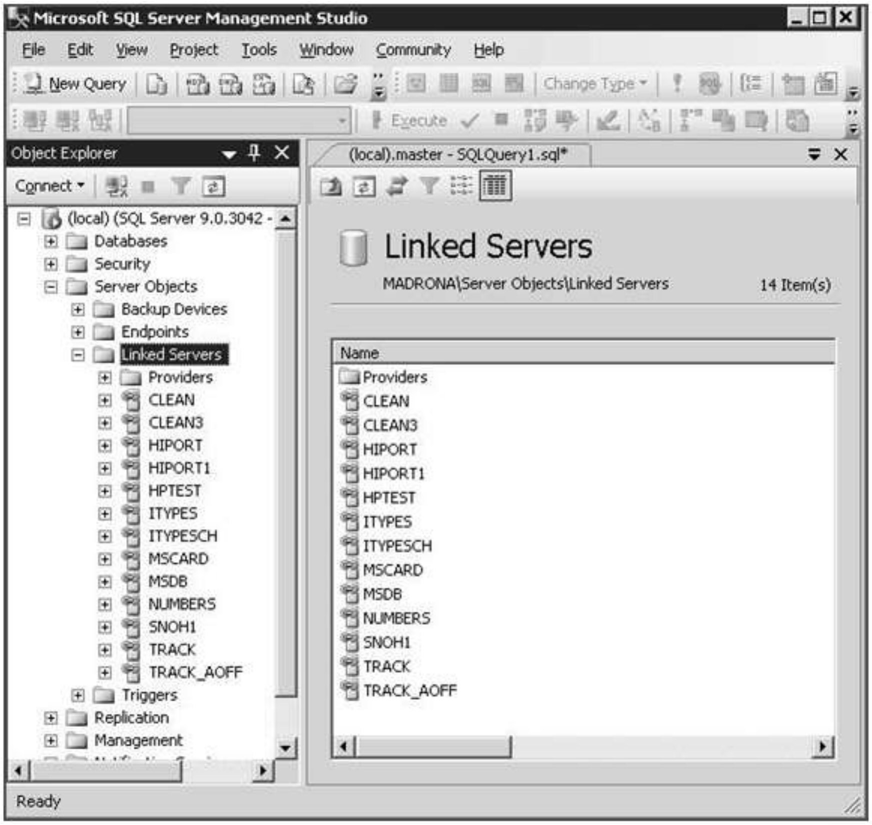
ServerPort=30009;User=xxx;UserPassword=xxx;Group=xxxt;Account=MINISOFT;

ImageDatabase0=ABC,PW,0,1,0

See Connection Properties (on page 37) for connection string parameters.

How do I create a Linked Server?

Run SQL Server Management Studio and expand out Server Objects. Right click on Linked Servers and select "New Linked Server…"



Linked Server - This is the name you will use to reference your Linked Server. It can be anything.

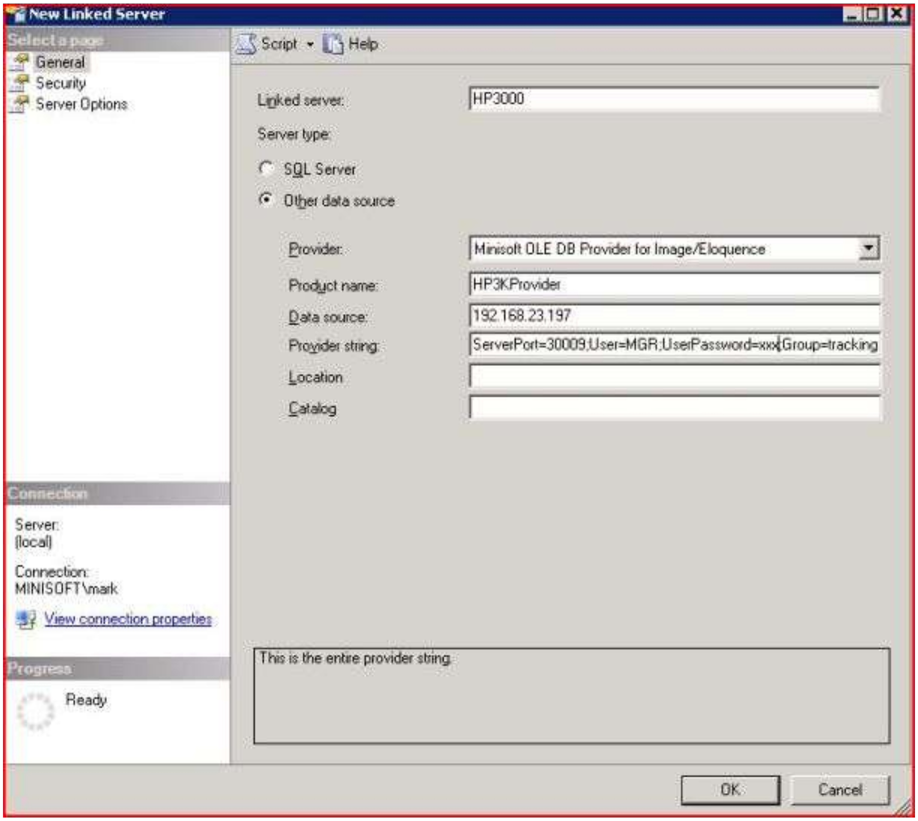
Server Type - Select “Other data source”.

Provider - Select the “Minisoft OLE DB Provider for Image/Eloquence”.

Product Name - This field can not be left blank, but the value can be anything. The suggested value is “HP3KProvider”.

Data Source - The IP Address or Host Name of the box running Image/Eloquence. This field can be left blank if the information is included in the “Provider String”.

Provider String - A list of properties separated by a semi-colon. At the very least it needs to consist of ServerPort, logon information, and database and/or schema file references. Please refer to xyz for a list of properties and Provider String examples.



Linked server: HP3000

Server type:

○ SQL Server

● Other data source

Provider: Minisoft OLE DB Provider for Image/Eloquence

Product name: HP3KProvider

Data source: 192.168.23.197

Provider string: ServerPort=30009;User=MGR;UserPassword=xxx;Group=tracking

Location:

Catalog:

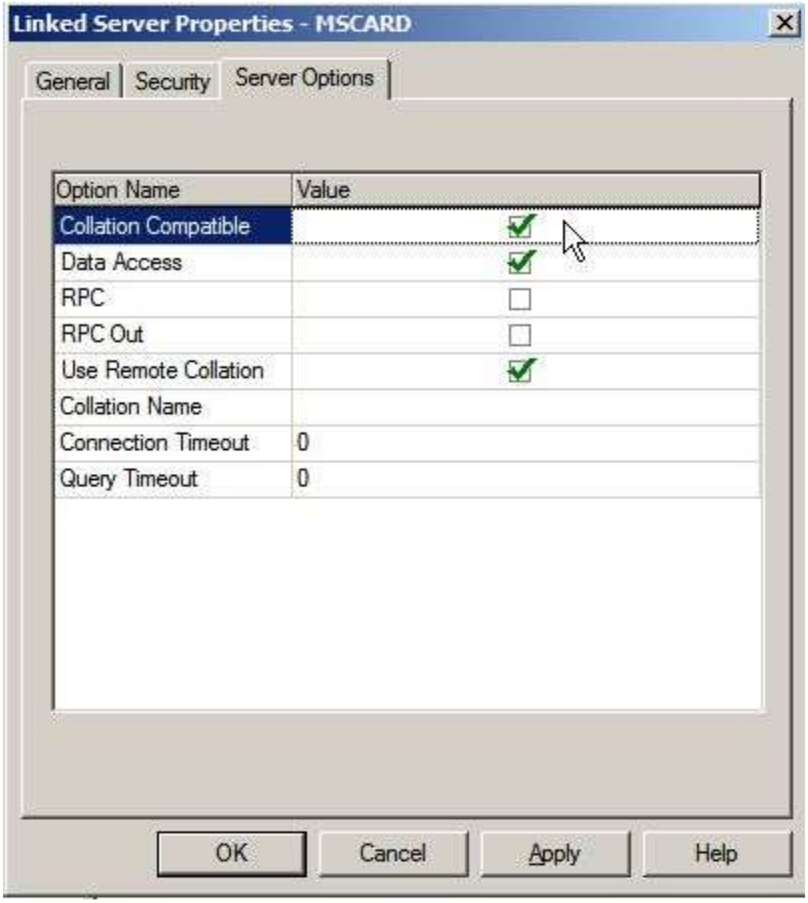
Server: (local)

Connection: MINISOFT\mark

This is the entire provider string.

How do I use character string searches in four part names with SQL Server 2005?

Each linked server will need to have the "Collation Compatible" Server Option checked.



Linked Server Properties - MSCARD

Option Name	Value
Collation Compatible	☑
Data Access	☑
RPC	□
RPC Out	□
Use Remote Collation	☑
Collation Name	
Connection Timeout	0
Query Timeout	0

A. Third-Party Indexing

If you use Schema Editor Files, you must recreate the links to your datasets any time the Indexing is changed. This step is required to pass the key information to the client application where it is stored.

All the TPI keys will appear as read-only items in the table.

If a TPI key has the same name as a TurboImage item, the TPI key item name will have "_TPI" appended to it. (This feature can be suppressed. See ImageDatabase<n> on page 40.)

In order to find records using a TPI key you must specify the TPI key item in the WHERE clause. You must update your WHERE clause on existing statements to use the TPI key, instead of the TurboImage item indexed by the key.

Specifying values for TPI keys

For X and U type items

If an item has a generic type key and the relational operator is =, the value of the selection criteria may contain the exact value or a value containing the @, ?, and # wildcard characters. The @ wildcard character is required to be at the end of the value. For example:

```
UNITED@
```

finds any values starting with UNITED.

If the key is a generic type key you may also use the following relational operators: >, >=, <, <=", " and LIKE.

If an item has a multiple-key type key and the relational operator is =, the value of the selection criteria may contain:

- A value only.

Example:
‘ABC Manufacturing’ finds any value that match exactly.

- A relational operator and a value.

  Example:
  ‘>B’ finds any values that are greater than B.

- Any of the two above combined with AND and/or OR.

  Example:
  ‘>=B AND <C’ finds any values greater than or equal to B and less than C.

Note: The value must *immediately* follow the relational operators (no spaces between them).

If the key is a multiple-key type key, you may also use the following relational operators: >=B AND ‘>=B AND, >=, <, <=”,” and LIKE.

If an item has a keyword type key, the value to the selection criteria may contain any word to be searched for. It may also contain the @, ?, and # wildcard characters. The @ wildcard character is required to be at the end of the value. You may also use the LIKE relational operator.

It is possible for an item to have more than one TPI key associated with it. If an item has a keyword type key and a generic or multiple key type key, then use @ as the first character of the selection criteria to specify a keyword search.

Examples:

’@UNITED’ finds any value with the word UNITED in it.

’UNITED@’ finds any value that starts with UNITED.

For numeric type items

You may use the following relational operators: =, >, >=, <, <=”, “ and BETWEEN. The selection criteria may only contain the value to search for.

Selection criteria for composite keys

Composite keys are always represented as X types. If all of the items that make up a composite key are X or U types than the composite key value is the concatenation of its composite items including any trailing or leading spaces. If any item of a composite key is not an X or U type, then the items value must be represented as an X type. The following shows how ODBC/32 represents composite items of various types:

This	Becomes
I1, K1, J1	X6
I2, K2, J2	X11
I4, K4, J4	X20
R2, E2	X13
R4, E4	X22
Zn	X(n+1)
Pn	X(n*2)

The value may contain a plus or minus sign, and must always be right justified and left padded with zeros. For example a composite key containing two items, where the first item is a J2 with the value of 179210 and the second is an X10 with the value of 1 ABC St, would have a value of 000001792101 ABC St.

B. Technical Support

Frequently Asked Questions

Why does starting the Schema Editor fail with a comment about OLEAUT32.DLL?

The Schema Editor may need updated 32-bit OLE files for some users of the original Win95. WinNT and OSR2 users do not need these files. Updated versions of these files are in use on most Win95 systems.

How do I run more than one version of ODBCSRVR?

You can start more than one version of the ODBCSRVR program at the same time by specifying both on the same MSSERVER000004 line.

Change the line:

```
!SETVAR MSSERVER000004 &
!”30006 0 ODBCSRVR.MM.MINISOFT S”
```

To:

```
!SETVAR MSSERVER000004 &
!”30006 0 ODBCSRVR.MM.MINISOFT S”+&
!”|31006 0 ODBCNEW.MM.MINISOFT S”
```

Port number 30006 in the DSN will start ODBCSRVR. Port number 31006 in the DSN will start ODBCNEW.

Note: Do not add spaces before or after the pipe (|) character.

Why do I see #DELETED in every field when I open a (MASTER) table?

Why can I open and see data in a (DETAIL) table, but searches return no data?

Some Image numeric type values are treated as signed values. If they are not signed and are key values, your results will not be as expected.

You must create a database view using the Schema Editor to set the type of the value to signed (NO).

Why do I get the following error message?

```
[Minisoft][3kodbc.dll] connect: Connection refused (#10061)
```

- ♦ Demonstration copies of the ODBC/32 driver expire. Please contact your Minisoft sales representative about getting an extension.
- ♦ To look at the expiration date of your software, log on to your server and type in the following commands:

```
HELLO MGR.MINISOFT,MM
FILE AF = MSSERVER
VINSTALL
```

- ♦ The updated ODBCSRVR was not uploaded with WS92 or NFT. Please delete the ODBCSRVR and upload with the recommended software.
- ♦ The job MSJOB.MM.MINISOFT must be streamed prior to making client connections.
- ♦ The port number specified on the client must match the port number in the MSJOB file for the ODBCSRVR process. As shipped, this number is 30006.
- ♦ The address given in the DSN is not that of the HP e3000 CPU. Most TELNET access to the HP e3000 is via an external box (DTC) and is a separate address from the HP e3000.
- ♦ A fire wall or proxy server is not allowing connections on port 30006. Contact your network administrator or use an allowed port.

Why do I get a message about not being able to start the server process?

The copy of ODBCSRVR on the host was not uploaded as a binary file with WS92 or NFT. You must delete the invalid file, then upload with WS92 or NFT as a binary file. You can test the upload by RUNning the ODBCSRVR and seeing a socket error message from the MPE prompt.

Why do I get a nonexistent permanent file error message?

One of the following files does not exist or was not completely qualified: SCHEMA, DATABASE or any of the tables specified by the SCHEMA.

Remember that the SCHEMA referenced by the Data Source Name (DSN) refers to a files created by the Schema Editor application and not the Image Schema file. Check that ALL the tables in your schema exist and are accessible.

Why is the list of available tables empty (or incomplete)?

The TurboImage password is incorrect. This password is case-sensitive and must be entered carefully. There can be several passwords that allow access to more or fewer of the datasets and fields.

If you are unaware of the password, see Chapter 3 on *Using DBUTIL to find database passwords*.

I cannot search on my TPI keys?

When upgrading to a version of ODBC/32 that supports TPI (1.1.0.6 or later) from an earlier version (1.1.0.5 or earlier), you must recreate the links to your datasets. This step is required to pass the key information to the client application where it is stored.

Tracing Facilities

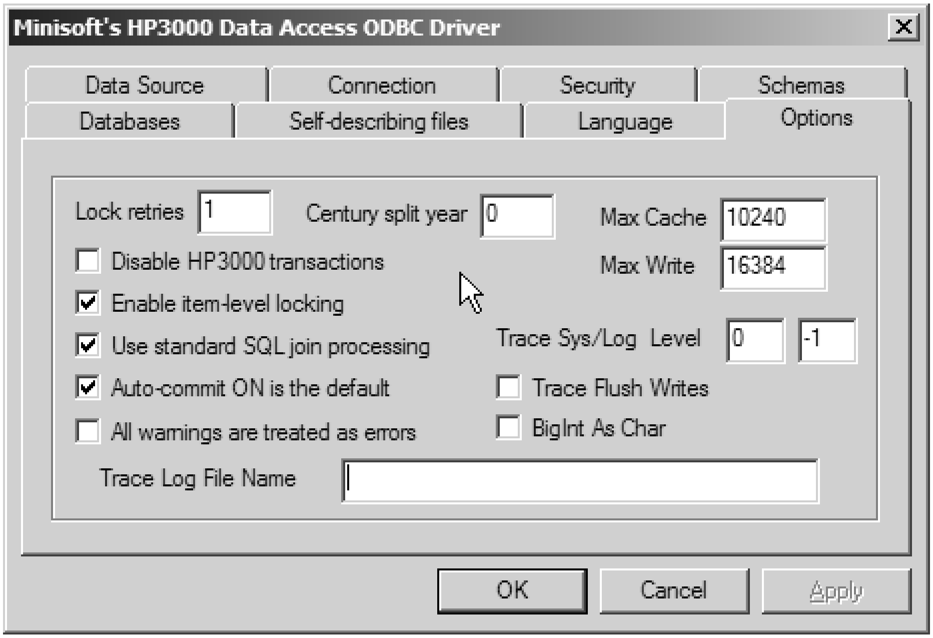
Minisoft support personnel may ask for tracing of unanticipated conditions. This document details what features are available for recording and how they are used. These features are available in version 2.2.6.2 or later.

Driver (Client) Side Tracing

Windows

There are four items on the Options Tab of the Data Source Configuration dialog that control client tracing on Windows based systems.

- Trace System Level
- Trace Log Level
- Trace Flush Writes
- Trace Log File Name



Lock retries 1

Century split year 0

Max Cache 10240

Max Write 16384

☐ Disable HP3000 transactions

☑ Enable item-level locking

☑ Use standard SQL join processing

☑ Auto-commit ON is the default

☐ All warnings are treated as errors

Trace Sys/Log Level 0 -1

☐ Trace Flush Writes

☐ BigInt As Char

Trace Log File Name

These items are detailed in the next section.

Unix and Java

Four values control the event recording features of the driver. They may be set in the DSN or from the current shell.

- TraceSysLevel
- TraceLogLevel
- TraceLogFlush
- TraceFileName

TraceSysLevel (Trace System Level)

Controls the priority level of messages that are reported to the Windows Application Event Log or Unix syslogd. The syslogd messages are recorded to "LOCAL6". Only messages of this value or lower, as shown below, will be logged. The value defaults to three (3) allowing serious errors to be recorded. (A value of −1 will disable tracing for release versions of the driver.)

TraceLogLevel (Trace Log Level)

Controls the priority level of messages that are record to a log file (TraceFileName). Only messages of this value or lower, as shown below, will be logged. The default value is five (5). Logging will not occur if TraceFileName is not provided. (A value of −1 will disable tracing for release versions of the driver.)

TraceLogFlush (Trace Flush Writes)

The default value is zero (0 or not checked). Setting this variable to one (1 or checked) will force each message to disc before processing continues. This will impact performance.

TraceFileName (Trace Log File Name)

Operating system dependant file name to which logging record will be appended. You must have write access to the file and directory. Be sure to use a fully qualified location to write the file.

Examples

Unix sample from the command line:

```
setenv TraceLogLevel 7
setenv TraceFileName /home/user/odbcfile.log
```

Unix sample of modified DSN (in odbc.ini):

```
[MSCARD]
driver = 3kodbc
Account = MINISOFT
DSN = MSCARD
Server = 192.168.23.70
Server Port = 30006
User = MGR
ImageDatabase0 = FXIJYO'OD&JXG'D'>
TraceLogLevel = 6
TraceFileName = /home/user/odbcfile.log
```

Modify the DSN in the Windows registry adding the two values:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI\MSCARD]
"TraceLogLevel"="7"
"TraceFileName"="C:\ODBCLOG.LOG"
```

Added as connection parameters:

```
url = "jdbc:MSJDBC:///?"
+ "Server=192.168.23.70&"
+ "Server Port=30007&"
+ "User=MGR&"
+ "User Password=PASSWORD&"
+ "Account=MINISOFT&"
+ "TraceFileName=/opt/minisoft/SQL_jdbc0.%p.txt&"
+ "TraceLogLevel=7&"
+ "TraceLogFlush=1&"
+ "Database0=MSCARD.PUB,DOALL,1,1";
```

TraceFileName

When the trace file name includes the characters %p, these two characters are replaced with the current process ID. Use this to create separate trace files for each process using the driver.

Logging Levels

Name	Value	Description
EMERG	0	(not currently used)
ALERT	1	(not currently used)
CRIT	2	(not currently used)
ERR	3	Indicates some failed SQL calls and communications problems.
WARNING	4	Triggered by any SQL error that generates a diagnostic record.

NOTICE	5	(not currently used)
INFO	6	Provides details of the which calls are made and some other important markers.
DEBUG	7	Shows the detailed contents of many calls.

Server (Host) Side Tracing

HP3000

MSJOB supports a number of options for tracing. Using the basic MSTRACE variable enables most processes for tracing to the job $STDLIST file. You can limit which application is traced by including the basic name for the application in the MSTRACE variable:

```
!SETVAR MSTRACEODBCSRVR 7
```

This will also work for an ODBCSRVR that has been renamed. Tracing of renamed files is based upon the original name. MSTRACEODBC2262 will not work for an ODBCSRVR that was renamed to ODBC2262. The output from an instance of a server (individual connection) may be redirected. You would commonly add the STDLIST= option to the run line for a server.

```
!FILE LP;DEV=LP,1
!SETVAR MSTRACEODBCSRVR 7
```

You can use an indirect file to start multiple instances or versions of ODBC. The indirect file would have the following format. Click here for more info.

```
30006 0 ODBCSRVR.EXE.MINISOFT;PRI=CS S W
2260 0 ODBC2260.EXE.MINISOFT;PRI=CS;INFO=' S W'
2262 0 ODBC2260.EXE.MINISOFT;PRI=CS;STDLIST=*LP;INFO=' S W'
```

In the sample above, the ODBCSRVR named ODBC2260 will run on port 2262 and each connection will use the file equation LP to create a spool file with output. The other ODBCSRVR processes will trace into the job $STDLIST.

Tracing in MSJOB is documented in (MSJOB - Configuration).

Unix

The default file name will be /opt/minisoft/odbcjdbc.%p.log, where %p is replaced by the PIN number of the server.

In inetd.conf (HPUX)

Add a /T parameter to the end of the odbcsrvr line in the /etc/inetd.conf file. You will have to restart inetd to re-read the configuration file.

```
odbcsrvr stream tcp nowait root /opt/minisoft/odbcsrvr.exe odbcsrvr.exe
 /S /T:7
```

As root, restart inetd using the command:

```
inetd -c
```

In /etc/xinetd.d/odbcsrvr (Linux)

Add a /T parameter to the end of the odbcsrvr line in the /etc/inetd.conf file. You will have to restart inetd to re-read the configuration file.

```
service odbcsrvr
        {
        port                    = 30006
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /opt/minisoft/odbcsrvr.exe
        server_args             = /S /W /T:7
        log_on_success          += DURATION USERID
        log_on_failure          += USERID
        disable                 = no
        }
```

As root, restart inetd using the command:

```
service xinetd reload
```

Using the command line

Set the environment variables as if you where configuring for the client (see above).

- TraceSysLevel
- TraceLogLevel
- TraceLogFlush
- TraceFileName

```
cd /opt/minisoft
export TraceLogLevel=7
export TraceLogFlush=1
export TraceFileName=/opt/minisoft/odbcfile.%p.log
/opt/minisoft/odbcsrvr.exe /T:15 /P:30016
```

Troubleshooting

First UNIX Connection

If you are having problems with connecting for the first time after an installation, follow these steps:

1. From a terminal logon as root
2. cd /opt/minisoft
3. ./odbcsrvr.exe /T /P:3333
4. Create/Modify a DSN to use port 3333
5. Start a connection using this DSN
6. Stop odbcsrvr using Control-C
7. Look for file with the name odbcjdbc.#.log in the current directory.

The log files will provide details about what is happening during the connection process. If the problems are not resolved after reviewing these files, please tar and send them to support@minisoft.com.

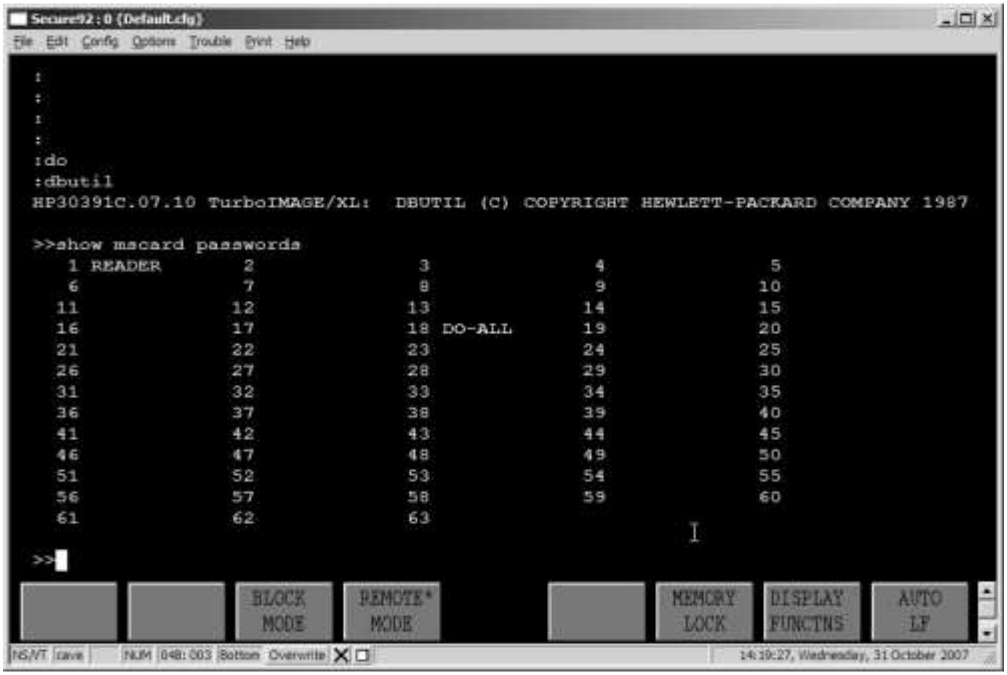
Using DBUTIL to find database passwords

1. Logon to the HP e3000 as the creator of the database in the group that contains the database. For example:

```
HELLO MGR.DBFILES,DATA
```

2. Run the utility DBUTIL as shown in Figure 34and enter the command:

```
show database passwords
```



Secure92 : 0 (Default.cfg)
File Edit Config Options Trouble Print Help
BLOCK MODE
REMOTE* MODE
MEMORY LOCK
DISPLAY FUNCTNS
AUTO LF
NS/VT cave NUM 048: 003 Bottom Overwrite
14:19:27, Wednesday, 31 October 2007

```
:
:
:
:
:do
:dbutil
HP30391C.07.10 TurboIMAGE/XL:  DBUTIL (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1987

>>show mscard passwords
   1 READER        2             3             4             5
   6               7             8             9            10
  11              12            13            14            15
  16              17            18 DO-ALL     19            20
  21              22            23            24            25
  26              27            28            29            30
  31              32            33            34            35
  36              37            38            39            40
  41              42            43            44            45
  46              47            48            49            50
  51              52            53            54            55
  56              57            58            59            60
  61              62            63

>>
```

Figure 34

3. Most users should select the highest numbered password shown.

C. SQL Functions

Using SQL Functions

Function must be entered in escaped ODBC syntax. The functions are entered in the form {fn function([parm_1[,parm_n]])}.

For example:

```
select LAST_NAME, FIRST_NAME,
{fn CONCAT(FIRST_NAME,{fn CONCAT(" ",LAST_NAME)})}
from CUSTOMER
where {fn LEFT(LAST_NAME,1)}="A"
```

SQL Function List

Function	Description

String Functions

ASCII	Returns the ASCII code value of the leftmost character of string_exp as an integer. {fn ASCII(string_exp)}
CHAR	Returns a character from the value of code (0 to 255). {fn CHAR(code)}
CHAR_LENGTH	Returns the length of the character string as an integer. {fn CHAR_LENGTH(string_exp)}
CHARACTER_LENGTH	Returns the length of the character string as an integer. {fn CHARACTER_LENGTH(string_exp)}

CONCAT	Returns a character string that consists of the two strings passed. {fn CONCAT(string_exp1, string_exp2)}
INSERT	Returns a character string where length characters have been deleted from string_exp1 beginning at start and where string_exp2 has been inserted into string_exp1, beginning at start. {fn INSERT(string_exp1, start, length, string_exp2)}
LCASE	Returns a string consisting only of lower case characters. {fn LCASE(string_exp)}
LEFT	Returns the number of characters requested from the left side of the given string. {fn LEFT(string_exp,count)}
LENGTH	Returns the length of the character string as an integer. {fn LENGTH(string_exp)}
LOCATE	Returns the position of a substring within a string as an integer. {fn LEFT (string_exp1,string_exp2[,start])}
LTRIM	Returns a character string except for any spaces on the left. {fn LTRIM(string_exp)}
OCTET_LENGTH	Returns the length in bytes of the value as an integer. {fn OCTET_LENGTH(string_exp)}
REPEAT	Returns string consisting of a given character the requested number of times. {fn REPEAT(string_exp,count)}
REPLACE	Search str_exp1 for occurrences of str_exp2 and replace with str_exp3. {fn LTRIM(str_exp1,str_exp2,str_exp3)}
RIGHT	Returns the rightmost count characters of string_exp. Returns the number of characters requested from the left side of the given string. {fn RIGHT(string_exp,count)}
RTRIM	Returns the characters of string_exp with trailing blanks removed. {fn LTRIM(string_exp)}
SPACE	Returns a character string consisting of count spaces. Returns the number of characters requested from the left side of the given string. {fn SPACE(count)}
SUBSTRING	Extracts one or more characters from a string. Returns the number of characters requested from the left side of the given string.

	{fn SUBSTRING(string_exp,start,length)}
UCASE	Converts strings to uppercase. Returns the number of characters requested from the left side of the given string. {fn UCASE(string_exp)}

Numeric Functions

ABS	Returns the absolute value of numeric_exp. {fn ABS(numeric_exp)}
CEILING	Returns the smallest integer greater than or equal to numeric_exp. The return value is of the same data type as the input parameter. {fn CEILING(numeric_exp)}
FLOOR	Rounds a number down to the nearest (smallest) whole number. {fn FLOOR(numeric_exp)}
MOD	Returns the remainder (modulus) of integer_exp1 divided by integer_exp2. {fn MOD(integer_exp1,integer_exp2)}
ROUND	Rounds a number (value1) down to the number of decimal digits specified in value2. {fn ROUND(value1,value2)}
SIGN	Returns a value indicating the sign of the provided value. {fn SIGN(value)}

Date Functions

CURRENT_DATE	Returns the current host system date. {fn CURRENT_DATE( )}
CURDATE	Returns the current host system date. {fn CURDATE( )}
DAYOFMONTH	Returns a number that consists of the Day portion of a given date. {fn DAYOFMONTH(date_exp)}
MONTH	Returns a number that consists of the Month portion of a given date. Returns a number that consists of the Day portion of a given date. {fn MONTH(date_exp)}
YEAR	Returns a number that consists of the Year portion of a given date. Returns a number that consists of the Day portion of a given date.

	{fn YEAR(date_exp)}

Misc Functions

IS_NULL	Returns true if the value is NULL. {fn IS_NULL(value)}
IS_NUMERIC	Returns true if the value represents a number. {fn IS_NUMERIC(value)}
COALESCE	Returns the first non-null value from the list provided. Used in joins that can return NULL values. {fn COALESCE(value1,value2)}

DECODE	Provides an IF THEN ELSE structure in the form,
	if (column==test)then value1 else value2.
	{fn DECODE(column,test,value1,value2)}

D. MSJOB on MPE

Controlling MSJOB

You may control MSJOB with either a Windows based application (Server Console) or from the MPE command line (MSJOBCMD).

Overview of MSJOB

MSJOB is used to contain the Minisoft Client/Server product server Listeners. Each product has one or more Listeners. These are controlled by parameters from a combination of SETVAR calls and indirect files.

Each Listener will start one Process for each connection established. Each Process has a unique PIN and is the son of the Listener that created it.

Parameters

MSTRACE

This can contain a number from zero (0) to seven (7). Each number gives sequentially more information about what is happening in the MSJOB processes.

```
!SETVAR MSTRACE 0
```

```
!SETVAR MSTRACE 7
```

Set to zero (0) to suppress all comments. A value of one (1) will output warnings. Use two (2) to display informational messages. Use three (3) to output both information and warnings. Detailed messages are printed when you use four (4). For a complete trace use seven (7).

Newer servers will accept server specific trace levels. For example;

```
!SETVAR MSTRACEDATASRVR 7
```

NOTE: Use of the ;STDLIST=??? in the MSSERVER###### variable will cause the trace data to be redirected to the specified file.

DEFAULT: No extra trace is generated.

MSTEMPFM

This parameter controls how middleman services manage MPE temporary files. This will primarily affect FrontMan and Custom servers. Your choices are to either share the MPE temporary file space among all processes in the same job or isolate the temporary file for each connection.

MSTEMPFM 0 temporary files are local to each process tree (connection). MSTEMPFM 1 will allow processes to share temporary files.

DEFAULT: Process local temporary files.

MSFILEEQ

Primarily controls the scope of MPE File Equations. Also controls the equivalent of logon UDCs.

!SETVAR MSFILEEQ 0

!SETVAR MSFILEEQ 1

!SETVAR MSFILEEQ 2

!SETVAR MSFILEEQ 3

When this value is set to zero or two, File Equations are Global in scope for the job.

When this value is set to one or three, File Equations are Local in scope. Each process in the job must set any needed file equations.

When this value is set to two or three, a command of MSFILEEQ is issued after the user is 'logged in'. The normal method of using this would be to have a command file or login UDC for each user that requires individual file equations.

DEFAULT: Zero (0), Global File Equations.

MSRUNPRI

Sets the priority the listening processes (MSSERVER and SERVER) will be running at. Each child process created for individual connections can be set at a different priority using the ;PRI=?? in the MSSERVER###### line.

!SETVAR MSRUNPRI "CS"

!SETVAR MSRUNPRI "DS"

!SETVAR MSRUNPRI "ES"

NOTE: This will override the job priority for MSSERVER and SERVER. As the two processes involved do not run for more than a fraction of a second each pass, no system performance degradation should be seen. Running these at too low a priority will result in connection timeout failures.

DEFAULT: Both MSSERVER and SERVER will run at the job priority.

MSSERVER######

The MSSERVER###### variable is used to specify: port number, maximum users, and RUN string for each Listener. Alternatively, MSSERVER###### can use an indirect file.

Each Listener can create Processes at a given priority.

```
!SETVAR MSSERVER000004 &
!  "30006 0 ODBCSRVR.MM.MINISOFT;PRI=DS S"&
!+"|31006 0 ODBCSRVR.MM.MINISOFT;PRI=CS S"
```

This will start two Listeners. Clients that connect to port 30006 will operate in the "D" queue. Clients that connect to port 31006 will operate in the "C" queue.

The second value will usually be zero (0). This value specifies the maximum number of concurrent connections on this port. Valid values are 0 to 255. Zero (0) indicates an indefinite number of connections.

Each connection can specify a different version of the server program (ODBCSRVR in this case). This is used while transitioning to a new version of the server when you can not change all the client software at the same time.

Indirect File

Each line of the file would contain the specification for connection. MSJOB would contain the following line:

```
!SETVAR MSSERVER000005 "^JDBC0001"
```

The file JDBC0001 can contain the following lines:

```
30007 0 JDBCSRVR.MM.MINISOFT;PRI=DS S
31007 0 JDBC2171.MM.MINISOFT;STDLIST=TRACEFIL;PRI=DS S
```

Control

Start MSJOB by streaming the file MSJOB.MM.MINISOFT.

MSJOB may be controlled by either Server Console or MSJOBCMD. These utilities allow you to view into and control MSJOB, the Listeners, and the Processes.

Sample

```
!JOB MSJOB,MGR.MINISOFT,MM;OUTCLASS=LP,1
!COMMENT
!COMMENT For use with MSSERVER 2.3.1.5 or later
!COMMENT For use with SERVER 2.3.1.4 or later
!COMMENT
!COMMENT MSTRACE 1 to output warnings
!COMMENT MSTRACE 3 to output information and warnings
!COMMENT MSTRACE 7 to output detailed trace
!COMMENT
!COMMENT MSTEMPFM 0 temp files are local to each process.
!COMMENT MSTEMPFM 1 will allow processes to share temp files.
!COMMENT
!COMMENT MSFILEEQ 0 File equations are job global
!COMMENT MSFILEEQ 1 File equations are process local
!COMMENT MSFILEEQ 2 File equations are job global (XEQ MSFILEEQ)
!COMMENT MSFILEEQ 3 File equations are process local (XEQ MSFILEEQ)
!COMMENT
!COMMENT MSRUNPRI sets the priority of the listener processes
!COMMENT Use the ;PRI= option for each server
!COMMENT
!SETVAR MSTEMPFM 0
!SETVAR MSFILEEQ 0
!SETVAR MSRUNPRI "CS"
```

```
!SETVAR TZ,"PST8PDT "
!COMMENT
!SETVAR MSSERVER000002 "30001 0 NFTSRVR.MM.MINISOFT;PRI=DS;INFO=' S'"
!SETVAR MSSERVER000004 "30006 0 ODBCSRVR.MM.MINISOFT;PRI=DS;INFO=' S'"
!SETVAR MSSERVER000005 "30007 0 JDBCSRVR.MM.MINISOFT;PRI=DS;INFO=' S'"
!SETVAR MSSERVER000008 "30002 0 DATASRVR.MM.MINISOFT;PRI=DS;INFO=' S'"
!RUN MSSERVER.MM.MINISOFT;INFO="30000 SERVER.MM.MINISOFT"
!EOJ
```

Server Console

The Minisoft server console application can be used to monitor and control the background MSJOB. You can view a list of the current users and optionally stop individual processes or the job.

The logon must be from a valid OP capable user in order to take full advantage of the console features.

The operator can selectively start or stop any licensed Client Server product. The operator may also stop any individual user process without affecting other users.

Operation

Enter the Host name or IP address. By default, MSSERVER listens on port 30000. You must supply a valid user with OP capability.

The top window will show the waiting servers.

After selecting the server you wish to view; the bottom window displays the active sessions including pin number, program executing, logon user, and cumulative CPU time for the process.

You can select a single server or process to stop. By selecting a server, all the processes that have started under it will be stopped

MSJOBCMD

You must have SM or OP capability to use this utility.

The 'loopback' feature of you network interface must be started. Use "NETCONTROL START;NET=LOOP"

This application is designed to run from a JOB or SESSION on the HP e3000 that is running the Minisoft Client Server listener (MSJOB). The purpose of this utility is to display the current users of MSJOB and to optionally stop the MSJOB.

Options

PARM=0

Display current listeners and connections.

PARM=1

Display current listeners and connections. If no current connections, then stop the job or listener.

PARM=2

Display current listeners and connections. Stop the job or listener regardless of the users.

PARM=3

Display current listeners and connections.

PARM=4

Start a listener using the INFO parameter. Use a format similar to the MSJOB indirect file.

INFO="5|30007|0|JDBCSRVR.MM.MINISOFT;PRI=DS S"

PARM=5

Kill all processes of a listener, leave the listener active.

INFO=(port number)

Port number to use for above commands. If not given, then PARM values apply to job.

Details

RUN MSJOBCMD;PARM=0 Will display the current users.

RUN MSJOBCMD;PARM=1 Will display the current users and optionally stop the job if the number of users is zero.

RUN MSJOBCMD;PARM=2 Will display the current users and stop the job regardless of the number of users.

RUN MSJOBCMD;PARM=2;INFO="30006" Will display the current users for the listener on port 30006 and stop the it regardless of the number of users.

Output JCWs

```
MSJOBUSERCOUNT
```

```
MSJOBSHUTDOWN
```

MSJOBUSERCOUNT is used to display the total number of active server connections on this server. The value will be -1 if there is an error.

MSJOBSHUTDOWN will be set to TRUE if a shutdown command is sent to the JOB. Otherwise it will be set to FALSE.

Input JCWs

```
MSJOBPORT
```

```
MSTIMEOUT
```

```
MSTRACE
```

Set MSJOBPORT to the value of your MSSERVER port. The default value is 30000.

Set MSTIMEOUT to the number of seconds to wait for a connection. The default is 60.

Set MSTRACE to 7 to display the version number. The default is zero.

IP Security

Overview

New in version 2.3.1.7 of SERVER.MM.MINISOFT is the ability to block connection based upon IP address.

Parameters

One MPE file per port number to be filtered. If the file exists block all connections except those included. Local Loop address is always accepted.
(127.0.0.1:255.255.255.255)

Sample

For a server configured to listen on port 30001:

```
:print ip30001
209.42.12.0:255.255.255.128
209.43.231.0:255.255.255.0
209.44.0.0:255.255.0.0
```

This would permit connections from:

209.42.12.1 to 209.42.12.127
209.43.231.1 to 209.43.231.254
209.44.0.1 to 209.44.255.254

Index

FINI