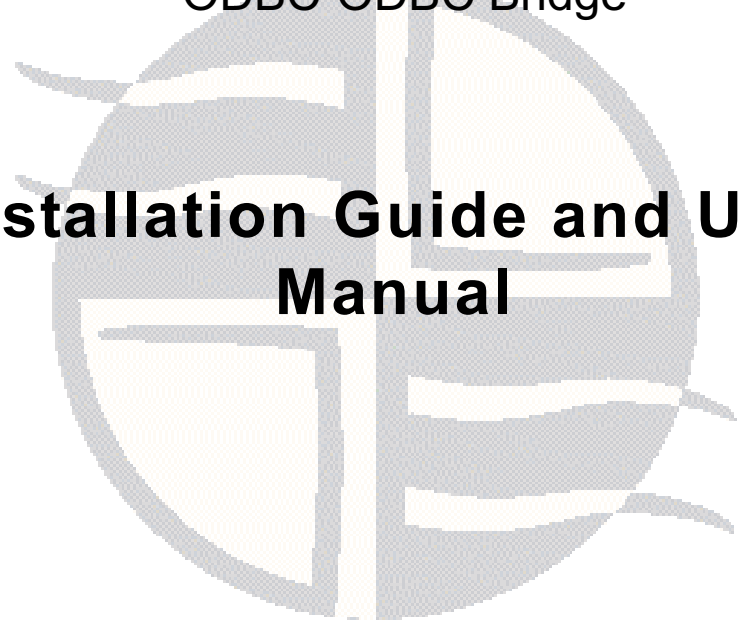
Easysoft Data Access

ODBC-ODBC Bridge



Installation Guide and User Manual

Version 26.

This manual documents version 2.6.n of the Easysoft ODBC-ODBC Bridge.

Publisher: Easysoft Limited

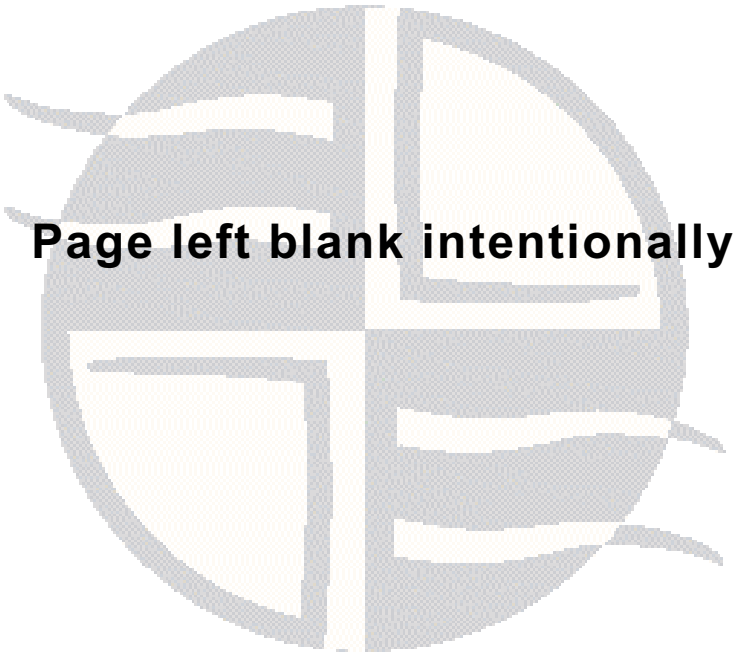
Thorp Arch Grange

Thorp Arch

Wetherby

LS23 7BA

United Kingdom



Page left blank intentionally

PREFACE

About this manual

This manual is intended to cover the full range of requirements for anyone wishing to install, use, or configure the Easysoft ODBC-ODBC Bridge (OOB). Supplementary information is provided for those wishing to build ODBC applications that link through a driver manager, but please note that this manual is not an ODBC programming manual.

Chapter Guide

Intended Audience

Sections written for the Microsoft Windows platforms require some familiarity with the use of buttons, menus, icons and text boxes, but should present no difficulties if you have any experience of Apple Macintosh computers, Microsoft Windows or the X Window System.

The Unix-based sections require experience of using a Unix shell and basic functions like editing a file. More complex activities are detailed more clearly, but it helps to understand how your system handles dynamic linking of shared objects.

NB

Several technical documents are installed in addition to this manual, including a detailed list of Frequently Asked Questions (`FAQ.txt`), installation, configuration and interfacing information. These are located in `<InstallDir>/easysoft/oob/doc` under Unix and `<InstallDir>\Docs` under Windows. Various tutorials are installed in subdirectories of this directory. For example, there's a subdirectory containing Perl tutorials. Please check all documentation thoroughly before contacting Easysoft with a query.

Displaying the Manual

This manual is available in the following formats:

- Portable Document Format (PDF), which can be displayed and printed using the Acrobat Reader, available free from Adobe at **http://www.adobe.com**.
- HTML (the format Easysoft recommend for viewing on screen).

Notational Conventions

Across the range of Easysoft manuals you will encounter passages that are emphasized with a box and a label.

A *note box* provides additional information that may further your understanding of a particular procedure or piece of information relating to a particular section of this manual:

NB	Note boxes often highlight information that you may need to be aware of when using a particular feature.

A *reference box* refers to resources external to the manual, such as a useful web site or suggested reading:

REF	For more manuals that use this convention, see the rest of the Easysoft documentation.

A *platform note* provides platform-specific information for a particular procedure step:

Linux	In Linux you must log on as the `root` user in order to make many important changes.

A *caution box* is used to provide important information that you should check and understand, prior to starting a particular procedure or reading a particular section of this manual:

Caution!	Be sure to pay attention to these paragraphs because Caution boxes are important!

Information has also been grouped within some chapters into two broad classes of operating system, Windows and Unix, for which side tabs are used to help you turn to the section relevant to you.

Typographical Conventions

To avoid ambiguity, typographic effects have been applied to certain types of reference:

- User interface components such as icon names, menu names, buttons and selections are presented in bold, for example:

  Click **Next** to continue.

  Where there is a chain of submenus, the following convention is used:

  Choose **Start > Programs > Command Prompt**.

- Commands to be typed are presented using a `monotype` font, for example:

  At the command prompt type `admin`.

- Keyboard Commands

  It is assumed that all typed commands will be committed by pressing the <*Enter*> key, and as such this will not normally be indicated in this manual. Other key presses are italicized and enclosed by angle brackets, for example:

  Press <*F1*> for help.

- File listings and system names (such as file names, directories and database fields) are presented using the `monotype` plain text style.

Contents

Trademarks

Throughout this manual, *Windows* refers generically to Microsoft Windows 2000, XP, 2003, Vista, 2008, 2008 R2, 7, 8 or 2012, which are trademarks of the Microsoft Corporation. The X Window system is specifically excluded from this and is referred to as *The X Window System* or just *X*.

Note also that although the name UNIX is a registered trademark of The Open Group, the term has come to encompass a whole range of UNIX-like operating systems, including the free, public Linux and even the proprietary Solaris. Easysoft use Unix (note the case) as a general term covering the wide range of Open and proprietary operating systems commonly understood to be Unix ‘flavors’.

Mac OS is a trademark of Apple Computer, Inc., registered in the U.S. and other countries.

Easysoft and Easysoft Data Access are trademarks of Easysoft Limited.

CHAPTER 1 INTRODUCTION

Introducing the Easysoft ODBC-ODBC Bridge

Easysoft Data Access is a suite of programs that add significant value to your investment in ODBC. With Easysoft software you can connect applications on more platforms to more database systems than ever before.

The Easysoft ODBC-ODBC Bridge (OOB) enables ODBC calls to be made across the network from any Windows or Unix system to an ODBC data source running under Windows or Unix, permitting heterogeneous client-server operation and allowing client access to data wherever it is stored.

Please note that if you want to connect to local data sources from a remote Java application, you will need the Easysoft JDBC-ODBC Bridge *instead of* this software.

Chapter Guide

Why ODBC?

Historically, corporate data was held on large, centralized computing resources that performed all the processing required on it. Changes to the business practice meant changes had to be made to the corporate mainframe system. Worse still was the problem of integrating two or more of these highly individual systems, for example in the event of a corporate merger.

As the desktop computer improved in power, users began to want to access corporate data in order to process it on their own desktop. The client-server architecture became a popular goal: the central computing resource (server) would produce a subset of its data for a user-friendly tool (client). The client would use desktop computing power to format and present the data.

Database application writers and their customers found themselves with a key problem: it was necessary to produce one version of a piece of software for each DataBase Management System (DBMS) they wished to use it with. Relational databases and SQL went part way toward alleviating the problem and for the first time there was a defined, open, standard language for querying databases.

In theory at least, it was possible to use the same language in dealing with databases from a variety of manufacturers and the X/Open consortium went on to define a Call Level Interface (CLI) so that programmers could effectively use SQL within their own programs.

ODBC is an API definition, compliant with ANSI SQL and X/Open's SQL CLI, which allows an application to be written without considering the intricacies of the particular database engine to which it connects.

An ODBC *driver* takes care of all the database-specific code, if necessary transforming the structure of the underlying system into a relational framework.

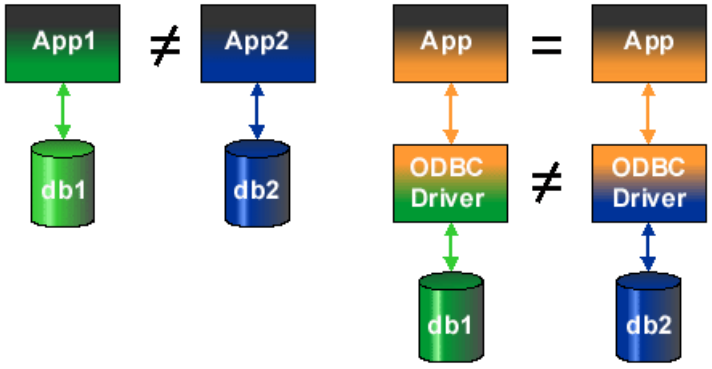


Figure 1: Before and after ODBC

Figure 1 on page 17 illustrates the principle of separating the driver from the application.

The left-hand side of the diagram shows how before ODBC, even if `App1` and `App2` were functionally equivalent, two programs were required, one for each DBMS.

The right-hand side of the diagram shows how ODBC permits the DBMS-specific parts of a program to be separated from the part that fulfils the functional requirement, enabling the completed application to be attached to any DBMS that has a corresponding driver.

Driver Managers

The barest ODBC system would include an ODBC-conformant driver accessing some data, and an ODBC-conformant application, linked to the driver library.

If commercial applications were distributed in this way, users would need to re-link their applications to their chosen driver whenever they wanted to access a different data source.

Instead, the application program is linked to a *driver manager*, which loads the required driver at runtime.

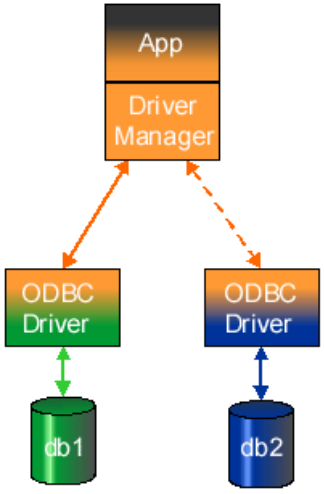


Figure 2: The Driver Manager as a dynamic linker

This approach provides three key results:

- Once developers have written applications to satisfy a business requirement, the application can be ‘plugged in’ to whatever database management system satisfies the technical demands.
- Administrators can connect a variety of applications (such as generic query tools) to their databases to browse and investigate the data.
- *Data access middleware* can be inserted between the ODBC application and driver to add strategic functionality such as joining heterogeneous databases into one data source or bridging a network.

Why use the Easysoft ODBC-ODBC Bridge?

The Easysoft ODBC-ODBC Bridge is data access middleware that allows an application running on one platform to access an ODBC data source on another platform.

An Example Business Requirement

A sales manager keeps the company sales figures in an Excel spreadsheet on his networked Windows NT PC. He enjoys the consistent user interface and flexibility of the Windows system.

The company also has an intranet running on an Apache server under Linux, which displays the latest sales figures and updates them daily.

The system administrator values the round-the-clock stability of the Linux platform and the ability to freely tailor and tune the server software.

At present the sales manager emails the latest figures to the web site administrator at the end of each day and then the web site administrator updates the intranet.

A Solution using the Easysoft ODBC-ODBC Bridge

The Easysoft ODBC-ODBC Bridge allows a script on the web server to run a SQL select query directly on the NT machine, removing the need for regular human intervention.

A small script along with a simple client program submits SQL to the Easysoft ODBC-ODBC Bridge, which passes it across to the spreadsheet software on the NT machine. The script is set up to trigger automatically at the end of the day.

Where the Easysoft ODBC-ODBC Bridge fits in

The Easysoft ODBC-ODBC Bridge comes in two parts, a client and a server.

The OOB Client appears to the ODBC application just as any other ODBC driver, and the OOB Server connects to the database engine as an ODBC application.

This terminology may seem a little confusing at first, so it is important to consider each element step by step.

First, **Figure 3 on page 21** shows the typical ODBC set up. The application ("`App`") connects through a driver manager to an ODBC driver, which interfaces to the DBMS. The interface between the driver manager and the driver is defined by the ODBC API.

The Easysoft ODBC-ODBC Bridge can be seen as a black box that takes ODBC calls in at the top and passes them out again at the bottom.

This box can be inserted at *A* and as far as the application or the database are concerned, nothing has changed:

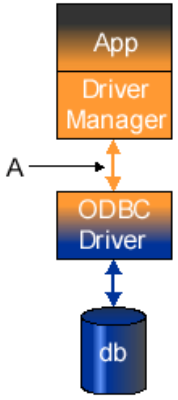


Figure 3: The typical ODBC configuration

What *has* changed is that now the driver manager, loads the OOB Client *and* the application and database no longer need to be on the same machine or software platform.

NB	Connecting an ODBC application to a networked database is not new. For example, Microsoft SQL Server allows ODBC client applications to connect to a remote SQL Server data source. With the Easysoft ODBC-ODBC Bridge, however, the database can be any ODBC data source and the client may be on Windows, Linux, Solaris or any other supported platform.

Inside this 'black box' is a client-server pair that handles all the network operations, passing function references and parameters one way and return values the other:

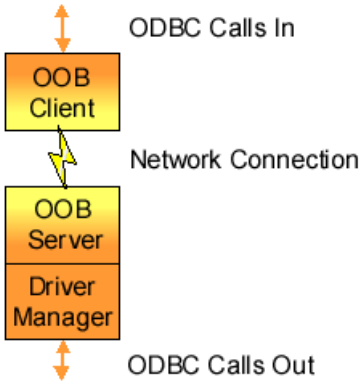


Figure 4: The Easysoft ODBC-ODBC Bridge

Figure 5 on page 23 shows the system set up across a network.

Comparing it with the model in **Figure 3 on page 21**, the database is replaced by a network connection on the client side and the OOB *Server* replaces the *ODBC application* on the server side.

From **Figure 5 on page 23** you can see that:

- to the application the OOB Client is a standard ODBC driver
- on the server the OOB Server is a standard ODBC application

NB	The OOB Server does not have to be on the same machine as your database if you have an ODBC driver for your database available on another machine.

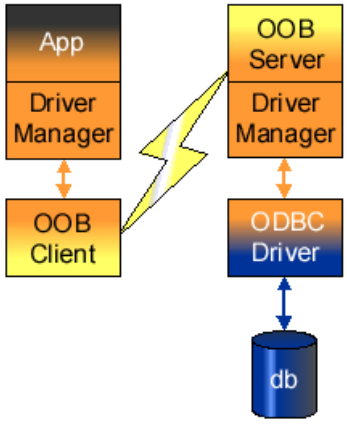
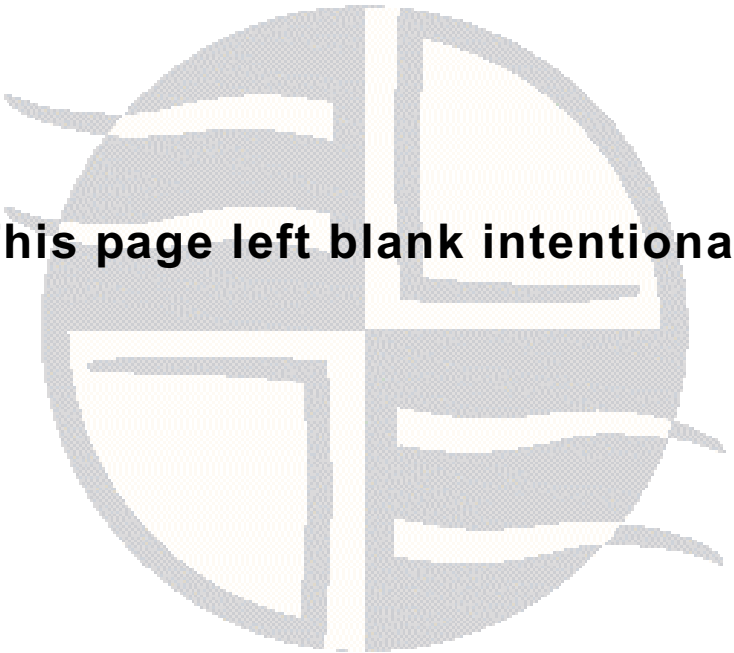


Figure 5: Bridging the network with the Easysoft ODBC-ODBC Bridge

REF	The definitive SQL CLI document is the **Open Group CAE Specification C451, ISBN 1-85912-081-4** (**http://www.opengroup.org/pubs/catalog/c451.htm**). The **Microsoft ODBC 3.0 Programmer's Reference, ISBN 1-57231-516-4** explains ODBC usage in some detail.



This page left blank intentionally

CHAPTER 2 INSTALLATION

Installing the Easysoft ODBC-ODBC Bridge

This section explains how to install, license and remove the Easysoft ODBC-ODBC Bridge (OOB) on supported Windows, Mac OS X and Unix platforms.

The Windows and Mac OS X installation can be carried out by anyone with local administrator privileges for the target machine.

The Unix installation assumes you are, or have available for consultation, a system administrator.

Chapter Guide

Obtaining the Easysoft ODBC-ODBC Bridge

There are three ways to obtain the Easysoft ODBC-ODBC Bridge:

- The Easysoft web site is available 24 hours a day at **http://www.easysoft.com** for downloads of definitive releases and documentation.

  Select **Download** from the Easysoft ODBC-ODBC Bridge section of the web site and then choose the platform release that you require.

  First time visitors must complete the new user form and click **Register**. Note that your personal Internet options may require you to login and click **Continue** if you have previously registered.

- The Easysoft FTP server is available 24 hours a day at **ftp://ftp.easysoft.com**, containing free patches, upgrades, documentation and beta releases of Easysoft products, as well as definitive releases.

  Change to the `pub/odbc-odbc-bridge` directory and then choose the platform release that you require.

- You can order Easysoft software on CD by email, telephone or post (see **Contact Details**).

What to install

The Easysoft ODBC-ODBC Bridge consists of the OOB Client and the OOB Server.

There are shared files which are needed to install either of these components and there are documentation and example files which you are recommended to keep for future reference.

Unix distributions also include the third-party unixODBC driver manager (see **"unixODBC" on page 243**).

As the OOB Client and the OOB Server will typically be installed on different machines with different operating systems, you will need to perform the basic installation procedure twice, probably in different environments. You should first decide which part of the Easysoft ODBC-ODBC Bridge you are installing on which machine.

It is also important to consider that it may be necessary for Easysoft to make a change to the protocol used between the client and server. The following table shows which versions are guaranteed compatible with a 'Y' and those versions which may not work together with a 'N'. The 'n' is the major number and the 'm' is the minor number as in an OOB release n.m.b (b, build numbers are not relevant).

	Server version		
Client version	n.m	n.m+1	n+1.m
n.m	Y	Y	N
n.m+1	N	Y	N
n+1.m	N	N	Y

Figure 6: Client Server version compatability.

For example:

- Client version 1.1 will work with Server version 1.1, 1.2, 1.3 ...
- Client version 1.1 might not work with Server version 2.0, 2.1...
- Client version 1.4 might not work with Server version 1.3.

NB Client version 1.0 will work with all Servers up to version 2.0 even though the table indicates otherwise.

Only the first two fields constitute the version number. The last digit is known as ‘build number’ and does not affect compatibility.

NB If you connect a Client and Server with mismatched protocols, the Easysoft ODBC-ODBC Bridge will detect and report this immediately.

The name of the Easysoft ODBC-ODBC Bridge distribution file varies from platform to platform, but is of the form:

- `odbc-odbc-bridge-x_y_z-platform.exe` (Windows)

– OR –

- `odbc-odbc-bridge-x.y.z-platform.tar.gz` (Unix)

– OR –

- `odbc-odbc-bridge-x.y.z-platform.dmg` (Mac OS X)

where "*x*" is the major version number, "*y*" is the minor version number and "*z*" is the build index.

"*platform*" will vary depending on the operating system distribution you require and you may come across files of the form:

- `odbc-odbc-bridge-x.y.z-platform-variation.tar`

within specific Unix platforms, where "*platform-variation*" refers to alternative versions available for a single platform (e.g. `-mt` for thread-safe versions).

NB	Select the highest release available for your platform within your licensed major version number (installing software of a different major version number requires a new Easysoft license).

Unix filenames may also be suffixed with `.gz` for a "`gzipped`" archive, `.bz2` for a "`bzip2ed`" archive, or `.Z` for a "`compressed`" archive.

NB	If you download a Unix file using Windows, the browser may change the filename. For example, if you download a `.gz` file and Windows changes the filename, it may not be obvious that the file is "`gzipped`". Use "`file` *filename*" to find out the file type of the downloaded file.

You can now download a file and begin the installation process.

Caution!

On Unix, as long as you stop any software either from Easysoft or using Easysoft drivers, it is safe to reinstall or upgrade the Easysoft ODBC-ODBC Bridge without uninstalling.

If you do uninstall, you should first back up any configuration data that you still need, as uninstalling some Easysoft products will result in this information being deleted (license details remain in place).

NB

If you are upgrading an OOB Server, you will need a new license from Easysoft to use the new Server. New licenses are available on request to customers with a support contract. For more information, contact **license@easysoft.com**.

Refer to the section relevant to your platform to continue:

- **"Installing on Windows" on page 31**
- **"Uninstalling on Windows" on page 43**
- **"Installing on Unix" on page 44**
- **"Uninstalling on Unix" on page 74**
- **"Installing on Mac OS X" on page 81**

Installing on Windows

- Execute the file distribution that you downloaded in **"Obtaining the Easysoft ODBC-ODBC Bridge" on page 26**.

Follow the on screen instructions.

64-bit Windows

64-bit Windows machines support both 32-bit and 64-bit ODBC drivers. A 32-bit application must be used with a 32-bit ODBC driver. A 64-bit application must be used with a 64-bit ODBC driver.

If your Windows application is 32-bit, you need to use it with a 32-bit OOB Client. If your Windows application is 64-bit, you need to use it with a 64-bit OOB Client.

If your target Windows ODBC driver is 32-bit, you need to use it with a 32-bit OOB Server. If your target Windows ODBC driver is 64-bit, you need to use it with a 64-bit OOB Server.

If you choose the **Typical** or **Complete** Setup types, the OOB installation program will install:

Both a 32-bit and a 64-bit OOB Client.

Both a 32-bit and a 64-bit OOB Server.

The **Custom** Setup type allows you to choose whether to install just the 32-bit OOB Client and Server or just the 64-bit OOB Client and Server. (You can also choose whether to install just the OOB Client and or just the OOB Server.)

When prompted to choose a Setup type, do one of the following:

- Select **Typical** or **Complete** to install the OOB Client and Server, documentation, tutorials and examples (all components).

  The OOB Client is required if you need to access remote ODBC data sources from this machine. The OOB Server is required if you need to access ODBC data sources on this machine from remote machines.

- Select **Custom** if you want to choose which components to install or choose where the the software is installed.

The Custom Setup dialog box has two panes. The left pane shows a tree view of the features that you can install and you can expand a feature to view its subfeatures. The right pane displays feature descriptions. When you click a feature or subfeature, the Feature description pane displays a description of the selection and its disk space requirements.

To add or remove a feature, click the arrow next to the feature name, and then choose one of the following options from the drop-down list:

- **Will be installed on local hard drive** Install the selected feature in the location shown under **Location**.
- **Entire feature will be installed on local hard drive** Install the selected feature and any subfeatures.
- **Entire feature will be unavailable** Do not install the selected feature and any subfeatures.

The default installation directory is `*drive*:\Program Files\Easysoft\Easysoft ODBC-ODBC Bridge`. To change the installation directory, click **Browse**.

To restore the default list of features to install, click **Reset**. To check that you have enough disk space to install the features you want, click **Disk Usage**.

OOB Server Configuration Settings

If you chose to install the OOB Server, the **Initial Server Configuration** dialog box is displayed:

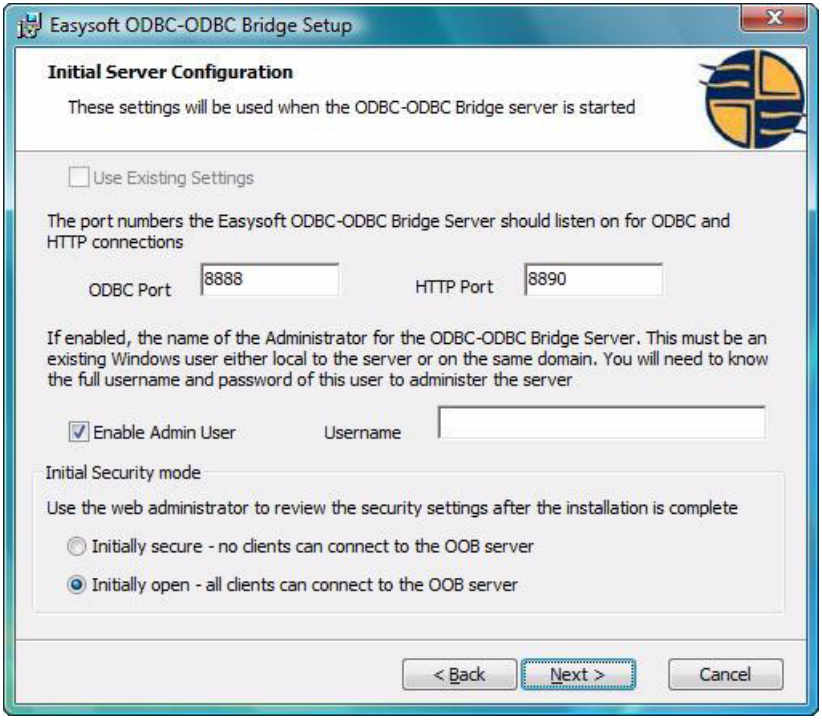


Figure 7: The Initial Server Configuration dialog box

If you are upgrading and want to use your existing OOB Server configuration settings, click **Use Existing Settings**. If you are not upgrading, **Use Existing Settings** is unavailable.

64-bit Windows

If you chose the **Custom** Setup type, the Initial Server Configuration dialog box is not displayed, refer to the note in **"Installation" on page 41** for the implications of this.

ODBC Port defines the port where the OOB Server will listen for OOB Clients. **HTTP Port** defines the port on which the OOB Server Web Administrator will listen for HTTP requests. Accept the default values unless you have a port conflict (see **"Choosing another Port or Service Name" on page 236**).

In the **Username** box, enter the login name of an existing Windows user account as the Server Administrator user name.

You also need to know the password for this user, because it will be required when you run the Web Administrator (see **"The Web Administrator" on page 197**).

To grant group access to the Web Administrator, you may wish to create a specific "`Administrator`" user.

To allow anyone to have access to the Web Administrator, leave the **Username** field blank or click to clear **Enable Admin User** to remove the requirement to log in.

Caution!

You should consider carefully whether you wish to allow *anyone* on your network to have access to the Web Administrator in order to change Easysoft ODBC-ODBC Bridge settings.

In the **Initial Security mode** section, the default value, **Initially open**, will allow any OOB client to access the OOB Server.

The **Initially secure** option places an asterisk on the **Denied Access** section of the Web Administrator **Security** page, which will cause all OOB Client connections to be rejected.

This should be removed after installation and the clients that require access to the OOB Server specified individually (see **"The Security Screen" on page 220**).

Updating Files That are in Use

To avoid rebooting your computer, the OOB installer prompts you when files that it needs to update are in use by another application or service. This frees the locked files and allows the installation to complete without a system restart.

On Windows Vista and later, the OOB installer uses the Restart Manager to locate the applications that are using files that need updating. These applications are displayed in the Files in Use dialog box. To avoid a system restart, choose **Automatically close applications and attempt to restart them after setup is complete**. The OOB installer then uses the Restart Manager to try to stop and restart each application or service in the list. If possible, the Restart Manager restores applications to the same state and with the same data that they were in before it shut them down.

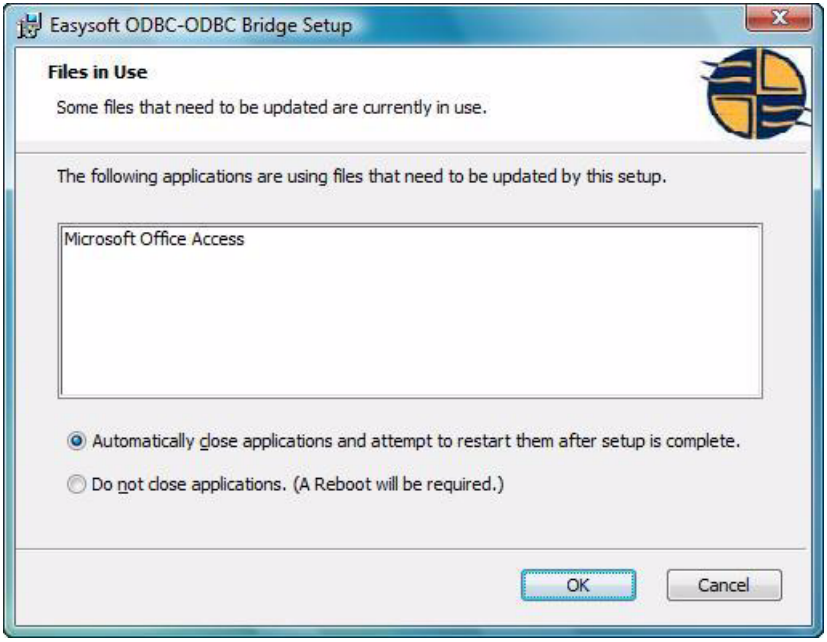


Figure 8: The Files in Use dialog box

On earlier versions of Windows, when the Files in Use dialog is displayed, manually shut down each application in the list and then click **Retry** to avoid a system restart.

LICENSING THE OOB SERVER

Only the OOB Server needs to be licensed.

By default, the install program starts the Easysoft License Manager (documented in the **Licensing Guide**), because you cannot use the OOB Server until a license is obtained.

The following types of license are available:

- A *free time-limited trial license*, which gives you free and unrestricted use of the product for a limited period (usually 14 days).
- A *full license* if you have purchased the product. On purchasing the product you are given an authorization code, which you use to obtain a license.

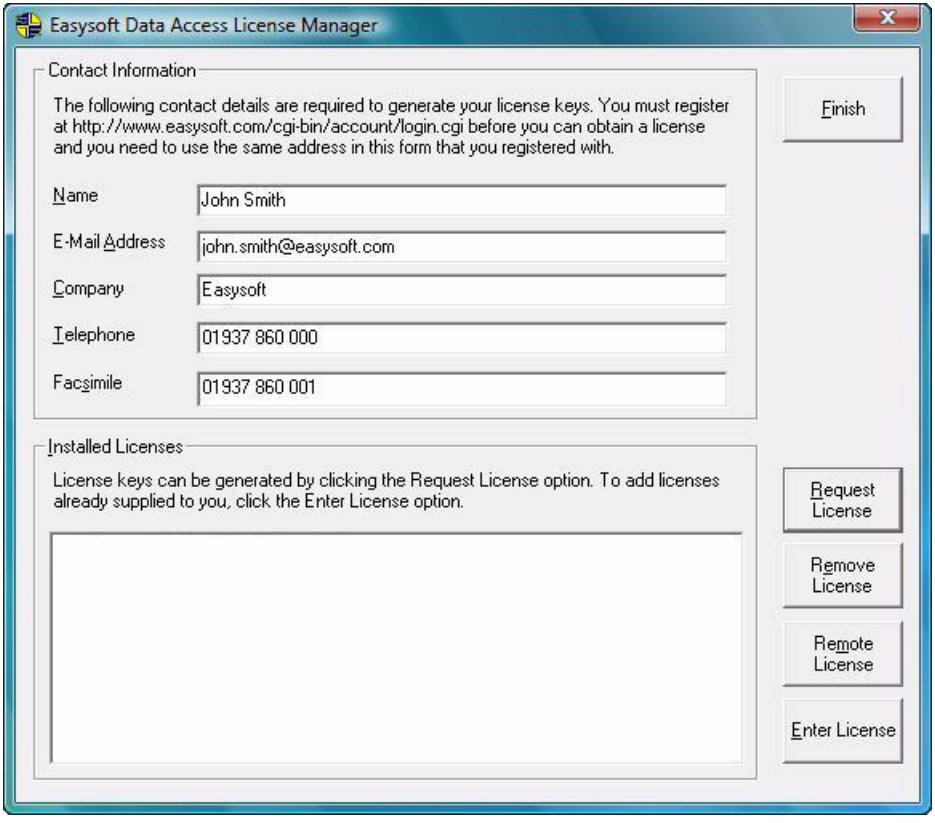


Figure 9: The License Manager window

1. Enter your contact details.

   You MUST enter the **Name**, **E-Mail Address** and **Company** fields. The e-mail address that you enter here must be the same as the one that you registered with.

   The **Telephone** and **Facsimile** fields are important if you require Easysoft to contact you by those methods.

2. Click **Request License**.

   You are asked for a license type.

3. For a trial license click **Time Limited Trial** and then click **Next**.

   The License Manager asks what software you are licensing.

   Select your required version of the Easysoft ODBC-ODBC Bridge from the drop-down list and then click **Next**.

   – OR –

   If you have obtained an authorization code for a purchased license, select **Non-expiring License** and then click **Next**.

   The License Manager requests your authorization code.

   Enter the authorization code and then click **Next**.

   The License Manager displays a summary of the information you entered and allows you to choose how to apply for your license.

4. Choose **On-line Request** if your machine is connected to the internet and can make make outgoing connections on port 8884.

The License Manager then sends a request to the Easysoft license server to activate your license key automatically. This is the quickest method and results in your details being entered immediately into our support database.

NB	Only your license request identifier and contact details as they are displayed in the main License Manager screen are sent to Easysoft.

The remaining three options (**Email Request**, **Print Request** and **View Request**) are all ways to obtain a license if your machine is offline (i.e. does not have a connection to the internet).

Each of these methods involves providing Easysoft with information including your machine number (a number unique to your machine) and then waiting to receive your license key.

Instead of emailing, faxing or telephoning your details to Easysoft, you can enter them directly at the Easysoft web site and your license key will be emailed to you automatically.

To use this method, click **View Request**, and then visit:

- **http://www.easysoft.com/support/licensing/trial_license.html**
  (trial licenses)
- **http://www.easysoft.com/support/licensing/full_license.html**
  (purchased licenses)

In the Licensing page, enter your machine number (and authorization code for purchased license), click **Submit** and your license key will be emailed to you.

NB	You can copy your machine number from the **View Request** dialog box using `CTRL-C` and then paste it into the License Generator by using `CTRL-V`.

When you receive the license key, you can activate it either by double-clicking the email attachment or by clicking **Enter License** on the License Manager main screen and pasting the license key into the dialog box.

A message tells you how many licenses have been added.

NB	If you use the **Email Request** option, the license key is emailed to the email address as displayed on the License Manager screen, not the `from:` address of your email.

For more information about the licensing procedure refer to the **Licensing Guide**.

NB	If you add a new license, the OOB Server service needs to be restarted in order to access the new details. During Setup, the OOB Server service is restarted automatically after a license has been added. If you add a license outside of Setup, you need to restart the service manually. For more information about restarting the OOB Server service, see **"To Start or Stop the Service in Windows" on page 227**.

5. Click **Finish** to quit the License Manager.

POST INSTALLATION

You should have an Easysoft program group with links to additional documentation, the product newsgroup and the Web Administrator.

64-bit Windows

Unless you specified otherwise with the **Custom** Setup type, the OOB installation program installs both a 32-bit and a 64-bit OOB Server. By default, the 64-bit OOB Server is not started automatically. If your target Windows ODBC driver is 64-bit, you need to use the 64-bit OOB Server. Refer to the note in **"Configuration" on page 228** for instructions on how to start 64-bit OOB Server.

If you chose the **Custom** Setup type, you will need to specify an `HTTPAdmin` user, before you can make changes to the OOB server via the Web Administrator. To do this refer to the note in **"Configuration" on page 198**.

CHANGING OR REPAIRING THE OOB INSTALLATION

The OOB installer lets you add or remove OOB components. You can add or remove the OOB Server, OOB Client or OOB documentation.

The installer can also repair a broken OOB installation. For example, you can use the OOB installer to restore missing OOB files or registry keys.

1. Do one of the following:
   - In Windows Vista and later, in **Control Panel**, open **Programs and Features**.
   - In earlier versions of Windows, in **Control Panel** open **Add or Remove Programs**.

2. Do one of the following:

   - In Windows Vista and later, right-click **Easysoft ODBC-ODBC Bridge**, and then click **Change** or **Repair**.
   - In previous of Windows, select **Easysoft ODBC-ODBC Bridge** and click **Change/Remove**.

OOB Demonstration Program

Easysoft provide a small client program which can demonstrate the OOB Client in action by connecting across the internet to the server `demo.easysoft.com`. Refer to **"The Demo.exe Client" on page 150** for more details.

Uninstalling on Windows

This section explains how to remove the Easysoft ODBC-ODBC Bridge from your system.

Caution! If you decide to remove a previous installation before upgrading to a new version a large amount of existing OOB Server configuration data may be deleted (machine access control details, for example).

1. Do one of the following:
   - In Windows Vista and later, in **Control Panel**, open **Programs and Features**.
   - In earlier versions of Windows, in **Control Panel** open **Add or Remove Programs**.
2. Do one of the following:
   - In Windows Vista and later, double-click **Easysoft ODBC-ODBC Bridge**.
   - In earlier of Windows, select **Easysoft ODBC-ODBC Bridge** and click **Change/Remove**.

   The uninstall process is complete.

   Any licenses you obtained for the Easysoft ODBC-ODBC Bridge and other Easysoft products are held in the Windows registry.

   When you uninstall, your licenses are not removed so you do not need to relicense the product if you reinstall or upgrade.

Installing on Unix

These instructions show how to install the Easysoft ODBC-ODBC Bridge on Unix platforms.

BEFORE YOU INSTALL

Requirements

The installation script has a minimal set of requirements:

- Bourne shell in `/bin/sh` (if your Bourne shell is not located there you may need to edit the first line of the install file).
- Various commonly used UNIX commands such as:

  `grep`, `awk`, `test`, `cut`, `ps`, `sed`, `cat`, `wc`, `uname`, `tr`, `find`, `echo`, `sum`, `head`, `tee`, `id`

  If you are missing any of these commands they can generally be obtained from the Free Software Foundation (**http://www.fsf.org**). As some machines have a broken `tee` command, the distribution comes with a `tee` replacement.
- Depending on the platform, you will need up to 22Mb of disk space free for the installed programs and up to 22Mb temporary space of the installation files themselves. If you install the unixODBC Driver Manager as well, these numbers increase by approximately 1.5Mb. Distributions not containing a GUI setup dialogue and QT are significantly smaller (usually 10Mb less).

- For Easysoft Licensing to work you must either install in `/usr/local/easysoft` or symbolically link `/usr/local/easysoft` to wherever you chose to install the software. The installation will do this automatically for you as long as you run the installation as someone with permission to create `/usr/local/easysoft`.
- An ODBC Driver Manager. OOB distributions contain the unixODBC Driver Manager but you can use an already installed unixODBC if you prefer.
- If you are only installing the client, you do not have to be the `root` user. However, if you are not `root`, it will not be possible to register the OOB Client with unixODBC, install sample SYSTEM DSNs or update the dynamic linker entries (only some platforms). If you are not `root` this will have to be done manually later. You must be `root` to install the server properly.

  Easysoft recommend you install all OOB components as the `root` user.

What you can install

This OOB distribution contains:

- the OOB Client
- the OOB Server
- the unixODBC Driver Manager (except on MAC OS X)

You may install the Client, the Server or both. However, most installations will install the Client on one machine and the Server on another machine. If you only want to access ODBC databases through drivers on remote machines from this machine, you only need the Client on this machine and the Server on the machine where your remote ODBC driver is located.

If you have ODBC drivers on this machine (e.g. Easysoft's Tetra/CS3, D-ISAM or Zortec ODBC drivers) which you want to access over the OOB from other machines you should install the server too.

A few examples help to illustrate this:

1. Accessing MS SQL Server on Windows from a UNIX machine.

   You should install the OOB Client on the UNIX machine.

   You should install the OOB Server on a Windows machine which has a MS SQL Server ODBC Driver (it does not have to be the machine running MS SQL Server).

2. Accessing D-ISAM files on UNIX from Windows

   You should install the OOB Client on the Windows machine which needs access to your D-ISAM files.

   You should install the OOB Server on your UNIX machine. In this particular case you will need an ODBC driver for D-ISAM on your UNIX machine (like Easysoft Data Access for D-ISAM).

   You will need an ODBC Driver Manager to use the OOB Client from your applications or for the OOB Server to access your ODBC drivers on this machine. OOB distributions contain the unixODBC Driver Manager (see **http://www.unixodbc.org**). Most (if not all) UNIX applications and interfaces (e.g. Perl DBD::ODBC, PHP, Python etc) support the unixODBC Driver Manager.

You do not have to install the unixODBC Driver Manager in this distribution as you can use an already installed unixODBC (whether that was installed with another Easysoft product, from your Operating System Vendor or even if you built it yourself). However, Easysoft ensure the unixODBC distributed with Easysoft ODBC Drivers has been tested with our drivers so we recommend you use it.

If you choose to use an already installed unixODBC Driver Manager the installation script will attempt to locate it. The installation looks in the standard places but if you have installed it in a non-standard location you will need to provide that location to the installation script when it prompts you. The installation primarily needs unixODBC's `odbcinst` command to install drivers and any data sources.

Where to install

This installation needs a location for the installed files. The default is `/usr/local`.

At the start of the installation you will be prompted for an installation path. All files are installed in a subdirectory of your specified path called "easysoft." For example, if you pick the default of `/usr/local`, the OOB will be installed in `/usr/local/easysoft` and below.

If you choose an install path different from the default then the installation will try to symbolically link `/usr/local/easysoft` to the easysoft in your chosen path. This allows us to distribute binaries with built in dynamic linker run paths. If you are not `root` or the path `/usr/local/easysoft` already exists and is not a symbolic link this will fail. For information about how this may be corrected manually, see **"Post installation steps for non-root installations" on page 68**. You should note that you cannot license Easysoft products until `/usr/local/easysoft` exists either as a symbolic link to your chosen install path or as the install path itself.

Changes made to your system

This installation installs files in subdirectories of the path requested at the start of the installation. Depending on what is installed, a few changes may be made to your system as outlined below:

Client:

1. If you choose to install the OOB Client into unixODBC then unixODBC's `odbcinst` command will be run to add an entry to your `odbcinst.ini` file. You can locate this file with "`odbcinst -j`" (`odbcinst` will be in `/usr/local/easysoft/unixODBC/bin` if you are using the unixODBC that comes with OOB.)

   The entry for OOB will look similar to this:

```
[OOB]
Description = Easysoft ODBC-ODBC Bridge
Driver = /usr/local/easysoft/oob/client/libesoobclient.so.1
Setup = /usr/local/easysoft/oob/client/libesoobsetup.so
UsageCount = 1
```

On some platforms, the OOB Client is distributed as two separate drivers, a thread-safe one requiring `pthreads` (OOB_r) and one which is not thread-safe (OOB) and does not require `pthreads`. If this distribution contains the thread-safe driver, an additional driver will be added to your `odbcinst.ini` file:

```
[OOB_r]
Description = Easysoft ODBC-ODBC Bridge (MT)
Driver = /usr/local/easysoft/oob/client/libesoobclient_r.so.1
Setup = /usr/local/easysoft/oob/client/libesoobsetup.so
UsageCount = 1
```

For information about removing these drivers, see **"Removing from unixODBC" on page 75**.

2. If you choose to add the demo data source to unixODBC, an entry will be added to your SYSTEM `odbc.ini` file. You can locate your SYSTEM `odbc.ini` file using `odbcinst -j`. The entry will look similar to this:

```
[demo]
Driver          = OOB
Description             = Easysoft ODBC-ODBC Bridge demo data source
SERVERPORT               = demo.easysoft.com:8888
TARGETDSN                = pubs
LOGONUSER                = demo
LOGONAUTH                = easysoft
TargetUser               = demo
TargetAuth               = easysoft
```

For information about removing this, see **"Removing from unixODBC" on page 75**.

If you had an OOB Server already installed on a machine when you installed the OOB Client, you would have been given the opportunity to test connection to that Server and enter all the details required to define this local data source tailored for your Server.

Once the test is successful, you will be asked to provide a data source name and whether you want this added to your SYSTEM `odbc.ini` file. The entry added will look like the one shown above in **2**.

Server:

1. If you installed the OOB Server and chose to run it under (`x`)`inetd` the installation will:

   a) add a service called `esoobserver` (or whatever you chose to call it) to your `/etc/services` file.

   b) update your `inetd` configuration. For a description of which files are affected, see **"Removing from inetd" on page 76** and **"Removing from xinetd" on page 77**.

Client and Server:

1. Dynamic Linker.

   On operating systems where the dynamic linker has a file specifying locations for shared objects (Linux, FreeBSD), the installation will attempt to add paths under the path you provided at the start of the install to the end of this list.

   On Linux, this is generally the file `/etc/ld.so.conf`.

   On FreeBSD, this is generally the file `/etc/defaults/rc.conf`.

Reinstalling or installing when you already have other Easysoft products installed

Each Easysoft distribution contains common files shared between Easysoft products. These shared objects are placed in `/usr/local/easysoft/lib`. When you run an installation, the dates and versions of these files will be compared with the same files in the distribution and only updated if the files being installed are newer or have a later version number.

You should ensure that nothing on your system is using Easysoft software before starting an installation because on some platforms, files in use cannot be replaced. If a file cannot be updated, you will see a warning during the installation. You may review all warnings after the installation in the file called "warnings" in the directory you unpacked the distribution into.

If the installer detects you are upgrading a product, the installer will suggest you delete the product directory to avoid having problems with files in use. An alternative is to rename the specified directory.

If you are upgrading an OOB Server, you will need a new license from Easysoft to use the new Server.

Gathering information required during the installation

During the installation you will be prompted for various pieces of information. Before installing, you should determine:

- If you have unixODBC already installed and where it is installed. The install searches standard places like `/usr` and `/usr/local`, but if you installed in a non-standard place and you don't install the unixODBC that comes with OOB you will need to know the location.
- If you install the OOB Client and you want to create a test data source and test it during the installation you will need:

a) the name or IP address of the machine where the OOB Server is running.

b) a username and password to logon to the operating system on the Server machine.

c) the name of a SYSTEM data source on the server machine you want to connect to.

d) a username and password (if required) to logon to the database engine defined in (**c**).

INSTALLATION

Unpacking the distribution

The OOB distribution for UNIX platforms except MAC OS X and QNX is distributed as a tar file. There are multiple copies of the same distribution with different levels of compression. You unpack the distribution as follows.

If the distribution file has been gzipped (i.e. the filename ends in `.gz`), then use:

`gunzip odbc-odbc-bridge-`*`x.y.z-platform`*`.tar.gz`

If the distribution file has been bzipped (i.e. the filename ends in `.bz2`), then use:

`bunzip2 odbc-odbc-bridge-`*`x.y.z-platform`*`.tar.bz2`

If the distribution file has been compressed (i.e. the filename ends in .Z), then use:

`uncompress odbc-odbc-bridge-`*`x.y.z-platform`*`.tar.gz`

You may have a distribution file which is not compressed at all (i.e. the filename ends in `.tar`).

To extract the installation files from the tar file use:

```
tar -xvf odbc-odbc-bridge-x.y.z-platform.tar
```

This will create a directory with the same name as the tar file (without the `.tar` postfix) containing further archives, checksum files, an install script and various other installation files.

Change directory into the directory created by unpacking the tar file.

License to use

There are two license agreement files provided in the distribution; one that applies if you are installing just the OOB Client, and another that applies if you are installing the OOB Server or both the OOB Client and Server. The license text can be found in the files `Client-License.txt` and `Server-Client-License.txt`. Determine which applies to you and be sure to understand the terms before continuing as you will be required to accept the license terms at the start of the installation.

Answering questions during the installation

Throughout the installation, you will be asked to supply the answer to some questions. In each case, the default will be displayed in square brackets and you need only press *<Enter>* to accept the default. If there are alternative responses, these will be shown in round brackets; to pick one of these type them and press *<Enter>*.

For example:

```
Do you want to continue? (y/n) [n]:
```

The possible answers to this question are "`y`" or "`n`". The default when you enter nothing and hit *<Enter>* is "`n`".

Running the installer

Before you run the installer, make sure you have read the **"Before You Install"** section. If you are considering running the installation as a non `root` user, we suggest you review this carefully as you will have to get a `root` user to manually complete some parts of the installation afterwards. Easysoft recommend installing as the `root` user. If you are concerned about the changes that will be made to your system, see **"Changes made to your system" on page 48**.

To start the installation run:

```
./install
```

You will need to:

- Confirm your acceptance of the license agreement with "`yes`" or "`no`".

  See **"License to use" on page 53**.

- Enter a location where the software is to be installed. Easysoft recommend taking the default here. See **"Where to install" on page 47**.

NB	If you are upgrading an OOB Server you will need a new license from Easysoft.

The next step is Locating or installing unixODBC.

Locating or installing unixODBC

Easysoft strongly recommend you use the unixODBC Driver Manager because:

- The installation is designed to work with unixODBC and can automatically add ODBC drivers and DSNs during the install.

- Most applications and interfaces that can use ODBC know about unixODBC. This means any new ODBC drivers or data sources you add with this installation will automatically become available to your applications and interfaces.
- unixODBC is currently maintained by Easysoft developer Nick Gorham. This means there is much greater experience with unixODBC within Easysoft and we will be able to provide better support for OOB running under unixODBC. It also means that if you find a problem in unixODBC it is much easier for us to facilitate a fix.
- The unixODBC package contains much more than a driver manager. The aim of the unixODBC project is to provide all the ODBC functionality available on Windows for UNIX operating systems. The unixODBC package may be built with the QT libraries to allow GUI configuration of DSNs and drivers. It also contains the GUI DataManager program which may be used to explore your ODBC data.
- OOB contains the code and shared object which is used by unixODBC's GUI ODBCConfig utility to add/delete and configure OOB DSNs.

The installation will start by searching for an installed unixODBC.

There are two possible outcomes here:

1. If unixODBC is located, a message will be output saying:

   `Found unixODBC under /path_to_unixODBC and it is version n.n.n`

2. unixODBC is not found.

   If unixODBC is not found in the standard places, you will be asked whether you have it installed.

If you have it installed, you need to provide the argument given to unixODBC's configure as --prefix. For example, if you built unixODBC with "`configure --prefix=/usr/local/unixODBC`," you enter "`/usr/local/unixODBC`". Generally, the path required is the directory above where `odbcinst` is installed. For example, if `odbcinst` is in `/opt/unixODBC/bin/odbcinst` the required path is `/opt/unixODBC`.

If you have not got unixODBC installed, you should install the unixODBC included with OOB.

If you already have unixODBC installed, you do not have to install the unixODBC included with OOB. However, you might consider doing so if your version is older than the one included with OOB.

The unixODBC in the OOB distribution is not built with the default options in unixODBC's configure line:

- `--prefix=/etc`

  This means the default SYSTEM `odbc.ini` file where SYSTEM dsns are located will be in `/etc/odbc.ini`.

- `--enable-drivers=no`

  This means other ODBC drivers that come with unixODBC are not installed.

- `--enable-iconv=no`

  This means unixODBC will not look for a `libiconv`. Warnings about not finding an iconv library were confusing our customers.

- `--enable-stats=no`

  Disables unixODBC statistics which uses system semaphores to keep track of used handles. Many machines do not have sufficient semaphore resources to keep track of statistics and they are only available in the GUI ODBC Administrator anyway.

- `--enable-readline=no`

  This disables `readline` support in `isql`. We disabled this because it ties `isql` to the version of `libreadline` on the machine we build on. We build on as old a version of the Operating System as we can for upwards compatibility. Many newer Linux machines no longer come with the older `readline` libraries and so enabling `readline` support renders `isql` unusable.

- `--prefix=/usr/local/easysoft/unixODBC`

  This installs unixODBC into `/usr/local/easysoft/unixODBC`.

Installing the OOB Client

You need to install the OOB Client if you want to access remote ODBC drivers (ODBC drivers on other machines) from this one. The OOB Client installation comprises of:

- At the "Install ODBC-ODBC Bridge Client (y/n) [y]:" enter "y" to install the OOB Client.
- Register OOB Client ODBC Driver with the unixODBC Driver Manager.

If unixODBC is now installed (either installed by this installation or an existing copy was found) the OOB Client ODBC Driver can be registered as an ODBC driver with the unixODBC Driver Manager.

Answering "`y`" to "`Install OOB into unixODBC (y/n) [y]:`" will add entries into unixODBC's `odbcinst.ini` file making the OOB Client available to your applications and interfaces (see **"Changes made to your system" on page 48**).

The default here is "`y`".

If you already have the OOB Client registered with unixODBC, you will see a warning that OOB is already registered and a list of the drivers unixODBC knows about. If you are installing OOB into a different directory than the one it was installed into before, you will need to edit your `odbcinst.ini` file after the installation and correct the Driver and Setup paths. unixODBC's `odbcinst` will not update them if a driver is already registered.

- Creating an OOB demo data source in unixODBC.

  If unixODBC is installed and you registered the OOB Client with unixODBC, you will be asked if you want to add a demo data source to your `odbc.ini` file. This is an OOB Client data source which points to an OOB Server running on `demo.easysoft.com`. You can use this DSN to test the OOB Client if you do not install an OOB Server yourself.

To use the demo data source, this machine will need access to `demo.easysoft.com` through port 8888 (which may be disabled in your firewall). If this machine does not have access to the net or is blocked by a firewall, there is still no harm in installing this DSN. It is useful as an example when you are creating your own data sources.

If you choose to install the OOB demo data source and a data source called "demo" already exists, the existing demo data source will be displayed and you have the option to replace it.

- Test Connection to OOB Server

  If you have already installed the OOB Server on a machine, you can now test the connection to it, display a list of tables and create a data source definition specific to your setup. Although the default is to perform the test, you do not have to.

  The installation guides you through this process step by step using `oobping` at each stage to test connection to the OOB Server, operating system authentication, remote SYSTEM DSN existence and database authentication. If at any time you want to abort this test, you can enter "`q`" at any prompt.

  Each successful step should display the output from the Server and then the OOB attributes required for that stage. For example:

```
Server (q=quit) : myserver
Using /usr/local/easysoft/bin/oobpings -h myserver -t 8888
================================================================
Host: myserver, Port: 8888
Attempting connection...OK
Examining Server...
      OOB Server Version: 2.0.0
      OOB Server Name: OOB
================================================================
ServerPort = gambantein:8888
```

Here you entered, `myserver` as the name of the machine where the OOB Server is installed, `oobping` was used to connect to the Server which returned Server version `2` and name `OOB`. Finally, the OOB attribute (`ServerPort`) required for this stage is displayed.

Subsequent steps add the operating system username/password, remote SYSTEM data source and database username/password until you have the complete OOB Client DSN to reproduce the connection.

If you successfully connect to a remote SYSTEM DSN, you can enter a name for this data source definition and it can be written to your system `odbc.ini` file.

Finally, this data source can be used to obtain a list of tables from the remote database (unixODBC's `isql` will be used for this).

- Test Perl setup

  If the installation detects you have Perl installed, it can run a series of tests on it making recommendations for using ODBC in Perl. You do not have to perform this test and if you are not going to use Perl there is nothing to be gained.

Installing the OOB Server

You install the OOB Server on the machine where you have an ODBC driver for the database you want to access. You need a license key to use the OOB Server and one can be obtained during the installation (see **"Licensing the OOB Server" on page 63**).

The OOB Server installation comprises of:

- At the "`Install ODBC-ODBC Bridge Server (y/n) [n]:`" enter "`y`" to install the OOB Server.

  All the server files will be installed.

- License

  "`Would you like to request a OOB Server license now (y/n) [y]: `"

  See **"Licensing the OOB Server" on page 63**.

- Install services and {x}`inetd` entries

  By default, the OOB Server runs under `inetd` or `xinetd` and needs an entry in your `services` file. You can run the OOB Server standalone without `inetd` but we recommend you start with an `inetd` setup.

At the question "Do you wish to install the services and inetd entries? (y/n) [y]: " enter "y". If you enter "n" here then you will either have to set up the OOB Server under inetd yourself or run it standalone.

The installation then attempts to find {x}inetd and services configuration files. If these are located successfully, it will check there is no other service called "esoobserver" and no other service using port 8888. If there is a conflict with either the service name or port, you will be given a chance to change (or overwrite) them. In this situation, you should pick any name you like for the service which is not already in use and an unused port. Once the service name and port are finalized, an entry will be added to the end of the {x}inetd and service configuration files. The installation will show you which files are being changed and which entries are being added. The final part of defining the server environment is creating a script in *install_path*/easysoft/oob/server. This will be run by {x}inetd when the OOB client connects to the specified port. Once this is complete, the {x}inetd daemon is asked to reread the configuration files so that may listen on the specified port on behalf of the OOB server.

You can see a list of changed files during the installation. Further details are available in **"Changes made to your system" on page 48**.

Licensing the OOB Server

Only the OOB Server needs to be licensed.

The program `/usr/local/easysoft/license/licshell` is used to obtain or list licenses.

Licenses are stored in the file `/usr/local/easysoft/license/licenses`. After obtaining a license, you should take a copy of this file in case something happens to it.

If you decide to install the OOB Server, the installation will ask you if you want to request an OOB Server license:

```
Would you like to request a OOB Server license now
(y/n) [y]:
```

You do not need to obtain a license during the installation, you can run `licshell` after the installation to obtain or view licenses.

If you answer yes to this, the installation will run the `licshell` script. The process of obtaining a license is best described in the **Licensing Guide** and on the Easysoft web site.

To obtain a license automatically, you will need to be connected to the Internet and allow outgoing connections to `license.easysoft.com` on port 8884. If you are not connected to the Internet or do not allow outgoing connections on port 8884, the License Client can create a license request file which you can:

1. Enter at:

   - **http://www.easysoft.com/support/licensing/trial_license.html**

     (to obtain a trial license)

   - **http://www.easysoft.com/support/licensing/full_license.html**
     (to obtain a purchased license)

2. Mail, fax or telephone Easysoft.

   Obviously, option 1 is quickest if you have a web browser and access to the Internet.

   Once the License Client has started you are presented with a menu of options which allow you to:

   ```
   [0] exit

   [1] view existing license

   [n] obtain a license for the desired product
   ```

   Obviously, if you have not got any other Easysoft products licensed then option [1] will not show any existing licenses.

   To obtain a license select one of the options from [2] onwards for the product you are installing. The License Client will then run a program that was installed for that product which generates a key which is used to identify the product and operating system (we need this key to license you)

NB	Some products in the list may not be licensable during the install, e.g. if you are installing the OOB and do not have the Easysoft JDBC-ODBC Bridge installed, you cannot obtain a license for the JDBC-ODBC Bridge. (In this case, the License Client will probably output a warning that the `siteinfo` command for the chosen product was not found).

   Once you have picked the product to license (ODBC-ODBC Bridge) you need to supply:

1. Your full name
2. Your company name

3. An email contact address. This (currently) **must** be the email address you registered on the Easysoft web site.
4. Your telephone number (you need to specify this if you telephone the license request to us).
5. Your fax number (you need to specify this if you fax the license request to us).
6. A reference number. When applying for a trial license, just hit *<Enter>* on this field as this field is used to enter a reference number we will supply you for full (paid) licenses.

   You will then be asked for a method of obtaining the license where the choices are:

   [1] Automatically by contacting the Easysoft License Daemon (this requires connection to the Internet and the ability to support an outgoing TCPIP connection to `license.easysoft.com` on port 8884).

   [2] Write information to file so you can:

   a) obtain your license at:

   - **http://www.easysoft.com/support/licensing/trial_license.html**

     (to obtain a trial license)

   - **http://www.easysoft.com/support/licensing/full_license.html**
     (to obtain a purchased license)

   b) fax, telephone it.

   The license request is output to `license_request.txt`.

   [3] Cancel request

If you choose to obtain the license automatically, the License Client will start a TCPIP connection to `license.easysoft.com` on port 8884 and send the details you entered at the prompts above and your machine number. No other data is sent. The data sent is transmitted as plain text so if you do not want this information possibly intercepted by someone else on the net you should choose [2] and telephone or fax the request to us. The License daemon will return the license key, print it to the screen and make it available to the installation script in the file licenses.out.

If you choose option [2], the license request is written to the file `license_request.txt` and you should exit the License Client via option [0] and complete the installation. Once you have mailed, faxed or telephoned the license request to us, we will return a license key which should add to the end of the file *`install_path`*`/easysoft/license/licenses`.

If during this process any warnings or errors are output, please mail the output to **support@easysoft.com** and we will rectify the problem.

POST INSTALLATION

Supplied documents and examples

The last part of the installation runs a post install script which lists resources available to you.

- Included documentation is installed in `/usr/local/easysoft/oob/doc`

  The OOB manual (this manual) in PDF format.

  Various files detailing how to get ODBC access with various applications and interfaces.

  `Changes.txt` - a list of all the changes in each OOB version.

  `FAQ.txt` - a text version of the OOB FAQ.

  `NEWS.txt` - a list of significant releases.

  The client and server licenses.

  Various notes per operating system.

  Example `odbc.ini` entries and descriptions of OOB-specific ODBC connection attributes in `example_odbc.ini` and `DSN_definition.txt`.

  The OOB performance white paper.

- Examples are installed in `/usr/local/easysoft/oob/examples`.
- Various Perl, PHP etc tutorials are installed in sub directories of `/usr/local/easysoft/oob/doc`.

There are many resources at the Easysoft web site.

Post installation steps for non-root installations

If you installed the OOB Client as a non-root user (not recommended), there may be some manual steps you will need to perform:

1. If you attempt to install the OOB Client under the unixODBC Driver Manager and you do not have write permission to unixODBC's `odbcinst.ini` file, the OOB driver cannot be added.

   You can manually install the OOB Client driver under unixODBC by adding an entry to the `odbcinst.ini` file. Run `odbcinst -j` to ascertain the `DRIVERS` file. Then append the lines from `oob.tmpl` (in the directory where the OOB distribution was untarred to) to the `odbcinst.ini` file.

2. As in step **1**, no example dsns can be added into unixODBC if you do not have write permission to the SYSTEM `odbc.ini` file. Run `odbcinst -j` to ascertain the name of the "SYSTEM DATA SOURCES" file then add your DSNs.

3. On machines where the dynamic linker has a configuration file defining the locations where it looks for shared objects (Linux/FreeBSD) you will need to add:

   install_path`/easysoft/lib`

   install_path`/easysoft/unixODBC/lib`

   The latter one is only required if you installed the unixODBC included with OOB. Sometimes after changing the dynamic linker configuration file you need to run a program to update the dynamic linker cache (e.g. `/sbin/ldconfig` on Linux).

4. If you did not install OOB in the default location, then you need to link `/usr/local/easysoft` to the easysoft directory in your chosen install path.

For example, if you installed in `/home/martin` the installation will create `/home/martin/easysoft` and you need to symbolically link `/usr/local/easysoft` to `/home/martin/easysoft`:

```
ln -s /home/martin/easysoft /usr/local/easysoft
```

5. If your system does not have a dynamic linker configuration file, you need to add the paths listed in step **3** above to whatever environment path the dynamic linker uses to locate shared objects. You may want to amend this in a system file run whenever someone logs in like `/etc/profile`.

   The environment variable differs per dynamic linker. Consult your `ld` or `ld.so man` page. It is usually:

   `LD_LIBRARY_PATH`, `LIBPATH`, `LD_RUN_PATH` or `SHLIB_PATH`.

   Testing the OOB server

   The Client side of the OOB comes with an `oobping` program which can be used to test connection to an OOB Server. There are two versions of `oobping` in `/usr/local/easysoft/bin`. `oobpings` is statically linked and should not require any libraries other than `libc` (this means you do not have to set any dynamic linker paths). `oobpingd` is a dynamically linked `oobping` (therefore smaller in size) which requires additional libraries in `/usr/local/easysoft/lib` and `/usr/local/easysoft/oob/client`). Generally speaking you should use `oobpings`.

   For full instructions on using `oobping` see **"Testing the OOB Server" on page 116**. To display a summary of usage options, run `oobpings` with no arguments.

A short summary is included here:

1. `cd /usr/local/easysoft/bin`
2. To test an OOB Server is installed, running and listening on a specific port use:

   ```
   ./oobpings -h myserver -p 8888
   ```

   For example:

   ```
   ./oobpings -h demo.easysoft.com -p 8888
   ```

   The `-p 8888` is optional as by default the OOB Server uses port 8888.
3. To test as in step **2** plus operating system authentication:

   ```
   ./oobpings -h demo.easysoft.com -p 8888 -u demo -p
   easysoft
   ```
4. To do a full test of an OOB Client DSN defined in your `odbc.ini` file use:

   ```
   ./oobpings -d "DSN=demo;"
   ```

   where "`demo`" is the name of your OOB Client DSN.

   You can also use `oobping` to test DSN-less connections by putting all the connection attributes in the `-d` argument. For example:

   ```
   ./oobpings -d
   'LogonUser=demo;LogonAuth=easysoft;TargetDSN=pubs;
   UID=demo;PWD=easysoft;ServerPort=demo.easysoft.com
   :8888;'
   ```

Testing the OOB client

The OOB client is a shared object containing entry points for the full ODBC 3.5 API, most of the ODBC 2.0 API (and a few others as well).

The OOB client only becomes functional when an application is linked with it directly or via a driver manager. You can use unixODBC's command line `isql` program to test the OOB Client and issue SQL to your database. OOB comes with an `isql` wrapper program called `isql.sh` which will set up the dynamic linker before running `isql`.

The `/usr/local/easysoft/oob/examples` directory contains a number of test applications and make files which may be used to try the OOB out.

Please consult the `INSTALL.txt` file in the `examples` directory.

Setting Dynamic Linker Search Paths

Your applications will be linked against an ODBC Driver Manager which will load the ODBC Driver you require. The dynamic linker needs to know where to find the ODBC Driver Manager shared object. The ODBC Driver Manager will load the OOB Client ODBC driver which is dependent on further common Easysoft shared objects; the dynamic linker needs to locate these too.

On operating systems where the dynamic linker has a file specifying locations for shared objects (Linux, FreeBSD), the installation will attempt to add paths under the path you provided at the start of the install to the end of this list; no further action should be required. For more information, see **"Dynamic Linker." on page 50**.

On other Unix platforms, there are two methods of telling the dynamic linker where to look for shared objects:

1. You add the search paths to an environment variable and export it.

   This always works and overrides **2** below.

2. At build time, a run path is inserted into the executable or shared objects. On System V systems, Easysoft mostly distribute OOB Client shared objects with an embedded run path which the dynamic linker can use to locate OOB Client shared object dependencies.

   For **2** above, the environment variable you need to set depends on the platform (refer to the platform documentation for `ld(1)`, `dlopen` or `ld.so(8)`).

Environment Variable	Platform
LD_LIBRARY_PATH	System V based operating systems and SCO, Linux, Solaris, FreeBSD, Irix, QNX.
DYLD_LIBRARY_PATH	MAC OS X
LIBPATH	AIX
SHLIB_PATH	HP-UX
LD_RUN_PATH	Many platforms use this in addition to those listed above.

Figure 10: Dynamic linker search path environment variables.

To use the OOB Client ODBC driver you need to add:

`<InstallDir>/easysoft/oob/client:<InstallDir>/easysoft/lib`

where *<InstallDir>* is the directory in which you chose to install the Easysoft ODBC-ODBC Bridge. If you accepted the default location, this is `/usr/local`.

An example of setting the environment path in the Bourne shell on Solaris is:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/easysoft/oob/client:/usr/local/easysoft/lib

export LD_LIBRARY_PATH
```

NB	The exact command you need to set and export an environment variable depends on your shell.

If unixODBC has been installed then you will also need to add

<InstallDir>`/easysoft/unixODBC/lib` to the dynamic linker search path.

Uninstalling on Unix

To uninstall the Easysoft ODBC-ODBC Bridge under Unix:

- If unixODBC is installed, the Easysoft ODBC-ODBC Bridge driver must be removed from its database.
- The entries for the Easysoft ODBC-ODBC Bridge must be manually removed from the Unix `services` and `{x}inetd` configuration files (this requires `root` access).

Caution!

This is only required if the Easysoft ODBC-ODBC Bridge is not going to be upgraded, or is to be upgraded using a different configuration.

If the Easysoft ODBC-ODBC Bridge is going to be upgraded with the existing configuration then do NOT make these edits.

- If the system has a dynamic linker (such as `ld.so` on Linux), the Easysoft ODBC-ODBC Bridge directories must be removed from the dynamic linker search path (this may require `root` access, depending on the mechanism used by the platform).
- The Easysoft ODBC-ODBC Bridge install directory tree must be removed (this requires the same privileges as the user who performed the installation, which is normally `root`).

A step-by-step guide follows:

1. Close down all client programs attached to your service.
2. Log in as `root`.

REMOVING FROM UNIXODBC

3. Check whether the Easysoft ODBC-ODBC Bridge is configured under unixODBC by typing:

   ```
   odbcinst -q -d
   ```

4. If "OOB" is returned in the output then remove the Easysoft ODBC-ODBC Bridge by typing:

   ```
   odbcinst -u -d -n OOB
   ```

   If a message is displayed about a reduced usage count, repeat this step until `odbcinst` states that the Easysoft ODBC-ODBC Bridge has been removed.

REMOVING THE SERVICE ENTRIES

5. Make a backup of the `/etc/services` configuration file.

6. Open `/etc/services` and look at the end for a line of the format:

   ```
   esoobserver 8888/tcp   # Easysoft ODBC-ODBC Bridge
   ```

   If more than one Easysoft ODBC-ODBC Bridge service has been created then there will be more than one line in the `services` file.

   Each such line should have a comment referencing the Easysoft ODBC-ODBC Bridge.

   `esoobserver` is the default name for an Easysoft ODBC-ODBC Bridge service and any additional Easysoft ODBC-ODBC Bridge services will display the alternative names given to them.

   Note down the names of the services you remove at this stage, so that if there are problems you can look them up in your `services` backup file and re-introduce them.

> **NB** You may wish to compare `/etc/services` with `/etc/services.pre_OOB` (which is installed with the Easysoft ODBC-ODBC Bridge) for details of entries that require removal.

7. You must remove all services that were configured for use with the Easysoft ODBC-ODBC Bridge. Delete all lines pertaining to all OOB Servers and save the file.

REMOVING FROM INETD

8. Make a backup of the `/etc/inetd.conf` configuration file.
9. Open `/etc/inetd.conf` in your editor. Look for a line in the file similar to the following:

   ```
   esoobserver stream tcp nowait root /bin/sh
   /bin/sh /usr/local/easysoft/oob/server/SERVER
   ```

   where `esoobserver` is the name as specified in the `services` file, so there should be one entry in `inetd.conf` for every entry in the `services` file.

   > **NB** You may wish to compare `/etc/inetd.conf` with `/etc/inetd.conf.pre_OOB` (which is installed with the Easysoft ODBC-ODBC Bridge) for details of entries that require removal.

10. Remove the lines pertaining to all OOB Servers and save the file.
11. Use `ps` to find the Process ID (PID) of the `inetd` process and send it a hangup signal:

    `kill -HUP` *`pid`*

REMOVING FROM XINETD

`xinetd` uses a configuration file of `/etc/xinetd.conf` (by default), which can either contain the names of the services and what to do with them, OR (on some Red Hat machines, for example) an `includedir` setting which points to a directory where those services are defined (one per file).

You should have been made aware of which method your machine uses from the original installation procedure (see **"Installing the OOB Server" on page 61**).

In the former case the Easysoft ODBC-ODBC Bridge install adds a service entry to `/etc/xinetd.conf and` in the latter case it creates a new service file called `esoobserver` containing the Easysoft ODBC-ODBC Bridge service settings.

12. There are therefore two ways to delete the Easysoft ODBC-ODBC Bridge service from `xinetd`.

    - make a backup of the `/etc/xinetd.conf` configuration file.
    - Open `/etc/xinetd.conf` in your editor. Look for a line in the file similar to the following:

      ```
      service esoobserver
      ```

      where `esoobserver` is the name as specified in the `services` file, so there should be one entry in `xinetd.conf` for every entry in the `services` file.

    – OR –

    - Delete the `esoobserver` file in the directory to which the `includedir` setting in `/etc/xinetd.conf` points.
    - Remove all the lines referring to the OOB Server and save the file.

13. Use `ps` to find the Process ID (PID) of the `xinetd` process and send it a Hangup signal:

    `kill -USR1` *`pid`*

    – OR –

    `kill -USR2` *`pid`*

REMOVING FROM THE DYNAMIC LINKER

Notify the dynamic linker that the shared objects are no longer available.

> **NB** This information only applies to systems with the `ld.so` dynamic linker (normally only Linux).

14. If you have the file `/etc/ld.so.conf` file, make a backup copy, e.g.

    ```
    cp /etc/ld.so.conf /etc/ld.so.conf.safe
    ```

15. Open `/etc/ld.so.conf` and manually remove the path to the Easysoft ODBC-ODBC Bridge client shared objects. The line is of the form:

    `<InstallDir>/easysoft/oob/client`

16. If you are not using any other Easysoft software then you may remove the path to the common Easysoft shared objects:

    `<InstallDir>/easysoft/lib`

17. If you are no longer using unixODBC then you can also remove the reference:

    `<InstallDir>/easysoft/unixODBC`

18. Run `/sbin/ldconfig` so that the dynamic linker re-reads the file and will no longer search the removed paths.

DELETING THE SOFTWARE

Finally, remove the software from your system's hard drive.

19. Change directory to:

    `<InstallDir>/easysoft/`

    `pwd`

    The system displays the current directory. Double-check that this is the directory under which you installed the Easysoft ODBC-ODBC Bridge.

Caution! Be very careful issuing the `rm -r` command as `root`. Normally `rmdir` will not remove directories that contain files, but `rm -r` will remove all subdirectories along with their contents. It is possible to effectively destroy your system and/or lose all user files by removing the wrong directory.

20. Remove the Easysoft ODBC-ODBC Bridge installation directory:

    `ls`

    Check that you are in the right directory.

    `rm -r oob`

    The system may ask you to confirm deletion for some files. You can confirm these as long as you are sure you are in the correct directory.

    `ls`

21. If you have no other Easysoft products on your system and you are not using any copy of unixODBC that may be in this directory, then you can delete the `easysoft` directory too.

    ```
    cd ..
    ```

    ```
    rm -r easysoft
    ```

    – OR –

    If there are other files in the directory tree (i.e. you have other Easysoft products installed) then you must not remove the `easysoft` directory, because it will contain your license keys and other important files.

22. If you left the Easysoft ODBC-ODBC Bridge distribution file on your system then you may wish to remove it at this point.

    The uninstall process is complete.

    Any licenses you obtained for the Easysoft ODBC-ODBC Bridge and other Easysoft products are stored in the *<InstallDir>*`/easysoft/license/licenses` file.

    After uninstalling the Easysoft ODBC-ODBC Bridge, unless you have deleted this file, you will not need to relicense the product when you reinstall or upgrade.

    However, for security purposes you may want to make a copy of *<InstallDir>*`/easysoft/license/licenses` before uninstalling.

Installing on Mac OS X

This section shows how to install the Easysoft ODBC-ODBC Bridge Client for Mac OS X. Note that you will also need to install the Easysoft ODBC-ODBC Bridge Server on the computer on which the remote ODBC driver is installed. For information about installing the OOB Server, see **"Installing on Windows" on page 31** and **"Installing on Unix" on page 44**.

The OOB Client for Mac OS X distribution file is a Mac Installer package. The package is contained in a disk image file.

The OOB Client distribution file size is 3 MB. You will need 8 megabytes (MB) of free disk space for the installed programs.

The OOB Client installer tries to:

- Install the OOB Client ODBC Driver into the iODBC Driver Manager.

  To do this, the installer adds an entry to the `odbcinst.ini` file similar to the following:

```
[ODBC Drivers]
Easysoft ODBC-ODBC Bridge = Installed

[Easysoft ODBC-ODBC Bridge]
Driver = /usr/local/easysoft/oob/client/libesoobclient.so
Setup = /usr/local/easysoft/oob/client/oob.bundle/Contents/MacOS/oob
```

  (To locate the `odbcinst.ini` file file, type `iodbc-config --odbcinstini` in a Terminal window.)

- Create an example System data source named "demo".

  demo is an OOB Client data source that lets you connect to an OOB Server running on `demo.easysoft.com`. You can use this data source to test the OOB Client. To use the demo data source, the computer on which you install the OOB Client will need access to `demo.easysoft.com` through port 8888 (which may be blocked by your firewall). Even if you cannot access `demo.easysoft.com`, demo is a useful example to refer to when creating your own data sources.

For the installer program to install the OOB Client into the iODBC Driver Manager and create the demo data source, the ODBC package included with Mac OS X must already be installed.

The installer program installs the OOB Client into `/usr/local/easysoft/oob` directory. When the installer finishes, look in the file `/usr/local/easysoft/oob/INVENTORY.txt` for information about the OOB Client directory structure and the files that are installed in these directories.

To install the OOB Client for Mac OS X

1. Log in as an administrator user.
2. Double-click the OOB Client for Mac OS X disk image file that you obtained from one of the sources described in **"Obtaining the Easysoft ODBC-ODBC Bridge" on page 26**.
3. In the open disk image, double-click the package file to start the OOB Client installer.

   The OOB Client installer welcome screen is displayed.

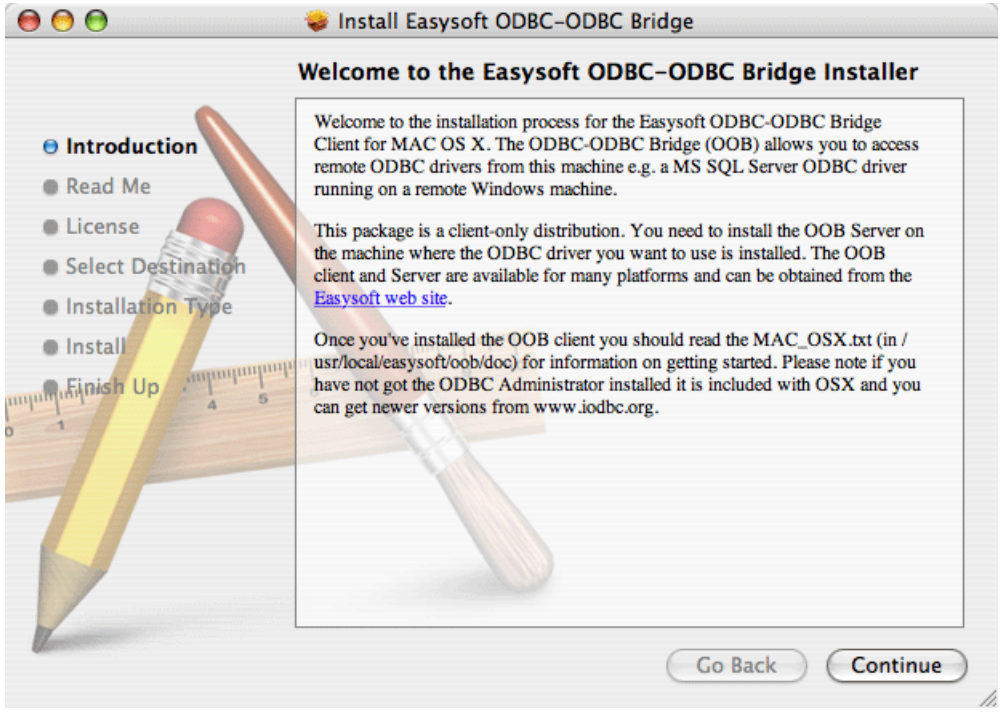


Figure 11: The Easysoft ODBC-ODBC Bridge Client for Mac OS X Installer Welcome Screen

4. When you have read the introductory text, click **Continue**.

   Follow the on screen instructions.

TESTING THE OOB CLIENT

The OOB client only becomes functional when an application is linked with it through an ODBC driver manager. You can use iODBC's command line iodbctest application to test the OOB Client and execute SQL. For more information about using iodbctest to test the OOB Client, see **"Testing Data Sources" on page 173**.

DOCUMENTATION AND RESOURCES

The OOB Client documentation is installed in the `/usr/local/easysoft/oob/doc` directory. For information about using the Easysoft ODBC-ODBC Bridge on Mac OS X, read the file MAC_OSX.txt in this directory. Various tutorials are installed in subdirectories of this directory. Example ODBC applications are installed in the `/usr/local/easysoft/oob/examples` directory.

There are also many resources available at the Easysoft web site.

Uninstalling on Mac OS X

There is no automated way to remove the OOB Client on Mac OS X. However, if you want to remove the software manually, Mac OS X provides a way of listing of the files installed by the OOB Client package. To do this, use the `lsbom` command to examine the OOB Client package's Bill of Materials (.bom) file.

> **Note** You do not need to remove the OOB Client before upgrading to a different version.

To list the files installed by the OOB Client package

1. In Terminal, change directory to `/Library/Receipts`.
2. In this directory, look for a subdirectory with the same name as the OOB Client package file. For example, `odbc-odbc-bridge-2.0.11-darwin-ppc.pkg`.
3. Change directory to *oob_client_package*`/Contents/Resources`
4. In this directory, look for the `.bom` file for the OOB Client package. For example, `odbc-odbc-bridge-2.0.11-darwin-ppc.bom`.
5. To display a list of files installed by the OOB Client package into the `/usr/local/easysoft` directory, type:

   `lsbom -s -f` *`oob_client_bomfile`*
6. To display a list of symbolic links created by the OOB Client package, type:

   `lsbom -s -l` *`oob_client_bomfile`*
7. To display a list of directories created by the OOB Client package, type:

   `lsbom -s -d` *`oob_client_bomfile`*

To remove the ODBC driver entry from odbcinst.ini

1. Log in as an administrator user.
2. Open ODBC Administrator in the `/Applications/Utilities` folder.
3. To remove the OOB Client ODBC driver entry from odbcinst.ini, in the **Drivers** tab, click Easysoft ODBC-ODBC Bridge in the list of drivers. Click **Remove.** Click **OK** when prompted.

   If the Drivers tab is locked, the Remove button will be unavailable. To unlock the Drivers tab, click the lock icon, and then type an administrator password when prompted.
4. If you want to remove your OOB Client data sources, click **OK** when prompted.
5. Click **Apply** to save your changes.

CHAPTER 3 CONNECTION

Connecting via the Easysoft ODBC-ODBC Bridge

An ODBC-compliant application is connected to a remote data source via the Easysoft ODBC-ODBC Bridge (OOB) as follows:

1. On the server, create a system data source for the target database.
2. On the client, configure a data source pointing to the server machine and the server data source created in **step 1 on page 87**.
3. From your application on the client machine, connect to the OOB Client data source created in **step 2 on page 87**.

If you have an internet connection you can omit the server sections and set up a client connection to the `demo.easysoft.com` server by using the recommended settings in the relevant client subsection.

Chapter Guide

The connection process

This section explains what happens when an ODBC application connects to a data source, and what happens when connecting through the Easysoft ODBC-ODBC Bridge.

Understanding the connection process will give an insight into why the connection might fail and what is required to connect an ODBC application to remote data sources.

To skip this explanatory section, proceed to **"Setting up the OOB Server" on page 97**.

In ODBC an application connects to a database by means of a data source description, which depends on the ODBC driver used to access the database and consists of a set of attribute and value pairs.

Usually, the application links with a driver manager that looks at the data source description in the connection string, loads in the required ODBC driver and then passes the connection string to the ODBC driver.

At its simplest the application passes a connection string which defines a data source name (DSN) to the ODBC driver (or driver manager), such as:

```
DSN=test_datasource;
```

In this case the driver manager looks at the **Driver** attribute in the data source to decide which driver to use, loads the driver and then the driver looks up the data source to retrieve all the other required attributes.

This information is found in the registry under Windows and in the user and system `odbc.ini` files under Unix and MAC OS X.

It therefore follows that before an application can connect to an ODBC driver you have to create a data source containing all the attributes the driver requires to describe the database (or alternatively, the application can pass all the attributes in the connection string).

Two data sources are required for the Easysoft ODBC-ODBC Bridge:

- a server data source on the machine where the OOB Server is installed and you have a native ODBC driver for your database, describing the database in whatever terms the ODBC driver requires (e.g. in Windows a dialog box is provided by the ODBC driver and accessed from the ODBC Administrator).
- a client data source on the machine where the application is running, which is an Easysoft ODBC-ODBC Bridge data source describing how to connect to the OOB Server (e.g. the address of the server), a user identity with which to access the server machine and a description of the server data source.

For example (omitting the driver manager differences):

Assume there is a Microsoft Access database and a Microsoft Access ODBC driver on a Windows NT machine called "`ntbox.easysoft.com`".

There is also an application on a Linux box called "`linuxbox.easysoft.com`", which wants to read data from the Microsoft Access database.

It would be necessary to install:

- the OOB Server on `ntbox`, and then create a system data source using the Microsoft Access ODBC driver to make the Microsoft Access database available to the OOB Server.
- the OOB Client on `linuxbox`, and then create a data source using the Easysoft ODBC-ODBC Bridge driver, so that the application on this machine can access the database on `ntbox`.

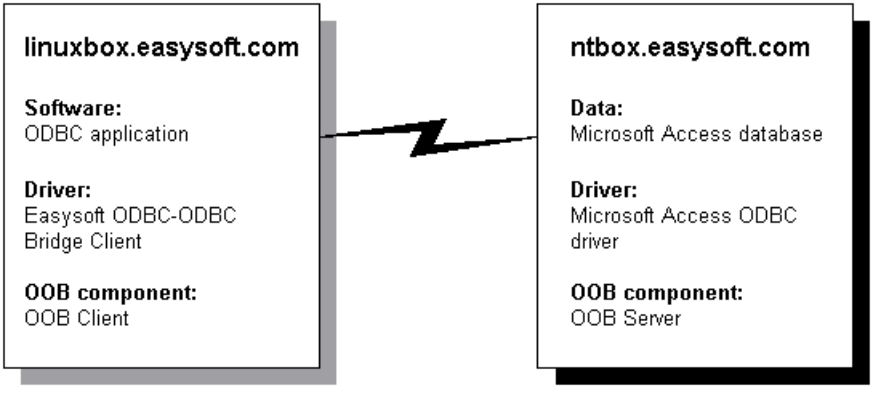


Figure 12: Linking a Linux client to an NT server

On the `ntbox` server machine the data source description for the Microsoft Access database might consist of:

```
[MyDB]

Description = my Access database

mdb file = c:\databases\msaccess\mydb.mdb
```

`MyDB` is the data source name and Microsoft Access needs the location of `mdb file` to define the database.

On the `linuxbox` client machine the Easysoft ODBC-ODBC Bridge data source might consist of:

```
[ntbox_mydb]

Description = Access mydb on ntbox

ServerPort = ntbox.easysoft.com:8888

LogonUser = my_ntusername

LogonAuth = my_ntpassword

TargetDSN = mydb
```

- `ntbox_mydb` is the data source name which you can refer to in the connection string passed to the OOB Client.
- `ServerPort` is the name of the server machine where the OOB Server is running and the port where the OOB Server is listening for connections.
- `LogonUser` and `LogonAuth` describe the user details under which the OOB Client will log on to the server.
- `TargetDSN` describes the data source on the server machine to which the OOB Server should connect.

The application on `linuxbox` can now call `SQLDriverConnect` passing a connection string of the form `"DSN=ntbox_mydb;"` or `SQLConnect` passing a data source name.

The OOB Client looks up the `ntbox_mydb` data source and retrieves all the other necessary attributes to connect to the server.

The OOB Client connects to the OOB Server, passing `my_ntusername` and `my_ntpassword` which are verified and then the OOB Server becomes the user `my_ntusername`.

At this stage the OOB Client produces a new connection string of the form "`DSN=mydb;`" in a call to `SQLDriverConnect` on the OOB Server and the process starts again in the server ODBC driver.

Note that the application is capable of passing any or all the attributes as part of the connection string (for example, the user name and password attributes can be included to avoid storing them on the server machine.

The full connection string for this example would be:

```
ServerPort=ntbox:8888;

LogonUser=my_ntusername;

LogonAuth=my_ntpassword;

TargetDSN=MyDB;
```

This is a simplified example of the connection process, but it does illustrate what actually happens.

WHEN THE DBMS REQUIRES AUTHENTICATION

When the server database requires authentication the ODBC defined `UID` and `PWD` attributes may be included in the connection string.

Alternatively, if the `TargetUser` and `TargetAuth` attributes are added to the Easysoft ODBC-ODBC Bridge Client data source, the OOB Client will pass these to the database as `UID` and `PWD`.

The OOB Client uses some simple rules to determine what to pass to the DBMS:

1. If `UID` and/or `PWD` exist in the connection string and `Override UID/PWD` (or `UseOOBDBAuth`) is *not* enabled (i.e. is not selected on the Easysoft ODBC-ODBC Bridge DSN dialog box or is set to 0 for that data source in `odbc.ini` on Unix platforms), then `UID` and/or `PWD` are passed to the DBMS unchanged. If `Override UID/PWD` is enabled, then any `UID` and/or `PWD` values in the connection string are ignored.
2. If not Rule 1, but the DSN contains `TargetUser` and/or `TargetAuth`, these are changed to `UID` and `PWD` and passed to the DBMS.
3. If not Rule 1 or 2, no `UID` or `PWD` will be in the connection string passed to the DBMS (implying that authentication is not required).

   On platforms where the OOB Client has specific support for a GUI environment, the ODBC driver may prompt you for database user name and password details, if `UID` or `PWD` are not specified or are incorrect.

   Note that the Easysoft ODBC-ODBC Bridge does server authentication first and then database authentication. A **Server Logon** dialog box will be displayed if you have failed to enter values into the **Username** and **Password** fields of the OOB Client DSN. In this case a valid user name and password for the server operating system must be entered:

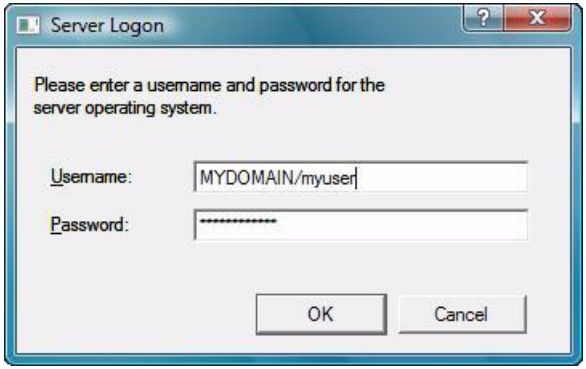


Figure 13: The Easysoft ODBC-ODBC Bridge Server Logon dialog box

The OOB Client ODBC driver will connect to the server and attempt a database connection, as described in **"The connection process" on page 88**. However, if the ODBC driver at the server end denies authentication and returns the ODBC state `28000`, the OOB Client ODBC driver displays the **Database Logon** dialog box.

This requires a valid database user name and password to be entered, which will prompt the OOB Client ODBC driver to make one further connection attempt with this new authentication:

Figure 14: The Easysoft ODBC-ODBC Bridge Database Logon dialog box

How ODBC driver managers fit into the connection process

This section explains how ODBC driver managers fit into the connection process when connecting via the Easysoft ODBC-ODBC Bridge.

You should read this section if you intend to integrate the Easysoft ODBC-ODBC Bridge with your own ODBC application.

In general, ODBC applications must be linked with either an ODBC driver or a driver manager.

When a program calls `SQLDriverConnect` (or `SQLConnect`), it passes in a connection string, which contains a list of connection attributes, normally including one of the following:

- a `DSN=` attribute which names the data source. Specifying a DSN (Data Source Name) implicitly tells the driver manager which driver to load because each DSN will contain a `DRIVER` attribute.
- a `DRIVER=` attribute which names the driver. Specifying a driver allows you to choose any database to which the driver has access.
- a `FILE=` attribute which allows the database attributes to be read from a file.

A connection string would look something like:

```
SQLDriverConnect("DSN=pubs;UID=demo;PWD=easysoft;")
```

where `pubs` is the data source name, `demo` is the database user name with which to connect to the database, and `easysoft` is the password for the database user name.

The driver manager examines the connection attributes and loads the required driver (the driver is either named in the `DRIVER=` attribute or is looked up in a database of DSNs). From then on the driver manager relays ODBC calls to the driver and passes the result back to its caller.

A configured Easysoft ODBC-ODBC Bridge driver (on the client side) appears as an ordinary ODBC driver connected through the driver manager like any other.

At the server end, the OOB Server acts like an ordinary ODBC-compliant application:

- In Windows, it takes the form of a network service, configured to start automatically.
- Under Unix, it is called by `inetd` or run standalone, and is linked to the driver manager of your choice.

The Windows platform has a well-established driver manager (`odbc32.dll`) to which programmers link their code

On Unix, many installations do not have a driver manager, so Easysoft distribute the unixODBC driver manager and recommend that you use it with the Easysoft ODBC-ODBC Bridge on Unix platforms (see **"unixODBC" on page 243**).

REF

The unixODBC driver manager is an open source project sponsored by Easysoft, rather than a commercial Easysoft product, and is fast becoming a standard across the data access community.

Other driver managers are available, but Easysoft believe unixODBC is demonstrably the most flexible and reliable open source solution. unixODBC driver manager distributions can be found at **http://www.unixodbc.org**.

Setting up the OOB Server

In terms of the Easysoft ODBC-ODBC Bridge, the server is the machine where the ODBC driver for your database is located. The database itself may also be on this machine, although it can be located elsewhere.

To allow remote machines to access your database, you first need to create a data source on the server machine to make the database available to the OOB Server.

Before setting up a data source on your server machine, you must have successfully installed and licensed the OOB Server on this machine (see **"Installation" on page 25**).

Once the OOB Server is successfully installed, go to the section appropriate to your server platform:

- **"Windows server setup" on page 98**
- **"Unix server setup" on page 109**

Windows server setup

The OOB Server for Windows can connect to any system data source configured on your machine, given the necessary information.

When creating the data source on your server, you should use the ODBC driver suitable for your database (for example, to connect to a SQL Server database, use the SQL Server ODBC driver to create the data source).

The instructions in this section describe how to create a data source for the Microsoft Northwind database, which is shipped with Microsoft Access.

You should follow the same procedure to connect to your own database on your server machine.

To follow this example, you should have on your computer:

- Microsoft Access
- the ODBC driver for Microsoft Access (almost all Microsoft Access installations have this)
- a Microsoft Access database to connect to, such as `northwind.mdb`.

The first step is to open the Microsoft ODBC Data Source Administrator:

1. In **Control Panel**, double-click **Administrative Tools** and then **Data Sources (ODBC)**

   The **ODBC Data Source Administrator** dialog box is displayed.

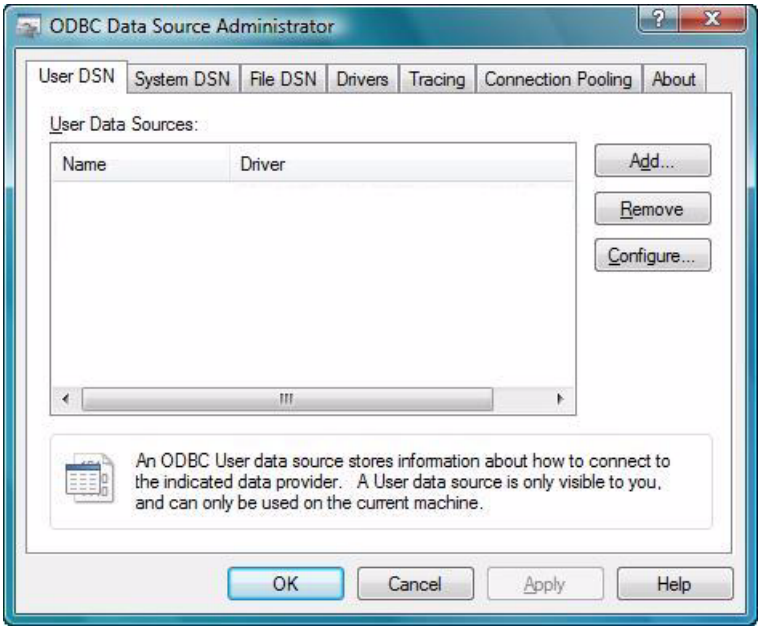


Figure 15: The ODBC Data Source Administrator User DSN tab

2. Select the **System DSN** tab:

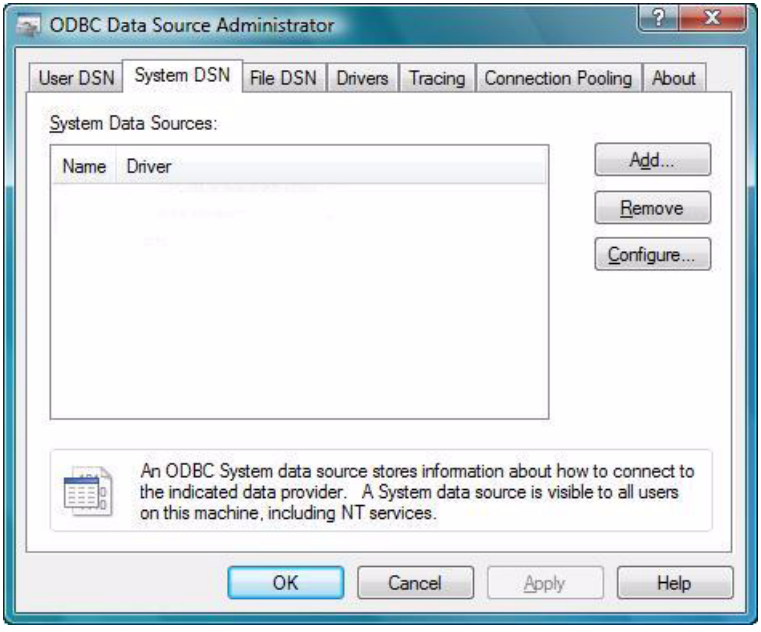


Figure 16: The ODBC Data Source Administrator System DSN tab

It is important to create a system DSN rather than a user DSN, which is only visible to the desktop user who created it.

Since the OOB Server runs as a service, user DSNs are not available to it.

3. Click **Add...** to add a new data source.

The **Create New Data Source** dialog box displays a list of drivers:

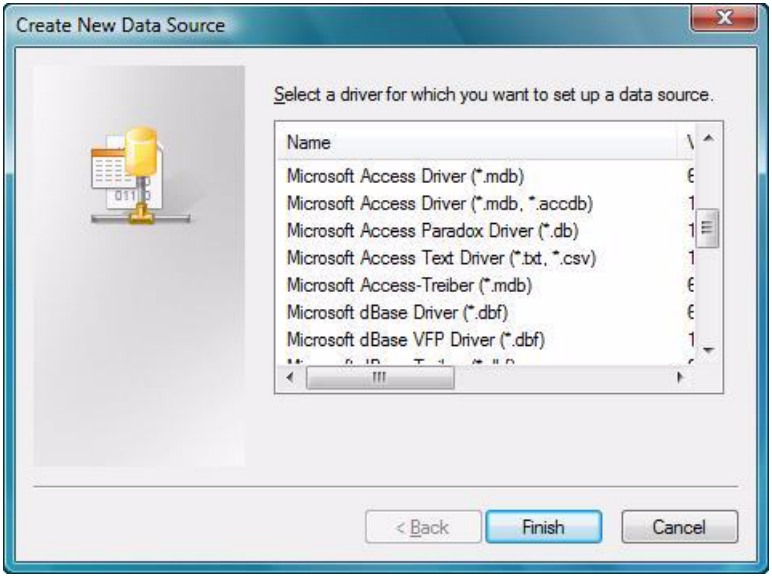


Figure 17: The Create New Data Source dialog box

4. Select `Microsoft Access Driver` and click **Finish**.

   The ODBC driver for Microsoft Access displays a dialog box for configuring the data source (this dialog box and the attributes you need to specify vary depending on the ODBC driver you are using).

NB	The Microsoft Access ODBC driver is NOT thread-safe unless run with Jet version 4. Previous versions require the OOB Server to be configured to run in multi-process mode, rather than the default multi-threaded mode (see the **Easysoft ODBC-ODBC Bridge FAQ**).

5. Enter your chosen name for this data source in the **Data Source Name** box (e.g. `"OOB Demo DSN"`).

6. In the **Description** field, enter something that would help a user faced with a choice of data sources (e.g. "`For demonstrating the OOB`"):

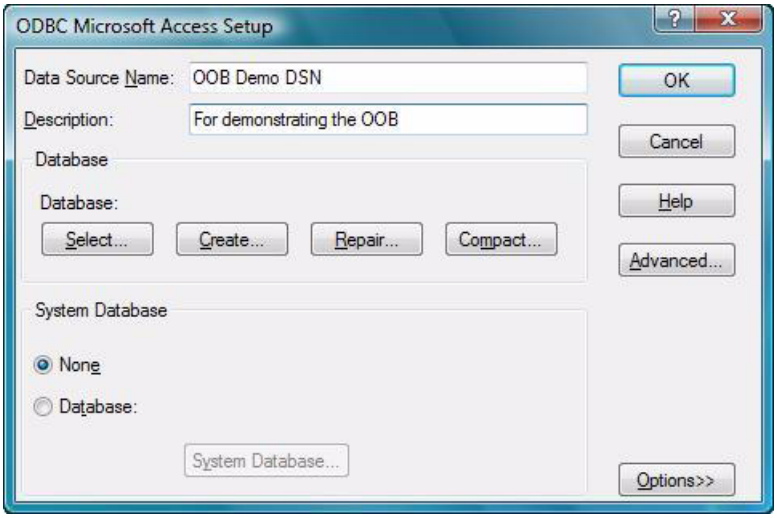


Figure 18: The ODBC Microsoft Access Setup dialog box

7. Click **Select...** to browse for the target database, select a database and click **OK**.

   If the `Microsoft Office\Office\Samples\Northwind.mdb` example database is not accessible, use any available database (preferably a small one).

NB	Note the data source name because it will need to be specified when a data source is created on the client machine.

8. Click **OK** to return to the ODBC Data Source Administrator window.

   Note:

   - the window now contains a line containing the new data source
   - the System DSN tab should be selected (if it is not, then remove the new data source, select the System DSN tab, and return to **step 3 on page 100**).

9. Click **OK**.

   You have now set up a system data source on your machine to a local database, making it visible to the OOB Server.

STARTING THE OOB SERVER IN WINDOWS

Before an OOB Client can connect to a data source on the server machine, the OOB Server must be running. In Windows, the

installation program configures the OOB Server to start automatically as a Service.

64-bit Windows

By default, only the 32-bit OOB Server is started automatically. (With the default configuration settings, both 32-bit and 64-bit OOB servers listening for incoming client connections on port 8888, it is not possible for both servers to run at the same time.) If you want to use the 64-bit OOB Server, you need to:

Stop the 32-bit OOB Server service (`Easysoft ODBC-ODBC Bridge Server`).

Start the 64-bit OOB Server service (`Easysoft ODBC-ODBC Bridge Server x64`).

To configure the 64-bit OOB Server service to start automatically when Windows starts:

1. In the Windows Services dialog box, double-click `Easysoft ODBC-ODBC Bridge Server`.

2. In the Easysoft ODBC-ODBC Bridge Server Properties dialog box, click Stop. In the Startup type list, choose **Manual**.

The 32-bit OOB Server service is no longer running and will not start when Windows starts.

3. In the Windows Services dialog box, double-click `Easysoft ODBC-ODBC Bridge Server x64`.

4. In the `Easysoft ODBC-ODBC Bridge Server Properties x64` dialog box, click Start. In the Startup type list, choose **Automatic**.

The 64-bit OOB Server service is now running and will start when Windows starts.

CHECKING THE OOB SERVER SERVICE UNDER WINDOWS

This procedure is not normally necessary, but should be followed if you are having problems connecting with the Easysoft ODBC-ODBC Bridge.

1. In **Control Panel**, double-click **Administrative Tools** and then **Services**.

   A list of your system's registered services is displayed.

NB	The `Services` dialog box looks different in the various versions of Windows, but the principles and the functionality are the same.

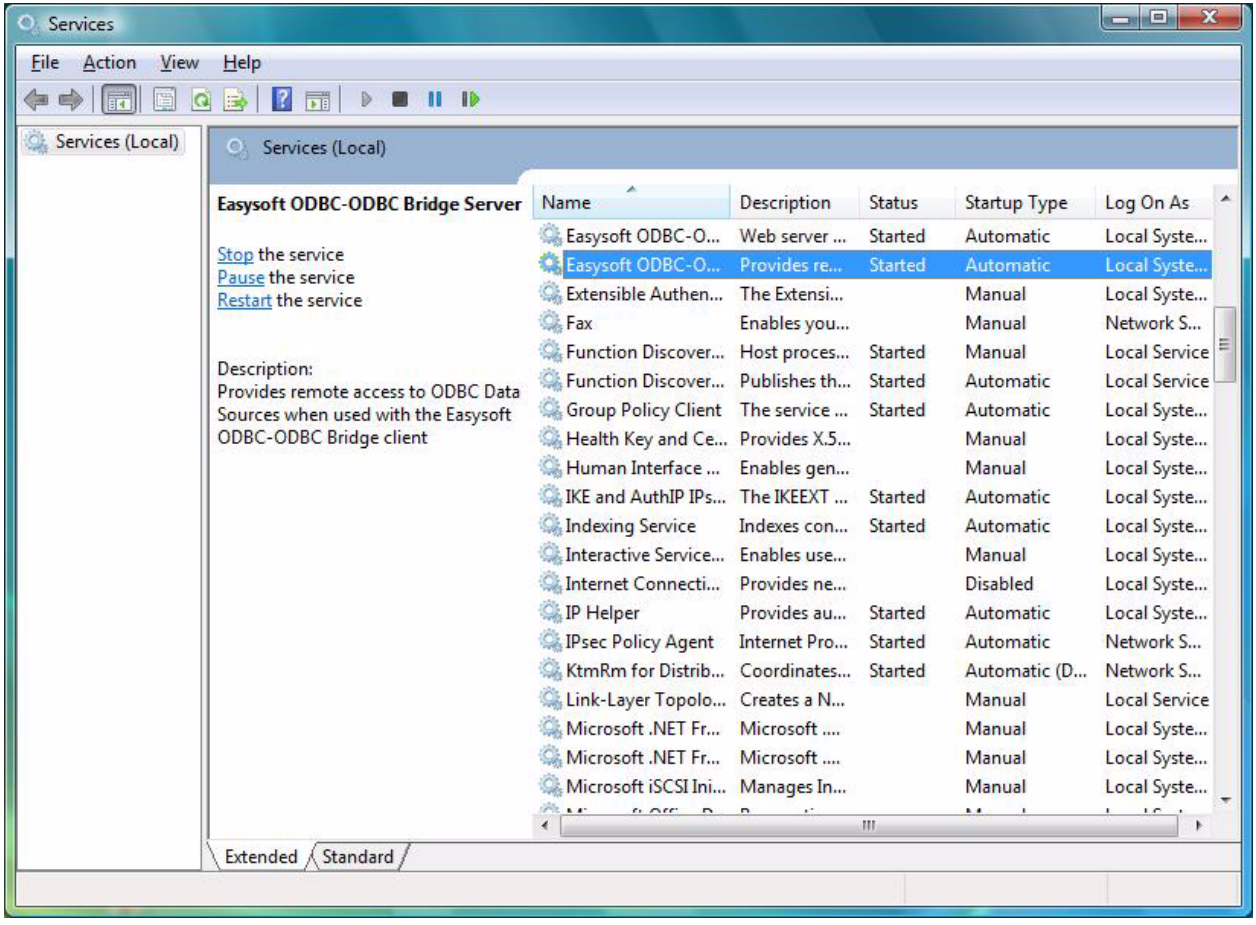


Figure 19: The Easysoft ODBC-ODBC Bridge Server Services entry

2. Find the entry for `Easysoft ODBC-ODBC Bridge Server`.
3. If the `Startup Type` field says `Disabled` then double-click `Easysoft ODBC-ODBC Bridge Server`.

4. In the resulting dialog box, select **Automatic** and then click **OK**.
5. If the `Status` field does NOT say `Started`, right-click `Easysoft ODBC-ODBC Bridge Server` and click **Start** to bring the Server on-line.
6. Click **Close**.
7. Close the **Control Panel**.

TESTING THE DATA SOURCE

You now have the OOB Server running on your Windows machine and a data source connecting to the database on the server.

Before setting up the OOB client, test that this data source is working, so that you can verify that the server side is functioning correctly, by running any other ODBC application on your Windows machine, linking to this data source and accessing its data.

Refer to the documentation supplied with your ODBC application if you are unsure how to link to a data source.

NB

If the database on your server machine is a Microsoft Access database then you cannot test the data source by linking to it from the Microsoft Access application.

You must connect to it via another ODBC application on your Windows machine, because although Microsoft Access is a multi-threaded application, the Microsoft Access ODBC driver is NOT thread-safe unless run with Jet version 4.

When using the Easysoft ODBC-ODBC Bridge and the Microsoft Access ODBC driver with a previous Jet version together, you must ensure that the OOB Server is running in multi-process mode (see **"MultiProcess" on page 212**).

Once you have established that the data source on your server machine is accessing data correctly, you can proceed to **"Testing the OOB Server" on page 116**.

Unix server setup

The OOB Server for Unix can connect to any system data source configured on the Unix machine, given the necessary information.

Easysoft recommend that you use the unixODBC driver manager, supplied with the Easysoft ODBC-ODBC Bridge, for setting up data sources on Unix.

> **NB** This section explains how to set up data sources using unixODBC as installed by the Easysoft ODBC-ODBC Bridge. If you choose to use a different driver manager, you should refer to the documentation with that driver manager for details of setting up data sources on Unix. For further information about using unixODBC, see **http://www.unixodbc.org**.

With unixODBC, you can create a data source by doing either of the following:

- directly adding the data source and its attributes into a configuration file (`odbc.ini`)
- if you are running an X server you can create a data source using the graphical ODBC Data Source Administrator (`ODBCConfig`). `ODBCConfig` is included with the unixODBC distribution supplied with OOB. You may already have it with an existing unixODBC distribution.

CREATING A DSN BY EDITING A CONFIGURATION FILE

With unixODBC, data sources are stored in a configuration file called `odbc.ini`.

System data sources are stored in the file output by *<UNIXODBCBINDIR>*`/odbcinst -j`. In the following sample output, `odbcinst -j` shows that System data sources are stored in `/etc/odbc.ini`.

```
unixODBC 2.2.12
DRIVERS............: /etc/odbcinst.ini
SYSTEM DATA SOURCES: /etc/odbc.ini
USER DATA SOURCES..: /home/tim/.odbc.ini
```

NB By default, you must be logged in as `root` to edit a system data source defined in `/etc/odbc.ini`, but user data sources created in an `.odbc.ini` file in a home directory are visible to an individual user only (i.e. the `Logon User` passed from the client to the server).

Each section starts with a data source name in square brackets [ ], followed by a number of *attribute=value* pairs.

The attributes that you need to specify vary depending on which ODBC driver you are using to connect to the local database.

A sample data source using the PostgreSQL driver is of the format:

```
[MAIN]
Description = Main data on Admin box
Driver = PostgreSQL
Database = main
```

```
Servername = localhost
UserName =
Password =
Port = 5432
Protocol = 6.4
ReadOnly = No
RowVersioning = No
ShowSystemTables = No
ShowOidColumn = No
FakeOidIndex = No
ConnSettings =
```

NB

unixODBC uses the `Driver` attribute to look up the driver in the `odbcinst.ini` file and locate the shared object to use as the ODBC driver.

The location of the `odbcinst.ini` file is shown by `odbcinst -j`.

Refer to the documentation with your ODBC driver for full details of the attributes required to define a data source.

CREATING A DSN USING THE ODBC DATA SOURCE ADMINISTRATOR

To create a data source using the graphical ODBC Data Source Administrator supplied with unixODBC:

1. On a machine running X, log in as `root`.
2. In a terminal emulator window, change into the `<InstallDir>/easysoft/unixODBC/bin` directory.
3. Type `./ODBCConfig` <*Enter*>.

   The ODBC Data Source Administrator opens.
4. Click the **System DSN** tab to create a data source which is available to any user or service that logs into this machine.
5. Click **Add** to create a new data source.

   The **Adding a New Data Source** dialog box displays the available drivers.
6. Select the driver required to connect to the database and click **OK**.

   A configuration dialog box specific to that driver is now displayed, such as the PostgreSQL data source:

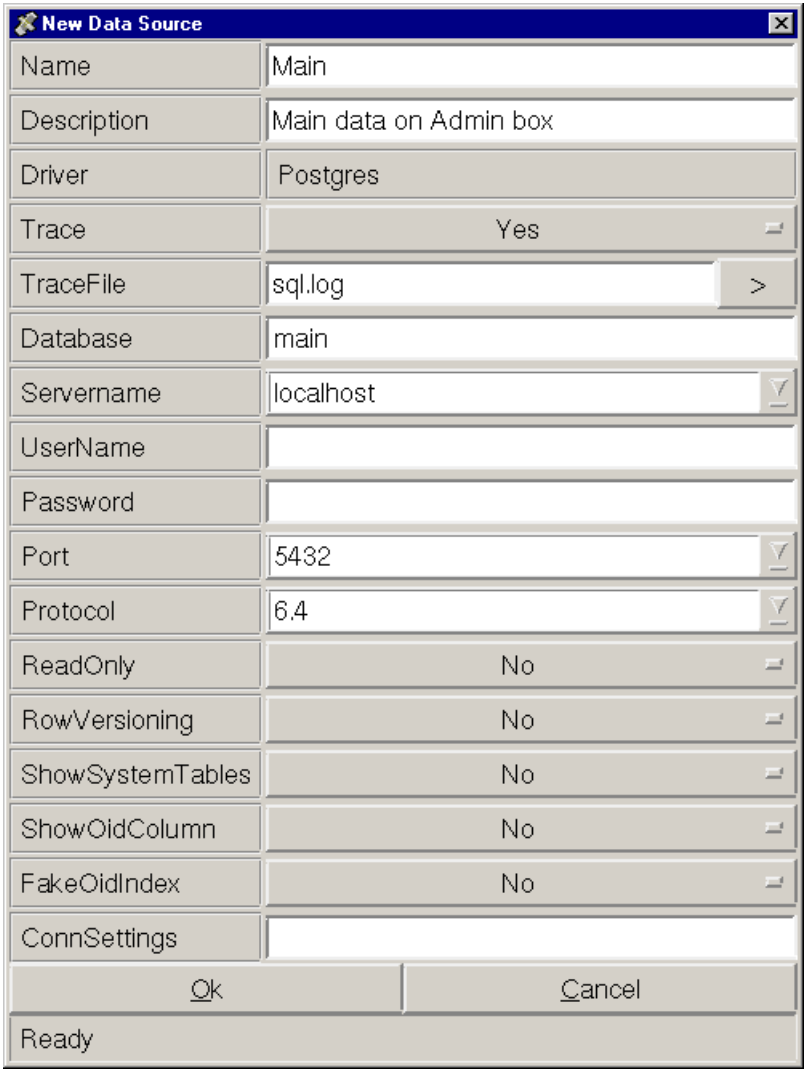


Figure 20: The Configuration dialog box for a PostgreSQL data source

Refer to the documentation with your ODBC driver for full details of the attributes you need to specify on this dialog box.

7. Click **OK** when you have specified the data source attributes you require and then close the ODBC Data Source Administrator.

TESTING THE DATA SOURCE

Before setting up the OOB Client, the new data source and the OOB Server setup should be verified with either an ODBC application available on your Unix machine or the unixODBC `isql` utility.

To use `isql` to test the data source:

1. Change into the *<InstallDir>*`/easysoft/unixODBC/bin` directory or the *<unixODBCInstallDir>*`/bin` directory.
2. Type:

   ```
   ./isql -v data_source_name
   ```

   For example, to connect to the PostgreSQL data source illustrated earlier in this section, you would type:

   ```
   ./isql -v main
   ```

NB	The `-v` option displays ODBC diagnostic messages.

NB	On some platforms, it may be necessary to run `isql` by using `isql.sh`. To do this, type `./isql.sh` instead of `./isql`. `isql.sh` is a wrapper program located in the *<InstallDir>*`/easysoft/unixODBC/bin` directory which sets up the dynamic linker before running `isql`.

If your server DBMS requires authentication, you should include the DBMS user name and password arguments in the `isql` command (see **"When the DBMS requires authentication" on page 92**).

3. Once connected, either:
   - type an SQL statement to query the data, such as:

     `select * from` *`tablename`*

     where *`tablename`* is a table in that database

   – OR –

   - type `help` to get a list of tables in the database.
4. To leave `isql` and return to the system prompt, press <*Enter*>.

   You can now proceed to **"Testing the OOB Server" on page 116**.

Testing the OOB Server

OOBPING

If you believe that there are problems with the connection between the application and the server then you can use `oobping` to help diagnose and fix any errors before proceeding to **"Setting up the OOB Client" on page 131**.

`oobping` is a small program distributed with the Easysoft ODBC-ODBC Bridge, and is a valuable tool for checking Easysoft ODBC-ODBC Bridge connectivity and diagnosing connection problems or connection timing issues:

In Windows distributions the `oobping.exe` program is located in the `<installdir>\Easysoft\Easysoft ODBC-ODBC Bridge` directory.

In Unix and Mac OS X distributions there are two versions of `oobping` located in the `/usr/local/easysoft/bin` directory.

- `oobpings` (a statically linked version not depending on anything other than `libc`)
- `oobpingd` (a dynamically linked version linked against the `libesoobclient` shared object)

NB	To use `oobpingd` you may need to set and export your `LD_LIBRARY_PATH/LD_RUN_PATH/LIBPATH` to include `/usr/local/easysoft/oob/client` and `/usr/local/easysoft/lib`.

`oobping` has the following command line:

```
oobping [-h host | -d ODBC_connection_string] {-t
port} {-u osuser -p ospassword} {-e}
```

where:

`-h host`

The name or IP address of the machine where the OOB Server is located.

`-d ODBC_connection_string`

An ODBC connection string consisting of a list of semi-colon separated `attribute=value` pairs, as defined by ODBC.

e.g. `DSN=test;UID=john;PWD=smith;`

If the `-u` and/or `-p` attributes are also specified as well as `-d` then "`LogonUser=xxx;LogonAuth=yyy;`" (where `xxx` and `yyy` are the values specified for `-u` and `-p`) will be added to the end of the connection string.

NB	When running `oobping` from a Unix shell, you may have to enclose the ODBC connection string, user name and password with single quotes. Do this to protect any spaces or special characters that the shell might interpret these settings contain. When running `oobping` from an MS-DOS window, use double quotes to protect spaces.

`-t port`

The port on which the OOB Server is listening.

`-u osuser`

A valid user name on the "host" operating system.

`-p ospassword`

A password for the user specified with the `-u` attribute.

`-e`

Show the amount of time (in seconds) to complete the requested operation.

WORKED EXAMPLES

The following examples illustrate the ways in which `oobping` can be used to investigate connection issues:

- **Example 1: Check the OOB Server is running on the correct machine and listening on the correct port**
- **Example 2: Check OOB Server authentication**
- **Example 3: Check connectivity to a specified remote data source**
- **Example 4: Checking connection times**

NB	The Easysoft ODBC-ODBC Bridge installation uses `oobping` to test the connection to the remote OOB Server.

Example 1: Check the OOB Server is running on the correct machine and listening on the correct port

`oobping` connects to port `8888` on the machine `myserver`, where `OOB Server version 1.1.0.00` is reported as running.

```
oobping -h myserver -t 8888


Host: myserver, Port: 8888
Attempting connection...OK
Examining Server...
    OOB Server Version: 1.1.0.00
    OOB Server Name: OOB
```

NB `oobping` defaults to port `8888`, so the `-t` attribute can be omitted.

If the wrong port is specified or some other service is listening on the specified OOB Server port, then you will get a variety of errors, such as when pointing `oobping` at an SMTP server on port `25`:

```
oobping -h myserver -t 25
Host: myserver, Port: 25


Failed to receive data
Packet (size=842149920) too big for buffer
(size=256)
```

If there is nothing listening on the specified port then you will get a connection refused error and you should check that the OOB Server is running and is configured to use the specified port.

e.g.

```
oobping -h myserver -t 8889
Host: myserver, Port: 8889


Connection refused, connect(), after 5 attempts
```

If you have specified access control rules in the OOB Server then you might see an error such as this:

```
oobping -h myserver -t 8888
Host: myserver, Port: 8888


Client denied access due to access control
rule.
```

Here the machine you are running `oobping` on has been denied access to the OOB Server and you should check the access control rules on the security page of the OOB Web Administrator.

Example 2: Check OOB Server authentication

Once you are sure a connection can be made to the OOB Server with `oobping` (as in example 1) you can check OOB Server authentication.

The `-u` and `-p` arguments to `oobping` allow you to specify a valid operating system user name and password.

> **NB**
>
> If you have disabled authentication in the OOB Server then any user name and password will work no matter what you enter.
>
> The values specified with `-u` and `-p` are equivalent to the OOB Client DSN attributes `LogonUser` and `LogonAuth`.

e.g.

```
    oobpings -h myserver -t 8888 -u 'A User' -p
'mypassword'

    Host: myserver, Port: 8888

    Attempting connection...OK

    Examining Server...

        OOB Server Version: 1.1.0.00

        OOB Server Name: OOB

    Trying to authenticate...OK
```

If there is something wrong with the user name or password then the output will look something like this:

```
    oobpings -h myserver -t 8888 -u 'A User' -p
'mypassword'

    Host: myserver, Port: 8888

    Attempting connection...OK

    Examining Server...

        OOB Server Version: 1.1.0.00

        OOB Server Name: OOB

    Trying to authenticate...Fail

    Authentication failure (error number 1326)
```

In this case the error returned by the remote user name/password authentication service is `1326` (as the server was on Windows, this is a Windows error code).

All the common Windows error codes can be found in the **Easysoft ODBC-ODBC Bridge FAQ**.

Example 3: Check connectivity to a specified remote data source

Once you have completed examples 1 and 2 you should have:

- the name of a server where an OOB Server is running
- the port it is listening on
- a valid operating system user name and password
- be enabled in the OOB Server access control rules

You can now check connectivity to a specified remote data source (DSN), which can be done in two ways with `oobping`.

The simplest way is to add a local DSN to your `odbc.ini` file and then specify the DSN in the `-d` argument as "`DSN=dsnname`". This method tests you have defined the local OOB Client DSN correctly and put the definition in the right file.

An alternative method is to specify all the connection attributes in the `-d` argument, not just the DSN.

e.g.

Suppose as a result of examples 1 and 2 you have the following information:

```
Server = myserver

Port = 8888

LogonUser = me

LogonAuth = mypassword
```

Now enter the name of the target DSN on `myserver` in the `TargetDSN` attribute.

NB	Note that the target DSN **MUST** be a remote SYSTEM data source, as you cannot access USER data sources.

If the database also needs login information then the database user name and password are specified using the `TargetUser` and `TargetAuth` attributes.

If you want to test a DSN in your `odbc.ini` file then it would look something like:

```
[mydsn]

ServerPort = myserver:8888

LogonUser = me

LogonAuth = mypassword
```

```
TargetDSN = mysystemdsn

TargetUser = dbusername

TargetAuth = dnpassword
```

> **NB** "Driver=OOB" must be included in the DSN in order to use the **unixODBC** driver manager, but this is not needed when using oobping.

You can run oobping as follows:

```
    oobping -d "DSN=mydsn;"

    Using Connection string :

    DSN=mydsn;

    Connected OK


    01000:1:5701:[NetConn:
032bc620][Microsoft][ODBC SQL Server Driver]

    [SQL Server]Changed database context to 'pubs'.

    01000:2:5703:[NetConn:
032bc620][Microsoft][ODBC SQL Server Driver]

    [SQL Server]Changed language setting to
us_english.

    OutConnectionString:


DSN=mydsn;UID=dbusername;PWD=dbpassword;SERVERPORT
=myserver:8888;
```

```
TARGETDSN=mysystemdsn;LOGONUSER=me;LOGONAUTH=mypas
sword;
```

```
    Connected to database:  pubs
    DBMS Name:  Microsoft SQL Server
    Driver Name:  esoobclient
    Driver Version:  01.00.0043
    Disconnecting
```

The ODBC connection string "`DSN=mydsn`" was passed via `oobping` to the OOB Client, but the OOB Client did not receive sufficient attributes in the connection string.

In order to define what it should do, the OOB Client looked up the DSN '`mydsn`' in the `odbc.ini` file, where it obtained the additional attributes `ServerPort`, `LogonUser`, `LogonAuth`, `TargetDSN`, `TargetUser` and `TargetAuth`.

The OOB Client then connected to the OOB Server on `myserver` via port `8888`, logged you in with `LogonUser/LogonAuth` values and finally connected via ODBC to the remote data source '`mysystemdsn`'.

NB	Although in this example the DSN pointed to Microsoft SQL Server and informational diagnostics reported the language as "`us_english`" and the database as "`pubs`", not all databases will return informational messages on the connection.

The `OutConnectionString` is a string returned by the OOB Client which you can use to connect to this data source again and the final messages show the database name, DBMS name, driver name and driver version.

Instead of specifying just the name of a DSN to `oobping` and defining the other connection attributes in the `odbc.ini` file you can specify all the connection attributes in one go and not use an `odbc.ini` file.

The `OutConnectionString` shows you what connection string you could have passed to the OOB Client to connect without a DSN (if you remove the "`DSN=mydsn;`") to produce the same result.

e.g.

```
    oobping -d
"UID=dbusername;PWD=dbpassword;SERVERPORT=myserver
:8888;
TARGETDSN=mysystemdsn;LOGONUSER=me;LOGONAUTH=mypas
sword;"
```

You might be slightly confused as to why `TargetUser/TargetAuth` is specified in the `odbc.ini` file, but `UID/PWD` in the connection string.

Strictly speaking, the ODBC defined attributes are `UID/PWD` and they are passed to the database for authentication.

As far as ODBC connection strings are concerned, `UID/PWD` and `TargetUser/TargetAuth` are synonymous in the Easysoft ODBC-ODBC Bridge, but if specified in the `odbc.ini` file you should always use `TargetUser/TargetAuth`.

Now you know how to use `oobping` with connection strings, here are some examples of common problems you might see:

```
    oobping -d
"UID=dbuser;PWD=dbauth;TargetDSN=test;LogonUser=me
;LogonAuth=mypassword"
    Using Connection string :


UID=dbuser;PWD=dbauth;TargetDSN=test;LogonUser=me;
LogonAuth=mypassword


    IM002:1:0:[Easysoft ODBC (Client)]
    Data source not found and no default driver
specified
    HY000:2:0:[Easysoft ODBC (Client)]
    general error: Missing attribute(s): SERVERPORT
```

The initial diagnostic, `IM002` is defined by ODBC, but as it is rather vague, the OOB Client added a secondary more helpful diagnostic.

Here the Easysoft ODBC-ODBC Bridge connection attribute '`ServerPort`' was omitted and so the OOB Client did not know which server to connect to.

```
    oobping -d
"ServerPort=myserver:8888;UID=dbuser;PWD=dbauth;Ta
rgetDSN=test;LogonUser=me;Log
onAuth=mypassword"
    Using Connection string :
```

```
ServerPort=myserver:8888;UID=dbuser;PWD=dbauth;Tar
getDSN=test;LogonUser=me;LogonAuth=mypassword
```

```
    28000:1:18456:[][Microsoft][ODBC SQL Server
Driver][SQL Server]
    Login failed for user 'dbuser'.
```

This Microsoft SQL Server error suggests the database password '`dbauth`' is not correct for the database user '`dbuser`' or that '`dbuser`' is not a valid user, and illustrates the importance of identifying which component is reporting an ODBC error.

You should examine the components displayed in `[]`, where the one furthest to the right is the reporting component and as you move left through the components you are moving closer to the OOB Client.

For example, if you specified a `TargetDSN` which did not exist on the server machine then the driver manager on the remote machine would be the component reporting the error and so the diagnostic error message (for Windows) would be:

```
    IM002:1:0:[][Microsoft][ODBC Driver Manager]
    Data source name not found and no default
driver specified
```

The other important thing to note here is that `oobping` outputs any ODBC diagnostics as:

```
    ODBC State : diagnostic sequence : ODBC native
error: text
```

The native error code is specific to the component reporting the error, and can be referenced in the Microsoft SQL Server documentation, which should tell you more accurately in which circumstances this error is reported.

Example 4: Checking connection times

`oobping` includes the `-e` option, which times the requested operation and can be invaluable in diagnosing a slow connection and determining which phase the problem is occurring in.

These operations may also use the `-e` switch.

e.g.

```
oobping -e -h myserver -t 8888
 Host: myserver, Port: 8888
 Attempting connection...OK
 Examining Server...
     OOB Server Version: 1.1.0.00
     OOB Server Name: OOB
 Time for execution: 0.16s
```

If this is repeated with the `-u` and `-p` attributes you can work out the extra time required to perform the operating system logon:

```
oobping -e -h myserver -u myosuser -p mypassword
 Host: myserver, Port: 8888
 Attempting connection...OK
 Examining Server...
     OOB Server Version: 1.1.0.00
     OOB Server Name: OOB
 Trying to authenticate...OK
 Time for execution: 0.52s
```

This example clearly demonstrates the extra time required to authenticate a user.

NB

When connecting to a server in part of a Windows domain, the server may need to contact the Primary Domain Controller (PDC). This can take a significant amount of time.

In addition, if you install the OOB Server on the same machine as Microsoft SQL Server, the OOB Server and SQL Server compete for CPU time. It is often quicker to put the OOB Server on a different Windows machine.

Setting up the OOB Client

The Easysoft ODBC-ODBC Bridge Client is the machine running the ODBC application with which you want to access the data on the server.

To allow an ODBC application on the client machine to access data on the remote server, you need to create a data source on the client.

This data source uses the Easysoft ODBC-ODBC Bridge Client driver and specifies the attributes required to connect to the data source on the remote server.

Before setting up a data source on your client machine, you must have successfully installed the OOB Client on this machine, and set up the server side of the Easysoft ODBC-ODBC Bridge.

Instructions for installing the OOB Client on Windows, Unix, and Mac OS X platforms are provided in **"Installation" on page 25**.

If you have not already set up the server side of the Easysoft ODBC-ODBC Bridge, go to **"Setting up the OOB Server" on page 97**.

Once the OOB Client is successfully installed and your server is set up correctly, go to the section appropriate to your appropriate client platform:

- **"Windows client setup" on page 132**
- **"Unix client setup" on page 153**
- **"Mac OS X client setup" on page 165**

Windows client setup

This section explains the steps you should take to connect an ODBC application on a Windows machine (where the OOB Client is installed) to the data source on your remote server (where the OOB Server is installed).

The instructions in this section show you how to connect to an example data source on a server at Easysoft, but you should follow the same procedure to connect to the data source on your own server.

The OOB Client is an ODBC driver. You configure a data source using the OOB Client in the same way that you configure a data source using any other ODBC driver, so many of the steps in this section are similar to those in **"Windows server setup" on page 98**.

64-bit Windows

64-bit Windows machines support both 32-bit and 64-bit ODBC drivers.

If you want to use a 32-bit ODBC application, you need to use the 32-bit OOB Client. If you want to use a 64-bit ODBC application, you need to use the 64-bit OOB Client.

There is both a 32-bit and a 64-bit version of the ODBC Data Source Administrator, which is used to configure ODBC data sources. To access the 32-bit ODBC Data Source Administrator, in the Windows Run dialog box, type:

```
%windir%\syswow64\odbcad32.exe
```

The 64-bit ODBC Data Source Administrator is located in Control Panel under Administrative tools.

OOB Client data sources created in the 32-bit ODBC Data Source Administrator will specify the 32-bit version of the OOB Client. OOB Client data sources created in the 64-bit ODBC Data Source Administrator will specify the 64-bit version of the OOB Client.

System data sources created in the 32-bit ODBC Data Source Administrator are only visible to 32-bit applications. If you want to create an OOB Client System data source for use with a 32-bit application, use the 32-bit ODBC Data Source Administrator therefore. Likewise, System data sources created in the 64-bit ODBC Data Source Administrator are only visible to 64-bit applications.

64-bit Windows

(The reason for this is that System data sources created in the 64-bit ODBC Data Source Administrator are stored in a registry key called `\HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI`. System data sources created in the 32-bit ODBC Data Source Administrator are stored in a registry key called `\HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\ODBC\ODBC.INI`. The WOW64 layer intercepts registry calls to `HKEY_LOCAL_MACHINE\Software` that are made by 32-bit applications, and then redirects them to the `HKEY_LOCAL_MACHINE\Software\WOW6432node` key.)

User data sources are visible to both 32-bit and 64-bit applications, irrespective of the version of ODBC Data Source Administrator they were created in. If a 64-bit application connects to an OOB Client User data source created in the 32-bit ODBC Data Source Administrator, the 64-bit version of the OOB Client will be used. Likewise, a 32-bit application that connects to a 64-bit OOB Client User data source will use the 32-bit version of the OOB Client.

Firstly, open the Microsoft Data Source Administrator:

1. In **Control Panel**, double-click **Administrative Tools**, and then double-click **Data Sources (ODBC)**.

   The ODBC Data Source Administrator opens.

2. To create a data source that is only available to the user currently logged into this machine, select the **User DSN** tab.

   – OR –

   To create a data source that is available to any user who logs into this machine, select the **System DSN** tab.

3. Click **Add...** to add a new data source.

The **Create New Data Source** dialog box is displayed, containing a list of drivers:

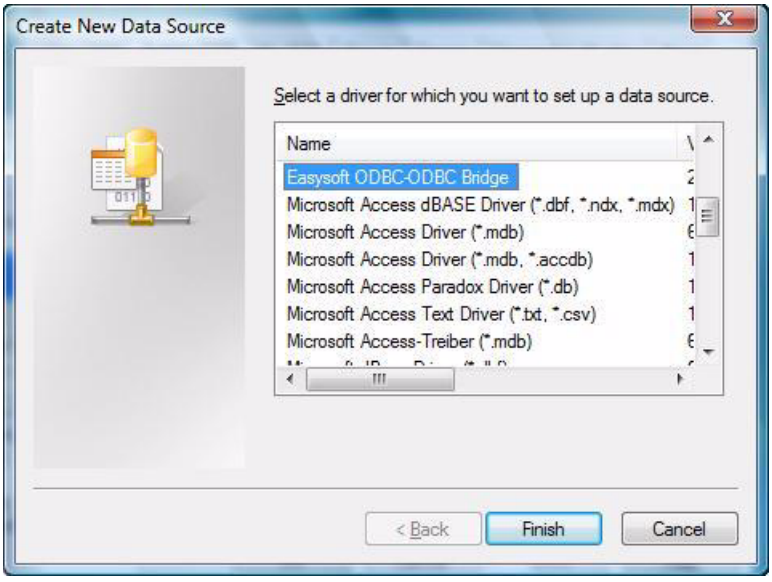


Figure 21: The Create New Data Source dialog box

4. Select `Easysoft ODBC-ODBC Bridge` and click **Finish**.

A a DSN configuration dialog box is displayed:

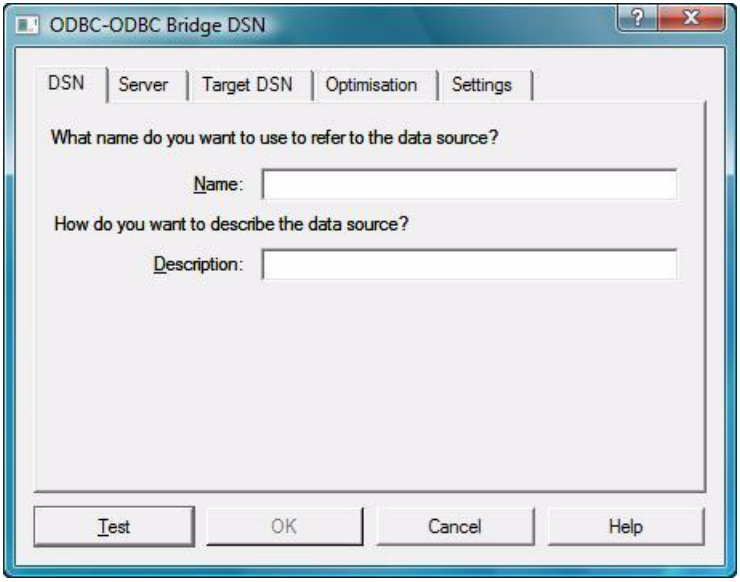


Figure 22: A blank Easysoft ODBC-ODBC Bridge DSN dialog box

THE EASYSOFT ODBC-ODBC BRIDGE DSN DIALOG BOX

The attributes on this dialog box are split into five tabs, arranged by functionality, from left to right:

- DSN The name and description that identify the data source.
- Server Settings and authentication details for the server that the OOB server is installed on.
- Target DSN Settings and authentication details for the remote System data source.

- Optimisation OOB performance settings.
- Settings Settings that let you override the default OOB behaviour.

The **Test** button allows you to check that the client is able to connect to the specified server data source.

> **NB** If you click **Test** before specifying a server and port (the **Servers** setting) or a remote System DSN (the **TargetDSN** setting), you'll be prompted to enter these required settings.

D S N

5. In the **Name** box, enter a name for this data source.

   Choose carefully because you will not be able to change this after pressing **OK**.

> **NB** A `demo` data source will probably already exist because it is set up during the OOB Client installation. It is required if you intend to use the `demo.exe` client application.

6. In the **Description** box, enter something that would help a user faced with a choice of data sources.

SERVER

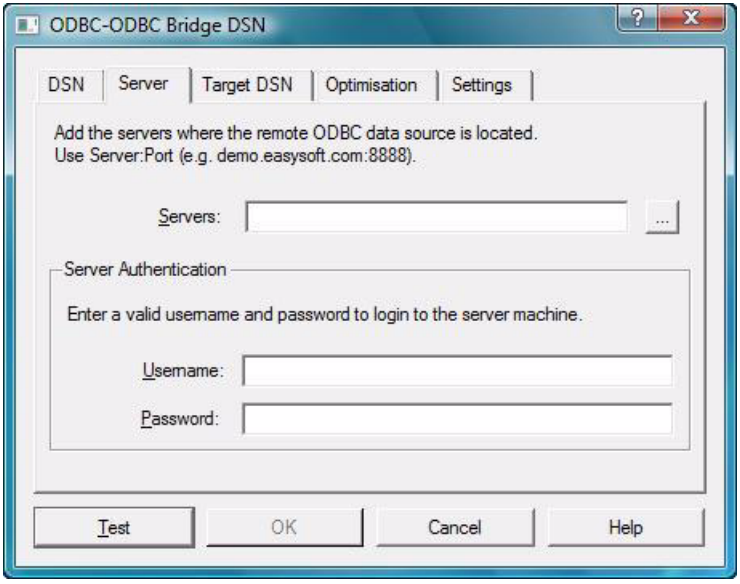


Figure 23: The Easysoft ODBC-ODBC Bridge DSN dialog box - Server tab

7. In the **Servers** box, type the name of the machine on which the OOB Server is running and the port that it is listening on.

   Use the following format:

   server:*port*

   The default port is 8888. Specify this port unless you know that the OOB Server is listening on another port.

   For example, if you are connecting to the Easysoft `demo` server, type:

   ```
   demo.easysoft.com:8888
   ```

If the remote System DSN is available on more than one OOB Server, you can define a primary OOB Server for the DSN and additional fallback OOB Servers. For more information about defining multiple OOB Servers, see **"Client support for Fallback OOB Servers" on page 189**.

8. In the **Username** and **Password** boxes, enter a valid logon account and password for the machine on which the OOB Server is running (if required).

   The OOB Server carries out all activities as this user.

NB	If your server is part of a Windows domain, you may need to include the domain name with the user name, using the format `domain/user name`. For example: `admin/John Smith`. If you have defined multiple OOB Servers for this data source, the username and password must be valid for each OOB Server that you specify.

   – OR –

   If you are connecting to the Easysoft `demo` server, enter `demo` and `easysoft` respectively in these boxes.

9. To test the connection to the OOB server and make sure the authentication details you entered are valid, click the **...** button.

10. In the Servers dialog box, click the **Test**.

    If the test succeeds, in the **Servers** list, a green tick displays next to the server.

    – OR –

    If a connection can't be made, in the **Servers** list, a red cross displays next to the server. The Test Log output shows the reason why the test failed in red text. For help on how to resolve the connection problem, click **Help**.

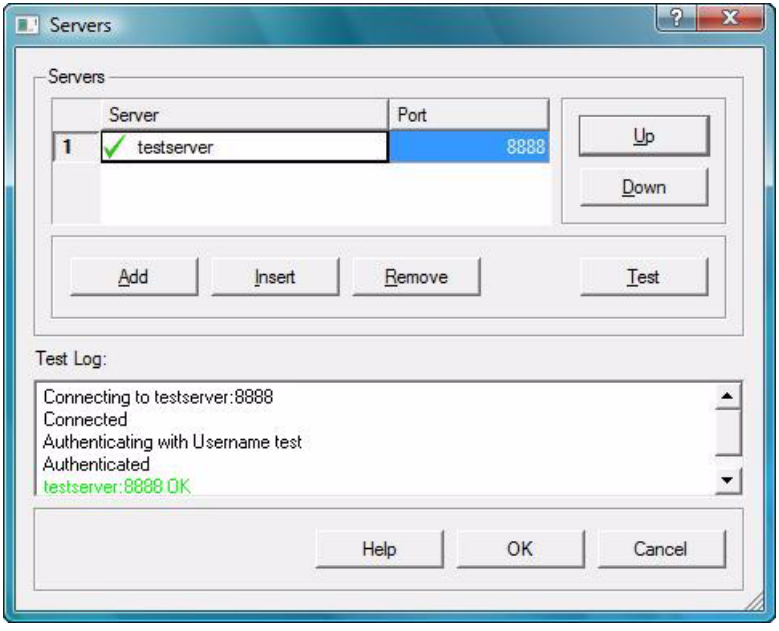


Figure 24: The Servers dialog box dialog box showing output from a successful connection to an OOB server.

TARGETDSN

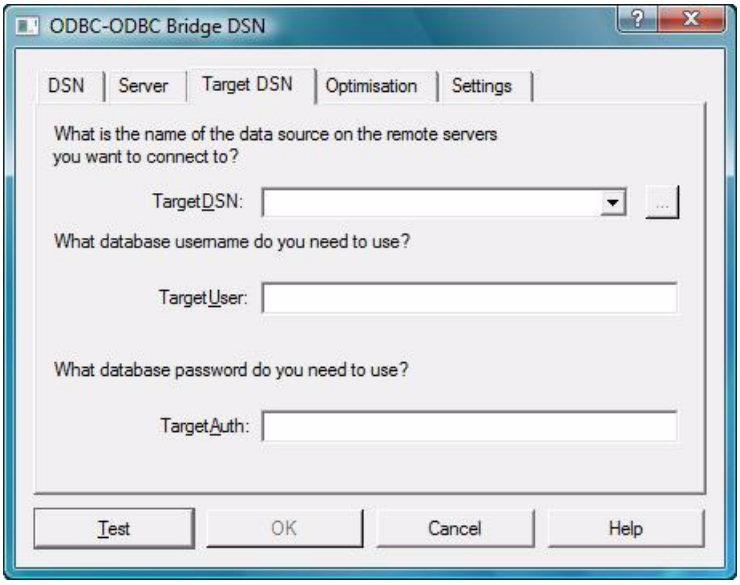


Figure 25: The Easysoft ODBC-ODBC Bridge DSN dialog box - Target DSN tab

11. In **TargetDSN**, enter the data source name on your remote machine.

    – OR –

    Click the **...** button to retrieve the data sources that are available on the remote machine, then choose one from the list. Note that the `AllowDSNBrowse` OOB Server configuration parameter controls whether you can retrieve the list of remote data sources. If this facility has been disabled, you'll need to enter the data source name instead.

– OR –

If connecting to the Easysoft `demo` server, enter `pubs`.

64-bit Windows

If you are connecting to a 32-bit Windows OOB Server, you must specify a 32-bit ODBC driver data source. If you are connecting to a 64-bit Windows OOB Server, you must specify a 64-bit ODBC driver data source.

If you attempt to use a 32-bit OOB Server with a 64-bit ODBC driver data source, or a 64-bit OOB Server with a 32-bit ODBC driver data source, the connection will fail with the error:

```
The specified DSN contains an architecture
mismatch between the Driver and Application
```

12. If your remote data source (i.e. the database itself) requires a user name and password apart from the user logon account for the machine, then enter these in **TargetUser** and **TargetAuth**.

    – OR –

    If you are connecting to the Easysoft demo data source, enter `demo` and `easysoft` in **TargetUser** and **TargetAuth**.

    – OR –

    If your data source does not need separate authentication details then leave these fields blank.

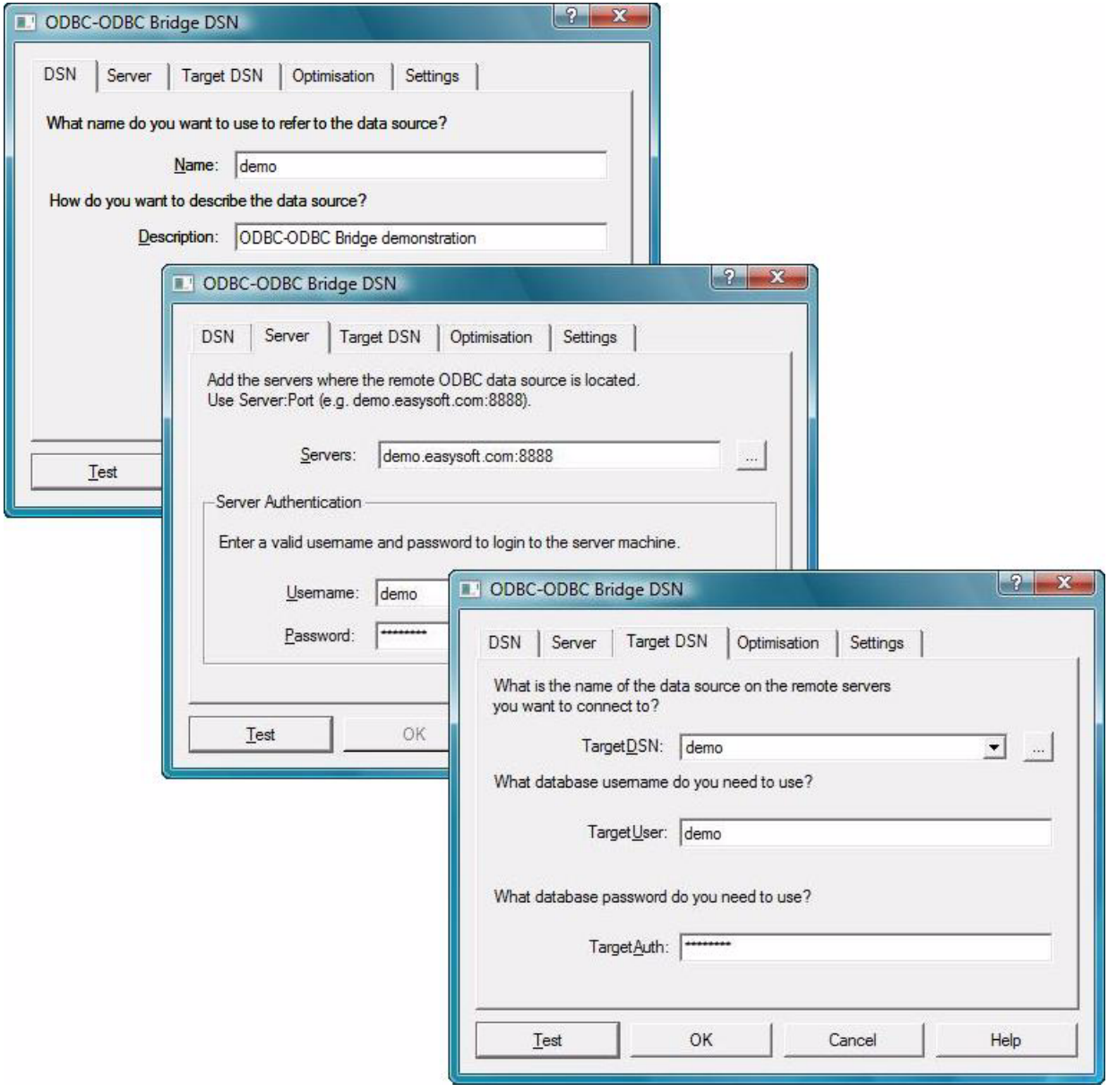


Figure 26: The DSN set up for the Easysoft demo data source

OPTIMISATION

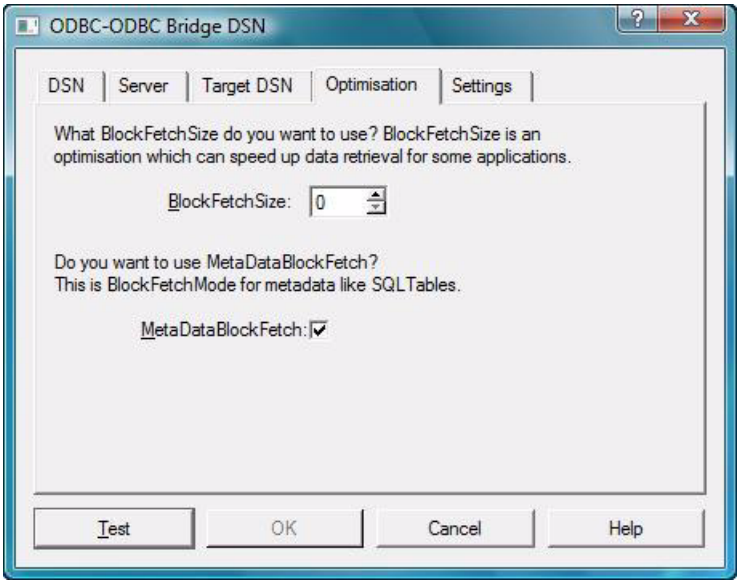


Figure 27: The Easysoft ODBC-ODBC Bridge DSN dialog box - Target DSN tab

For more information about the options on the Optimisation tab, see **"Attribute Fields" on page 174**. If you are connecting to the Easysoft demo data source, leave the options set to their default values.

SETTINGS

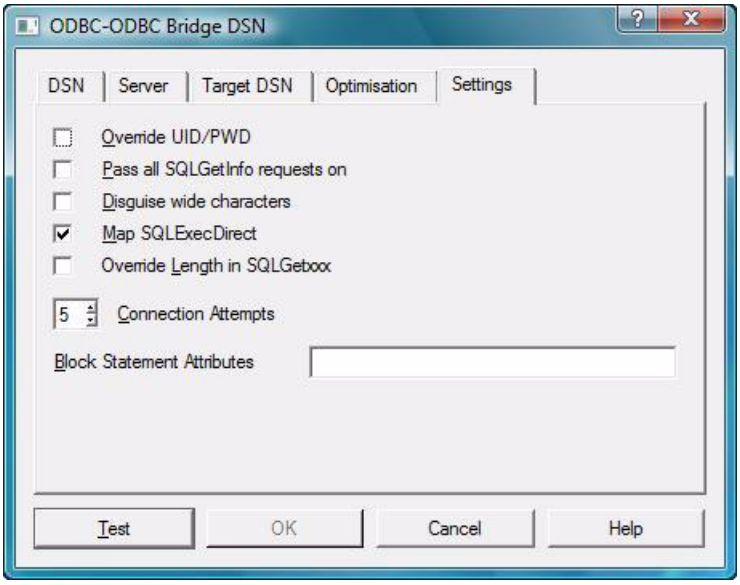


Figure 28: The Easysoft ODBC-ODBC Bridge DSN dialog box - Settings tab

For more information about the options on the Settings tab, see **"Attribute Fields" on page 174**. If you are connecting to the Easysoft demo data source, leave the options set to their default values.

CHECK YOUR VALUES

13. Now click **Test**.

    The OOB Test dialog box displays, showing the current dialog box settings.

14. In the OOB Test dialog box, click **Test**.

Using the current dialog box dialog box settings, the OOB Client tests that:

- A connection can be made to the OOB server on the specified server and port.
- The server operating system authentication details are valid.
- The remote System DSN exists.
- The database authentication details are valid (if required).

For a comprehensive description of the test process, click the Help button. Note that you can test the server connection and authentication details separately. To test just your server settings, see **step 10 on page 140**.

15. If a stage in the connection process cannot be successfully completed, the OOB client stops the test and displays diagnostic output. The output identifies the test that failed and provides the reason for the failure. For information on how to solve connection problems, click **Help**.

    – OR –

    If you see "`Test Successful`" the connection process has been successfully completed.

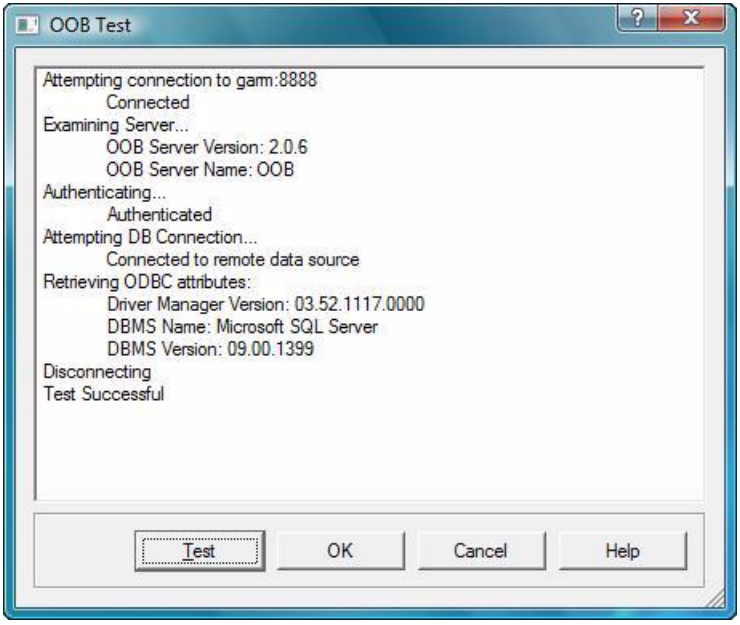


Figure 29: The OOB Test dialog box showing test results generated from valid DSN settings

16. Click **OK** in the test box and **OK** in the DSN dialog box.

    The connection has been made.

CONNECTING A CLIENT APPLICATION IN WINDOWS

You should now have a data source on your Windows machine that connects through the Easysoft ODBC-ODBC Bridge to a second data source on the remote server machine (either your own server or `demo.easysoft.com`).

To demonstrate that the Easysoft ODBC-ODBC Bridge is functioning correctly you can connect an example ODBC application to the local data source.

You will need experience of Microsoft Access to complete this section.

Caution!

If you are using the Easysoft ODBC-ODBC Bridge across the internet (e.g. to contact the Easysoft `demo` server) then Microsoft Access is *`not`* recommended, as its very heavy use of ODBC calls generates significant network traffic when using the Easysoft ODBC-ODBC Bridge. **"The Demo.exe Client" on page 150** describes a more efficient client program.

1. Start Microsoft Access (for example) and create a blank database.
2. Do one of the following:
   - In Access 2007, on the **External Data** tab, in the **Import** group, click **More**. Click **ODBC Database**. 
   - In earlier versions of Access, select **File > Get External Data > Link Tables**.

     The **Link** dialog box displays the existing databases on your system.
3. Do one of the following:
   - In Access 2007, click **Link to the data source by creating a linked table**, and then click **OK**.
   - In earlier versions of Access, from the **Files of type** drop-down list, choose **ODBC Databases**.

   The Microsoft ODBC driver manager displays the **Select Data Source** dialog box:

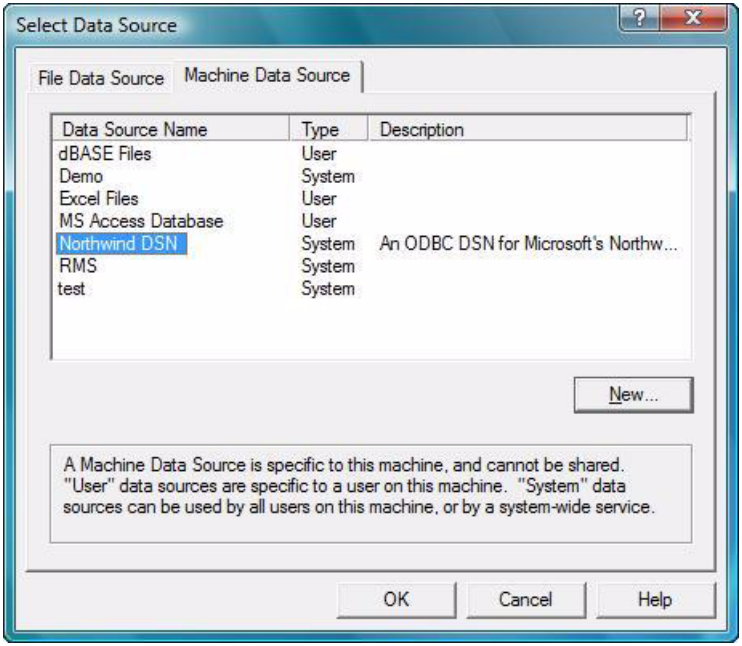


Figure 30: The Select Data Source dialog box Machine Data Source tab

4. Click the **Machine Data Source** tab.

   Find the local data source you created, somewhere in the list. Note that your description of the data source is displayed beside it.

5. Select your data source and click **OK**.

   Microsoft Access connects to the OOB data source and the OOB client relays the ODBC API calls to the OOB Server. Microsoft Access will retrieve a list of tables in the remote data source and display them in a window.

6. Click on a table and then click **OK**.

After a short wait, you are returned to the **Database** window.

7. Double-click any table to open and browse it.

THE DEMO.EXE CLIENT

An application called `demo.exe` is included with the Easysoft ODBC-ODBC Bridge that can be used to verify that a connection has been made to an ODBC data source.

The `demo.exe` program lets you test your OOB Client installation by connecting to the demo data source.

Note that you can also use `demo.exe` to connect to any Easysoft ODBC-ODBC Bridge data sources you have created yourself.

The source code is included in the demonstration program to give developers an insight into creating simple ODBC client applications.

NB

The `demo.exe` client application will not work without a local data source having been configured.

Refer to the section **"Windows client setup" on page 132** and create the demo data source. Note that the data source must be named `demo` for some parts of `demo.exe` to work.

1. Start an MS-DOS window.
2. Change directory to the `Examples` directory within the `easysoft` install directory.

   e.g.

   ```
   Program Files\Easysoft\Easysoft ODBC ODBC Bridge
   \Examples
   ```

3. Execute the `demo.exe` program:

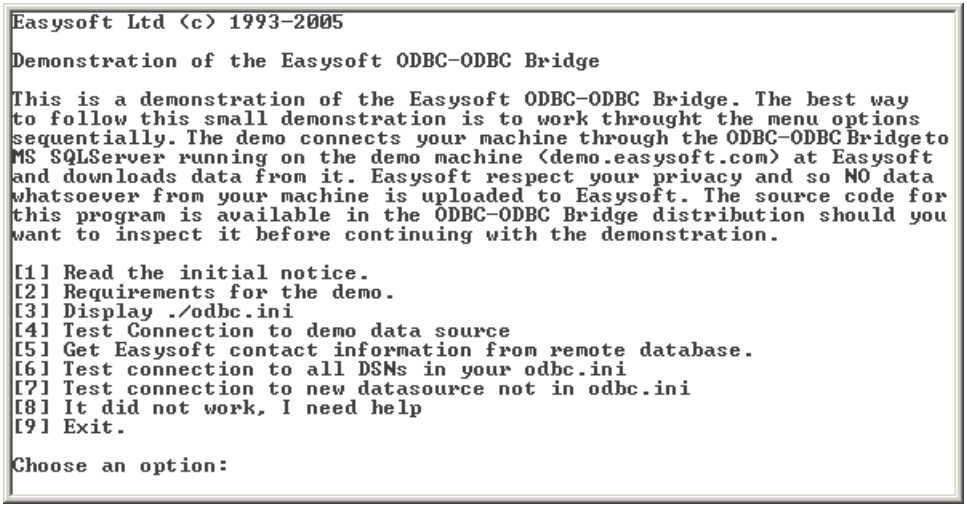


Figure 31: The demo.exe program

The demonstration program is self-explanatory: select an option, press <*Enter*> and then follow the information displayed on screen.

[1] displays the initial notice.

[2] displays the requirements for the demo.

[3] this option is not supported on Windows.

[4] tests the connection by contacting `demo.easysoft.com` through the Easysoft ODBC-ODBC Bridge.

[5] retrieves a table from the database on `demo.easysoft.com`.

NB	Option 5 will only retrieve data when your demo data source is configured to connect to the `demo.easysoft.com` server. You will probably receive errors if you configure your demo data source to connect to another database.

[6] attempts to connect to all user data sources one at a time. Whether the connection fails or succeeds is displayed in a Status column. If the connection fails, diagnostics are provided.

NB	If a data source points to a machine that cannot currently be contacted, the connection attempt may take a couple of minutes to time out.

[7] builds up a connection string to connect to the OOB Client ODBC driver but with values you specify to demo.exe (you do not need to create a data source). You are prompted for each value in turn (the default values build up a connection string to the demo data source on `demo.easysoft.com`). The connection string is displayed in full before demo.exe attempts to connect using that connection string.

[8] displays suggestions as to what may be the problem with your connection.

[9] exits from the `demo.exe` program.

Unix client setup

This section explains the steps you should take to connect an ODBC application on a Unix machine (where the OOB Client is installed) to the data source on your remote server (where the OOB Server is installed).

To connect the OOB Client in Unix to a remote data source, you need to define the data source on the remote machine by specifying its attributes in a data source on the local (client) machine.

You can create a data source for the OOB Client on the local Unix machine either by:

- directly adding the data source and its attributes to a configuration file (`odbc.ini`). To locate the `odbc.ini` file where System data sources are defined and the `odbc.ini` file where User data sources are defined, use the `<UNIXODBCBINDIR>/odbcinst -j` command.
- if you are on a machine running X, you can create a data source using the graphical ODBC Data Source Administrator (`ODBCConfig`).

Driver manager functionality in the Easysoft ODBC-ODBC Bridge

The most important driver manager functionality provided by the Easysoft ODBC-ODBC Bridge is the storing of data source attributes in order to be able to connect given a minimal connection string.

When an application connects through the OOB Client, it can pass in as little information as just a data source name (DSN).

The OOB Client will search for a configuration file in this sequence:

- $ODBCINI (i.e. it looks for the file whose name is stored in the environment variable `ODBCINI`)
- *`current-dir`*`/odbc.ini`
- *`current-dir`*`/.odbc.ini`
- `$HOME/odbc.ini`
- `$HOME/.odbc.ini`
- `/etc/odbc.ini` (for System data sources only)
- *`dir-where-unixODBC-stores-DSNs`* (for System data sources only)

For the last location, the OOB client will attempt to load the unixODBC `libodbcinst` shared object and use `SQLGetPrivateProfileString()`. In this way, if you built unixODBC yourself with a different `--sysconfdir` option value, OOB can still find your DSNs.

When unixODBC is installed with the Easysoft ODBC-ODBC Bridge, `--sysconfdir` is set to `/etc`, but if you have built and configured your own unixODBC then it will be whatever you specified with the `--sysconfdir` configuration option, or `/usr/local/etc` if omitted.

NB	If you are running Apache/PHP or ColdFusion with the Easysoft ODBC-ODBC Bridge, Easysoft recommend you use `/etc/odbc.ini`. Putting the file in the web server document tree risks making it publicly accessible, and putting it in the ‘current directory’ may be meaningless as the web server can be started from any directory.

CREATING A DSN BY EDITING A CONFIGURATION FILE

With unixODBC, data sources are stored in a configuration file called `odbc.ini`.

Use the *<UNIXODBCBINDIR>*`/odbcinst -j` command to locate the `odbc.ini` file where System data sources are defined and the `odbc.ini` file where User data sources are defined. The following lines show some sample `odbcinst -j` output:

```
unixODBC 2.2.12

DRIVERS............: /etc/odbcinst.ini

SYSTEM DATA SOURCES: /etc/odbc.ini

USER DATA SOURCES..: /home/tim/.odbc.ini
```

You usually have to be logged in as `root` to edit the system `odbc.ini`.

To add a data source, open the relevant odbc.ini file in a text editor, then add a new section.

Each section of the `odbc.ini` file starts with a data source name in square brackets [ ] followed by a number of *`attribute`*=*`value`* pairs.

NB	Attribute names in `odbc.ini` are not case sensitive.

The Driver attribute names the ODBC driver in the `odbcinst.ini` file to use for this data source. When the Easysoft ODBC-ODBC Bridge is installed into unixODBC it places an OOB entry into the `odbcinst.ini` file so you should always have `Driver = OOB` in your Easysoft ODBC-ODBC Bridge data sources. On some platforms, the OOB Client is distributed as two separate drivers, a thread-safe one requiring pthreads (OOB_r) and one which is not thread-safe (OOB) and does not require pthreads. If this distribution contains the thread-safe driver, an additional driver will be added to your `odbcinst.ini` file. To use the thread-safe driver, specify `Driver = OOB_r` in the data source definition.

To configure your DSN in your `odbc.ini` file, you will need to specify:

- The name of the machine on which the OOB Server is running and the port that it is listening on (`ServerPort`)
- The name of a user on the machine on which the OOB Server is running (`LogonUser`)
- The password for the user specified by `LogonUser` (`LogonAuth`)
- The System DSN on the remote machine (`TargetDSN`)
- The user name that will be supplied to the database to authenticate the connection, if required by your database (`TargetUser`)
- The password for the user specified by `TargetUser` (`TargetAuth`)

Other optional attribute values may be set in the `odbc.ini` file. For more information about the available attributes, see **"Attribute Fields" on page 174**.

NB	Don't forget that any attributes not specific to the Easysoft ODBC-ODBC Bridge are passed through to the remote data source, so you can effectively set up the remote data source from the local machine.

AN EXAMPLE odbc.ini FILE

If you have a Linux box called `linus.mydomain` and Microsoft SQLServer and the OOB Server running on a remote Windows NT machine called `ntbox.mydomain` with a Windows NT user name of `myname` and your password of `mypassword`.

Imagine you have set up a system data source on `ntbox` (see **"Windows server setup" on page 98**) called `myNTdsn` which requires database authentication `dbuser` and `dbpassword`.

You want to access data in Microsoft SQLServer on `ntbox` from `linus` using your Perl or PHP script.

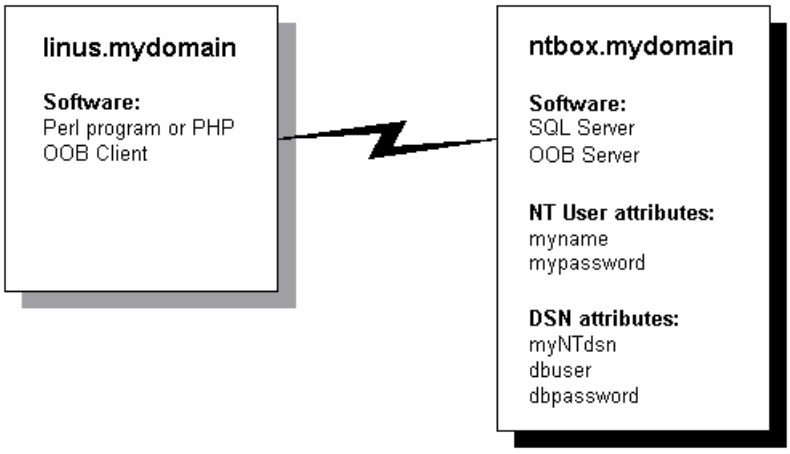


Figure 32: A Linux client connected to an NT server

The Linux box `odbc.ini` file would contain a section of the format:

```
[localdsn]
Driver=OOB
ServerPort=ntbox.mydomain.com:8888
LogonUser=myname
LogonAuth=mypassword
TargetDSN=myNTdsn
TargetUser=dbuser
TargetAuth=dbpassword
```

NB If using a driver manager at the client end, unixODBC looks up the `Driver=` entry in the `odbcinst.ini` file to locate the shared object to use as the ODBC driver.

When your application connects through the OOB Client, it needs to pass in the DSN `localdsn`, which the OOB Client then uses to access the correct section in the `odbc.ini` file.

CREATING A DSN USING THE ODBC DATA SOURCE ADMINISTRATOR

To create a data source using the graphical ODBC Data Source Administrator:

1. On a machine running X, log in as `root`.
2. In a terminal emulator window, change into the *<InstallDir>*`/easysoft/unixODBC/bin` directory.
3. Type `./ODBCConfig` and press *<Enter>*.

   The ODBC Data Source Administrator opens.
4. Click the **System DSN** tab to create a data source which is available to any user or service that logs into this machine.

   If you create a user DSN, only the `root` user will be able to access it because you logged in as `root` when you started your session. User DSNs will be placed in the `.odbc.ini` file in the home directory of the current user (i.e. `$HOME/.odbc.ini`).
5. Click **Add** to create a new data source.

   The **Adding a New Data Source** dialog box lists the available drivers.
6. Select the Easysoft ODBC-ODBC Bridge driver and then click **OK**.

   On some platforms, the **ODBC-ODBC Bridge DSN** dialog box is displayed:

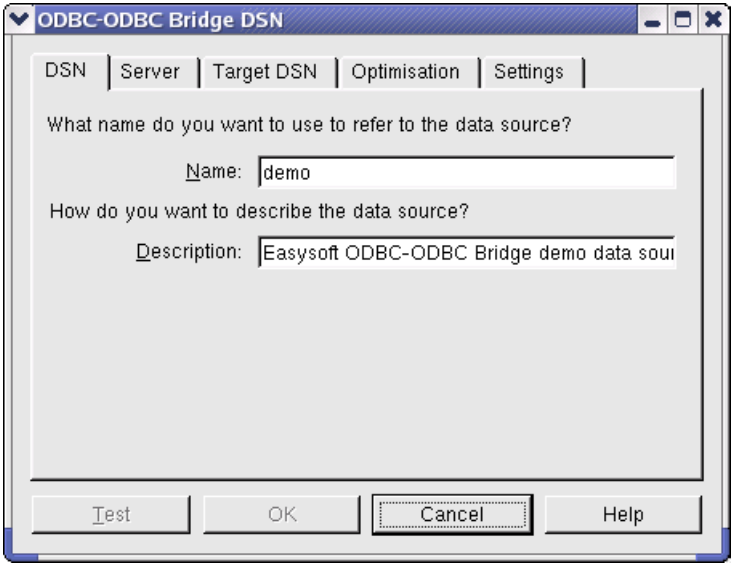


Figure 33: A blank Easysoft ODBC-ODBC Bridge DSN dialog box

For information about adding and testing a data source by using this dialog box, see **"The Easysoft ODBC-ODBC Bridge DSN dialog box" on page 136**.

On other platforms, the **Data Source Properties** dialog box is displayed.

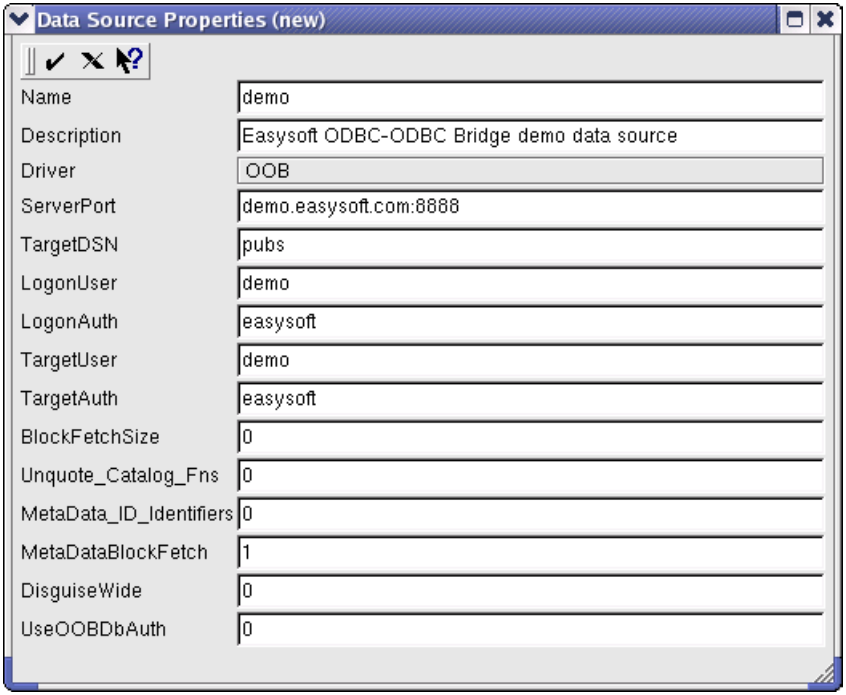


Figure 34: The Data Source Properties dialog box

The options that you can set on this dialog box are the same as the attributes that you can specify when defining a data source in the `odbc.ini` file.

Refer to **"Creating a DSN by editing a configuration file" on page 155** for information about each setting and details of how they are handled by the Easysoft ODBC-ODBC Bridge.

7. Click **OK** when you have specified all the data source attributes and then close the ODBC Data Source Administrator.

Once you have a data source on your Unix client connecting to a data source on your server, you can run an application on your client machine and connect to the data source on the server machine.

THE DEMO APPLICATION

The distribution includes a simple ODBC program for connecting to any ODBC data source in order to test that it is working.

This program can be used in the event that your own remote ODBC data source is not yet configured.

Note that unlike other ODBC applications, demo is not linked to an ODBC Driver Manager. demo is linked directly with the OOB Client. You can therefore only use it to test data sources defined in the sample `./odbc.ini` file or those where the OOB attributes are entered directly into demo. oobping is a more flexible and more recent program to use to test the Easysoft ODBC-ODBC Bridge. For more information about oobping, see **"oobping" on page 116**.

To run the demo program:

1. Change into the `<InstallDir>/easysoft/oob/examples` directory.
2. Type:

   ```
   ./demo
   ```

   The source code for this demo program is in the `examples` subdirectory.

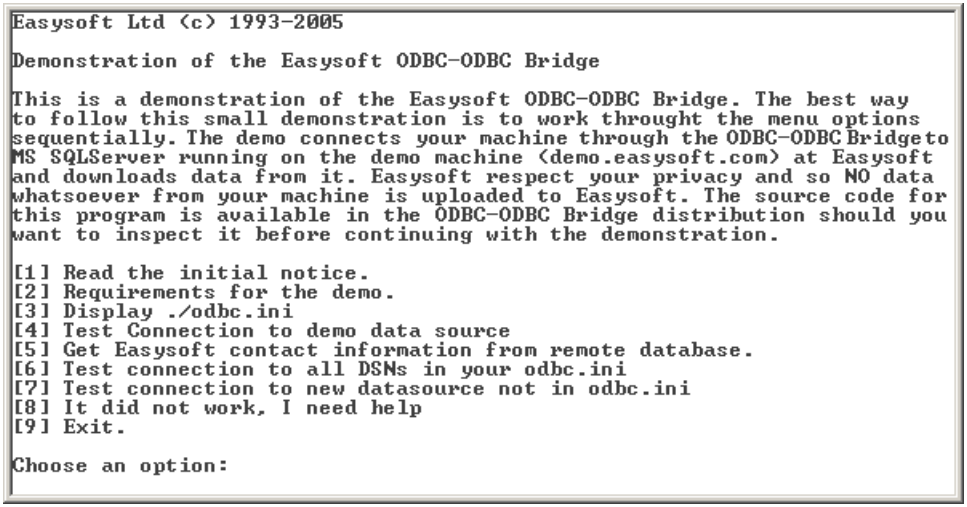


Figure 35: The demo.exe program on Unix

The demonstration program is self-explanatory: select an option, press <*Enter*> and then follow the information displayed on screen.

[1] displays the initial notice.

[2] displays the requirements for the demo.

[3] displays the contents of `./odbc.ini`.

[4] tests the connection by contacting `demo.easysoft.com` through the Easysoft ODBC-ODBC Bridge.

[5] retrieves a table from the database on `demo.easysoft.com`.

NB	Option 5 will only retrieve data when your demo data source is configured to connect to the `demo.easysoft.com` server. You will probably receive errors if you configure your demo data source to connect to another database.

[6] attempts to connect to each data source in the local `odbc.ini` file. Whether the connection fails or succeeds is displayed in a Status column. If the connection fails, diagnostics are provided.

NB	If a data source points to a machine that cannot currently be contacted, the connection attempt may take a couple of minutes to time out.

[7] builds up a connection string for the OOB Client ODBC driver but with values you specify to the demo program. You are prompted for each value in turn (the default values build up a connection string to the demo data source on `demo.easysoft.com`). The connection string is displayed in full before the demo program attempts to connect using that connection string.

[8] displays suggestions as to what may be the problem with your connection.

[9] exits from the demo program.

Mac OS X client setup

This section shows you how to connect an ODBC application on a Mac OS X computer (where the OOB Client is installed) to a data source on a remote server (where the OOB Server is installed).

To access remote data from an ODBC application on the client computer, you need to create an ODBC data source on the client. This data source uses the OOB Client ODBC driver and specifies the attributes required to connect to the remote data source.

To create an OOB Client data source on Mac OS X, use ODBC Administrator. ODBC Administrator is a component of iODBC that provides a GUI with which you can add, modify, delete and examine data sources.

Data sources are stored in the odbc.ini file. User data sources are stored in `~/Library/ODBC`. To find out the directory where System data sources are stored, in a Terminal window, type the following command:

```
iodbc-config --odbcini
```

iodbc-config is a script that outputs iODBC configuration information. The `--odbcini` option prints the system wide odbc.ini file path. The following line shows some example output from the previous command:

```
/Library/ODBC/odbc.ini
```

Note	To add, remove or edit System data sources, you need to be logged in as an administrator user. Depending on the permissions on the system wide odbc.ini file, you may have to log in as the root user to add, remove or edit System data sources. Otherwise, the changes that you make will not be saved in the odbc.ini file. By default, the root user account is not active. For information about enabling the root user and the implications of using the root account, in the Mac OS X Help, search for "root user."

To add an OOB Client data source

1. Open ODBC Administrator in the `/Applications/Utilities` folder.

   ODBC Administrator is displayed.

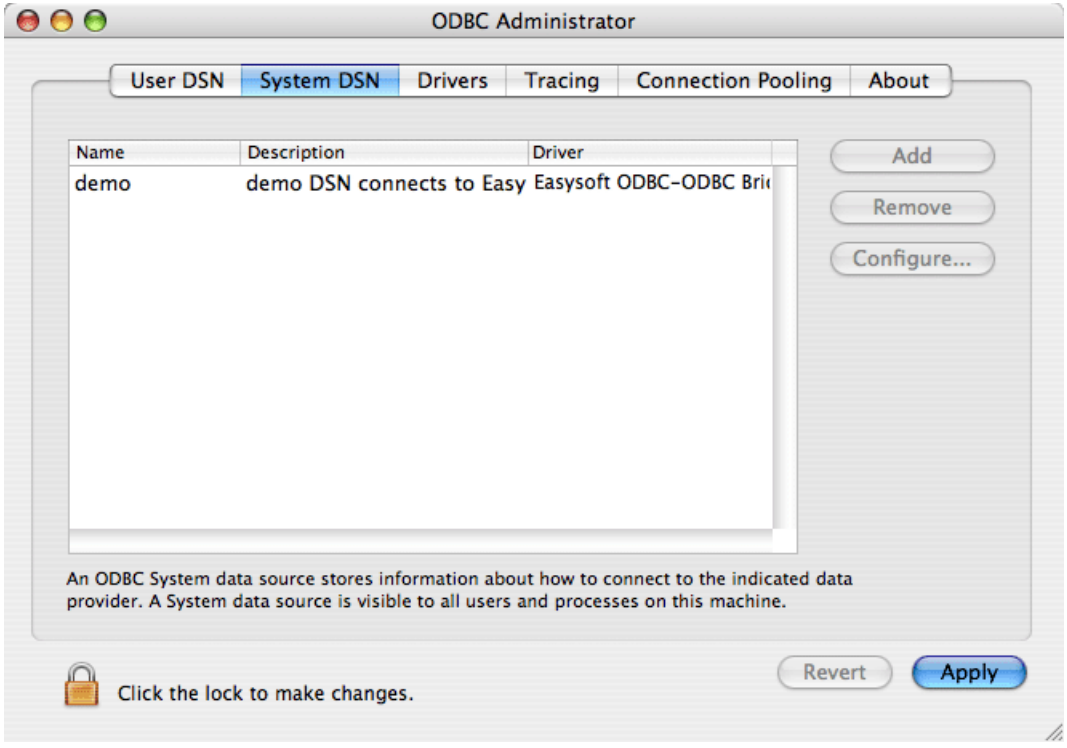


Figure 36: The Mac OS X ODBC Administrator

2. Do one of the following:

   - To add a data source that is only visible to the user currently logged in to this computer, in the **User DSN** tab, click **Add**.
   - To add a data source that is visible to all users of this computer, click the **System DSN** tab, and then click **Add**.

   If the System DSN tab is locked, the Add button will be unavailable. To unlock the System DSN tab, click the lock icon, and then type an administrator user name and password when prompted.

The Choose A Driver dialog prompts you to choose the driver for which you want to set up the data source.

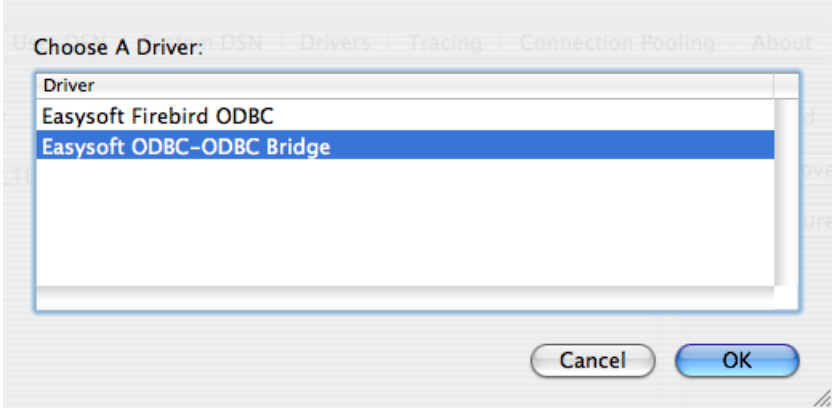


Figure 37: The Mac OS X ODBC Administrator Choose A Driver dialog box

3. From the list of ODBC drivers, choose Easysoft ODBC-ODBC Bridge, then click **OK**.

   The OOB Client for Mac OS X DSN dialog box is displayed.

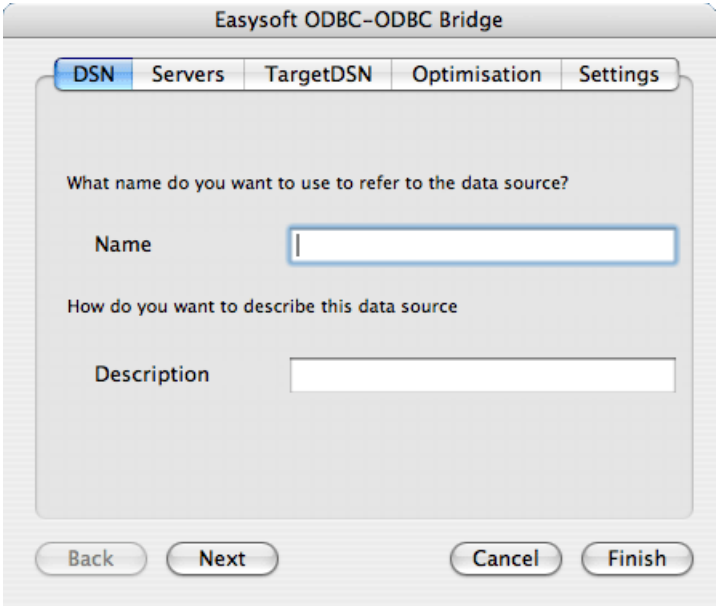


Figure 38: The OOB Client for Mac OS X DSN dialog box

The dialog box uses the following tabs to organise data source attributes:

- DSN The name and description that identify the data source.
- Server Settings and authentication details for the server that the OOB Server is installed on.
- Target DSN Settings and authentication details for the remote System ODBC data source.
- Optimisation OOB performance settings.
- Settings Settings that let you override the default OOB behaviour.

4. In the DSN tab, in the **Name** box, type the data source name.

Note that the OOB Installer program creates a sample System data source named "demo". If you want to create your own test System data source, use a different name. Otherwise, the demo data source will be overwritten.

5. In the **Description** box, type some text to describe the data source.

   Some applications display the description text to help users differentiate between different data sources.

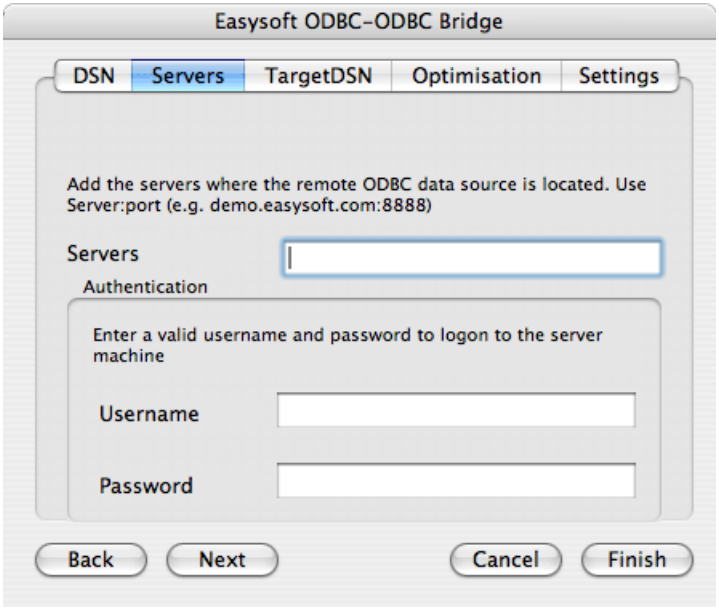


Figure 39: The OOB Client for Mac OS X DSN dialog box—Servers tab

6. In the **Servers** tab, in the **Servers** box, type *server*:*port*.

   server is the fully qualified domain name or IP address of the server on which the OOB Server is running. *port* is the port that the OOB Server is listening on. The default port is 8888.

For example, to connect to the Easysoft `demo` server, type:

```
demo.easysoft.com:8888
```

If the remote System DSN is available on more than one OOB Server, you can define a primary OOB Server for the DSN and additional fallback OOB Servers. For more information about defining multiple OOB Servers, see **"Client support for Fallback OOB Servers" on page 189**.

7. In the **Username** and **Password** boxes, type the username and password to use to log in to the server on which the OOB Server is running (if required).

   If the username and password are valid, the OOB Server process changes to become the specified user.

   For example, to connect to the Easysoft `demo` server, type `demo` in the **Username** box and `easysoft` in the **Password** box.

> **Note**
>
> If your server is part of a Windows domain, you may need to include the domain name with the user name, using the format `domain/`*`user name`*. For example: `admin/John Smith`.
>
> If you have defined multiple OOB Servers for this data source, the username and password must be valid for each OOB Server that you specify.

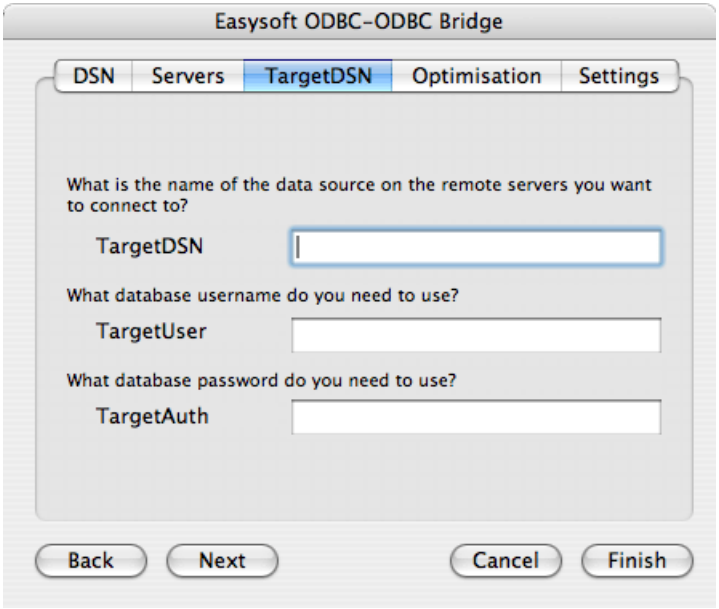


Figure 40: The OOB Client for Mac OS X DSN dialog box—TargetDSN tab

8. In the **TargetDSN** tab, in the **TargetDSN** box, type the name of a System data source on the remote server.

   For example, if you are connecting to the Easysoft `demo` server, type `pubs`.

9. If the database requires a username and password, type these in the **TargetUser** and **TargetAuth** boxes. Otherwise, leave these boxes blank.

   For example, if you are connecting to the Easysoft `demo` data source, in the **TargetUser** box, type `demo`. In the **TargetAuth** box type `easysoft`.

10. For more information about the options on the **Optimisation** tab and the **Settings** tab, see **"Attribute Fields" on page 174**. If you are connecting to the Easysoft `demo` data source, leave the options set to their default values.
11. Click **Finish**.
12. In ODBC Administrator, click **Apply** to save your new data source.

TESTING DATA SOURCES

iodbctest is a command line ODBC application that is supplied with iODBC. Use it to test your OOB data sources. If a connection can successfully be made, you can query the remote database by executing SQL statements in iodbctest.

To test an OOB Client data source

1. Open Terminal in the `/Applications/Utilities` folder.
2. At the shell prompt, type:

   ```
   iodbctest
   ```

3. Type DSN=*`name`*

   where *`name`* is the name of an OOB data source that you want to test.

   If the connection succeeds, iodbctest prompts you to type some SQL.

   If the connection fails, iodbctest displays an error to help you identify what the problem is. For help on troubleshooting the problem, see FAQ.txt in the `/usr/local/easysoft/oob/doc` directory.

4. To exit iodbctest, type `quit`.

Attribute Fields

This section lists the attributes which can be set for the Easysoft ODBC-ODBC Bridge in a table showing:

- the label of the attribute on the Easysoft ODBC-ODBC Bridge DSN dialog box.
- the entry required when editing the Unix `odbc.ini` file.
- the string to be used in a call to `SQLDriverConnect` or in a connect string for ADO type use.

Attributes which are text fields are displayed as "`value`".

Attributes which are binary fields can contain either 0 (to set to off) or 1 (to set to on) and are displayed as "`0|1`".

DSN

The name of the User or System data source, as used by the application when calling the `SQLConnect` or `SQLDriverConnect` functions.

Interface	Value
DSN Dialogue Box	Name
odbc.ini file (Unix)	[value]
Connect String	DSN=value

DESCRIPTION

A single line of descriptive text which may be retrieved by certain applications to describe the data source..

Interface	Value
DSN Dialogue Box	Description
odbc.ini file (Unix)	Description=value
Connect String	Not Used

SERVERS

The name or IP address of the remote host on which the OOB Server is running.

The port on this machine where the OOB Server (or an agent on its behalf such as inetd, for more information see **"The mechanics of inetd and the server" on page 235**) is listening for incoming connections. The default port is 8888. Specify this port unless you know that the OOB Server is listening on another port. Separate the host name and port with a colon (:).

If the remote System DSN is available on more than one OOB Server, you can define a primary OOB Server for the DSN and additional fallback OOB Servers. For more information about defining multiple OOB Servers, see **"Client support for Fallback OOB Servers" on page 189**..

Interface	Value
DSN Dialogue Box	Servers
odbc.ini file (Unix)	ServerPort=server:port [, server:port...]
Connect String	SERVERPORT=server:port

USERNAME

The name of a user on the machine on which the OOB Server is running (if required). The OOB Server changes to this user when an incoming connection is made.

On Windows, you may need to include the domain name in this attribute. If so, use the format `domain/user name`. For example: `admin/John Smith`

If you have defined multiple OOB Servers for this data source, the user name and password must be valid for each OOB Server that you specify.

Interface	Value
DSN Dialogue Box	Username
odbc.ini file (Unix)	LogonUser=value
Connect String	LOGONUSER=value

PASSWORD

The password for the user specified by `Username`.

Interface	Value
DSN Dialogue Box	Password
odbc.ini file (Unix)	LogonAuth=value
Connect String	LOGONAUTH=value

TargetDSN

The System DSN on the remote machine.

Interface	Value
DSN Dialogue Box	TargetDSN
odbc.ini file (Unix)	TargetDSN=value
Connect String	DSN=value

TargetUser

The user name that will be supplied to the database to authenticate the connection. This may not be required by your database.

Interface	Value
DSN Dialogue Box	TargetUser
odbc.ini file (Unix)	TargetUser=value
Connect String	UID=value

TargetAuth

The password for the user specified by `TargetUser`.

Interface	Value
DSN Dialogue Box	TargetAuth
odbc.ini file (Unix)	TargetAuth=value
Connect String	PWD=value

BLOCKFETCHSIZE

If you set `BlockFetchSize` to a value greater than 0, the Easysoft ODBC-ODBC Bridge determines whether to perform an optimization which retrieves multiple rows of data instead of one row at a time. The value you specify with `BlockFetchSize` is the number of rows to retrieve in one go. The maximum value is 100.

This optimization is not performed if the application itself binds columns in the result set. If your ODBC application uses cursors or positioned updates/deletes, you should not set this to greater than 1. Refer to question "**I only want to retrieve data from the server, are there any tricks to speed it up in read-only mode**" in the **Easysoft ODBC-ODBC Bridge KB** for more information about this value.

The default value 0 means that block fetches should not be done by the OOB client.

Interface	Value
DSN Dialogue Box	BlockFetchSize
odbc.ini file (Unix)	BlockFetchSize=value
Connect String	BLOCKFETCHSIZE=value

METADATABLOCKFETCH

When ON (set to 1 or checked), the `MetaDataBlockFetch` option enables blockfetching for meta data without affecting other result sets.

This will increase the speed of retrieving metadata, such as lists of tables or columns in a data source. This option is enabled by default, but needs to be disabled for a few ODBC drivers that do not support it (see "**Why do I not get a list of all the tables and columns in a database?**" the **Easysoft ODBC-ODBC Bridge KB** for a list of these drivers).

Interface	Value
DSN Dialogue Box	MetaDataBlockFetch
odbc.ini file (Unix)	MetaDataBlockFetch=0|1
Connect String	METADATABLOCKFETCH=0|1

OVERRIDE UID/PWD

When ON (set to 1 or checked), the `Override UID/PWD` option causes the Easysoft ODBC-ODBC Bridge to ignore the `UID` and `PWD` values passed from the application in the connection string and use the `TargetUser` and `TargetAuth` values for `UID`/`PWD`.

This option may need to be selected for some software (e.g. Crystal Info which passes an empty `PWD` field and expects the driver to issue a prompt dialog box), but this is not always possible when the driver is on a remote machine.

The default for `Override UID/PWD` is off (0).

Interface	Value
DSN Dialogue Box	Override UID/PWD
odbc.ini file (Unix)	UseOOBDBAuth=0|1
Connect String	USEOOBDBAUTH=0|1

PASS ALL SQLGETINFO REQUESTS ON

When ON (set to 1 or checked), `SQLGetInfo` requests for the ODBC driver name, version and ODBC version are sent to the remote ODBC driver instead of returning the OOB Client name, version and ODBC version. This may be useful in situations when you want to know the remote ODBC driver so the application can take alternative actions.

The default for `Pass all SQLGetInfo requests on` is off (0).

Interface	Value
DSN Dialogue Box	Pass all SQLGetInfo requests on
odbc.ini file (Unix)	GetInfoPassThru=0|1
Connect String	GETINFOPASSTHRU=0|1

DISGUISE WIDE CHARACTERS

When ON (set to 1 or checked), column types described by the ODBC driver as SQL_Wxyz are disguised for applications that do not understand wide characters. For example, some versions of StarOffice need this enabled when connecting to Microsoft SQLServer.

The default for `Disguise wide characters` is off (0).

Interface	Value
DSN Dialogue Box	Disguise wide characters
odbc.ini file (Unix)	DisguiseWide=0|1
Connect String	DISGUISEWIDE=0|1

MAP SQLEXECDIRECT

When ON (set to 1 or checked), the Easysoft ODBC-ODBC Bridge maps calls to `SQLExecDirect` to `SQLPrepare`/`SQLExecute` for ODBC 2.0 applications.

Between ODBC 2.0 and 3.0 a change in the allowable state transitions occurred. In ODBC 2.0 an application could call `SQLDescribeParam` even in the executed state (i.e. after the call to `SQLPrepare` **and** after the call to `SQLExecute`). In ODBC 3.0, this is no longer possible. The Easysoft ODBC-ODBC Bridge works around this limitation for ODBC 2.0 applications by mapping `SQLExecDirect` to `SQLPrepare`/`SQLExecute` and caching the parameter descriptions in between).

The Easysoft ODBC-ODBC Bridge has always performed this mapping for ODBC 2.0 applications but `Map SQLExecDirect` allows you to turn it off.

The default for `Map SQLExecDirect` is on (1).

Interface	Value
DSN Dialogue Box	Map SQLExecDirect
odbc.ini file (Unix)	MapExecDirect=0|1
Connect String	MapExecDirect=0|1

OVERRIDE LENGTH IN SQLGETXXX

When ON (set to 1 or checked), the OOB Client overrides the *BufferLength* argument value in `SQLGetInfo`, `SQLGetConnectAttr`, `SQLGetStmtAttr` and `SQLGetDescField` requests for integer information types. The OOB Client replaces the *BufferLength* value supplied by the application with the size of either an SQLINTEGER or an SQLSMALLINT.

The ODBC specification says that ODBC drivers should ignore the *BufferLength* argument supplied by the application for integer information types. However, some ODBC drivers incorrectly expect to receive a buffer length for integer information types from the application. If an application passes a buffer length smaller than the size of an SQLINTEGER or an SQLSMALLINT, these ODBC drivers may return SQL_SUCCESS_WITH_INFO with SQLSTATE 01004 (Data truncated), instead of the integer attribute value. To work around this problem, enable `Override Length in SQLGetxxx`.

The default value OFF (0) means that the Easysoft ODBC-ODBC Bridge passes the *BufferLength* value to the remote ODBC driver unchanged.

Interface	Value
DSN Dialogue Box	Override Length in SQLGetxxx
odbc.ini file (Unix)	OverrideLength=0|1
Connect String	OVERRIDELENGTH=0|1

CONNECTION ATTEMPTS

The number of times to attempt a connection before failing. Each time a connection attempt fails the OOB client waits (0.1 * connect attempt) seconds before trying again.

If you're using multiple server definitions in a single DSN you may find lowering this setting from the default of 5 an advantage as it reduces the time from the client failing to connect to the first server and starting a connection attempt to the second and subsequent servers.

Interface	Value
DSN Dialogue Box	Connection Attempts
odbc.ini file (Unix)	ConnectAttempts=value
Connect String	CONNECTATTEMPTS=value

BLOCK STATEMENT ATTRIBUTES

A comma separated list of statement attributes that the OOB Client will block in `SQLSetStmtAttr` calls. Any statement attribute you specify with `Block Statement Attributes` will not be passed to the remote ODBC driver. Instead, the OOB Client blocks them and returns SQL_SUCCESS.

To block a statement attribute, specify the ODBC assigned statement attribute value. Look up these values in the ODBC specification and the SQL header files. For example, the values to use for the SQL_ATTR_MAX_ROWS and SQL_ATTR_METADATA_ID attributes are 1 and 10014. To block both these attributes, specify the following `Block Statement Attributes` value:

```
1,10014
```

Usually, you should not block any statement attributes as this may prevent your application from working correctly. The `Block Statement Attributes` setting was introduced to work around a problem with the Allbase ODBC driver which was setting SQL_ATTR_MAX_ROWS on all statements when it was set on one particular statement.

By default, the OOB Client does not block statement attributes.

Interface	Value
DSN Dialogue Box	Block Statement Attributes
odbc.ini file (Unix)	IgnoreStmtAttrs=*`statement_attribute`* [, *`statement_attribute`*...]
Connect String	IGNORESTMTATTRS =*`statement_attribute`* [, *`statement_attribute`*...]

ENCRYPT COMMUNICATION

Whether to use encryption to protect data passed between the OOB client and server.

When set to `No Encryption (0)`, the channel between OOB client and server is not encrypted.

When set to `Encrypt If Available (1)`, the channel between OOB client and server is encrypted if the OOB server is capable of this (i.e. is a version that supports encryption and has been configured with a valid key and certificate file.) If the OOB server is not capable of encryption, the connection between OOB client and server will still succeed, but the channel between client and server will be unencrypted.

When set to `Encrypt Required (2)`, the channel between OOB client and server is encrypted if the OOB server is capable of this. Otherwise, the connection will fail. The OOB client will not attempt to validate the certificate used by the OOB server.

When set to `Encrypt Required + Validate (3)`, the channel between OOB client and server is encrypted if the OOB server is capable of this. Otherwise, the connection will fail. In addition, you need to specify the certificate used by the OOB server with the `Certificate File` attribute. If the OOB client is unable to verify the certificate, the connection will fail.

Note that the Test button on the Windows OOB client data source configuration dialog box cannot be used to check whether encryption has been successfully set up, you need to use to connect to the OOB client data source in your application to do this.

Interface	Value
DSN Dialogue Box	Encrypt Communication
odbc.ini file (Unix)	Encrypt= `0 | 1 | 2 | 3`
Connect String	ENCRYPT = `0 | 1 | 2 | 3`

CERTIFICATE FILE

The path to a file containing the certificate used by the OOB server to encrypt the connection between it and the OOB client. For example, `C:\Cert.pem`. Use this attribute if `Encrypt Communication` is set to `Encrypt Required + Validate (3)`.

Interface	Value
DSN Dialogue Box	Certificate File
odbc.ini file (Unix)	CertFile=*path*
Connect String	CERTFILE=*path*

UNQUOTE_CATALOG_FNS

When ON (set to 1 or checked), `Unquote_Catalog_Fns` removes the double quotes from around names in the calls to catalog functions such as `SQLColumns`. This is a workaround for a problem in Applixware which can incorrectly quote arguments to the catalog functions.

The default for `Unquote_Catalog_Fns` is off (0).

Interface	Value
DSN Dialogue Box	Not available
odbc.ini file (Unix)	Unquote_Catalog_Fns=0|1
Connect String	UNQUOTE_CATALOG_FNS=0|1

METADATA_ID_IDENTIFIER

When ON (set to 1), `MetaData_ID_Identfier` causes the OOB Client to call `SQLSetStmtAttr` with the `SQL_ATTR_METADATA_ID` attribute to set it to `SQL_TRUE`. This causes ODBC 3.0 drivers to treat strings in metadata functions as literals. This attribute is useful if you have an application you cannot change which assumes all strings in metadata calls are treated as literal but are going to an ODBC driver containing tables etc with special characters in them (e.g. _). Applixware's AXData sometimes needs this attribute defining.

The default for `MetaData_ID_Identifier` is OFF (0)

Interface	Value
DSN Dialogue Box	Not available
odbc.ini file (Unix)	MetaData_ID_Identifier=0|1
Connect String	METADATA_ID_IDENTIFIER=0|1

DECASNUMERIC

When ON (set to 1), `DecAsNumeric` causes the OOB Client to map an `SQL_DECIMAL` type in a `SQLBindParameter` call to an `SQL_NUMERIC` type. The `DecAsNumeric` attribute was introduced to work around a problem with Oracle Heterogeneous Services and Microsoft Access. When Oracle attempted to bind a type as a SQL_DECIMAL, the Microsoft Access ODBC driver returned the error "Optional feature not implemented".

The default for `DecAsNumeric` is OFF (0)

Interface	Value
DSN Dialogue Box	Not available
odbc.ini file (Unix)	DecAsNumeric=0|1
Connect String	DECASNUMERIC=0|1

CLIENT SUPPORT FOR FALLBACK OOB SERVERS

By default, the OOB Client is passed a data source name (DSN) which defines a single remote server with which to connect.

If the client fails to connect to the specified server then it tries 4 more times by default, pausing (`0.1 * connection attempts`) seconds between each attempt (the number of additional attempts can be configured with the parameter `ConnectAttempts`).

Each client DSN can also define multiple servers and ports. If the client fails to connect to the first server in the list it retries with each server in the list until either a connection is made or the list is exhausted.

You specify the servers on which the remote System DSN is available when configuring the client DSN. On Windows, Mac OS X and some Unix platforms, you configure client DSNs in the Easysoft ODBC-ODBC Bridge DSN dialog box. For more information about configuring client DSNs, see pages **"Windows client setup" on page 132**, **"Mac OS X client setup" on page 165** and **"Creating a DSN using the ODBC Data Source Administrator" on page 159**.

To add multiple servers by using the Easysoft ODBC-ODBC Bridge DSN dialog box on Windows

1. In the Easysoft ODBC-ODBC Bridge DSN dialog box, click the **Server** tab.
2. Click the **...** button.
3. In the **Server** box, type the host name or IP address of the primary OOB Server.
4. In the **Port** box, type the port on which the primary OOB Server is listening.

The default port is 8888. Specify this port unless you know that the OOB Server is listening on another port.

5. Click **Add**.
6. In the **Server** box, type the host name or IP address of a fallback OOB Server on which the remote System DSN is also defined.
7. If the fallback OOB Server is listening on a different port to the default, in the **Port** box, type the port. Otherwise skip this step.
8. Do one of the following:
   - To add another fallback OOB Server, click **Add**, then repeat steps 6 and 7.
   - If you've finished adding OOB Servers, click **Test** to test the connection process for each server in the list.

     If a connection can't be made to a particular server, in the **Servers** list, a red cross displays next to the server. The Test Log output shows the reason why the test failed in red text. For help on how to resolve the connection problem, click **Help**.

NB	You **must** enter values in the **Username** and **Password** boxes in the **Server** tab before testing, unless authentication is disabled in the OOB Server.

9. Click **OK**.

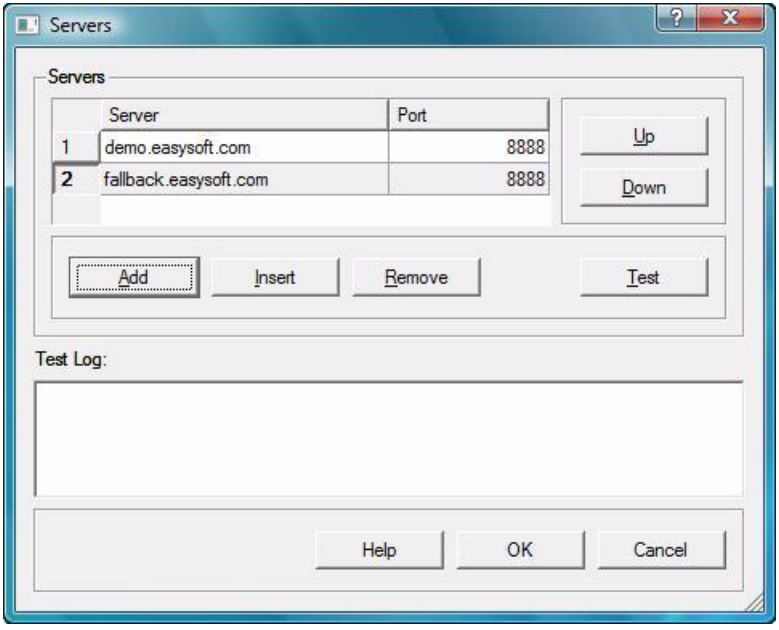


Figure 41: Multiple OOB Servers defined in the Servers dialog box

The OOB Client tries to connect to OOB Servers in the order they are defined in the list. To reorder the OOB Servers, in the **Servers** list, click the OOB Server whose position you want to change. Then use the **Up** or **Down** buttons to move the OOB Server up or down the list.

When you add a new OOB Server, its added to the bottom of the list. If you want to add a server in a different position in the list, click the server above which you want the new server to appear, then click **Insert**.

To add multiple servers by using the Easysoft ODBC-ODBC Bridge DSN dialog box on Mac OS X

1. In the Easysoft ODBC-ODBC Bridge DSN dialog box, click the **Servers** tab.
2. In the **Servers** tab, in the **Servers** box, type *primaryserver*:*port*,*fallbackserver*:*port*.

   where *primaryserver* is the fully qualified domain name or IP address of the primary OOB Server on which the remote System DSN is defined, *port* is the port on which the OOB Server is listening (the default is 8888), and *fallbackserver* is the name or IP address of another OOB Server on which the remote system DSN is defined.

   Separate each *fallbackserver*:*port* you specify with a comma.

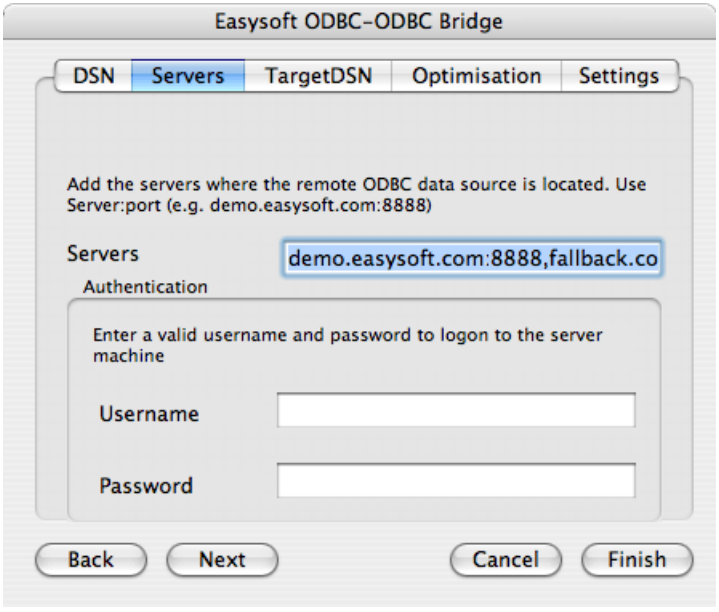


Figure 42: Multiple OOB Servers defined in the Easysoft ODBC-ODBC Bridge DSN dialog box on Mac OS X

3. Click **Finish**.

On Unix, you configure client data sources by editing an odbc.ini file. (For more information about configuring client data sources on Unix, see page **"Creating a DSN by editing a configuration file" on page 155**.)

To add multiple servers by editing odbc.ini

In the section for the remote System DSN, specify the primary OOB server and each fallback OOB server with the `ServerPort` attribute. Use the following format for the `ServerPort` entry:

```
ServerPort =
primaryserver:port,fallbackserver:port
[,fallbackserver:port...]
```

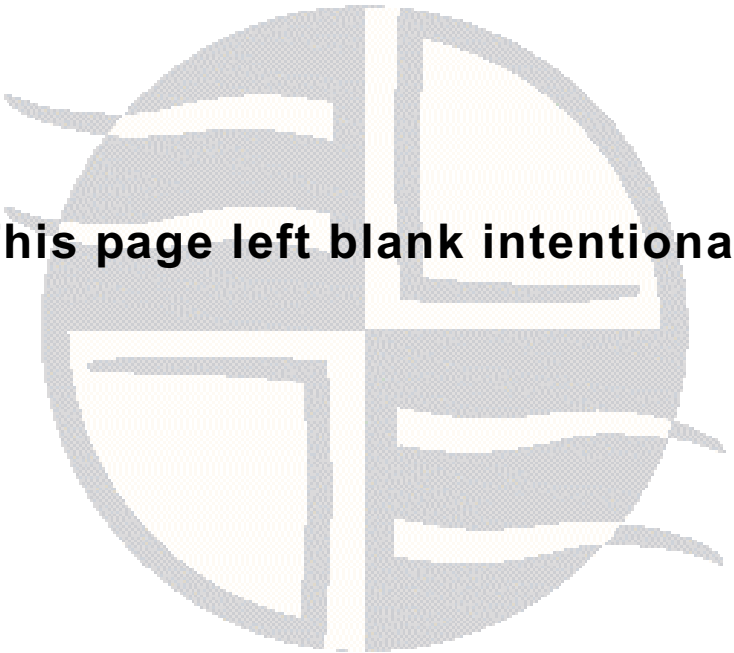
where *`primaryserver`* is the name or IP address of the primary OOB Server on which the remote System DSN is defined, *port* is the port on which the OOB Server is listening (the default is 8888), and *`fallbackserver`* is the name or IP address of another OOB Server on which the remote system DSN is defined.

For example:

```
[data_source_name]

Driver = OOB

TargetDSN = demo

LogonUser = user name

LogonAuth = password

ServerPort =
demo.easysoft.com:8888,fallback.easysoft.com:8888
```

Assuming an example primary OOB Server named `demo.easysoft.com` and a fallback OOB Server named `fallback.easysoft.com`, the OOB Client will attempt a connection to port 8888 on `demo.easysoft.com`. If this fails and after `Connection Attempts` additional attempts a connection has still not been made then the client will try to connect to the backup server `fallback.easysoft.com`.

There may be an advantage in lowering `Connection Attempts` from the default of 5 if multiple server definitions are being used in a single DSN, as this will reduce the time taken from the client failing to connect to the first server and starting a new connection attempt to the second and subsequent servers. For more information about the `Connection Attempts` setting, see **"Connection Attempts" on page 183**.



This page left blank intentionally

CHAPTER 4 CONFIGURATION

Configuring the Easysoft ODBC-ODBC Bridge

This section explains the configuration options available for the Easysoft ODBC-ODBC Bridge (OOB) Server. The configuration for Unix subsection also covers how the OOB Server for Unix can be run as a standalone server process without `{x}inetd`.

Chapter Guide

The Web Administrator

There are a number of configurable parameters for the OOB Server, irrespective of platform. There are platform-specific ways to change these settings, but the OOB Server also provides a simple HTTP server, known as the Web Administrator, that provides a web-browser based interface to use for updating these parameters.

The Web Administrator is available in the OOB Server for Windows and in the OOB Server for Unix when run as a standalone server process without `{x}inetd` (see **"Configuring the Server as Standalone" on page 236** for details).

This section describes the Web Administrator and shows you how to change the server configurable parameters.

In order to run the Web Administrator you will need:

- a Windows machine running the OOB Server or a Unix machine with the OOB Server running standalone

In order to make changes in the Web Administrator you will need:

- an HTTPAdmin user name and password unless the OOB Server administrative user name is disabled.

(Note that it's also possible to configure the Web Administrator so that an HTTPAdmin user name and password need to be supplied before any Web Administrator page can be accessed. For more information, see **"HTTPAuthAll" on page 214**.)

NB

To obtain the HTTPAdmin user name that was entered for your OOB Server when it was installed (HTTP authentication will have been disabled if this field was left blank) display the `HTTPAdmin` setting on the **Configuration** screen (see **"HTTPAdmin" on page 206**). This setting is hidden by default, but may be reset. The HTTPAdmin user name and password are required in order to make changes through the Web Administrator. If this value is not a valid user on the server system, then you must change it. The value is case sensitive.

HTTPAdmin is a registry setting on Windows. If you forget the HTTPAdmin user name or it is invalid, you cannot change it in the Web Administrator and will need to edit the registry. For information about the registry key where Web Administrator settings are stored, see **"OOB Configuration Parameters in the Registry" on page 233**.

Start the Web Administrator by opening the URL `http://`*oobhost*`:8890/` from a web browser (substituting your machine name in place of `oobhost` and the required port address if you have specified a port other than the Easysoft default of `8890`).

NB

To obtain the port number for the Web Administrator you need to refer to the `HTTPPort` setting on the **Configuration** screen (see **"HTTPPort" on page 205**).

If HTTPAdmin is set to anything other than `"disabled"` then some pages in the Web Administrator are protected with HTTP authentication.

If you click on a link to a protected page you will be prompted for the HTTPAdmin user name and password by your browser for the ESOOBServer realm:

Figure 43: The Enter Network Password dialog box

The web page returned is generated by the server process and displays runtime statistics for the latest run of the server:

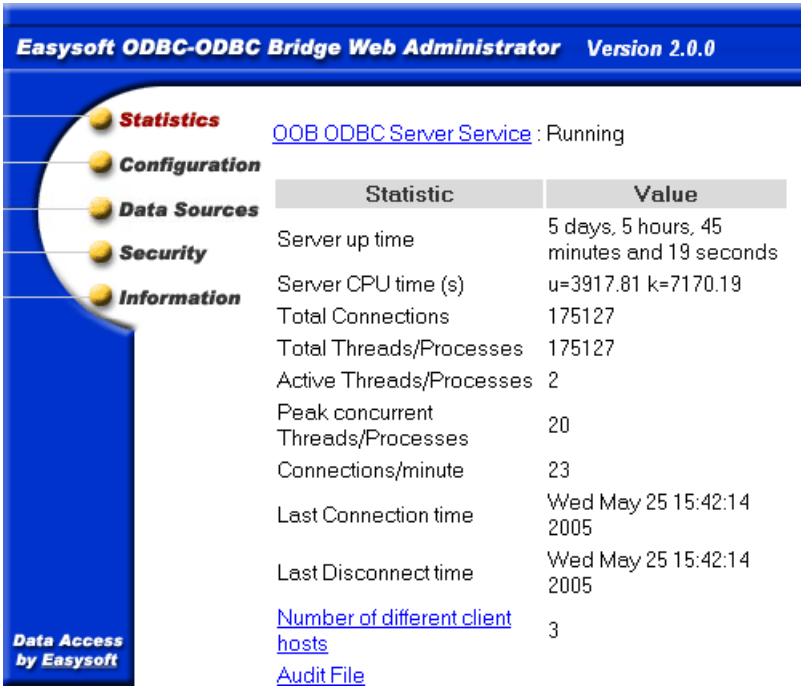


Figure 44: The Statistics screen of the Web Administrator

The following additional screens can be accessed from the **Statistics** screen:

- **Configuration** defines OOB Server features.
- **Data Sources** displays data source and driver information. For more information, see **"Browsing System Data Sources in the Web Administrator" on page 280**.

- **Security** allows a user to define the IP addresses of those machines which have access to the OOB Server enabled or disabled. It also allows a user to define the machines or users that can access a particular DSN. For more information, see **"The Security Screen" on page 220**.
- **Information** provides access to various Easysoft locations providing documentation and user support. For more information see **"Easysoft ODBC-ODBC Bridge Resources" on page 285**.

Client Hosts displays details of client machines which have connected to the OOB Server.

The **Statistics** and **Data Sources** screens display information, whilst the **Configuration** and **Security** screens allow you to modify the server's runtime parameters.

THE CONFIGURATION SCREEN

Figure 45: The Configuration screen of the Web Administrator

This section explains the parameters that are configurable via the Web Administrator **Configuration** screen. These parameters apply either to the OOB Server itself, or as a default value for all DSNs accessed via the server.

These settings, and some additional settings, can also be edited via the registry in Windows (see **"OOB Configuration Parameters in the Registry" on page 233**) or in a file in Unix (see **"Changing Server Configurable Parameters under Unix" on page 238**).

Note that enabling the `HideSensitive` attribute will result in some sensitive settings not being displayed (see **"HideSensitive" on page 212**).

You can modify the server settings by clicking **Change** and then typing the HTTP Admin user name and password when prompted. This is the user name that you entered during the installation process and the password for that user on the system where the OOB Server is running.

NB	You are only asked for this if the HTTP authentication has been enabled by the HTTP Admin user name being previously entered as something other than "`disabled`".

Keys are not case-dependent, so for example `Port`, `PORT` and `pOrT` are treated as the same key. Values are case-dependent if the operating-system is, so it is best to match case where possible.

The **Configuration** screen contains the following fields (click **Submit** to make your changes):

Parameter	Description
Port	The port where the OOB server is listening for incoming OOB client connections. The OOB server will need restarting for the changes to take effect.
IPAddress	The IP address the OOB Server listens on. This is only relevant if you have multiple NICs and only want the OOB server to listen on one. The default is for the server to bind to port 8888 on `INADDR_ANY` which means it is listening on all local interfaces.

Parameter	Description
HTTPPort	The port on which the OOB Server will listen for HTTP requests. The default is port 8890 but it may be any port not in use on your machine. On Unix, if the `Flags` configurable option `HTTP_SERVER` attribute is set (see **"HTTP_Server" on page 212**) then the OOB Server will listen on the specified port for HTTP requests as well as acting in its normal role serving the OOB Client. On Windows, the Web Administrator is a separate service that is started automatically by the Windows Service Manager. The Web Administrator listens on the specified port for HTTP requests. You may use the URL `http://machine_name:HTTPPort` where `machine_name` is the name (or IP address) of the machine running the OOB Server and `HTTPPort` is the port number to communicate with the OOB Server from your browser. It can show statistics, DSNs and the current values of configurable parameters. On Windows, the Web Administrator needs to be restarted in Service Manager for the new port to take effect.
Timeout	The inactivity timeout in seconds (the default is 7200 - 2 hours). Currently, this option is only applicable to the Windows OOB Server. The OOB Server starts a new thread (or process) for each client that connects. If there has been no communication in `Timeout` seconds the thread or process exits. This causes the client to be disconnected. To disable timeouts, set this field to 0. Disable this field if using persistent connections from PHP/mod_perl etc.

Parameter	Description
RetryCount	The number of additional attempts (after the first attempt fails) the server will make to get a resource. The resources influenced by this are thread/process creation and license slots for new connections.
RetryPause	The time in seconds between each retry attempt (see RetryCount).
HTTPAdmin	The username of the person allowed to make changes to the OOB server via the Web Administrator. This must be a valid username in the operating system the server is running on. If set to the string "`disabled`" (omit the quotes) then HTTP authentication is not required (this does not mean the OOB Server stops authenticating incoming ODBC connections).  The "eye" symbol displayed against the `HTTPAdmin` parameter denotes that the value of the field is only visible to the user via the **Change Configuration** screen, which may require the entry of a user name and password.
KeyFIle	The path to the file containing the private key for the certificate specified with `CertFile`. You need to restart the OOB server before this setting takes effect.
CertFile	The path to the file containing the certificate used to secure the connection between OOB client and server. For example, `/home/myuser.cert.pem`. You need to restart the OOB server before this setting takes effect.

Parameter	Description
HTTPNetWorkAccess	The network or IP address for clients allowed to use the Web Administrator. For example 194.131.236.4 only allows one machine but if the netmask was 255.255.255.0 then setting 194.131.236 would allow anyone on that network. The default is all browser clients have access to the Web Administrator although they may be required to enter a username/password if `HTTPAdmin` is set to anything other than "disabled".
MaxBookMarkSize	The size of the largest bookmark the OOB can handle (it defaults to 32 bytes). ODBC 2.0 uses fixed length bookmarks of 4 bytes. In ODBC 3.0 bookmarks are all variable in size. If you find an ODBC driver that needs more than 32 bytes for a bookmark please contact **support@easysoft.com**, otherwise the default should be fine.
Path	The installation path of the Easysoft ODBC-ODBC Bridge. This is a read-only parameter for information only.
Logging	A bitmask specifying the events to be logged by the OOB Server. `Logging` slows the Easysoft ODBC-ODBC Bridge down considerably and should only be set as directed by Easysoft support. The number may be specified as decimal (e.g. in the format `2047`)or hexadecimal (e.g. in the format `0x7ff`).
LogDir	The directory where log and audit files are created (see Logging and AuditODBCAccess). It defaults to *drive*:\Program Files\Easysoft\Easysoft ODBC-ODBC Bridge\Logs\ in Windows and /tmp in UNIX. This is also the directory where the OOB Server might write an esoob.exception file if an unhandled exception is caught.

Parameter	Description
MaxThreadCount	The maximum number of concurrent connections from a single client. If `MaxThreadCount` is set to 0, there is no limit. The default is 0. You can use this parameter to prevent people from swamping your machine with ODBC connections.
MaxClientConnect	The maximum number of concurrent connections from a single client (where a client is defined as the IP address of the machine where the client is running). If `MaxClientConnect` is set to 0, there is no limit. The default is 0. You can use this parameter to prevent people from swamping your machine with ODBC connections.
NetRCVLOWAT	The size of the receive low-water mark in bytes. Possible values are: `0`, do not set it at all, leave at system default. `-1`, set to internal OOB default (the default setting) `n` (n > 0), set to this value
NetRCVBUF	The size of the receive queue buffer for the socket in bytes. Possible values are: `0`, do not set it at all, leave at system default. `-1`, set to internal OOB default (the default setting, which is 16K) `n` (n > 0), set to this value
NetSNDBUF	The size of the send queue buffer in bytes. Possible values are: `0`, do not set it at all, leave at system default. `-1`, set to internal OOB default (the default, which is 16K) `n` (n > 0), set to this value

Parameter	Description
NetConnectRetryCount (Deprecated)	The number of times the OOB client will attempt a connection to an OOB Server. Each time a connection attempt fails the OOB client waits (0.1 * connect attempt) seconds before trying again. Possible values are: `1`, set to internal OOB default (the default, which is 5) `n` (n >= 0), set to this value If you're using multiple server definitions in a single DSN you may find lowering this setting from the default of 5 an advantage as it reduces the time from the client failing to connect to the first server and starting a connection attempt to the second and subsequent servers.

Parameter	Description
ListenBacklog	Defines the maximum length the queue of pending connections may grow to. It is a value passed to the BSD sockets call listen(). If a connection request arrives with the queue full the client will probably get an error of ECONNREFUSED and the OOB Server will be totally unaware of the connection attempt. If this happens the OOB client will use ConnectAttempts to make multiple connection attempts. Possible values are: 1 <= n <= 100 (the default is 20) You should not usually need to change this value as the default is sufficient for all but the very busiest of OOB Servers and having to set a high value is an indication that the server machine is underpowered. In addition, on Windows operating systems, the backlog queue is in non-pageable system memory. For some versions of the Windows operating system (NT workstation) changing this value has no effect since it is hard-wired at 5. For more information about the ConnectAttempts setting, see **"Connection Attempts" on page 183**. For more information about the listen backlog queue, see the **OOB FAQ**.

Parameter	Description
Affinity	If the OOB Server is running on a multiple CPU Windows Server machine you will see a check box for each CPU (labelled CPU_0 to CPU_N). The OOB Server holds the `Affinity` value as a mask of processors in your machine which the OOB Server can run on. Initially, the affinity value is 0 which means the OOB Server will make no changes at all and the Web Administrator will show all CPUs as available. However, if you make any changes to the CPU check boxes the OOB Server will call `SetProcessAffinityMask` which will define which processors the threads in the OOB Server are allowed to run on. In this way you can bind the OOB Server to particular CPUs in your machine.
Flags	A bitmask of operational flags, split into check boxes, one for each bit in Flags.

The following table lists and describes each Flags setting. If you're changing the Flags setting by editing `oobserver.ini` on Unix rather than by using the Web Administrator, use the hexadecimal value instead of the flag name. To set multiple flags, add the hexadecimal values for the flags you want.

Name	Hex Value	Description
Authentication_Disabled	0x1	If set then authentication is disabled in the OOB Server. The default is off. Although setting this parameter should be considered as a security risk, on some high hit servers in a controlled environment where you do not need to authenticate the connections this can save a considerable amount of time during connections (e.g. Windows NT can use between 0.25 and 0.75 seconds of the connection time for authentication)

Name	Hex Value	Description
HTTP_Server	0x2	If set then the Web Administrator is made available as soon as the Unix OOB Server is started. The default is on. This option is not applicable to the Windows OOB server. On Windows, the Web Administrator is a service that is started automatically by the Windows Service Manager.
MultiProcess	0x4	If set then the OOB Server starts a new process for each incoming connection, rather than a new thread. The default is off, so the OOB Server starts a new thread for each connection. Enable `MultiProcess` if an ODBC driver which is not thread-safe is being used or if your ODBC driver leaks memory. Currently on non-Windows platforms the server always starts new processes as it is NOT multi-threaded and so this flag cannot be changed.
MetaDataBlockFetch_Disabled (Deprecated)	0x8	If set prevents the OOB client from automatically starting block-fetch-mode for read-only metadata result-sets. The default is off.
HideSensitive	0x10	If set causes the Web Administrator to hide sensitive parameters on the **Configuration** screen. Currently the parameters removed when `HideSensitive` is set are `HTTPAdmin` (see **"HTTPAdmin" on page 206**) and `Authentication_Disabled` (see **"Authentication_Disabled" on page 211**). All parameters are always shown in full on the password protected **Change Configurable** screen. The default is on.

Name	Hex Value	Description
ReverseLookup	0x20	If set causes the OOB Server to call `gethostbyaddr()` on the connecting client's IP address to obtain the client's machine name. On machines where DNS is not set up properly this can cause problems and in any case adds time to the connection. `ReverseLookup` currently only affects the statistics, number of different clients page where "unknown" will be displayed instead of the machine name for connecting client's machine names if ReverseLookup is off. The default is off.
AuditODBCAccess	0x40	If set the OOB Server audits all ODBC connections to a log file (named `esoob_access.log` in the `LogDir` directory) which may be viewed through the Web Administrator. The default is off.
FreeStmtsOnDisconnect	0x80	If set then the OOB Server frees all existing statements in the driver before calling `SQLDisconnect`. By default this flag is off as this function should be executed by the ODBC driver, rather than the application, as detailed in the ODBC specification. Setting this flag fixes a potential crash in Navision's ODBC driver if `SQLDisconnect` is called from Perl when there are existing statements.
ShowProcessTime	0x100	If set the OOB Server retrieves the user and kernel CPU times for the main OOB Server process and displays them on the **Statistics** screen. The values are only updated once every five seconds. The default is on.
AutoUpdateConfiguration (Deprecated)	0x200	If set the OOB Server automatically updates its internal copy of the configuration settings from the registry when five seconds elapse without an incoming connection. Disabling this feature reduces the number of accesses to the registry, but means that the server has to be restarted to pick up configuration changes. The default is off.

Name	Hex Value	Description
HTTPAuthAll	0x400	If set then all pages in the Web Administrator are protected by HTTP authentication and you will be prompted for the HTTPAdmin username and password before any pages are returned to you (the realm is "ESOOBServer"). Note that setting HTTPAuthAll has no effect if HTTPAdmin is to "disabled".
ConnectionPooling	0x800	If set then the OOB Server automatically sets the `SQL_ATTR_CONNECTION_POOLING` attribute to `SQL_CP_ONE_PER_DRIVER` to enable connection pooling in the driver manager for the ODBC driver. Note that ConnectionPooling is meaningless when the OOB Server is running in MultiProcess mode (see **"MultiProcess" on page 212**).
LogFailingSQL	0x1000	If set then when any call to `SQLPrepare`/`SQLExecDirect` fails the SQL is logged to the file `oob_sql.log` in the `LogDir`. As some DBMS only parse the SQL at `SQLExecute` time sometimes you do not find out the SQL is incorrect until `SQLExecute` is called. To catch these SQL errors too the OOB Server saves the first 1K of any SQL passing `SQLPrepare` successfully so it may be logged if `SQLExecute` fails. For this reason turning this flag on will have some impact on the speed of the server and memory use. This is extremely useful when developing applications or web services and can be used to monitor errors. The default is off.
AllowDBBrowse	0x2000	If set the data sources listed on the **Data Sources** screen may be browsed to get lists of tables, columns in tables and rows of table data. The default is off.

Name	Hex Value	Description
AutoDenyNonOOBClients	0x4000	If set then any clients connecting to the OOB Server port and sending initial data down the socket which is not the EasyRPC protocol will be added to the internal denied ACL. Clients automatically added in this way are not visible on the security page and are removed from the ACL when the OOB Server is restarted or when the ACLs are changed on the security page. The default is off.
AllowDSNBrowse	0x8000	If AllowDSNBrowse is not enabled then OOB Clients cannot obtain a list of SYSTEM DSNs from the OOB Server. This affects the ODBC API function `SQLBrowseConnect()` and the population of the TargetDSN dropdown in the OOB Client Setup dialogue. The default is on.
NoStatusArrayResize	0x10000	This is a workaround for an issue in the SQL Server ODBC driver, which can write off the end of the parameter status array (SQL_ATTR_PARAM_STATUS_PTR) when the SQL_ATTR_PARAMSET_SIZE is reduced. For details about when this issue can happen, see the OOB FAQ "Why do I get an Access Violation in MS SQL Server ODBC Driver?". The default is off.
WriteStatsOnExit	0x20000	If set, the contents of the **Statistics** screen are written to file when the OOB Server service (on Windows) or the Standalone process (Unix) exits. For more information, see **"Termination statistics" on page 268**. The default is off.

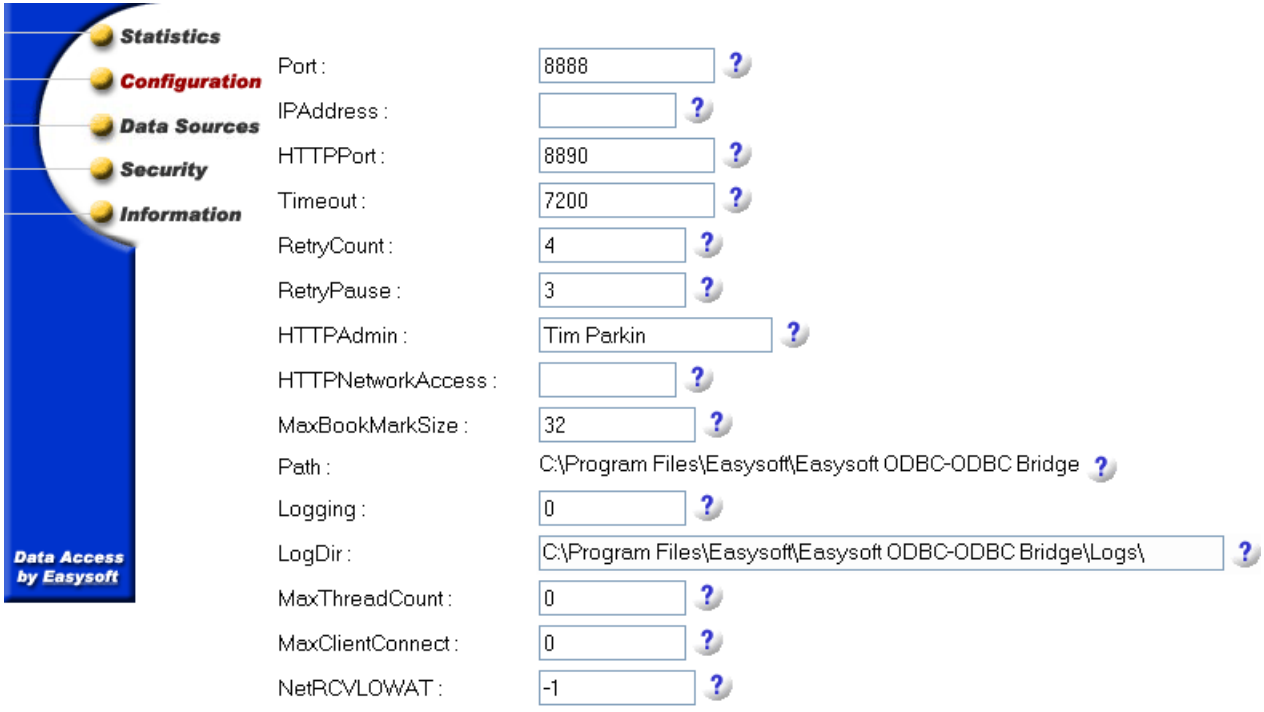


Figure 46: The Change Configuration screen of the Web Administrator

64-bit Windows

The OOB Server parameters in the **Configuration** screen apply to both the 32-bit and 64-bit OOB Server, apart from Port. The Port setting defines the port on which the 32-bit OOB Server listens for incoming client connections. If you change the Port value, the 32-bit OOB Server will listen on the new Port; the 64-bit Server will continue to listen on the default port, 8888. (When the 32-bit and 64-bit OOB Servers are listening on different ports, it is possible for both Servers to run at the same time.)

If you do not want the 64-bit OOB Server to listen on the default port, you need to change the value of this Registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Easysoft ODBC-ODBC Bridge\Configuration\System\Settings\Port64
```

SUPPORT FOR MULTIPLE NETWORK INTERFACE CARDS

By default, the OOB Server will start to listen on port `8888` on all network interfaces, but if the server has network interface cards (NICs) on multiple networks the OOB Server can be associated with a particular NIC by specifying an IP address.

This will be necessary if the server is running on multiple networks and access to the OOB Server is only required from one network.

The `IPAddress` configurable parameter on the Web Administrator **Configuration** page is usually an empty field, signifying that the OOB Server is listening on all local interfaces.

Change this field to the IP address of a particular NIC to restrict access to a particular network.

SPECIFIC SUPPORT FOR MULTIPLE PROCESSORS (PROCESS AFFINITY)

On multiple processor Windows machines, you can allocate particular processors on which to run the OOB Server. This is known as process affinity.

If your server has multiple CPUs then the Web Administrator **Configuration** screen will show each configured processor as `CPU_n` (where `n` is a number between zero and the number of configured processors minus one).

Initially, the OOB Server starts with its process affinity set to the default (available to run on all processors). However, unchecking the `CPU_n` check boxes on the change configuration page changes the OOB Server affinity so that it is restricted to a subset of the configured processors.

Note that the OOB Server only sets process affinity when starting up, so any changes made will only affect the server after it has been restarted.

CONNECTION POOLING

Automatic connection pooling in the OOB Server process may be enabled from the Web Administrator **Configuration** page (see **"ConnectionPooling" on page 214**).

If connection pooling is enabled then the OOB Server sets the `SQL_ATTR_CONNECTION_POOLING` attribute to `SQL_CP_ONE_PER_DRIVER` to enable connection pooling in the driver manager for the ODBC driver.

If connection pooling is enabled then connections through the ODBC driver to the database are held open by the driver manager for use the next time the application uses the same DSN.

This is only of any real benefit if the OOB Server is running in multi-threaded mode (the default), as the pooled connections are per process.

Enabling connection pooling in the OOB Server can vastly reduce connection times, especially through ODBC drivers connecting to remote databases.

EXPORTING SERVER CONFIGURATIONS

OOB Server configuration is stored in the registry (in Windows) or in a plain text file (on Unix systems).

It is simple to safeguard this configuration or copy it to another OOB Server on Unix, but normally more difficult on Windows.

In Windows, at the bottom of the **Configuration** screen in the Web Administrator there is an **Export** button.

Clicking **Export** writes the entire OOB Server configuration out to the file `LogDir\oob.reg` where `LogDir` is an OOB Server configurable parameter defined on the **Configuration** screen.

This file is in `RegEdit 4` format, so it can be copied to other servers where double-clicking on it will install that OOB Server configuration into the registry.

THE SECURITY SCREEN

The Web Administrator **Security** screen displays and allows the user to change the set of hosts that are allowed to connect to the OOB Server.

If they have not already been entered, you are then prompted for the HTTPAdmin user name and password before the **Security** screen is displayed:

Figure 47: The Security screen of the Web Administrator

In addition to the user names and passwords of your system and of the database management system, the Easysoft ODBC-ODBC Bridge provides another layer of security by using access control lists.

To add an IP address to a list, type the address into either the **Allowed Access** or **Denied Access** boxes and click **Add**.

Unix administrators will recognize this mechanism from the `hosts.allow` and `hosts.deny` files.

NB	Though the approach is similar, the rules for determining whether or not a host should be allowed to connect are different from those for `hosts.allow` and `hosts.deny`. Please read the rest of this section for more information.

Easysoft ODBC-ODBC Bridge Security takes the form of two lists of IP addresses.

When a host attempts to connect to the OOB Server, access is only granted if:

- the `allowed` list is empty and the IP address is not in the `denied` list
- the IP address is in the `allowed` list and not in the `denied` list.

The lists can be edited via the Web Administrator (Windows and Unix when run standalone), their relevant flat files (Unix), or the registry (Windows),

List entries must consist of full dotted quad notation IP addresses. For example, `194.131.236.4`. Entries may be followed by an optional "/" and a network mask. For example, `192.168.5.0/255.255.255.0` specifies the private class C address `192.168.5`. You can use the address/mask notation to specify any network address. If you omit the network mask it defaults to `255.255.255.255` i.e. an exact match for the specified IP address.

To delete a list entry, click the entry you want to delete.

If you have just installed the OOB Server and selected the "initially secure" option then the "Denied Access" column will contain `0.0.0.0/0.0.0.0` (which denies everyone access to the OOB Server).

ACCESS CONTROL PER DSN

The Easysoft ODBC-ODBC Bridge provides these methods of controlling access to data sources:

- Authentication of clients who must provide a valid user name/password for the host operating system.
- Access control lists that restrict access to the DSN by client IP.

The **DSN Access Control** section of the **Security** screen provides the additional facility to build access control lists to give you even more control over specific data sources.

With DSN access control lists you can define exactly which machines and users access which DSNs.

Figure 48: The Web Administrator Access Control table

The DSN `Allowed Access` list and `Denied Access` list define the hosts and users which may access a particular DSN.

The server consults the `Deny Access` list first, and access is denied if a client address/user name matches an entry in this list.

If there is no match for the client/user name in the `Denied Access` list then the Server consults the `Allowed Access` list, and if the `Allowed Access` list is empty the client/user is allowed access.

If there are any entries in the `Allowed Access` list then the client/user is only allowed access if it matches an entry in this list.

The entries in either list must consist of either IP addresses or user names. IP addresses must be full dotted quad notation. For example, `194.131.236.4`. Entries may be followed by an optional "/" and a network mask. For example, `192.168.5.0/255.255.255.0` specifies the private class C address `192.168.5`. You can use the address/mask notation to specify any network address. If you omit the network mask it defaults to `255.255.255.255` i.e. an exact match for the specified IP address.

Any user name added to either list must match the `LogonUser` attribute used by the OOB Client to log the OOB Server thread/process into that account before allowing access to a DSN.

NB	As a special case (to avoid having to name every user when adding an IP address to the DSN allow list) the deny list is checked for the current user, and allows access to the DSN if no matching entry is found.

SECURING THE WEB ADMINISTRATOR FROM CLIENTS OUTSIDE YOUR NETWORK

The `HTTPNetworkAccess` configurable parameter can be used to restrict access to the Web Administrator by setting it to the network or IP address of clients that are to be allowed access.

The IP address may be followed by an optional "/" and a network mask. For example, `194.131.236.4` only allows one machine to access the Web Administrator. `194.131.236/255.255.255.0` allows anyone on that network.

Browser clients not permitted access will receive an "`HTTP 403 Forbidden`" error. By default, all browser clients have access to the Web Administrator, although they may be required to enter a user name/password if `HTTPAdmin` is set to anything other than `"disabled"`.

PREVENTING DATA SOURCES BEING BROWSED IN THE ADMINISTRATOR

The Easysoft ODBC-ODBC Bridge is not only capable of displaying a list of system data sources, but can also allow those data sources to be browsed (see **"Viewing data sources, tables, columns and data" on page 281**).

This is a powerful, but potentially dangerous feature, as anyone with a browser and access to the Web Administrator might be able to see data in your database.

Although you can protect all of the Web Administrator pages with HTTP authentication, the `AllowDBBrowse` parameter allows the user to define a more precise configuration.

By default `AllowDBBrowse` is not set, which means that data sources shown on the **Data Sources** page are listed but not browsable. If `AllowDBBrowse` is turned on then the data sources become hyper links allowing you to browse down through the DSNs, tables, columns and data.

Configuring the OOB Server under Windows

The OOB Server under Windows is installed as a service, configured to be started automatically by the Windows Service Manager when the machine boots.

A `Services` entry is also added to the Windows Easysoft ODBC-ODBC Bridge menu.

> **NB** The OOB Server writes status, diagnostic, and security information to the Windows application event log. To examine this information, use Event Viewer to view the application event log. For more information, see **"Event Logging On Windows" on page 279**.

The OOB Server runs with administrative privileges, allowing it to become the user specified in the client DSN.

When a user connects, the OOB Server becomes the user specified in the `LogonUser` attribute so that it acts with the specified user's permissions.

You can configure the service like any other service. For instance, you can set it up not to start automatically on system boot.

You can also explicitly stop the service, as follows:

Choose:

1. Open **Administrative Tools** in **Control Panel**.
2. Open **Services**.

   The Service Manager displays a list of services, along with their status and their startup configuration.

TO START OR STOP THE SERVICE IN WINDOWS

1. In the **Services** dialog box, ensure that the `Easysoft ODBC-ODBC Bridge Server` entry is selected.
2. If the service does not have `Started` recorded in the **Status** column, you can click **Start** to run the service.

   – OR –

If the service has `Started` recorded in the **Status** column, you can click **Stop** to stop the service. Click **Yes** if the Service Manager asks you to confirm your action.

64-bit Windows

By default, only the 32-bit OOB Server is started automatically. (With the default configuration settings, both 32-bit and 64-bit OOB servers listening for incoming client connections on port 8888, it is not possible for both servers to run at the same time.) If you want to use the 64-bit OOB Server, you need to:

Stop the 32-bit OOB Server service (`Easysoft ODBC-ODBC Bridge Server`).

Start the 64-bit OOB Server service (`Easysoft ODBC-ODBC Bridge Server x64`).

To configure the 64-bit OOB Server service to start automatically when Windows starts:

1. In the Windows Services dialog box, double-click `Easysoft ODBC-ODBC Bridge Server`.

2. In the Easysoft ODBC-ODBC Bridge Server Properties dialog box, click Stop. In the Startup type list, choose **Manual**.

The 32-bit OOB Server service is no longer running and will not start when Windows starts.

3. In the Windows Services dialog box, double-click `Easysoft ODBC-ODBC Bridge Server x64`.

4. In the `Easysoft ODBC-ODBC Bridge Server Properties x64` dialog box, click Start. In the Startup type list, choose **Automatic**.

The 64-bit OOB Server service is now running and will start when Windows starts.

TO CONFIGURE AUTOMATIC/MANUAL/DISABLED STATUS

When installed, the Easysoft ODBC-ODBC Bridge Server is configured to run automatically when Windows starts. Windows offers two other options for services.

The Manual startup mode dictates that the service should start only when a user explicitly runs the service.

If you need to stop the service, for example if you need to restructure your data sources or you have a security policy stating that services should not be available outside allotted times. Windows provides the Disabled state for this.

1. In the **Services** dialog box, double-click the `Easysoft ODBC-ODBC Bridge Server` entry.
2. In the Service configuration dialog box, in the **Startup Type** section, choose the relevant option:
   - If you want the service to be started automatically by Windows, select **Automatic**. This is the recommended option.

   – OR –

   - If you want the service to be available only when started by a user on the local machine, select **Manual**.

   – OR –

   - If you want to disable the service, select **Disabled**.
3. Click **OK** to close the dialog box.

To Set Up Recovery Actions For The OOB Server

If you are running the OOB Server as a service in multi-threaded mode (the default) and an exception is caught by the OOB Server it will shut itself down. The reason for this is that the exception could have occurred anywhere (most commonly, the ODBC driver you are using) and there is no way for the OOB Server to correct the problem and no way for it to protect the other threads in the service.

On later versions of Windows it is possible to set the service manager to monitor services and restart them for you after a specified time:

1. In the **Services** dialog box, double-click the `Easysoft ODBC-ODBC Bridge Server` entry.
2. In the Service configuration dialog box, click the **Recovery** tab.
3. Click the recovery action you want in **First failure**, **Second failure**, and **Subsequent failures.**

User Accounts and the Server

Windows distinguishes two types of service with respect to their privileges:

- services running in the System Account have privileged access to resources on the local machine
- services set to run as a specific user are given the set of privileges assigned to that user.

The OOB Server is designed to be run in the System Account. As soon as a client connects, authentication takes place using the `LogonUser` and `LogonAuth` attributes. The server calls the `LogonUser()` and `ImpersonateLoggedOnUser()` Windows APIs, so that all subsequent actions are attempted with only the privileges of `LogonUser`.

This is the default configuration of the OOB Server. The other option is to set up the server to always run with a specific user's permissions. This is **not recommended**, but if you do want to do this, the procedure is as follows.

Caution!

Running the OOB Server under a named account for added security is counterproductive because you open up security holes by doing this. Firstly, anyone with the OOB client can now connect to your data and secondly, to administer the OOB Server via HTTP you will have to disable HTTP authentication. Disabling HTTP authentication means anyone can change the OOB Server settings.

Easysoft recommend you run the OOB Server in the System Account.

1. Go into the OOB Web Administrator and disable authentication.
2. In the OOB Web Administrator, disable HTTP authentication by setting `HTTPAdmin` to "`disabled`" (omit the quotes).
3. Be sure that the user account you wish to use is set up and that you know its password. It may be an existing end-user's account, or one created specifically for the OOB Server.

Caution!

For the OOB Server to function correctly when running as a specific user, that user needs two important access rights: `Log On as a Service`, which is needed for the service to start, and `Act as Part of Operating System`, which is needed so that the service can authenticate connecting users for the OOB Server and for the Web Administrator.

4. In the **Services** dialog box, double-click the `Easysoft ODBC-ODBC Bridge Server` entry to open the Service configuration dialog box.

5. In the **Log On** tab, in the **Log On As** section, select **This Account**.
6. In the **This Account** text box, type the user name or click **...** to browse for a user name.
7. In the **Password** box, type the user account password.
8. Type the same password in the **Confirm Password** box.
9. Click **OK** to close the dialog box and commit your changes.

You may have noticed the **Allow Service to Interact With Desktop** checkbox. This has no effect on the OOB Server.

OOB Configuration Parameters in the Registry

OOB and Web Administrator configuration settings are stored in the registry. The settings are saved under this registry key:

```
\HKEY_LOCAL_MACHINE\SOFTWARE\Easysoft ODBC-ODBC Bridge\Configuration\System\Settings.
```

"The Configuration Screen" on page 202 describes the available configuration settings.

64-bit Windows

OOB and Web Administrator configuration settings are saved under this registry key:

```
\HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Easysoft ODBC-ODBC Bridge\Configuration\System\Settings
```

Configuring the OOB Server under Unix

On Unix systems, the OOB Server is by default installed as a service under `inetd`, but it may be configured in its standalone mode.

When the OOB Server is installed as an `inetd`-controlled service, entries are added to the `inetd.conf` and `services` files. Examples of these entries might be

- in `inetd.conf`:

  ```
  esoobserver stream tcp nowait root /bin/sh
  /bin/sh /usr/local/easysoft/oob/server/server
  ```

- and in `services`:

  ```
  esoobserver 8888/tcp # Easysoft OOB Server
  ```

These are just examples and may differ if you picked a different port, service name or installation path.

THE MECHANICS OF `inetd` AND THE SERVER

`inetd` handles incoming network connections and starts the relevant program to service each incoming request.

When `inetd` starts (or is sent a `SIGHUP`), it reads `/etc/inetd.conf` for a list of what services to supply and to configure its handler for each service.

It then starts listening on the ports defined in `/etc/services` for each service configured in `inetd.conf`.

When a new connection is made, `inetd` starts the relevant program and hands over control.

The following example applies to the default installation: when a client connects to port `8888`, `inetd` uses the information read from `/etc/inetd.conf` to decide which program to run and with what arguments.

NB	Note that `/bin/sh` is repeated. The first time it appears (in the sixth position on the line) it is the name of the executable to run. The second time it appears it is the value to be passed as the *zeroeth* argument to `/bin/sh`. Normally this is the same as the name of the executable.

In the default installation, the arguments on the line cause `/bin/sh` to run the OOB Server startup script `<InstallDir>/easysoft/oob/server/server`.

The script then sets any necessary environment variables required by the dynamic linker and runs `esoobserver`.

The script passes an argument of `inetd` to the server executable, which notifies the server that it is running under `inetd` rather than at the command line.

The OOB Server inherits the socket from `inetd` and begins communicating with the OOB Client. `inetd` returns to listening on port 8888 for any new connections.

NB	The Web Administrator feature of the OOB Server does not run when the OOB Server is configured under `inetd`, because `esoobserver` itself is not run until an OOB Client connects.

CHOOSING ANOTHER PORT OR SERVICE NAME

If you have a port conflict or a service name conflict, or for some other reason you want to change the port or service name of the OOB Server, then you will need to edit the configuration files manually.

The `/etc/services` file maps service names to ports. The `/etc/inetd.conf` file lists the services that `inetd` will handle requests for.

After making the necessary amendments you will have to send SIGHUP to the `inetd` process to make it re-read the files. This can be achieved with the command:

`kill -HUP` *pid*

where *pid* is the process ID of `inetd`. You need to be `root` to edit the configuration files and send `inetd` a SIGHUP.

CONFIGURING THE SERVER AS STANDALONE

By default, on Unix the OOB Server is installed as a service with entries in the `/etc/services` and `/etc/inetd.conf` files (`inetd` is informed of the configuration file changes).

However, the OOB Server will also run standalone. When run as standalone, the OOB Server is faster at accepting connections than the `inetd` started server. So if you are making a lot of small connections to the OOB Server, starting it standalone (without `inetd`) is preferable. Also, the Web Administrator is only available when the OOB Server is running standalone.

Note that the OOB Server no longer uses `inetd` to listen on the socket when running as standalone, so you cannot run `tcp` wrappers. You can use the facilities within the Easysoft ODBC-ODBC Bridge to set up access control.

To start the OOB Server in standalone mode:

1. Comment out the esoobserver entry in the `/etc/services` file.
2. Force `inetd` to re-read the files using:

   ```
   kill -HUP inetd
   ```

3. Start the server with:

   path`/esoobserver standalone`

   This notifies the server that it must listen on the port itself and fork its own child process when a connection is made (the OOB Server reads the `port` and `httpport` settings in *path*`/oobserver.ini` for its configuration).

NB	You should start the OOB Server as the `root` user otherwise it will be unable to change to the `LogonUser` specified in the OOB Client data source. However, if you do not start the OOB Server as `root` then no logon authentication will be performed and all clients will run on the server as the user who started the OOB Server.

You could add a line to one of your `rc` scripts to have the OOB Server start up as standalone every time the machine boots.

The file you need to edit depends on your operating system so consult your system administrator for further information.

When the OOB Server is started standalone, by default it will fork a process to handle HTTP requests on port 8890 (you can change this in `esoobserver.ini`).

You can then use your browser to go to the Web Administrator (see **"The Web Administrator" on page 197**).

The `HTTPAdmin` configuration parameter controls whether you're prompted for a password before making changes in the Web Administrator. The default value for `HTTPAdmin` is "disabled" which means that you're not prompted before accessing Web Administrator pages that let you change configuration settings. If you do want to control access to these Web Administrator pages, edit the `HTTPAdmin` value. Specify a user name that exists on the server the OOB server is running on. If you do want to do this, the OOB Server must be running as `root` on some machines to validate the user name and password. For more information, see **"HTTPAdmin" on page 206**.

CHANGING SERVER CONFIGURABLE PARAMETERS UNDER UNIX

If you're running the standalone OOB Server you can use the Web Administrator to examine and change OOB configuration settings. If your OOB server runs under `inetd` or you prefer a configuration file to a web-based interface, you'll need to edit the following file to configure the OOB:

<InstallPath>`/easysoft/oob/server/esoobserver.ini`.

If <*InstallPath*> is anything other than `/usr/local/` then there will be a symbolic link `/usr/local/easysoft` to the real `easysoft` directory.

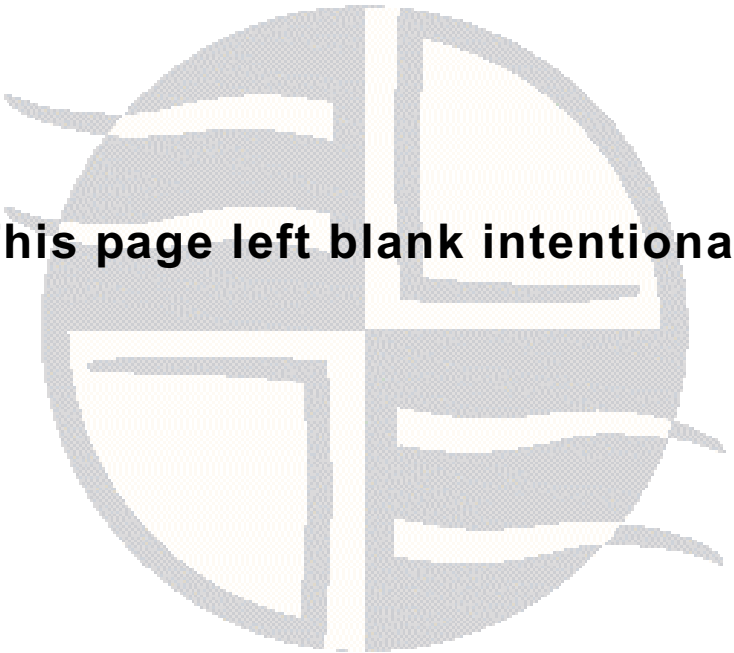
`esoobserver.ini` entries have the following format:

```
{Settings}

attribute = value
```

"The Configuration Screen" on page 202 describes the available configuration settings.

Note that the bitmask values are given either in decimal or in hexadecimal and the access control lists are separated simply by spaces.

Set `Flags` by entering the sum of the corresponding bitmask values of your required options.



This page left blank intentionally

CHAPTER 5 INTERFACING

Using the Easysoft ODBC-ODBC Bridge with other software

This section provides information about third-party programming languages and applications that are commonly used with the Easysoft ODBC-ODBC Bridge (OOB) to access remote data sources. There are tutorials on many of these applications and interfaces in the <*InstallDir*>`/easysoft/oob/doc` directory and these are updated on the Easysoft web site at **http://www.easysoft.com**.

Chapter Guide

- **Snort**
- **SQLPlus_hsODBC**
- **StarOffice**

Introduction

EXAMPLE PROGRAMS

The Easysoft ODBC-ODBC Bridge comes with several example programs which are installed into the `examples` subdirectory. The example programs are provided to help you get started writing applications that will load the OOB Client.

The C examples can be compiled and run like any C program, but for the others you may need to rebuild or reconfigure the application or interface with support for an ODBC driver manager.

INFORMATION FOR WINDOWS DEVELOPERS

Some of the programming languages, tools and applications referred to in this section (and in the files supplied in the `docs` subdirectory) are available on Windows as well as on Unix platforms, and work equally well with the Easysoft ODBC-ODBC Bridge in both environments.

However, the Windows versions of these products are usually pre-built and so you are unlikely to need the information about building or configuring them with the Easysoft ODBC-ODBC Bridge.

Since most of the following information is more likely to be used by the Unix community, all references to directory paths are given in their Unix format, even though the information may also apply to Windows.

unixODBC

The Easysoft ODBC-ODBC Bridge (OOB) does not require a driver manager at the client end although there are many advantages to using one:

- ability to pick a DSN and have the correct ODBC driver dynamically loaded for you.
- ability to work with multiple ODBC drivers.
- support for DSN-less connections.
- support and conversions between ODBC 2 applications and ODBC 3 drivers and vice versa.
- one central place for all your ODBC driver and data source configuration.
- automatic installation of OOB Client ODBC Driver and data source

The unixODBC driver manager is Open Source (LGPL) and was developed by Nick Gorham (an Easysoft developer). It is available as part of the unixODBC project being jointly developed by a number of people and headed by Nick Gorham.

REF For more information on unixODBC, see **http://www.unixodbc.org**.

If you have multiple ODBC drivers you probably need a Driver Manager.

There are currently two driver managers available as Open Source for Unix, unixODBC (**http://www.unixodbc.org**) and iODBC. If you want or need to use a driver manager then Easysoft recommend the unixODBC driver manager. There are a number of reasons for this:

1. The unixODBC project started by Peter Harvey is now maintained by Nick Gorham who is an Easysoft developer. This means there is much greater experience with unixODBC within Easysoft and we will be able to provide better support for OOB running under unixODBC. It also means that if you find a problem in unixODBC it is much easier for us to facilitate a fix.
2. Easysoft test all their ODBC drivers with unixODBC and although they may work with iODBC it is more difficult for us to support it as we are less familiar with it.
3. The unixODBC package contains much more than a driver manager. The aim of the unixODBC project is to provide all the ODBC functionality available on Windows for Unix operating systems. The unixODBC package may be built with the QT libraries to allow GUI configuration of DSNs and drivers. It also contains the GUI DataManager program which may be used to explore your ODBC data. OOB contains the code and a shared object which is used by unixODBC's GUI ODBCConfig utility to add/delete and configure OOB DSNs.
4. The OOB installation for Unix can automatically install itself using the unixODBC installer program to run under unixODBC.
5. The OOB installation can automatically configure a demo data source in unixODBC for OOB.
6. Many (if not most) applications and interfaces know about unixODBC.

A Simple OOB Client in C

In **"The Demo Application" on page 162**, you ran a simple C program that connected through the Easysoft ODBC-ODBC Bridge.

Other small programs are provided in the `examples` directory, and documented in the file `EXAMPLES.txt`.

One of these is `getdata`, a simple C program that reads `stdin` for an SQL statement and passes it to a data source for execution.

NB	The syntax in this example serves as a guide for Unix platforms. Windows developers should refer to the manual for their chosen development platform.

Before you can use the program you must edit the call to `SQLDriverConnect()` to indicate a valid DSN.

You can then build the program by finding a suitable makefile in the `examples` subdirectory:

1. Change to the examples directory by typing:

   `cd` *<InstallDir>*`/easysoft/oob/examples`

2. Look for a makefile by typing:

   ```
   ls Make*
   ```

3. Edit the makefile appropriate for your platform (e.g. `Makefile.linux`), removing the comment from the second line and making sure that the installation path is correct (for example, `#INSTALLPATH=/usr/local` might become `INSTALLPATH=/opt/`).

   Note that you should remove the "`#`", whether or not you change the directory name.

4. Make the examples:

   ```
   make -f Makefile.linux
   ```

   The `getdata` program, the source to the demo program, and the makefile are enough to show you how to create C programs that connect through the Easysoft ODBC-ODBC Bridge.

   If you need more information, please refer to an ODBC Programmer's Manual.

Apache/PHP

Building Apache with PHP and the Easysoft ODBC-ODBC Bridge allows your web content to be closely integrated with a database running on a machine separate from your web server.

This is particularly useful for offices that have databases that would benefit from up-to-date information from a web server located on a different machine.

For information about building Apache and PHP with the Easysoft ODBC-ODBC Bridge and details of which versions of Apache and PHP the Easysoft ODBC-ODBC Bridge has been proven to work with, please refer to the Apache/PHP document **Apache_PHP.txt** in *<InstallDir>*`/easysoft/oob/doc`.

Also check the INSTALL files in the Apache and PHP distributions, because they may provide more up-to-date information specific to Apache and PHP.

REF Apache can be found at **http://www.apache.org** and PHP at **http://www.php.net**.

If you want to access multiple ODBC drivers from PHP the best method is to install the unixODBC driver manager and then tell the unixODBC driver manager about your ODBC drivers.

You can build PHP with an ODBC driver and other database drivers such as Oracle or MySQL.

Applixware

Applixware Office is a multi-platform suite of applications.

For information about using Applixware Office with the Easysoft ODBC-ODBC Bridge, and details of which versions of Applixware Office the Easysoft ODBC-ODBC Bridge has been tested with, please refer to the Applixware document **Applixware.txt** in *<InstallDir>*`/easysoft/oob/doc`.

REF For information about Applixware Office, visit **http://www.vistasource.com** or **http://www.applix.com**.

Lotus Notes / Domino

Lotus Notes combines e-mail, calendar management, group scheduling, contact and task management in one client application. Lotus Domino turns Lotus Notes into an application and messaging server.

For information about using Lotus Notes / Domino with the Easysoft ODBC-ODBC Bridge, and details of which versions of Lotus Notes / Domino the Easysoft ODBC-ODBC Bridge has been tested with, please refer to the Lotus Notes / Domino document **LotusNotesDomino.txt** in <*InstallDir*>`/easysoft/oob/doc`.

mnoGoSearch (formerly UDMSearch)

mnoGoSearch is a web site indexing tool and search engine you can install on your web site. Both tools use a relational database backend to store the metadata and execute searches.

Using mnoGoSearch with the Easysoft ODBC-ODBC Bridge allows you to store the mnoGoSearch index information in a remote ODBC data source (e.g. running mnoGoSearch on your Unix web server, but storing the indexed web site information in your Microsoft SQLServer database).

For information about using mnoGoSearch with the Easysoft ODBC-ODBC Bridge, and details of which versions of mnoGoSearch the Easysoft ODBC-ODBC Bridge has been tested with, please refer to the mnoGoSearch document **mnoGoSearch.txt** in *<InstallDir>*`/easysoft/oob/doc`.

Also check the INSTALL file in the mnoGoSearch distributions because it provides instructions for building mnoGoSearch with the Easysoft ODBC-ODBC Bridge.

REF The official home page for mnoGoSearch with links to the latest downloads, language dictionaries and documentation is at **http://search.mnogo.ru**.

mxODBC

mxODBC is a database API for the Python scripting language which provides an interface to ODBC data sources.

For information about running mxODBC with the Easysoft ODBC-ODBC Bridge, and details of which versions of mxODBC the Easysoft ODBC-ODBC Bridge has been tested with, please refer to the mxODBC document **mxODBC.txt** in *<InstallDir>*`/easysoft/oob/doc`.

When building mxODBC, there is a configuration section specifically for the Easysoft ODBC-ODBC Bridge which makes the build very straightforward.

REF mxODBC and instructions for building it are available at **http://www.egenix.com/products/python/mxODBC/**.

OpenOffice.org

OpenOffice.org is the open source project through which Sun Microsystems has released the technology for the StarOffice Productivity Suite. OpenOffice.org can use ODBC data sources in its various applications to link to external data.

For information about using OpenOffice.org with the Easysoft ODBC-ODBC Bridge, and details of which versions of OpenOffice.org the Easysoft ODBC-ODBC Bridge has been tested with, please refer to the OpenOffice.org document **OpenOffice.org.txt** in <*InstallDir*>`/easysoft/oob/doc`.

REF For product information about OpenOffice.org, visit **http://www.openoffice.org/product**

Perl DBI DBD::ODBC

DBI is the database interface module for Perl.

When DBD::ODBC (the ODBC database driver for Perl DBI) is built with an ODBC driver manager it enables access to ODBC drivers and when using the Easysoft ODBC-ODBC Bridge, it provides access to ODBC data sources on other machines.

For information about building Perl DBI with the DBD:ODBC module and the Easysoft ODBC-ODBC Bridge, and details of which versions of Perl DBI the Easysoft ODBC-ODBC Bridge has been proven to work with, please refer to the Perl document **Perl_DBI_DBD_ODBC.txt** in <*InstallDir*>`/easysoft/oob/doc`.

There are also some Perl examples in the <*InstallDir*>`/easysoft/oob/examples` directory.

REF

The Perl home page is at **http://www.perl.com**.

Information about DBI (the database interface module for Perl) can be found at **http://dbi.perl.org**.

Links to other useful sites can be found in the Perl document **Perl_DBI_DBD_ODBC.txt** within <*InstallDir*>`/easysoft/oob/doc`.

Various Perl tutorials are installed in the <*InstallDir*>`/easysoft/oob/doc/perl_tutorials` directory. The Perl tutorials are updated on the Easysoft web site.

Rexx/SQL

Rexx/SQL provides Rexx programmers with a consistent, simple and powerful interface to SQL databases.

For information about using Rexx/SQL with the Easysoft ODBC-ODBC Bridge, and details of which versions of Rexx/SQL the Easysoft ODBC-ODBC Bridge has been tested with, please refer to the Rexx/SQL document **Rexx_SQL.txt** in *<InstallDir>*`/easysoft/oob/doc`. Also check the INSTALL files in the Rexx/SQL and Easysoft ODBC-ODBC Bridge distributions because they may provide more up-to-date information.

REF

The Rexx/SQL home page at **http://www.lightlink.com/hessling** contains download links, documentation, mailing list instructions and links to other Rexx resources.

The Rexx/SQL anonymous FTP site is mirrored in the US at **ftp://ftp.lightlink.com/pub/hessling/REXXSQL**.

You will need a Rexx interpreter to run the Rexx/SQL test code and your Rexx programs using Rexx/SQL.

The Rexx interpreter called Regina was used by Easysoft during the testing of the Easysoft ODBC-ODBC Bridge with Rexx/SQL and is available at **ftp://ftp.lightlink.com/pub/hessling/Regina**.

You may use `./configure` (help in Rexx/SQL) to see which Rexx interpreters are supported.

Please ensure that you install a Rexx interpreter before attempting to build Rexx/SQL.

Snort

Snort is an open source network intrusion detection system. Snort alerts and logs may be written to a database by using the database plugin. If snort is built against the unixODBC driver manager and the ODBC-ODBC Bridge client is installed, snort can then log to remote ODBC data sources.

For information about using Snort with the Easysoft ODBC-ODBC Bridge, and details of which versions of Snort the Easysoft ODBC-ODBC Bridge has been tested with, please refer to the Snort document **snort.txt** in <*InstallDir*>`/easysoft/oob/doc`.

SQLPlus_hsODBC

hsODBC (Heterogeneous Services) allows an Oracle listener to connect to an ODBC driver.

Once you have configured hsODBC to use the Easysoft ODBC-ODBC Bridge, you can use SQLPlus to connect to any remote database for which you have an ODBC driver (e.g. Microsoft SQL Server).

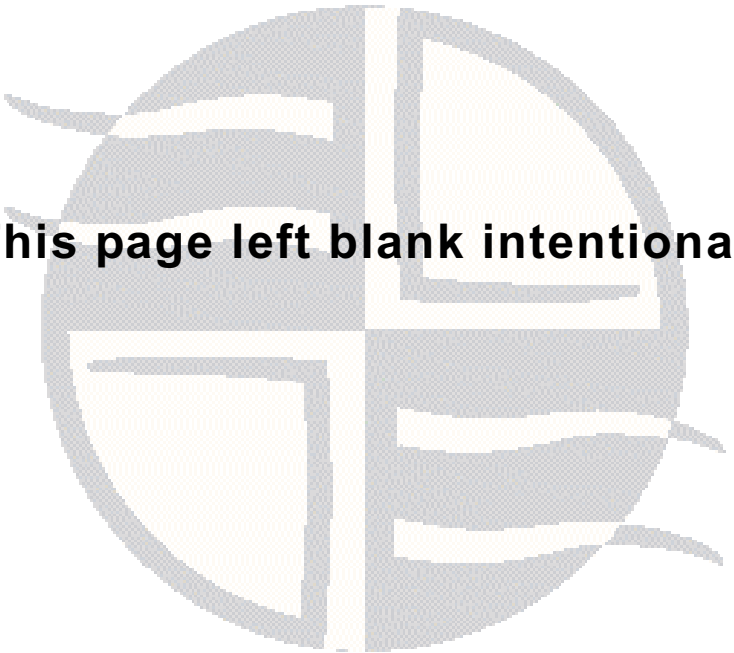
For information about running hsODBC with the Easysoft ODBC-ODBC Bridge and details of which versions of hsODBC the Easysoft ODBC-ODBC Bridge has been tested with, please refer to the hsODBC document **SQLPlus_hsODBC.txt** in *<InstallDir>*`/easysoft/oob/doc`.

StarOffice

StarOffice is a multi-platform suite of applications.

For information about using StarOffice with the Easysoft ODBC-ODBC Bridge, and details of which versions of StarOffice the Easysoft ODBC-ODBC Bridge has been tested with, please refer to the StarOffice document **StarOffice.txt** in *<InstallDir>*`/easysoft/oob/doc`.

REF For product information about StarOffice, go to **http://www.sun.com/products/staroffice**.



This page left blank intentionally

CHAPTER 6 REPORTING AND STATISTICS

The Easysoft ODBC-ODBC Bridge provides a suite of performance management reports and statistics that allow you to monitor and manage your system.

Chapter Guide

Introduction

One of the most frequent requests that Easysoft receive from enterprise customers is for greater reporting and statistics, so that they can see exactly what the OOB Server is doing at any time.

The Web Administrator **Statistics** screen displays a number of statistics detailing server up time, total connections, total successful connections, active threads/processes, peak concurrent threads/processes and the time of the last connect or disconnect:

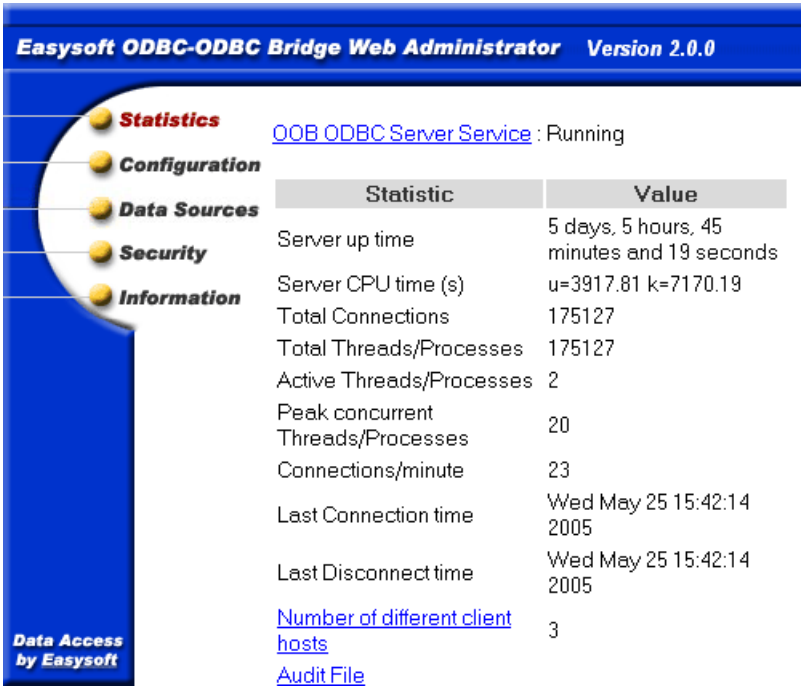


Figure 49: General data on the Web Administrator Statistics screen

THE STATISTICS SCREEN

The Web Administrator **Statistics** screen provides the following information:

- **Server up time -** The time in days, hours, minutes and seconds since the OOB Server was started.
- **Server CPU time(s) -** This is only visible if the `ShowProcessTime` option on the **Configuration** screen (see **"ShowProcessTime" on page 213**) is selected. One or more values are shown. If only one value is shown, it is the total CPU time consumed by the OOB Server. If two times are shown, the first is the user time and the second is the kernel time. Note that the CPU times shown include CPU time consumed by the ODBC Driver Manager, any ODBC drivers and any child processes.
- **Total Connections -** The total number of connections (or attempted connections) to the OOB Server. This includes connections dropped due to invalid license keys or insufficient license slots, port scanners or anyone using `telnet` to access the OOB Server ODBC port.
- **Total Threads/Processes -** The total number of threads or processes that the OOB Server has created during its execution. Connections denied access because of a Server access control rule (but not a DSN access control rule) or because the value of either `MaxThreadCount` (see **"MaxThreadCount" on page 208**) or `MaxClientConnect` (see **"MaxClientConnect" on page 208**) has been exceeded do not count because the OOB Server does not start a thread/process for these.

- **Active Threads/Processes -** The total number of active threads or processes the OOB Server has created to handle ODBC connections. However, this number may exceed the actual active count, as the OOB Server only looks for exited threads and processes when five seconds has elapsed without any connections or every 10 connections (to give preference to incoming connections). Note that if `MaxThreadCount` (see **"MaxThreadCount" on page 208**) or `MaxClientConnect` (see **"MaxClientConnect" on page 208**) is set to anything other than 0 then the OOB Server has to reap exited threads and processes every time a new connection arrives. In this case, the active count will always be up to date immediately after an ODBC connection from a client.
- **Peak concurrent Threads/Processes -** The maximum value recorded in the `Active Threads/Processes` field.
- **Connections/minute -** The Server up time (in minutes) divided by the number of `Total Threads/Processes`. This is a very useful measure of how busy the OOB Server itself is. However, if you are using persistent connections from your client application you should realise that this number is likely to always be low, since connections are being reused by your client.
- **Last Connection time -** The time of the last ODBC connection.
- **Last Disconnect time -** The time of the last disconnect.
- **Number of different client hosts -** The number of different client machines which have connected to the OOB Server (where a client machine is identified by its IP address). You can click on this link to get a list of IP addresses or machine names. Machine names are only displayed if you have `ReverseLookup` enabled (see **"ReverseLookup" on page 213**).

- **Audit File -** A link to another page which shows entries from the last audit trail file. As the audit trail file is renamed every day you will also see links to older audit trail files. Graphs of connections per hour and connections per minute are available for each audit file. Note that an audit trail file is only produced if `AuditODBCAccess` is enabled (see **"AuditODBCAccess" on page 213**).
- **DSN statistics -** Shows the first ten DSNs accessed over the Easysoft ODBC-ODBC Bridge, the number of times a connection to this DSN has been opened, the total time (in seconds) connections to this DSN have been open, the connections per minute and the average time per connection. Note that the total time is concurrent time (i.e. if the same DSN is open twice concurrently for ten seconds, Total time shows 20 seconds).

Changing the refresh frequency

The Web Administrator uses a set of template files into which the dynamic data is inserted before sending it back to your browser.

The template file for the **Statistics** screen is `index.html`, which is located in the `templates` directory wherever you installed the OOB Server.

Edit the `index.html` file and near the top you will see:

```
meta http-equiv="refresh" content="60"; URL=/index.html
```

Change the 60 (the refresh time in seconds) to your preferred setting.

NB	Note that setting the refresh time to a very low value will increase the workload on the OOB Server process which handles HTTP requests. As this may reduce the response time to the Easysoft ODBC-ODBC Bridge ODBC server thread, times much less than 60 seconds are not recommended.

DSN STATISTICS

The Easysoft ODBC-ODBC Bridge records each DSN to which it is connected and displays a table of DSN statistics:

DSN	Connections	Total Time (s)	Connections/minute	Average Time per connection (s)
cinemadev	55	256	0	4.65
cinema	53459	144557	49	2.70
web	4455	5399	4	1.21
webdev	108	282	0	2.61
company	1	2	0	2.00

Figure 50: DSN data on the Web Administrator Statistics screen

The information shown in the table per DSN is:

- **Number of connections to this DSN -** By comparing this number for each DSN you can see which is the busiest data source on your server.
- **Total time connected to this DSN -** This is concurrent time so two concurrent connections, connected for the same 5 seconds registers as 10 seconds total time.

- **Connections per minute to this DSN -** The server up time (in minutes) divided by the number of connections to this DSN. This includes connections that were refused by the server for various reasons e.g. access control or authentication.
- **Average time per connection -** The total time connected to this DSN divided by the number of connections to this DSN. If you know your client applications connect, issue some SQL and then disconnect, this is a good measure of how much processing there is for each connection. This number is likely to be higher for client applications using persistent connections, as there may be long periods of connected time where nothing is happening.

HOST STATISTICS

From the Web Administrator **Statistics** screen there is a link called **Number of different client hosts** which takes you to a table of statistics per connecting client:

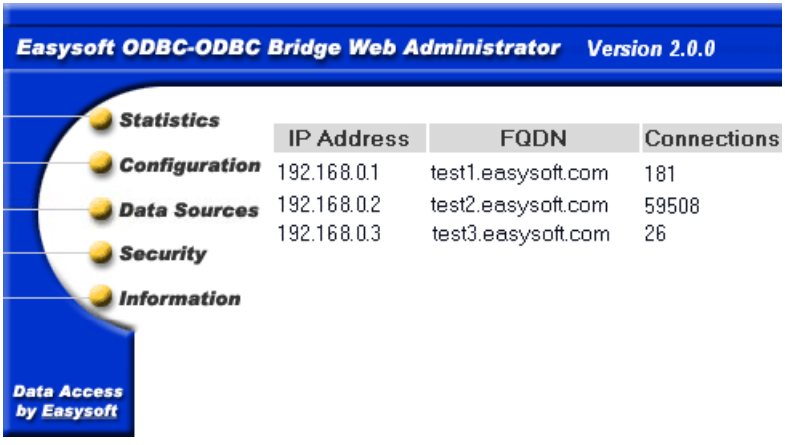


Figure 51: The Web Administrator Client Hosts screen

The Client Hosts screen lists the first 20 different clients that have connected to the OOB Server. These connection attempts may not have been successful (e.g. not authenticated or denied access). The entry for each client shows:

- **IP Address -** The IP address of the client.
- **FQDN -** The Fully Qualified Domain Name of connecting clients. This is displayed as "`unknown`" unless `ReverseLookup` is enabled (see **"ReverseLookup" on page 213**).
- **Connections -** The number of connections per client. If you have more than one connecting client then this allows you to see which client is putting the greatest load on your server.

TERMINATION STATISTICS

If you want the contents of the Web Administrator **Statistics** screen to be written to file, set the `WriteStatsOnExit` server parameter. When `WriteStatsOnExit` is set, statistics are written on exit to a file called `esoob_stats_<dd>_<mm>_<yyyy>_<hhmmss>.stats` where `<dd>` is the day number, `<mm>` is the month number, `<yyyy>` is the year number and `<hhmmss>` is the time in hours, minutes and seconds.

This file will be written into the directory defined by the server configurable parameter `LogDir`:

```
Server up time 0 days, 18 hours, 5 minutes and 23 seconds

Server CPU time(s):                u=92.66 k=242.72

Total Connections:                 58966

Total Threads/Processes:           58966
```

```
Active Threads/Processes:          4

Peak concurrent Threads/Processes: 74

Connections/minute:                54

Last Connection time:              Wed Jul 04 08:55:13 2001

Last Disconnect time:              Wed Jul 04 08:55:12 2001


DSN  Connections Total   Connections Average

DSN              Time(s) per minute  Time per

DSN                                  connection(s)


dsn1 3217        4153    2           1.29

dsn2 55566       212747  51          3.83

dsn3 42          8060    0           191.90

dsn4 111         101     0           0.91


Number of different client hosts: 3

192.168.0.1 (test1.easysoft.com)

192.168.0.2 (test2.easysoft.com)

192.168.0.3 (test3.easysoft.com)
```

NB In pre-2.1 versions of the OOB, connection statistics were always written to file when the OOB Server service exited. To restore the previous behaviour, enable the `WriteStatsOnExit` server parameter.

Log of Failing SQL

The OOB Server can keep a log of any failed `SQLPrepare`, `SQLExecute` or `SQLExecDirect` functions.

To enable this feature check the `LogFailingSQL` parameter on the Web Administrator **Configuration** screen (see **"LogFailingSQL" on page 214**), which will create a file called `oob_sql.log` when the first failing SQL occurs.

This file is located in the directory defined by the `LogDir` parameter on the Web Administrator **Configuration** screen (see **"LogDir" on 207**).

Any future SQL commands that fail are appended to the file once it has been created and you can rename or delete this file at any time.

The failing SQL log file has two types of entry:

[a] an entry showing a piece of SQL which failed:

```
date time "DSN" SQL text
```

e.g.

```
12-04-2002 10:32:55 "test" insert into mytable values (1,2)

---------- -------- ------ --------------------------------

   date         time     DSN name         the SQL text which failed
```

An error may be for a number of reasons, such as the SQL failing to parse correctly, or the database engine being unable to comply (a duplicate key error, for example):

```
31-07-2001 11:24:24 "example" update Title set Count=Count+1
where id = 1

01-08-2001 09:08:08 "web" select columnnoexist from stores

01-08-2001 09:05:05 "test" update sales set stor_id = 8042
where stor_id = "fred"

01-08-2001 09:06:06 "test" select * from tablenoexist
```

The DSN name field is extremely useful when identifying problems.

In this example the "`web`" and "`test`" DSNs both point at the same database, but "`web`" is the live version of the application and "`test`" is used by software developers.

As the client applications are on different machines, the `TargetDSN` used by the OOB Client is set to "`web`" on one client and to "`test`" on the other.

Failing SQL in the log to "`test`" would not be uncommon as this is under development, but failures for "`web`" indicate a problem a real user might have experienced running live software.

[b] an entry logging the first database error retrieved AFTER a piece of SQL failed:

```
date time SQLState="xxxxx", NativeError=nnnn, Msg="xxxxxxxx"
```

e.g.

```
12-04-2002 10:32:55 SQLState="23000", NativeError=2627
Msg="[Microsoft][ODBC SQL Server Driver][SQL Server]
Violation of PRIMARY KEY constraint"
```

The failing SQL file is written to as follows:

1. each piece of SQL executed with either `SQLPrepare`, `SQLExecute` or `SQLExecDirect` is documented where those APIs fail.
2. an entry is written in the format in [a] if an `SQLPrepare`, `SQLExecute` or `SQLExecDirect` function fails.
3. if [2] occurs then an entry is written in the format in [b] if the application calls `SQLError` or `SQLGetDiagRec`.
4. once the application is finished with the SQL in [1] (or executes more SQL) then the previous SQL is no longer available to log.

   This means that failing SQL is written to one line of the failing SQL log, followed by the error text returned from the database engine on the next line.

   However, this is NOT always the case, as the failing SQL statements and the error text can get mixed up if the server is very busy (for example, if multiple concurrent connections all fail at the same time).

Auditing

The OOB Server records all activity to an audit trail file if the `AuditODBCAccess` parameter is checked (see **"AuditODBCAccess" on 213** or **"Changing Server Configurable Parameters under Unix" on page 238**) on Unix).

The **Audit File** link on the Web Administrator **Statistics** screen displays a page showing:

- the first page from the current audit trail file.
- links for the current audit trail file to display the previous page, the next page, the first page and the last page.
- a link to each recorded audit trail file.
- two graph links for each audit trail file, one showing connections per minute and another connections per hour over the period of that audit trail file (a new file is opened each day), which show when the OOB Server is busiest (so that better decisions can be made for scheduling administration tasks that require taking the server down, for example) and highlight unexpected bursts in activity (e.g. a search robot cataloging your web site).

Audit File Description

The current Audit File is called `esoob_access.log` and will be placed in the directory specified by the server configurable parameter `LogDir`.

Here is an example of a few lines from the audit file:

```
Mon, 02 Jul 2001 15:01:07 GMT CONNECT
192.168.0.1(test1.easysoft.com)

Mon, 02 Jul 2001 15:01:12 GMT DISCONNECT
192.168.0.1(test1.easysoft.com)

Fri, 06 Jul 2001 13:10:46 GMT AUTH_DENIED
192.168.0.1(test1.easysoft.com)

Fri, 06 Jul 2001 14:34:36 GMT REFUSED_BY_DSN_RULE
test(Martin Evans)

Fri, 06 Jul 2001 15:18:34 GMT REFUSED_MAXCLIENTCONNECT
192.168.0.3(unknown)

Fri, 06 Jul 2001 15:19:33 GMT REFUSED_MAXCONCURRENT
192.168.0.2(test2.easysoft.com)

Mon, 09 Jul 2001 13:04:02 GMT REFUSED_BY_RULE
192.168.0.1(test1.easysoft.com)
```

The fields in each line of the audit file are:

- day - the day of the week
- day number - the day of the month
- month - the month name
- year - the year
- time - hours, minutes and seconds (colon separated)
- timezone - the time zone
- event - an event code (see **"Graph Generation" on page 277**)
- event arguments - details of the client, DSN or user name (see **"Graph Generation" on page 277**)

The events and their descriptions are:

- **CONNECT -** An OOB Client has connected to the OOB Server. This does not mean the client gained successful access to the ODBC data source, but that the server accepted the connection. If the OOB Server `Authentication_Disabled` parameter was not set (see **"The Configuration Screen" on page 202**) then the client was authenticated by the operating system and also completed any access control test based on its IP Address. The last field of the audit file shows the client IP address and also the client FQDN in brackets if the `ReverseLookup` parameter is enabled (see **"ReverseLookup" on page 213**).
- **DISCONNECT -** An OOB Client disconnected from the OOB Server. The audit file does not indicate why the client disconnected. The client may have called `SQLDisconnect`, simply been interrupted, or the OOB Server could have timed out the connection because it was inactive for the number of seconds set in the `TimeOut` parameter (see **"The Configuration Screen" on page 202**).
- **AUTH_DENIED -** This event happens when an OOB Client connects to the OOB Server, OOB Server authentication is enabled and the client passes an invalid user name/password (`LogonUser/LogonAuth`) pair for the operating system. The OOB Server has turned down the connection request.

- **REFUSED_MAXCLIENTCONNECT -** The connection attempt has been turned down. The OOB Client attempting connection to the OOB Server already has too many open connections. The OOB Server limits the connections from a particular client to `MaxClientConnect` (see **"The Configuration Screen" on page 202**).
- **REFUSED_MAXCONCURRENT -** The connection attempt has been turned down. There are already too many open connections as defined by `MaxThreadCount` (see **"The Configuration Screen" on page 202**).
- **REFUSED_BY_RULE -** The connection attempt has been turned down due to an client access control rule defined in the server on the Web Administrator **Security** screen.
- **REFUSED_BY_DSN_RULE -** The connection attempt has been turned down due to a DSN access control rule. These rules are defined in the server on the Web Administrator **Security** screen.
- **SERVER_SUSPENDED -** The service was suspended.
- **SERVER_RESUMED -** The service was resumed.

DAILY RENAMING

The audit files are renamed once a day at midnight. At midnight the current audit trail file (`esoob_access.log`) is renamed to `esoob_access_<dd>_<mm>_<yyy>.log` where `<dd>` is the day number, `<mm>` is the month number and `<yyyy>` is the year number.

The audit file for each day should be visible in the Web Administrator.

GRAPH GENERATION

The Web Administrator is capable of analysing a particular audit file and producing graphs of connections per minute and connections per hour for any selected day.

These graphs are accessed from the **Audit File** link on the Web Administrator **Statistics** screen.

Connections are displayed as blue bars and warning events are displayed as coloured warning bars on top of the relevant blue bar:

failed connects (Purple bar):

- AUTH_DENIED

attempts denied access due to access control lists of server limits (Red bar):

- REFUSED_MAXCLIENTCONNECT
- REFUSED_MAXCONCURRENT

authentication failures (Orange bar):

- REFUSED_BY_RULE
- REFUSED_BY_DSN_RULE

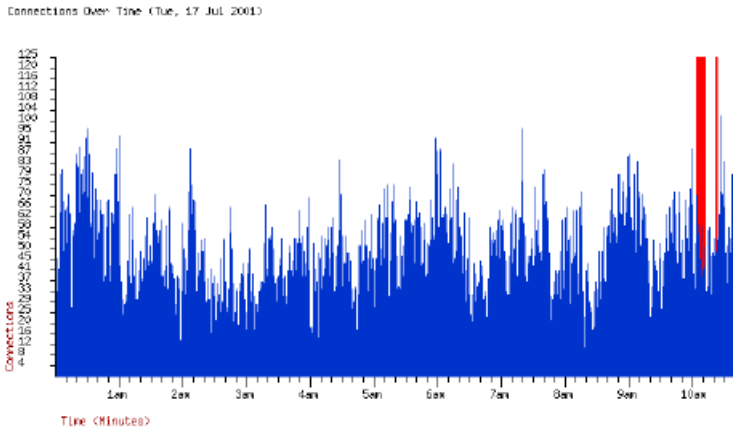


Figure 52: The Web Administrator Connections per Minute graph

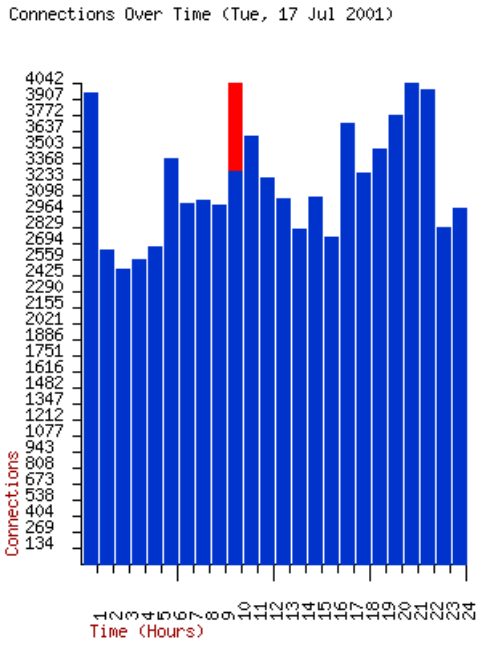


Figure 53: The Web Administrator Connections per Hour graph

EVENT LOGGING ON WINDOWS

The Windows OOB Server uses the application event log to record status, diagnostic, and security information. Events logged by the OOB Server include the following:

- The OOB Server was started, stopped, paused, or resumed.
- An attempt was made to access the OOB Server or a data source that was disallowed by an access control rule.
- An attempt was made to access a Web Administrator page that was disallowed because an invalid user name or password was supplied.
- The OOB Server caught an exception.

Much of the information written to the application event log is also written to the OOB Server log files (by default, these are stored in the *drive*:\Program Files\Easysoft\Easysoft ODBC-ODBC Bridge\Logs\ directory). Refer to the application event log if:

- The standard OOB auditing mechanism wasn't enabled for the period that you're interested in.
- You prefer to examine log file entries by using Event Viewer rather than a text editor.

To view the application event log

1. In **Control Panel**, open **Administrative Tools**, then open **Event Viewer**.
2. In the left hand pane of the **Event Viewer** window, click **Application**.

Browsing System Data Sources in the Web Administrator

The Web Administrator **Data Sources** screen displays the ODBC system data sources found by the server and their configured drivers. Note that if the `AllowDSNBrowse` parameter is disabled, no data source or driver information will display in this page.

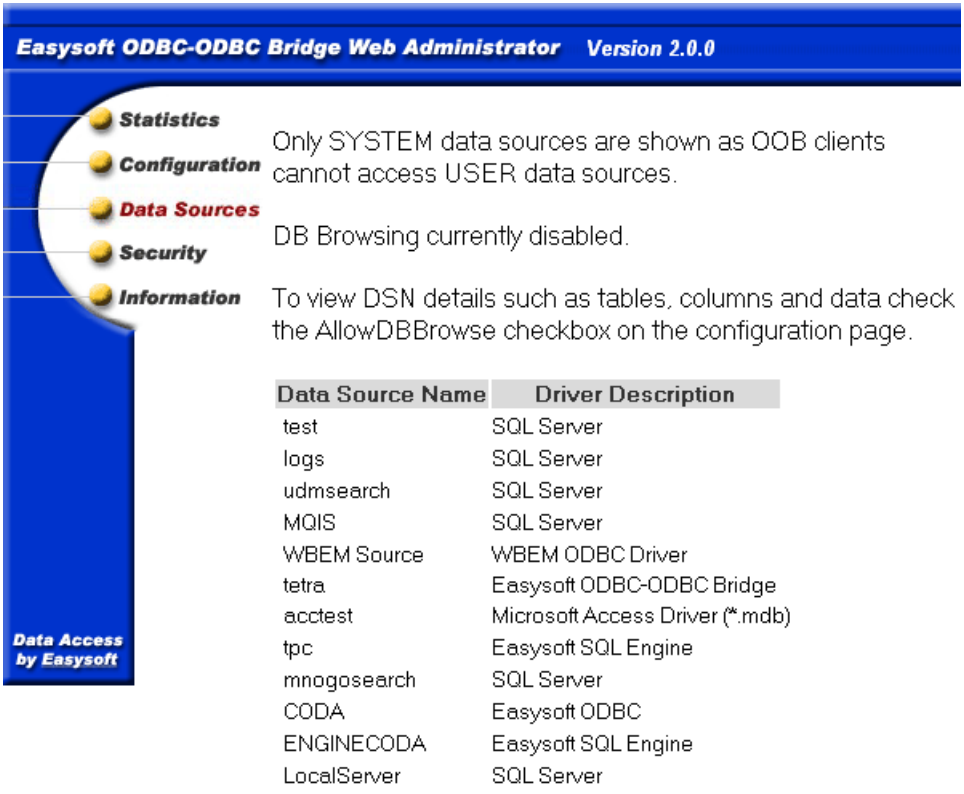


Figure 54: The Data Sources screen of the Web Administrator

VIEWING DATA SOURCES, TABLES, COLUMNS AND DATA

The Web Administrator facility lets you browse data sources for lists of tables, table composition and rows of data in the tables.

In order to browse system data sources on the machine where the OOB Server is running you must enable the `AllowDBBrowse` configurable parameter on the Web Administrator **Configuration** screen.

This converts the list of data source names on the `Data Sources` page into links to further pages, allowing you to browse down through the data source and its tables, columns and data.

You may be prompted for a database user name and password when first clicking on a DSN.

The OOB Server attempts an initial connection without the ODBC UID/PWD attributes, but if that fails with an authentication error (`28000`) an authentication challenge will be issued, where the realm is the name of the DSN.

NB	DSNs cannot be browsed using trusted connections.

Figure 55: The DSN Realm Enter Network Password dialog box

Having gained access, DSN details can then be displayed, as in the following example:

The initial screen displays the tables and views in the DSN `test` (i.e. it is the result set generated by calling `SQLTables` when connected to a DSN called `test`):

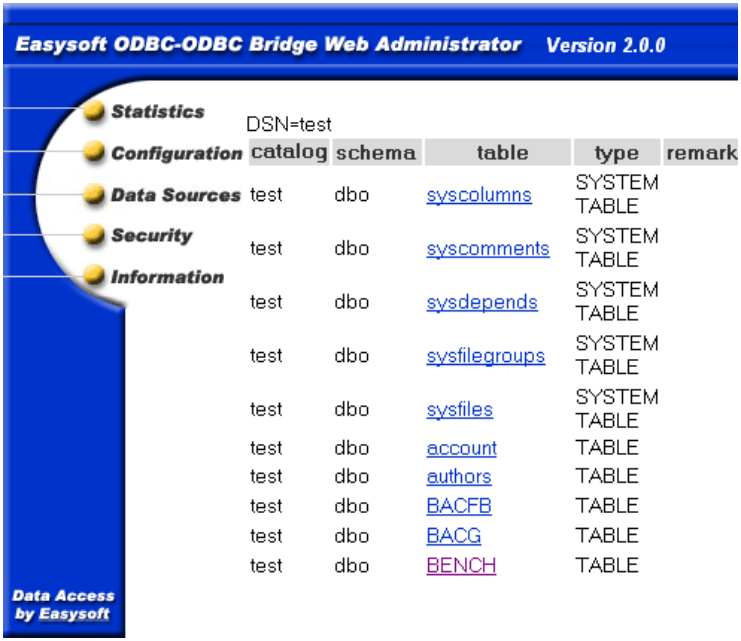


Figure 56: The Web Administrator Data Sources screen DSNs

Click on **BENCH** to display a list of the columns of data in that table (generated by calling `SQLColumns` on table `BENCH` in DSN `test`):

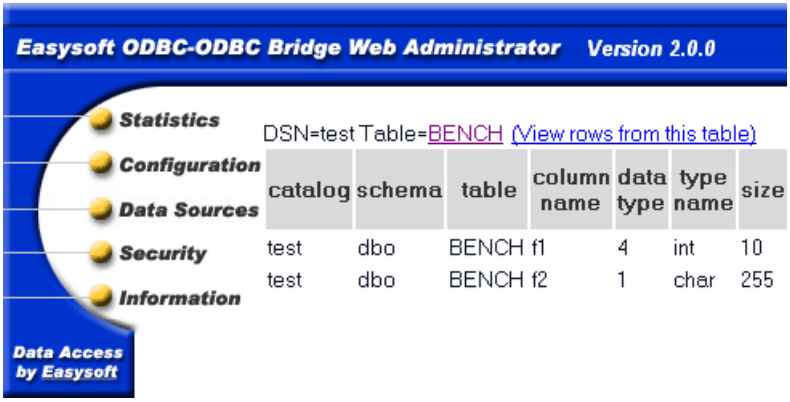


Figure 57: The Web Administrator Data Sources screen DSN data

Click on **View rows from this table** to display the first ten rows in that table (generated by a `"select * from BENCH"` statement):

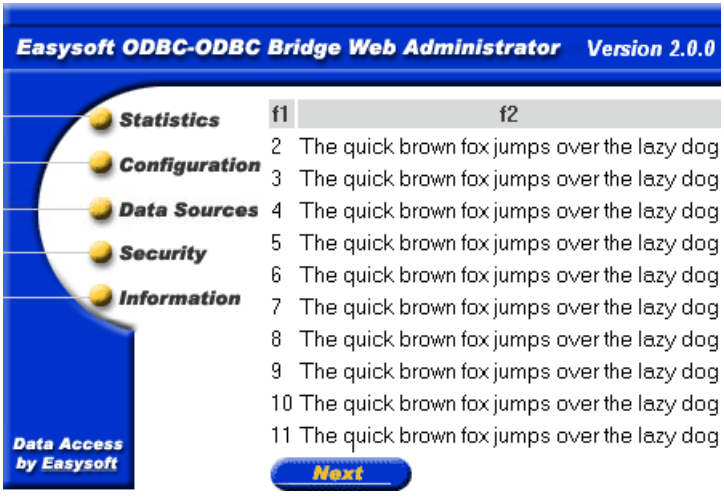


Figure 58: The Web Administrator Data Sources screen row data

Click **Next** or **Prev** to move forwards or backwards in the result set by ten rows at a time.

Easysoft ODBC-ODBC Bridge Resources

The Web Administrator **Information** screen displays a list of links to Easysoft support resources:

Figure 59: The Information screen of the Web Administrator

- **OOB FAQ -** a list of Frequently Asked Questions.
- **OOB news group -** a forum for posting questions and answering questions with other users.
- **Latest OOB Changes/Fixes -** a document listing any recent product updates.

- **OOB User Guide (pdf) -** the User Guide in Adobe Portable Document Format.
- **OOB User Guide (online) -** the User Guide in HTML format.
- **Easysoft Contact Details**.

VIEWABLE LICENSES

To view all your installed Easysoft licenses, click **Information** on the main web page followed by the **Installed Licenses** link:

Figure 60: The Web Administrator Licenses screen

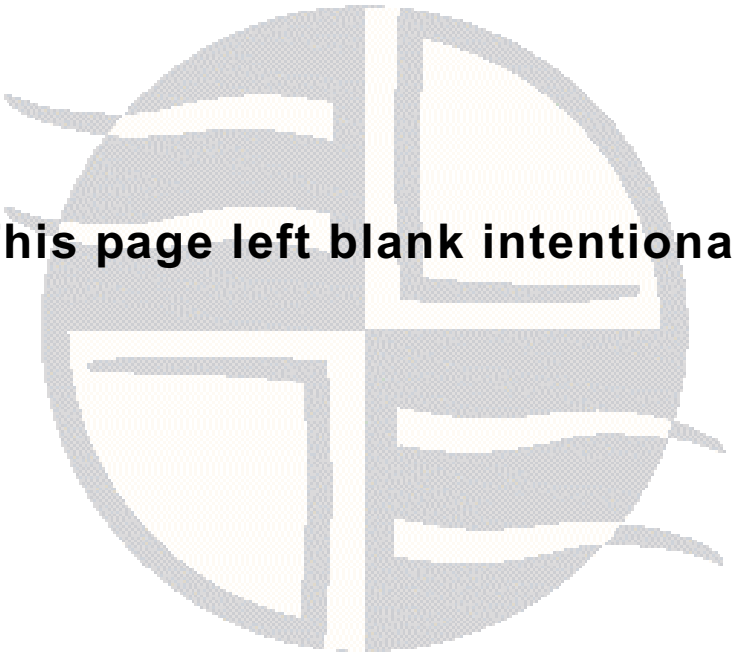
The table shows four columns:

- **Product -** The name of the product this license applies to.
- **Version -** The version of the product.
- **Created -** The date the license was installed.
- **Expires -** The date the license expires or the string "`Never`", meaning the license does not expire.
- **Quantity -** A product specific limit which is either the maximum allowable number of concurrent connections or the string "`unlimited`", meaning that there is no limit on the number of concurrent connections.

EXPORTING LICENSES

On Windows the **Information** page also allows you to export the registry entries in `RegEdit 4` format to a file called `LogDir\easysoft_licenses.reg` where `LogDir` is an OOB Server configurable parameter defined on the **Configuration** screen.

Click the **Export** button to export your licenses. If your registry becomes damaged you can restore the license entries by double clicking on the `easysoft_licenses.reg` file.



This page left blank intentionally

APPENDIX A TECHNICAL REFERENCE

Additional information for the Easysoft ODBC-ODBC Bridge

This section contains extra information relating to the deployment of the Easysoft ODBC-ODBC Bridge (OOB).

It documents where the Easysoft ODBC-ODBC Bridge API differs from other ODBC APIs, diagnostic functionality and tracing issues.

Appendix Guide

ODBC versions supported

The Easysoft ODBC-ODBC Bridge supports most of ODBC 2.0 and ODBC 2.5, and all of ODBC 3.0 and 3.5 with the exception of `SQL_IS_POINTER` (see **"Unsupported ODBC 3.5 functionality" on page 290**).

All the ODBC 2.0 and 2.5 API functions required to run Perl DBD:ODBC, PHP and mxODBC are present, but the full API of these deprecated versions is not supported, as new applications will be written using the ODBC 3.5 API.

Unsupported ODBC 3.5 functionality

SQL_IS_POINTER

The use of driver-specific pointer types (`SQL_IS_POINTER`) is not supported in calls to:

- `SQLSetConnectAttr()`
- `SQLSetStmtAttr()`
- `SQLSetDescField()`

Modifications to the API

SQLBrowseConnect()

The semantics of `SQLBrowseConnect()` are slightly difficult in the context of the OOB. Normally, `SQLBrowseConnect()` provides an iterative method of discovering and enumerating the attributes and attribute values required to connect to a data source. Each call to `SQLBrowseConnect()` returns successive levels of attributes and attribute values. When all levels have been enumerated a connection to the data source is completed and a complete connection string is returned by `SQLBrowseConnect()`.

This process works fine when `SQLBrowseConnect()` is called for a driver on the local machine, but when you introduce a bridge there are in effect two levels of browsing. The first level will browse the local data sources (defined in your `odbc.ini` file or in the registry), but these point to another real data source on a remote machine.

`SQLBrowseConnect()` only supports the browsing of Easysoft ODBC-ODBC Bridge data sources on the local machine. If the user tries to connect to one of these then they will be prompted for Easysoft ODBC-ODBC Bridge attributes such as `SERVERPORT`, `LOGINUSER`, `LOGINAUTH`, `TARGETUSER` and `TARGETAUTH`.

Once sufficient attributes have been defined to allow a bridge connection to the server, the server side of the Easysoft ODBC-ODBC Bridge will return a list of DSNs retrieved by calling `SQLDataSources()`. The browse stops here and so the remote data source must already be set up with sufficient information to allow a connection.

This implementation avoids complications with possible clashes of attributes between the Easysoft ODBC-ODBC Bridge and the remote ODBC driver which would make it impossible to return a final connection string which allows a later connection without browsing.

Understanding ODBC diagnostic messages

The Easysoft ODBC-ODBC Bridge works in conjunction with a number of other types of software, such as ODBC applications, driver managers, drivers and DBMSs.

If a diagnostic (or *error*) message is displayed when a user connects via the Easysoft ODBC-ODBC Bridge, the error does not necessarily lie in the Easysoft ODBC-ODBC Bridge component of your configuration and the message should usually indicate where the problem lies.

For example, examine the following diagnostic messages:

```
[Easysoft ODBC (Client)]Invalid authorization specification
```

This error was produced when the `LogonUser/LogonAuth` attributes were invalid and the connection attempt has been refused. Only the Easysoft ODBC-ODBC Bridge was involved in this process.

```
[Easysoft ODBC (Server)][Microsoft][ODBC Driver Manager]
Data source name not found and no default driver specified
```

This error was produced by the Microsoft ODBC driver manager on the OOB Server machine when the `TargetDSN` attribute specified a DSN which does not exist on the server. You can see that the last item in square brackets was the `"ODBC Driver Manager"` and hence it is that component which generated the error text.

The text is also prefixed with `"[Easysoft ODBC (Server)]"`, which means that the error occured with the driver manager at the server end.

```
[Easysoft ODBC (Server)][Microsoft][ODBC SQL Server
Driver][SQL Server]
Login failed for user 'demo'.
```

This error was produced when the `TargetUser/TargetAuth` specified at the OOB Client was passed through the DBMS which refused the connection.

The last item in square brackets was `"SQL Server"` and so you know that SQLServer turned down the connection attempt.

Therefore, if you encounter any error messages when using the Easysoft ODBC-ODBC Bridge, please note the last item specified in square brackets, because that is the component that generated the error text and the error may not lie in the Easysoft ODBC-ODBC Bridge.

If you are integrating the Easysoft ODBC-ODBC Bridge with your own ODBC application, it is worthwhile implementing diagnostics in your application because then your error messages will indicate in which component the problem lies.

The next section explains how to implement ODBC diagnostic messages.

Implementing ODBC diagnostics

This section explains how to implement ODBC diagnostics in your own ODBC application, so that the application can display diagnostic messages if an error occurs at any point in the connection process.

STATUS RETURNS

All ODBC APIs return a status value which may be used to check whether the function succeeded or not.

In C you can test the return value from an ODBC function using the macro `SQL_SUCCEEDED`:

```
SQLRETURN fsts;

/* Assume environment has already been allocated
*/

SQLHENV  envh;

SQLHDBC  dbch;

fsts = SQLAllocHandle(SQL_HANDLE_DBC, envh,
&dbch);

if (!SQL_SUCCEEDED(fsts))

{

/* an error occurred allocating the database
handle */

}

else

{
```

```
/* Database handle allocated OK */
}
```

The macro SQL_SUCCEEDED is defined as:

```
#define SQL_SUCCEEDED(rc) (((rc)&(~1))==0)
```

Virtually all ODBC functions can return two values which indicate success:

- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO

Both of these returns cause the `SQL_SUCCEEDED` macro to result in 1. If a function returns `SQL_SUCCESS_WITH_INFO` it means that the call succeeded, but an informational message was produced.

For example, with some drivers you might set the cursor type, prepare a statement and then execute it. When `SQLExecute` is called the statement is acted upon, but the driver might change the cursor type to something else.

In this case, `SQLExecute` would return `SQL_SUCCESS_WITH_INFO` and the driver would add a diagnostic indicating the cursor type had been changed.

You should note that a few ODBC functions return a status which fails the `SQL_SUCCEEDED` macro, but do not necessarily indicate an error. For example, `SQLFetch` can return `SQL_NO_DATA` indicating there is no further rows in the result set), this is not necessarily an error.

OBTAINING DIAGNOSTICS

When an ODBC function returns an error or `SQL_SUCCESS_WITH_INFO` then the driver will associate a diagnostic with the handle used in the ODBC call.

You can obtain the diagnostic to find out what failed by calling `SQLGetDiagRec` with the handle you used in the ODBC call that failed.

The driver may associate multiple diagnostic records with a handle. You can call `SQLGetDiagField` and request the `SQL_DIAG_NUMBER` attribute to find out how many diagnostics exist.

Alternatively, as diagnostic records start at 1, you can repeatedly call `SQLGetDiagRec` asking for record 1, then 2 (and so on) until `SQLGetDiagRec` returns `SQL_NO_DATA`.

As, an example, the following C function takes a function name string, handle type and handle and retrieves all the diagnostics associated with that handle.

```
void extract_error

    (char *fn,

     SQLHANDLE handle,

     SQLSMALLINT type)

{

    SQLINTEGER   i = 0;

    SQLINTEGER   native;

    SQLCHAR      state[ 7 ];

    SQLCHAR      text[256];

    SQLSMALLINT  len;
```

```
    SQLRETURN    ret;

    fprintf(stderr,"\n"

                "The driver reported the following
diagnostics whilst running "

            "%s\n\n",

            fn);

    do

    {

        ret = SQLGetDiagRec(type, handle, ++i, state,
&native, text, sizeof(text), &len );

       if (SQL_SUCCEEDED(ret))

            printf("%s:%ld:%ld:%s\n", state, i, native,
text);

    }

    while( ret == SQL_SUCCESS );

}
```

Using this example, which attempts to allocate a database handle, you could use `extract_error` as follows:

```
SQLRETURN fsts;

/* Assume environment has already been allocated */

SQLHENV   envh;

SQLHDBC   dbch;

fsts = SQLAllocHandle(SQL_HANDLE_DBC, envh, &dbch);

if (!SQL_SUCCEEDED(fsts))
```

```
{
  extract_error("SQLAllocHandle for dbc", envh,
SQL_HANDLE_ENV);
  exit(1);
}
else
{
/* Database handle allocated OK */
}
```

ODBC 2.0 applications will use `SQLError` instead of `SQLGetDiagRec`.

DIAGNOSTIC FIELDS

When you call `SQLGetDiagRec` you can retrieve three diagnostic fields:

- State
- Native error code
- Message text

The state is a five character SQLSTATE code. The first two characters indicate the class and the next three indicate the subclass. SQLSTATE codes provide detailed information about the cause of a warning or error.

REF

For the definitive SQL CLI document consult the **Open Group CAE Specification C451, ISBN 1-85912-081-4** (**http://www.opengroup.org/pubs/catalog/c451.htm**).

The **Microsoft ODBC 3.0 Programmer's Reference, ISBN 1-57231-516-4** explains ODBC usage in some detail.

The native error code is a code specific to the data source. This number is often extremely useful to the driver developers in locating an internal error or state.

If you are reporting a bug in the Easysoft ODBC-ODBC Bridge ODBC driver for which you obtained an error you should always quote the ODBC function called, the error text and this native number.

The message text is the text of the diagnostic.

This string takes one of two forms:

- For errors and warnings that do not occur in a data source the format is:

  ```
  [vendor-identifier][ODBC-component-identifier]component-supplied-text
  ```

- otherwise it is:

  ```
  [vendor-identifier][ODBC-component-identifier][data-source-identifer] data-source-supplied-text
  ```

See **"Understanding ODBC diagnostic messages" on page 292** to review some example messages.

Tracing

There are three ways to trace the ODBC calls an application makes through the driver manager and the OOB Client ODBC driver:

1. tracing in the driver manager may be turned on.
2. tracing in the OOB Client ODBC driver may be turned on.
3. an application can turn tracing on via the ODBC API `SQLSetConnectAttr` (...,`SQL_ATTR_TRACE`,...). The trace filename may also be specified with the `SQLSetConnectAttr` attribute `SQL_ATTR_TRACEFILE`.

   Starting tracing in the driver manager is platform-specific:

   [1a] Windows:

   Start the ODBC driver manager administration interface via **Start Menu > Control Panel > ODBC Data Sources**.

   Click on **Tracing**, ensure the specified filename is valid and click **Start Tracing Now**.

   [1b] Unix:

   If you are using the unixODBC driver manager then tracing is enabled in the `odbcinst.ini` file (usually `/etc/odbcinst.ini`).

   To enable tracing you must add two attributes to the [ODBC] section (if you do not have an `[ODBC]` section, create one):

   ```
   Trace = Yes

   TraceFile = /path/filename
   ```

e.g.

```
[ODBC]
Trace = Yes
TraceFile = /tmp/sql.log
```

Make sure that the user who is running the application to be traced has write permission to `TraceFile` (and to the directory containing it), or you will not get any tracing at all.

Driver manager trace files show all the ODBC calls applications make, their arguments and return values, but OOB Client ODBC driver tracing is specific to the Easysoft ODBC-ODBC Bridge and is of most use when making a support call.

You can enable OOB Client ODBC tracing by method [3] or by a platform-specific method:

[2a] Windows

Update the **Registry** by running `regedit` and edit the key:

`HKEY_LOCAL_MACHINE\SOFTWARE\EASYSOFT ODBC-ODBC BRIDGE\CONFIGURATION\SYSTEM\SETTINGS`

Edit the `Logging` string value and set it to `0xffffff`, which turns on all tracing (see **Easysoft ODBC-ODBC Bridge FAQ** for tracing bitmask values).

A trace file named `esoobclient.log` is written to the `LogDir` directory (which is defined in the same registry key).

[2b] Unix:

OOB Client ODBC driver tracing is enabled by setting the `Logging` attribute value to a non-zero value. Set the `Logging` value in the `{Settings}` section of the `odbc.ini` file in the **current working directory**. For example, to enable OOB Client tracing, your `odbc.ini` file in the current working directory should look something like this:

```
[DSN1]
Attribute = Value
.
.
.
{Settings}
Logging = 0xnnnnnn
```

The DSNs are optional, as they may be defined in another `odbc.ini` file (e.g. `/etc/odbc.ini`).

The `Logging` value is a bitmask and the values are listed in the **Easysoft ODBC-ODBC Bridge FAQ**.

A trace file named `esoobclient_`*`PID`*`.log` is created in `/tmp`, where *`PID`* is the process ID.

APPENDIX B GLOSSARY

Terms and definitions

API

Application Programmer Interface. An API is a published set of function calls and constants allowing different programmers to utilize a ready-written library of subroutines.

Application

An Application Program ("`Application`" or "`App`") is a program that *applies* the computer to solving some real-world problem. In ODBC terms, it is the program connecting to the data source.

Authorization code

You must have an authorization code for the Easysoft product you wish to license in order to obtain a purchased license. When you purchase a product your authorization code is emailed to you. You do not need an authorization code to obtain a trial license.

Bitmask

A value which, when written out in binary, has a meaning assigned to each digit, which can be 0 or 1. This is a very efficient way of storing a number of *flags* in a small amount of memory.

When viewed in decimal it is a single number resulting from adding up the values of the individual bits. The bits are worth 1, 2, 4, 8, 16, 32 and so on.

Client/Server

The name given to the architecture whereby one process (the *server*) keeps track of global data, and another task (the *client*) is responsible for formatting and presenting the data. The client requests queries or actions be performed on the data by the server. Often these processes run on different *hosts* across a local-area network.

Column

The vertical dimension of a table. Columns are named and have a *domain* (or *type*). The term *column* might refer to only the *definition* of a column (i.e. its name and type), or to all the data in it.

Connection String

ODBC *driver managers* accept a connection string when a client connects. Ideally it contains all necessary attribute values to make the connection to a data source, but provision is made for the driver to negotiate with the application or the user for any missing information.

Data Source

In ODBC terms, a data source is a database or other data repository coupled with an ODBC Driver, which has been given a Data Source Name (see **"DSN" on page 305**) to identify it to the ODBC Driver Manager.

DLL

Dynamic Link Library. Windows' mechanism for shared object code. See also **"Shared Object" on page 307**.

Download

The transfer of data from a remote machine (on the internet, for example) to your local machine. Mechanisms for achieving this include FTP and the World Wide Web.

Driver

See **“ODBC driver” on page 306**.

Driver Manager

Software whose main function is to load ODBC drivers. ODBC applications connect to the Driver Manager and request a *DSN*. The Driver Manager loads the driver specified in the DSN’s configuration file. In Windows, the ODBC Data Source Administrator is used to set up the Driver Manager.

DSN

Data Source Name. This is a name associated with an ODBC data source. Driver Managers, such as unixODBC or the Microsoft Windows Driver Manager, use the Data Source Name to cross-reference configuration information and load the required driver.

Field

A placeholder for a single datum in a record, for example you can have a Surname field in a Contact Details record. Called a *cell* in Microsoft Access.

Flags

Single-bit values, representing ‘Yes’ or ‘No’. When more than one flag is present, they are normally stored in a *bitmask*.

Host

A computer visible on the network.

HTTP

HyperText Transfer Protocol. The means of transferring web pages.

HTTPAdmin

An NT user name, valid on the machine the server is running on. This is the only user allowed to amend settings, or display certain items, through theWeb Administrator.

Middleware

Software that is placed between the *client* and the *server* to improve or expand functionality.

ODBC (Open DataBase Connectivity)

A standard *API* for connecting application programs to relational database systems through a suitable *driver*. ODBC is available on a wide number of platforms and the Easysoft ODBC-ODBC Bridge allows the database and the application to reside on different machines across the network.

ODBC driver

Software that accesses a proprietary data source, providing a standardized view of the data to ODBC.

Operating System

A collection of software programs, APIs and working practices that control and integrate the execution of system functions on behalf of application programs.

Platform

The term *platform* normally covers the hardware and operating system as a unit. For example; a PC running Microsoft Windows, a PC running BSD Unix, and a Sun running Solaris are three different platforms.

Server

A computer, or *host*, on the network, designed for power and robustness rather than user-friendliness and convenience. Servers typically run round-the-clock and carry central corporate data.

– OR –

A process performing the centralized component of some task, for example extracting information from a corporate database. See **"Client/Server" on page 304**.

Shared Object

A piece of object code (i.e. a program fragment) for loading and executing by other programs.

SQL (Structured Query Language)

A standard language for interacting with relational database systems, based on Relational Theory.

System Data Source

In the context of ODBC under Microsoft Windows, a data source which can be accessed by any user on a given system. See also **"User Data Source" on page 308**.

Table

A data set in a relational database, composed of rows and columns.

For example, the following table has two columns (`vendor` and `name`) and two rows (one for the Easysoft ODBC-ODBC Bridge and the other for MySoft's ODBC client software) :

software	
vendor	name
Easysoft	Easysoft ODBC-ODBC Bridge
MySoft	My ODBC Client Application

The term *table* can also apply to just the definition of the table, without its data.

User Data Source

An ODBC Data Source with access limited to a specific user on a given system. See also **"System Data Source" on page 307**.

Web Administrator

The OOB Server under Windows runs a simple web server which can be used to display and amend certain server statistics and parameters.

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

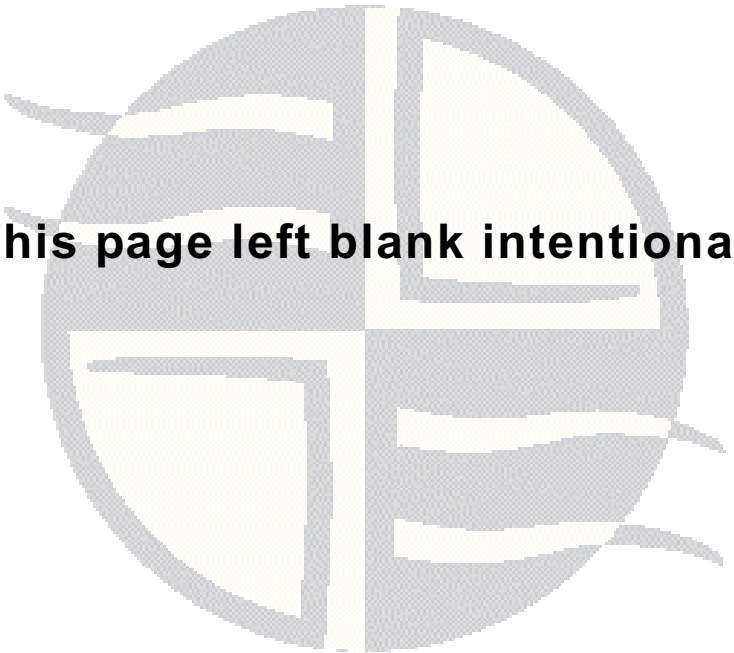
S

T

U

W

X



This page left blank intentionally