Easysoft® Data Access

XML-ODBC Server

Installation Guide and User Manual

Document Version 1.1.0

Publisher: Easysoft Limited

Thorp Arch Grange
Thorp Arch
Wetherby
LS23 7BA
United Kingdom

CONTENTS

LIST OF FIGURES

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 XML-ODBC Server.

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`). These are located in the `Documentation` subdirectory underneath the installation directory on both Windows and Unix. 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 onscreen).

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 website 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, NT or XP, 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 UNIX System Laboratories, 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’.

Easysoft and Easysoft Data Access are trademarks of Easysoft Limited.

1 INTRODUCTION

Introducing the Easysoft XML-ODBC Server

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 XML-ODBC Server is a single solution running on any supported Windows or Unix variant, which will allow you to use XML to access all your ODBC data sources by using existing ODBC drivers.

Databases can be stored either locally or on any other computer platform across a TCP/IP network for which there is a Windows or Unix ODBC driver.

Chapter Guide

Why XML?

XML is an acronym for Extensible Markup Language, designed to improve the functionality of the Web by providing more flexible and adaptable information identification.

You do not have to be a programmer to use or learn XML, as it is a set of rules for designing text formats that let you structure your data, rather than a programming language.

XML is intended to make it easy to define document types, author them and share them across the Web. XML is not just for Web pages, but can be used to store any kind of structured information and pass it between different computer platforms.

It is called extensible because it does not have a fixed format like HTML (a single, predefined markup language), but is a metalanguage (a language for describing other languages) which lets you design your own customized markup languages for limitless different types of documents.

XML uses tags (words bracketed by '<' and '>') and attributes (of the form `name="value"`) to delimit and describe pieces of data, and leaves the interpretation of the data to the application that reads it.

However, although XML can be read with a text editor, the rules for XML files are strict, so that a forgotten tag or an attribute without quotes makes an XML file unusable.

A large and growing community of people who are experienced in the technology support XML and there is a wide range of tools available to help you. Since XML is license-free, you can incorporate it into your own solutions without paying anybody and without being tied to a single vendor.

Why ODBC?

ODBC is an API definition 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 database-specific code and is a distinct program inserted between a database and an application:

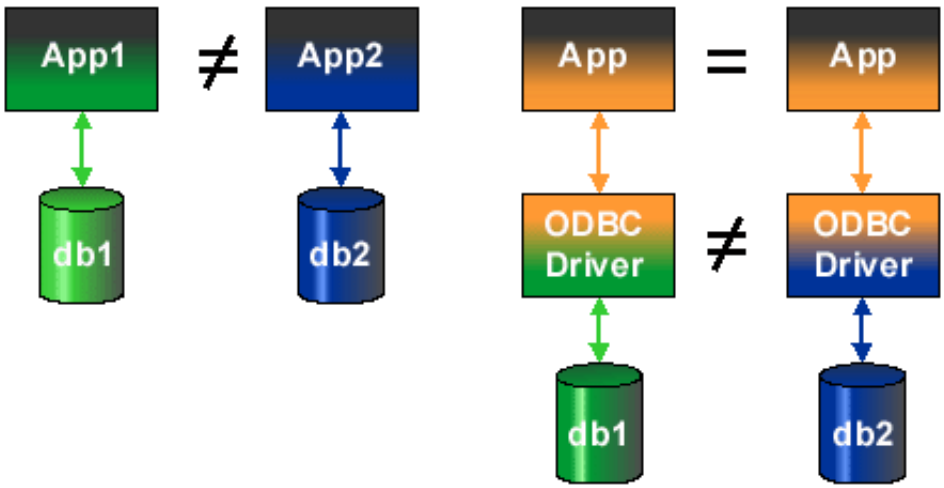

Figure 1: ODBC driver architecture

Before ODBC, even if App1 and App2 were functionally equivalent, two separate programs were required, one for each database.

ODBC permits the parts of a program that are specific to a database to be separated from the part that fulfils the functional requirement.

This means that a completed application can be attached to any database for which there is a corresponding driver available.

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 and initializes the required driver at runtime:

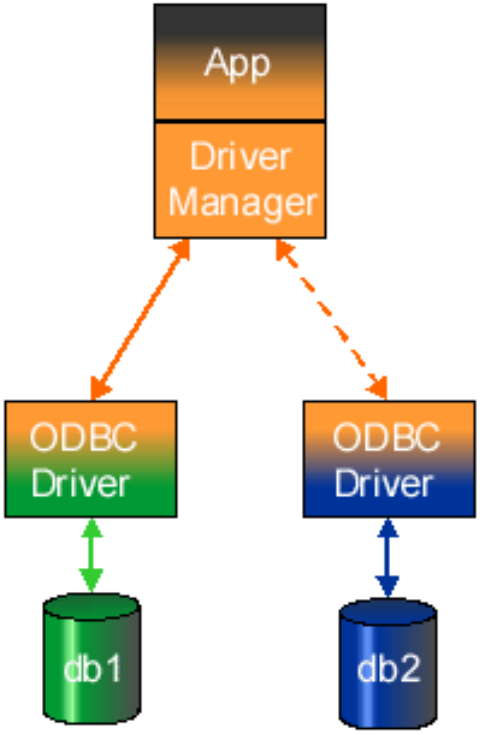

Figure 2: The Driver Manager as a dynamic linker

The Easysoft XML-ODBC Server

OVERVIEW

The Easysoft XML-ODBC Server is a server which allows client applications to access any ODBC compliant database using simple requests in XML over TCP/IP.

The server runs as a service which, once installed on a machine which has ODBC drivers, provides access to those ODBC drivers via XML requests.

By using a specified port, a client application can send a SQL request in XML to the Easysoft XML-ODBC Server, which is then executed via a standard ODBC data source and the results returned to the client, again in XML.

You may issue XML requests containing any valid SQL, allowing you to select, insert, update and delete from any of the tables to which you have access.

In addition, the Easysoft XML-ODBC Server will also accept requests containing multiple SQL statements to be executed as a single transaction, allowing you to perform multiple updates, inserts or deletes as part of a single operation.

The Easysoft XML-ODBC Server can be used with any ODBC driver available for Windows, Linux and major Unix platforms.

NB	Windows 9x is not supported by the Easysoft XML-ODBC Server.

The Easysoft XML-ODBC Server service listens on a port for simple XML requests in ASCII/UTF-8, using an extremely straight forward protocol which allows client applications to be developed for any platform and language with access to TCP/IP.

See **“Developing Client Applications for the Easysoft XML-ODBC Server” on page 79** for sample C, Perl, PHP, Java and Visual Basic client application code included with the Easysoft XML-ODBC Server.

FEATURES

- Simple installation and configuration.
- The target database can be hosted on any ODBC compliant database system, including Microsoft SQL Server, Microsoft Access, Oracle, InterBase and IBM DB2.
- Multithreaded, high performance service available on Windows, Linux and major Unix platforms.
- Supports ODBC versions 2.0 to 3.52.
- Fully thread-safe, supporting any native ODBC drivers.
- Compatible with Zebedee and other TCP/IP encryption solutions.
- Includes unixODBC, the open source ODBC Driver Manager for non-Windows platforms.
- No special client code is required, only access to TCP/IP and a small amount of socket code in any language.
- Includes sample C, Perl, PHP, Java and Visual Basic client applications, with example SAX based parsing of XML output in PHP and Perl, an example XSLT application for Perl.
- Simple XML requests allow you to update, delete, insert or query local and remote ODBC data sources.
- Supports complete SQL syntax of the target ODBC driver.
- Support for transactions and stored procedures.

- Full remote server configuration using a control port and simple XML requests.
- Both SQL Request and Control ports protected by access control lists based on IP address.
- Multiple XML output formats for result sets:

  Short format with column list followed by row list

  Long format with column name as an attribute on each column in the row list

  TableTags format for import into Microsoft Office
- XML is output in UTF-8 format, but with full UTF-16 support for Unicode data sources, with data accurately output as UTF-8 in the result set.
- Column data in the result set may be requested as CDATA or PCDATA.

ODBC Driver and Version support

The Easysoft XML-ODBC Server is an ODBC application which requires ODBC 3.0 in order to provide Unicode support (either in the ODBC driver manager or directly in the ODBC driver) for the returning of result sets in UTF-8.

Easysoft software expects all ODBC drivers to be used behind an ODBC driver manager and therefore direct Unicode support in the ODBC driver is not required.

This means you should be able to use ODBC 2.0 drivers just as well as ODBC 3.0 (or above) drivers.

However, if you are using an ODBC driver you will not be able to retrieve non-ASCII characters from your driver (this is not a limitation of Easysoft XML-ODBC Server, but of the ODBC driver used).

INSTALLATION

2

Installing the Easysoft XML-ODBC Server

This section covers the tasks necessary to install and remove the Easysoft XML-ODBC Server.

A Windows installation can be carried out by anyone with administrator privileges.

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

Chapter Guide

Obtaining the software

There are three ways to obtain the Easysoft XML-ODBC Server:

- 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 **Free Trials** page on the Easysoft XML-ODBC Server section of the website, 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**. It contains free patches, upgrades, documentation and beta releases of Easysoft products, as well as definitive releases. The FTP site is useful if you have a slow connection or if you want to write a script to retrieve the file. Change to the `pub/xml-odbc` directory and then choose the platform release that you require.
- If you have an extremely slow connection you can order Easysoft software on CD by email, telephone or post (see **Contact Details**).

What to install

The name of the install file varies from platform to platform, but you can expect something of the form:

- `xml-odbc_`*`x_y_z`*`.msi` (Windows)

– OR –

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

where "*x*" is the major version number, "*y*" is the minor version numbers and "*z*" is the build index, which is incremented when minor changes are made. "*platform*" will vary depending on the operating system distribution you require.

Within your licensed major version number, you should go for the highest release available for your platform. To install software of a different major number requires a new license.

There are documentation and example files which you are recommended to keep for future reference.

The Unix installation also provides the unixODBC driver manager (Windows already has one).

Whichever medium you choose, you should now download the file and begin the installation process. The exact installation process depends on your operating system.

Refer to the section relevant to your platform to continue:

- **"Installation under Windows" on page 23**
- **"Uninstalling under Windows" on page 31**
- **"Installation under Unix" on page 33**
- **"Uninstalling under Unix" on page 45**

Installation under Windows

For Windows, the Easysoft XML-ODBC Server installation comes as an executable with a name of the form `xml-odbc_x_y_z.msi`.

If there is a choice of files then you should download the file with the highest version number.

1. From the web, click to download the installation file.

   – OR –

   In your FTP client, switch to `binary` mode and `get` the installation file.

   – OR –

   If you have a CD, go to the folder containing the distribution file.

Caution! Please shut down other Windows programs before installing. In particular, if Microsoft Outlook is running there can be a pause of up to several minutes when InstallShield is started.

2. Double-click on the `xml-odbc_x_y_z.msi` file.

The **Welcome** dialog box is displayed:

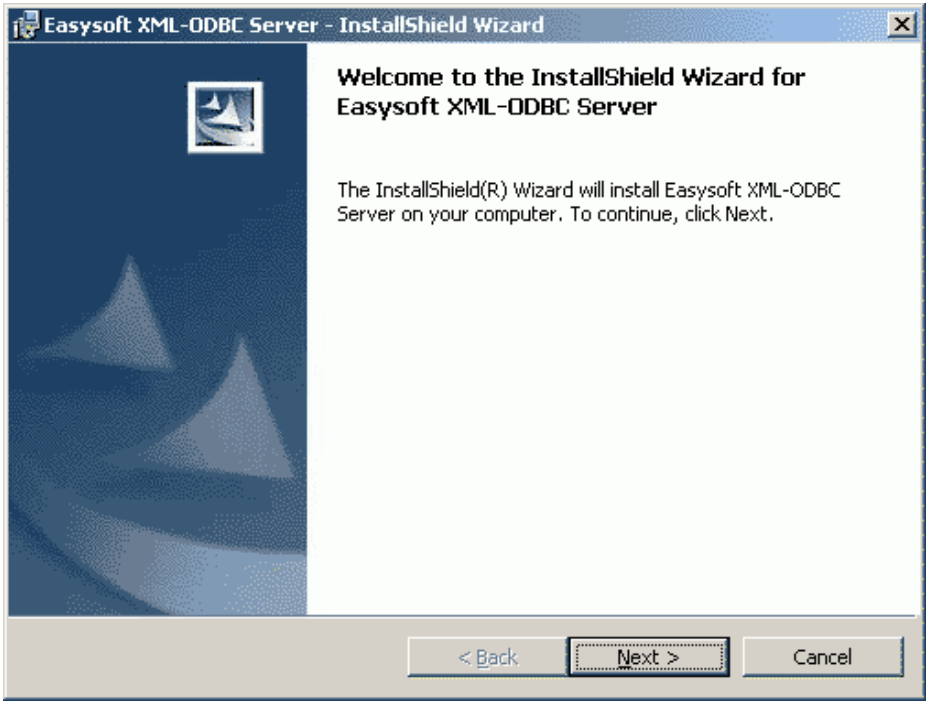

Figure 3: The Welcome dialog box

3. Click **Next**.

The **License Agreement** dialog box is displayed:

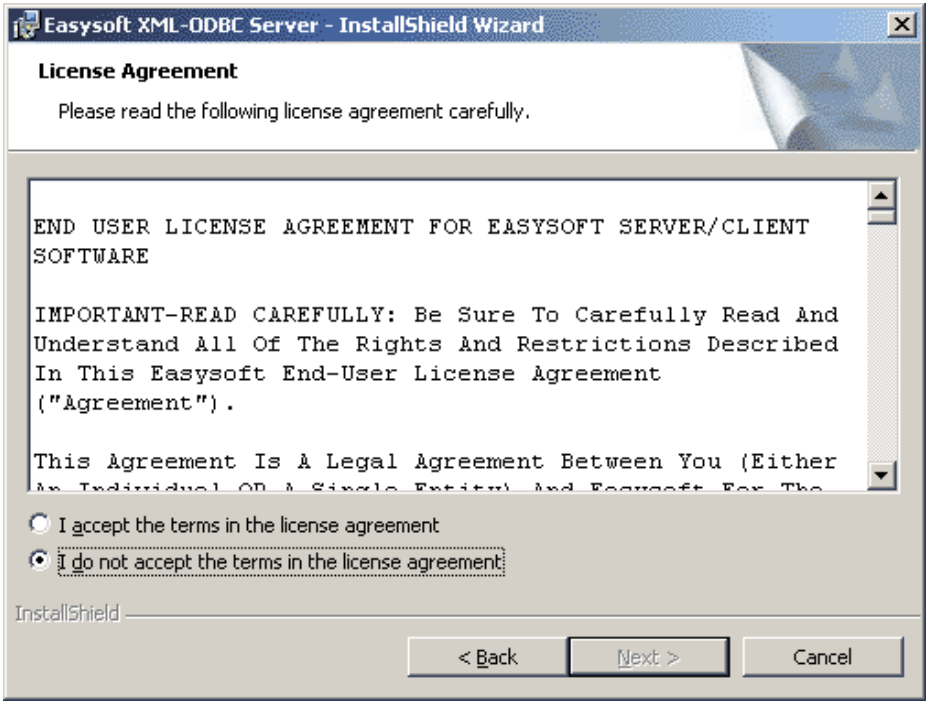

Figure 4: The License Agreement dialog box

4. Click **I accept the terms in the license agreement** and then click **Next** to continue with the installation if you accept the License Agreement.

   – OR –

   If you do not agree with the License Agreement click **Cancel** to exit the installation and then refer to the **Easysoft License Agreement** for more information (see **Contact Details** if you have any questions about the licensing procedure).

The **Customer Information** dialog box is displayed:

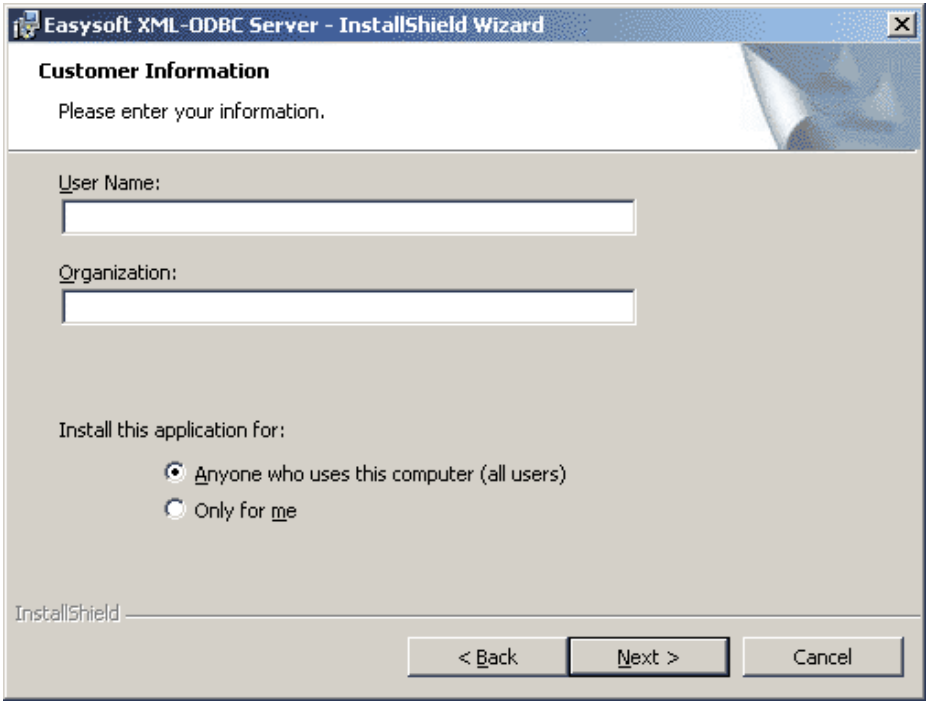

Figure 5: The Customer Information dialog box

5. Enter your name and the name of your company. Then click **Next** to continue.

NB	The name and company that you enter here will become the defaults in the License Manager later on.

The **Setup Type** dialog box is displayed:

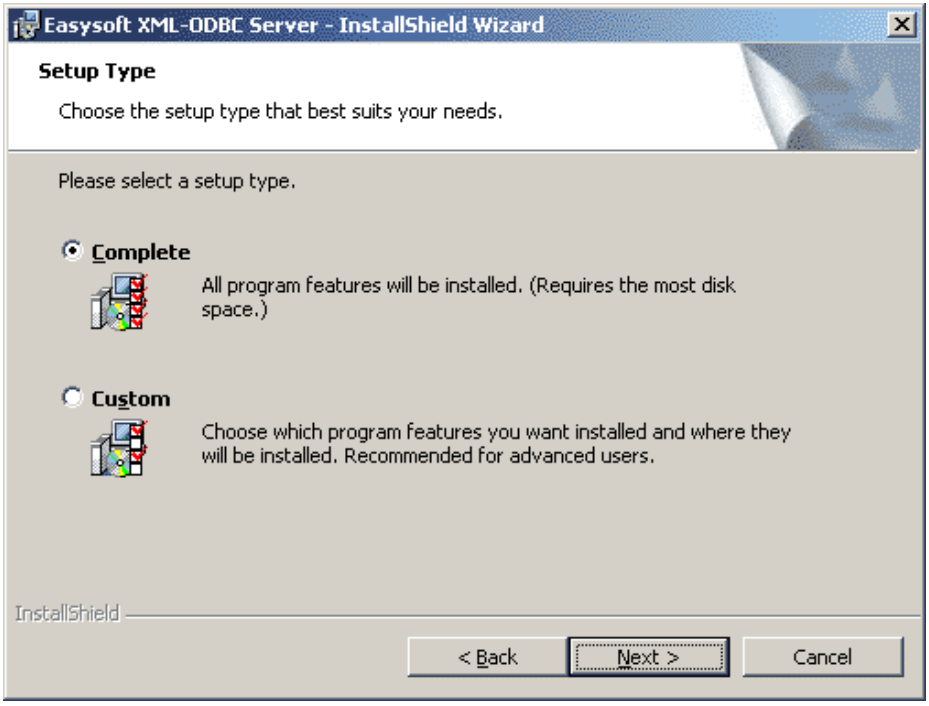

Figure 6: The Setup Type dialog box

6. Select **Complete** to install all components (the server, License Manager, documentation and client examples). This is recommended for developers and administrators.

   – OR –

   Select **Custom** to select the components to be installed.

7. Click **Next** to continue. If you selected **Custom**, a checklist of items to install is displayed. Select the items to install and click **Next**:

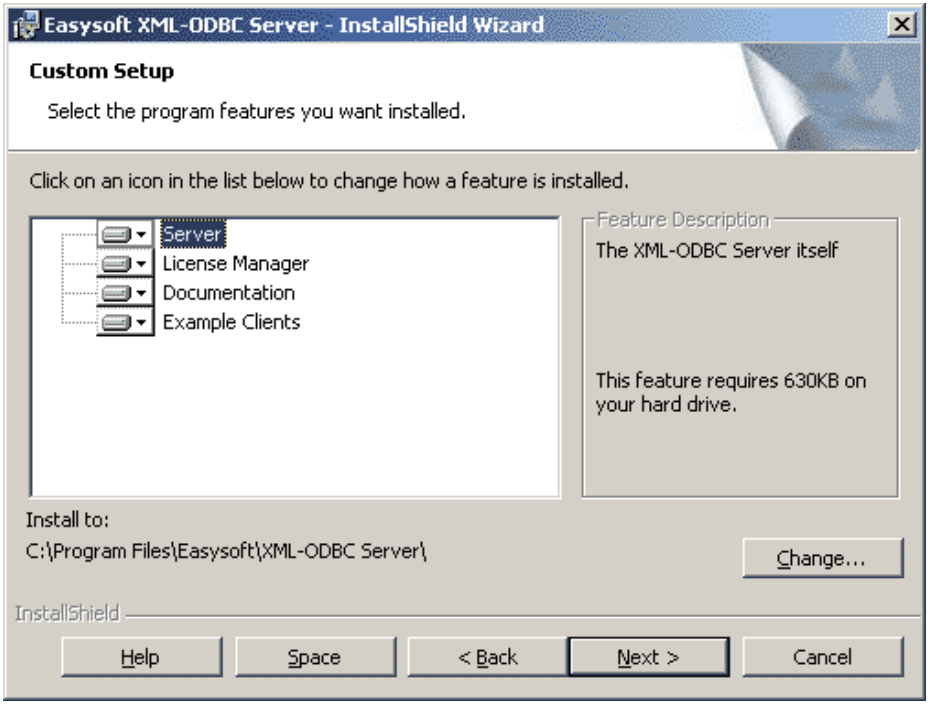

Figure 7: The Custom Setup dialog box

The **Ready to Install the Program** dialog box is displayed:

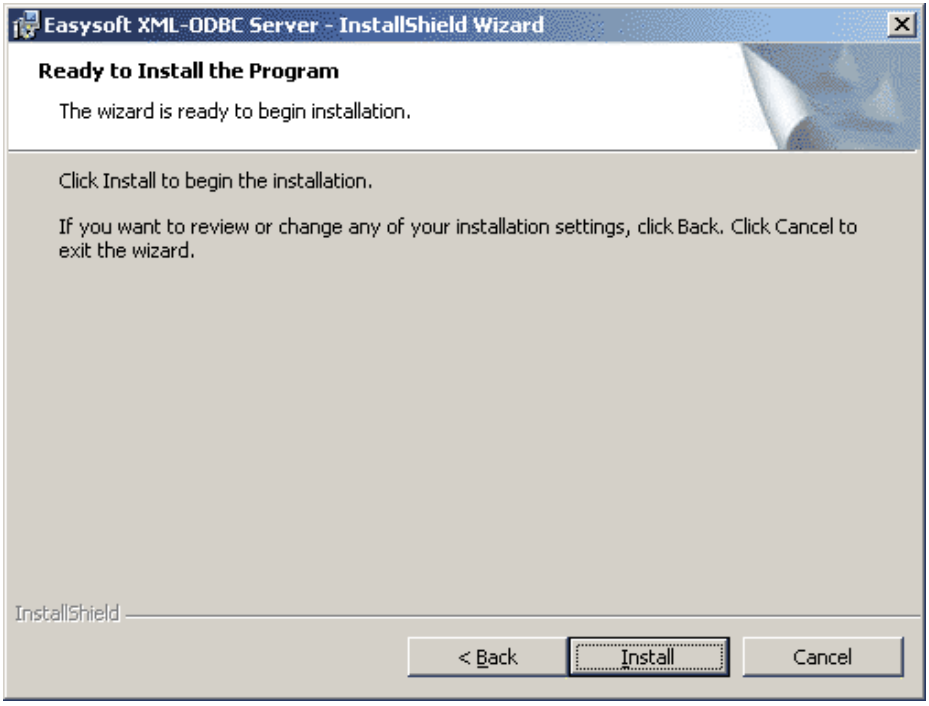

Figure 8: The Ready to Install the Program dialog box

8. Click **Install**.

   There is now a short wait while the relevant Easysoft XML-ODBC Server components are copied and configured.

The **InstallShield Wizard Completed** dialog box is then displayed:

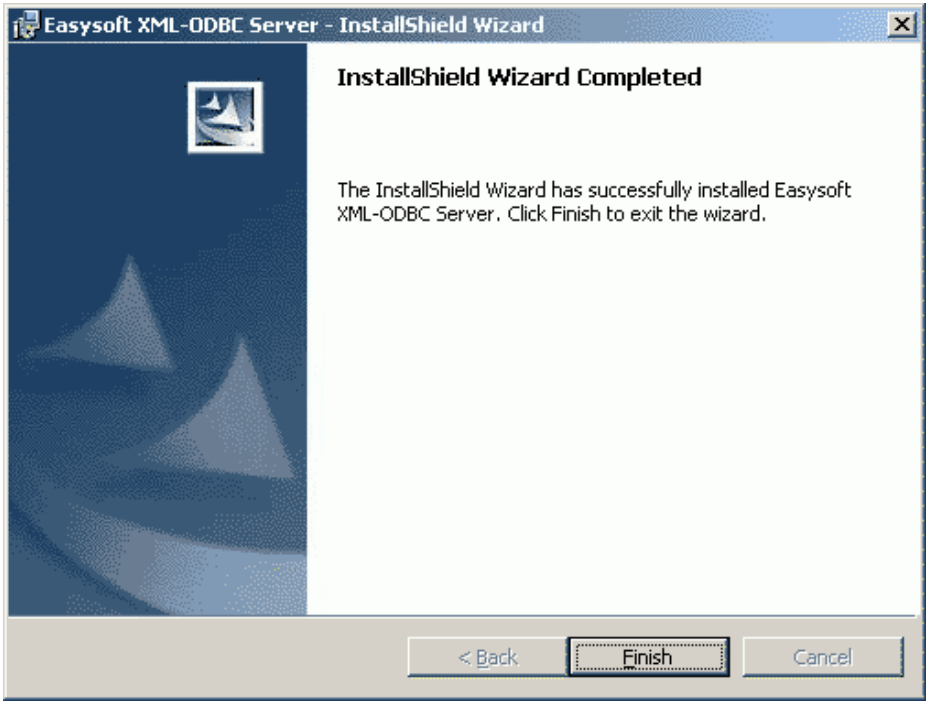

Figure 9: The InstallShield Wizard Completed dialog box

9. Click **Finish**.

   The installation is now complete (although you may be asked to restart your machine if certain files have been overwritten).

   A new **Programs>Easysoft>Easysoft XML-ODBC Server** program group is added with links to additional documentation, product newsgroups and the Easysoft web site.

Uninstalling under Windows

This section explains how to remove the Easysoft XML-ODBC Server from your Windows system.

1. Select **Start > Settings > Control Panel** and then double-click the **Add/Remove Programs** icon.

   You are then presented with a list of applications that can be automatically removed.

2. Select **Easysoft ODBC-XML Server** and click the **Add/Remove** button.

3. Click **Yes** to confirm that you wish to remove the Easysoft XML-ODBC Server and all its components.

   The system begins to remove all the components.

   You will be prompted for further advice on the removal of shared DLLs if they seem not to be required.

NB	Under Windows, install/uninstall procedures incorporate a mechanism in the registry to determine whether or not shared files are still required by other programs. Sometimes this database can become out-of-date if, for instance, the user deleted an application directly without using **Add/Remove Programs** or if the registry was 'repaired' after a system crash.

4. It is generally safer not to remove shared DLLs, but if you feel confident with the registry (i.e. your system has had relatively few programs installed or removed, and you are confident that DLLs are not shared by any other products on your machine) then click **Yes** or **Yes to All** to continue.

   – OR –

If you have any doubts (e.g. uninstall procedures have failed in the past) then click **No** or **No to All**.

The uninstall process is now complete and all the Easysoft XML-ODBC Server components removed from your system.

NB	If files have been created in any of the installation directories then these directories will not be removed. In this case, the uninstall program will issue a warning and you can click **Details** to find out what directories remain.

Note that any license codes that you have obtained for either the Easysoft XML-ODBC Server or any other Easysoft products are held in the registry and these are not removed when the Easysoft XML-ODBC Server is uninstalled.

If you want to, you can manually remove these entries from

`HKEY_LOCAL_MACHINE\`

`SOFTWARE\`

`Easysoft\`

`Info\`

`Products\`

The key for Easysoft XML-ODBC Server is `2006000000000001`. You can delete this key if you no longer want to use the product, but do not delete this key if you plan to reinstall the Easysoft XML-ODBC Server, or you will have to re-license.

Installation under Unix

PREPARATION

Because this section covers a range of platforms, the install process may vary from system to system. It is worth having a pen and paper to hand in case you need to note any details down for future reference.

The first stage is to get the installation archive onto your system. For Unix systems, the filename is of the following form:

`xml-odbc-`*`x.y.z.platform`*`.tar`

This filename may be suffixed with `.gz` for a "`gzipped`" archive, `.bz2` for a "`bzip2ed`" archive, or `.Z` for a "`compressed`" archive. "*`platform`*" may be any supported platform.

> **NB** If you download a Unix file using Windows, the browser may corrupt the filename. For example, if you download a `.gz` file and Windows corrupts 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.

1. Log on to your Unix machine as the `root` user.
2. If you have not already done so, consult **"What to install" on page 22** and **"Obtaining the software" on page 21**, and copy or download the file into a working directory such as your home directory or `/tmp`.
3. Change to the directory in which the archive file resides.

   The next step is to extract the installation files from the archive.

EXTRACTING THE INSTALLATION FILES

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

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

   – OR –

   If the archive has been `bzip2ed` (i.e. the filename ends in `.bz2`), then use:

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

   – OR –

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

   `uncompress xml-odbc-`*`x.y.z.platform`*`.tar.Z`

   – OR –

   If the archive has not been compacted at all (i.e. the filename ends in `.tar`), then the archive is ready for extraction.

5. Extract the files from the archive:

   `tar -xvf xml-odbc-`*`x.y.z.platform`*`.tar`

   The `tar` program creates a directory with the same name as the `tar` file (without the final `.tar`) containing further archives, checksum files, a script called `install` and a text file called `INSTALL`.

   It also contains a versioned directory to ensure that any common Easysoft components that may have been installed with other Easysoft products are not overwritten unless the new files are more recent.

If you do not wish to keep the original downloaded archive you can now delete it safely.

6. Change into the newly created directory.
7. Check through the `INSTALL` file before continuing.

   The `INSTALL` file gives full installation instructions for the Unix-literate and, if you are confident in the use and administration of your system, you may follow these instructions instead of working through the remainder of this section.

Caution!

If you have previously installed the Easysoft XML-ODBC Server on the system, be sure to remove it before proceeding, otherwise there may be conflicts after installation which could prove difficult to resolve (see **"Uninstalling under Unix" on page 45**). You will need approximately 5MB of free space for the Easysoft XML-ODBC Server.

8. There are two license files provided in the archive: one that applies if you are installing just the Easysoft XML-ODBC Server sample client applications and another that applies if you are installing the either the Easysoft XML-ODBC Server or both the Easysoft XML-ODBC Server sample client applications and the Easysoft XML-ODBC Server.

Caution!

You must read and accept the terms of the license applicable to your installation in order to use the software. The license texts can be found in the files `Client-License.txt` and `Server-Client-License.txt`. These are initially located in your newly created install directory and then copied into the "`Documentation`" subdirectory during the install process. Determine which applies to you, and be sure to understand its terms before continuing.

BEGINNING THE INSTALLATION

The illustrations in the following section are taken from a session installing the Easysoft XML-ODBC Server as the `root` user on a Linux glibc system.

Although the precise output will differ for other platforms, the installation process is essentially the same.

9. Once you have taken care of any matters arising from the `INSTALL` text file, begin the installation by typing the following from within the unpacked directory:

   ```
   ./install
   ```

```
Installation procedure for Easysoft XML-ODBC

Copyright (c) 1993-2002 Easysoft Limited

Please read the file INSTALL.txt before attempting to install.

                                IMPORTANT NOTICE

This installation may be used to install Easysoft XML-ODBC.

The installation of this SOFTWARE and its associated LICENSE is subject to
                        the Easysoft End-User License Agreement.
Your use of the SOFTWARE and LICENSE shows your acceptance of the terms
                                and conditions.

Before proceeding you should read the Client-License.txt if you are only
installing the Clients and the Server-Client-License.txt if you are
installing the Server.

Do you accept the license? (q=quit to read license, yes, no):
```

Figure 10: The Unix License Manager introduction screen

10. If you have read and agree to the License Agreement, type `yes` to continue.

> **NB** You must type `yes`, not `y`, to continue.

11. The script pauses to allow you to read its output so far. Up to this point it has checked the following:

    - that you have the minimum set of Unix programs it requires

- the platform you are running
- any platform-specific components, such as the version of the C runtime library

12. Press *<Enter>* to continue.

    The script checks the archive package, with three possible outcomes:

    - The files are checked and they pass
    - The files are checked and they fail
    - The files are not checked because some component required for the check is not found

13. If the check failed because of missing components, enter "`y`" to continue regardless or "`n`" to quit and investigate the missing components.

    – OR –

    If the check was carried out and the files failed then the files have been damaged:

    - return to **"Obtaining the software" on page 21** and download the install archive again

    – OR –

    - call the Easysoft support team (see **Contact Details** for more information)

    If the files passed the check then you are asked for a directory in which to place the `easysoft` installation directory tree.

    The default is `/usr/local`, which would be the normal location to install software for system-wide use.

If you wish to install the software in another directory then specify a directory name here.

If you have any other Easysoft products already installed, then choose the same directory that you chose for the other Easysoft product(s).

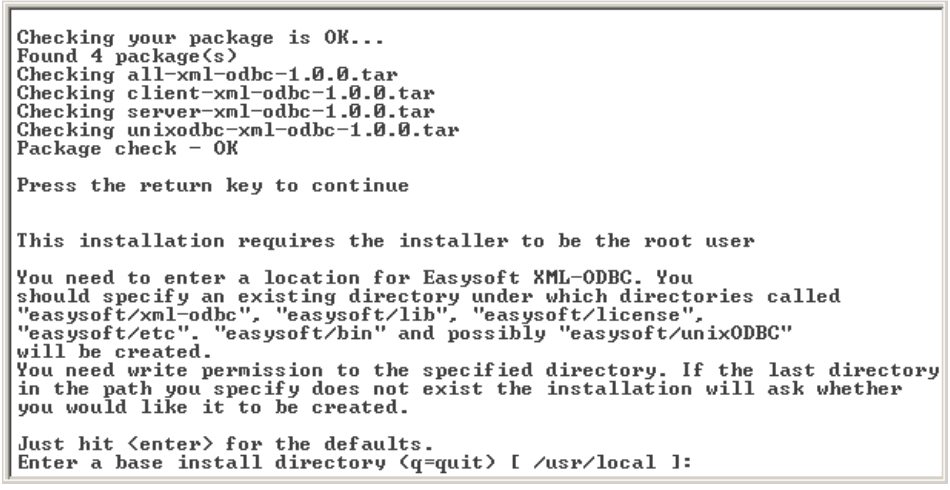

Figure 11: The Unix License Manager data entry screen

The installation script will create a directory called "`easysoft`" under the directory you specify here, into which all the Easysoft XML-ODBC Server files will be placed.

NB	The installation script can accept a path to a directory that does not exist, provided a parent directory exists. For example, if the directory `/usr/local` exists on your machine, the script can accept the non-existent directory `/usr/local/xml` and will create it, but will not create `/usr/local/xml/xmlodbc`. The script always creates an "`easysoft`" directory beneath the specified path.

14. If you have `root` permission and want a typical system-wide installation, press *<Enter>* to place the `easysoft` directory in `/usr/local/`.

    – OR –

    If you do not have `root` privileges, or wish to select a custom installation directory, type the desired directory and press *<Enter>*.

    The script now creates the "`easysoft`" directory under the directory you specified. If your chosen directory does not exist, you will be asked whether or not the script should create it.

15. Enter y or n.

16. If the chosen directory already contains an "`easysoft`" directory, the script warns that you may be installing over a previous Easysoft XML-ODBC Server installation (this conflict can also arise if you have another Easysoft product installed). Normally you should continue with the installation to install into the existing "`easysoft`" directory. Enter y or n.

    If you chose a directory other than `/usr/local` the script creates a symbolic link `/usr/local/easysoft`, pointing to the "`easysoft`" directory in the directory you specified, in order to ensure that licensing will work.

17. Press *<Enter>* to continue.

INSTALLING UNIXODBC

The script now offers to install the copy of the unixODBC driver manager contained in the Easysoft XML-ODBC Server distribution.

The unixODBC driver manager enables the Easysoft XML-ODBC Server ODBC application to load whichever driver is required to access the data source at runtime.

If you do not have the unixODBC driver manager on your system then the Easysoft XML-ODBC Server will not function correctly.

The Easysoft XML-ODBC Server install routine will examine your system in an attempt to locate an already installed version of unixODBC, but you MUST install the unixODBC driver manager UNLESS you already have version 2.2.2 (or greater) installed, as the Easysoft XML-ODBC Server takes advantage of essential routines that this version introduced for handling Unicode characters (see **"The unixODBC driver manager and Unicode" on page 159**).

Having the unixODBC driver manager installed multiple times on your machine does not do any harm, but if applications are linked against different versions which have been built with different `sysconfdir` settings, then each driver manager may link to a different data source.

NB	unixODBC is an open source project sponsored by Easysoft and other industry members. It is rapidly becoming the standard driver manager across the Unix data access community. Comprehensive documentation can be found at **http://www.unixodbc.org**.

18. Press *<Enter>* to continue and then choose whether to install unixODBC.

NB	If you would prefer, you can download the entire unixODBC source distribution from **ftp://ftp.easysoft.com/pub/beta/unixODBC** and install it independently.

19. If you do not wish to install unixODBC, enter n.

    – OR –

If you wish to install unixODBC, enter y. The script pauses at this point. Press *<Enter>* to continue and the script extracts the unixODBC files.

NB	Depending on your platform this installation of the unixODBC driver manager may not contain the GUI components of unixODBC.

20. The script pauses. Press *<Enter>* to continue.

INSTALLING THE DOCUMENTATION

The Easysoft XML-ODBC Server documentation files are extracted.

INSTALLING THE SAMPLE CLIENTS

The script asks you whether you want to install the sample client files for the Easysoft XML-ODBC Server.

21. Enter n to skip installing the sample client files.

    – OR –

    Enter y to install the sample client files. The script pauses at this point. Press *<Enter>* to continue and the script extracts the sample client files.

22. Press *<Enter>* to continue.

INSTALLING THE SERVER SOFTWARE

The script asks whether you wish to install the Easysoft XML-ODBC Server.

This allows you to make data sources on the local machine available to remote client applications using the Easysoft XML-ODBC Server.

23. If you wish to skip the Easysoft XML-ODBC Server install, enter `"n"` and go to **"Completing the installation" on page 44**, which presents a final checklist.

    – OR –

    Enter `"y"`, and continue with the next step.

24. The script pauses at this point. Press *<Enter>* to continue.

    The files necessary for the Easysoft XML-ODBC Server are extracted.

25. Press *<Enter>* to continue.

    The script now begins to configure the Easysoft XML-ODBC Server under the `{x}inetd` “super-server” and asks whether you would like to install the `services` and `inetd/xinetd` entries.

    NB Some new Linux distributions (in particular) use `xinetd`, rather than `inetd`. The installation recognizes and correctly handles the configuration of either. For more information about `xinetd` visit the web site at **http://www.xinetd.org**.

26. If you want to skip these changes, enter n and the installation terminates. Go to **"Completing the installation" on page 44**.

    – OR –

    Press *<Enter>* to continue.

    The script now amends the `/etc/services` and `{x}inetd` configuration files.

27. If either of these files is missing or protected against writing, then you must supply their locations as prompted.

NB	You must use full path and file names in order to specify the relevant configuration file, and not just the file names themselves.

    The script looks for an existing entry in the `services` file for the default service name of the Easysoft XML-ODBC Server. If one already exists then you are prompted for a decision.

28. If there is no service name conflict, then go to **"Completing the installation" on page 44**. If there is a service name conflict, read on.

    If you are reinstalling the Easysoft XML-ODBC Server (to replace an older installation), enter `r` to replace the existing entry.

    – OR –

    If you want to have two different Easysoft XML-ODBC Server services running beside one another and you wish to define a new service name for the Easysoft XML-ODBC Server then type `d`, press *<Enter>* and then enter your chosen new name for the service.

29. The script looks for an existing entry in the `services` file for the Easysoft XML-ODBC Server default port.

    If an entry already exists then you are asked to choose another port. Enter another port number and the script will check whether the new value conflicts with any other services.

30. If the script cannot determine what shell to use for the `{x}inetd` configuration file you must enter one at the prompt. It will now be read again so that any changes take effect, and the Easysoft XML-ODBC Server startup script is installed.

COMPLETING THE INSTALLATION

31. Press *<Enter>* if necessary to return to the shell prompt.

    You can remove the installation files.

    Unless you specified the installation directory (at **step 14 on page 39**) to be within the temporary directory, then you can safely remove the temporary directory and all its contents.

NB	If any warnings were generated during the installation these were appended to the file `./warnings` and a message will be output containing a list of all the warnings. You should review this file and satisfy yourself that none of the warnings have adversely affected your installation. You should mail this file to Easysoft support if you are unsure. In particular, one warning you may see is caused by the installation attempting to untar a file which is in use. If you see this warning it will be necessary to make sure all applications using the file in question are stopped and the installation is rerun.

Uninstalling under Unix

This section explains how to remove the Easysoft XML-ODBC Server from your Unix system.

Although there is no automated method of removing the Easysoft XML-ODBC Server, the operation is quite simple.

NB	To uninstall the Easysoft XML-ODBC Server requires the same privileges as the user who performed the installation (normally `root`).

There are four steps to uninstalling the Easysoft XML-ODBC Server under Unix:

- Manually removing the Easysoft XML-ODBC Server entries from the Unix `services` and `{x}inetd` configuration files.

NB	You only need to do this if you do not intend to upgrade the Easysoft XML-ODBC Server software, or if you intend to upgrade using a different configuration. If you intend to upgrade the Easysoft XML-ODBC Server software and you want the configuration to remain the same, do *not* make these edits.

- Removing the Easysoft XML-ODBC Server documentation (if installed).
- Removing the Easysoft XML-ODBC Server sample clients (if installed).
- Removing the Easysoft XML-ODBC Server server software (if installed).

A step-by-step guide follows:

1. Close down all client programs attached to your service.

2. Log in as `root`.

Removing the service entries

3. Make a backup of the `/etc/services` configuration file.
4. Open `/etc/services` in your editor.
5. Look at the end of the file for two lines of the format:

   ```
   esxmlodbcbserver    8895/tcp
   ```

   and

   ```
   esxmlodbcbserver_c 8896/tcp
   ```

   Note that you may have specified a different name to the default of `esxmlodbcserver` during the Easysoft XML-ODBC Server installation.
6. Delete both service entries and save the file.

Removing the inetd entry

7. Make a backup of the `/etc/inetd.conf` configuration file.
8. Open `/etc/inetd.conf` in your editor.
9. Look at the end of the file for two lines of the format:

   ```
   esxmlodbcserver stream tcp nowait root /bin/sh /bin/sh
   /usr/local/easysoft/xml-odbc/server/server
   ```

   and

   ```
   esxmlodbcserver_c stream tcp nowait root /bin/sh /bin/sh
   /usr/local/easysoft/xml-odbc/server/serverc
   ```

   Note that, as with the `services` file, you may have specified a different name to the default of `esxmlodbcserver` during the Easysoft XML-ODBC Server installation.

There should be one entry in `inetd.conf` for every entry in the `services` file.

10. Delete both `inetd` entries and save the file.
11. Use the `ps` command to find the Process ID (PID) of the `inetd` process and send a Hangup signal to tell it to re-read the configuration files:

    `kill -HUP` *pid*

REMOVING THE XINETD ENTRY

`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 Server software" on page 41**).

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

There are therefore two ways to delete the Easysoft XML-ODBC Server service from `xinetd`:

12. Make a backup of the `xinetd` configuration file, open it in your editor, look for two lines similar to the following:

```
service esxmlodbcserver
service esxmlodbcserver_c
```

(where `esxmlodbcserver` is the name as specified in the `services` file, so there should be one entry in `xinetd.conf` for every entry you saw in the `services` file), delete both entries and save the file.

– OR –

Delete the `esxmlodbcserver` file in the directory to which the `includedir` setting in `xinetd` points.

13. Use the `ps` command to find the Process ID (PID) of the `xinetd` process and send either a SIGUSR1 or SIGUSR2 signal (consult the `xinetd(8)` man page for the correct option) to tell it to re-read the configuration files:

    `kill -USR1` *pid*

    – OR –

    `kill -USR2` *pid*

REMOVING THE DOCUMENTATION

14. Change directory to *<InstallDir>*`/easysoft/xml-odbc` and delete the directory tree "`Documentation`" (where *<InstallDir>* is the installation path specified at install time).

15. You can also delete the `INVENTORY.txt` and `README.txt` files from *<InstallDir>*`/easysoft/xml-odbc`.

REMOVING THE SAMPLE CLIENTS

16. Change directory to *<InstallDir>*`/easysoft/xml-odbc` and delete the directory tree "`clients`" (where *<InstallDir>* is the installation path specified at install time).

REMOVING THE SERVER SOFTWARE

17. Change directory to `<InstallDir>/easysoft/xml-odbc` and delete the directory tree "`server`" (where `<InstallDir>` is the installation path specified at install time).

18. If you have no other Easysoft products on your system you can delete the entire "`easysoft`" directory tree, including the "`xml-odbc`" directory beneath it.

    ```
    rm -r <InstallDir>/easysoft
    ```

    – OR –

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

    ```
    rm -r <InstallDir>/easysoft/xml-odbc
    ```

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.

19. If you left the installation files on your system then you may wish to remove them at this point.

    The uninstall process is complete.

    Any licenses you obtain for the Easysoft XML-ODBC Server and other Easysoft products are stored in the `<InstallDir>/easysoft/license/licenses` file.

After uninstalling the Easysoft XML-ODBC Server, 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 the Easysoft XML-ODBC Server.

RUNNING THE SERVER

3

Running the Easysoft XML-ODBC Server as a service

This section explains the how the Easysoft XML-ODBC Server runs as a service to which external client applications can connect.

Chapter Guide

Introduction

Once it has been successfully installed, the Easysoft XML-ODBC Server acts like an ordinary ODBC-compliant application and runs as a service.

In Windows it takes the form of a network service, controlled by the Windows service control manager and configured to start automatically.

On Unix, the Easysoft XML-ODBC Server is a either a service under {x}`inetd` (the default) or a standalone process, and is linked to the driver manager of your choice.

If you enable the control port for the Easysoft XML-ODBC Server then clients may obtain Easysoft XML-ODBC Server statistics, but only if the server is running as a standalone process (Unix) or under the service control manager (Windows).

Starting the server from the command line

You do not need to install a Easysoft XML-ODBC Server client somewhere in order to issue requests to the server and have it return the results in XML.

The Easysoft XML-ODBC Server program can be run directly from the command line with a request in a file.

e.g.

Suppose you have a file called `request.xml` in the current directory which contains a valid Easysoft XML-ODBC Server request.

On Windows:

```
xmlodbcserver.exe file request.xml out.xml
```

On Unix:

```
./esxmlodbcserver file request.xml out.xml
```

The output is written to `out.xml` (or use "`-`" to direct to standard out instead).

There is also a "`pipe`" method allowing an XML request to be piped in to the Easysoft XML-ODBC Server program and output to standard out.

You can run `xmlodbcserver/esxmlodbcserver` with "`-?`" as an argument to get a full usage summary, although only the "`file`", "`pipe`", "`siteinfo`" and "`version`" methods should be used directly.

UNIXODBC

You are required by Easysoft to use the unixODBC driver manager on Unix platforms in order to run the Easysoft XML-ODBC Server.

unixODBC is an open source project sponsored by Easysoft which is fast becoming the standard across the data access community.

REF	unixODBC is not an Easysoft product, but the unixODBC driver manager is included with the Easysoft XML-ODBC Server distribution. **http://www.unixodbc.org/** is the home page for unixODBC. The distribution found here contains the unixODBC driver manager plus a set of ODBC drivers.

For details of the Easysoft XML-ODBC Server connection, go to the section appropriate to your server platform:

- **"Running under Windows" on page 54**
- **"Running under Unix" on page 60**

Running under Windows

Before a client application can connect to a data source on the server machine, the Easysoft XML-ODBC Server must be running.

Under Windows 2000, NT and XP, the installation program configures the Easysoft XML-ODBC Server to start automatically as a service.

NB	The Easysoft XML-ODBC Server writes errors and startup information to the **Application** section of the **Event Log**. If you suspect that the Easysoft XML-ODBC Server is not functioning correctly you are recommended to check the **Event Log** with the **Event Viewer** before pursuing other lines of enquiry.

CHECKING THE SERVICE

This procedure is only necessary if you are having problems connecting with the Easysoft XML-ODBC Server.

1. Choose the `Services` icon from **Start > Settings > Control Panel**.

2000/	Choose **Start > Settings > Control Panel** and open the `Administrative Tools` icon followed by the `Services` icon.
XP	Choose **Start > Settings > Control Panel > Administrative Tools > Services**.

You are presented with a dialog box containing all your NT system's registered services.

2000/ XP	The appearance of the `Services` dialog box is different in Windows 2000 and Windows XP, but the principles and the functionality are the same.

Find the entry for `Easysoft XML-ODBC Server`:

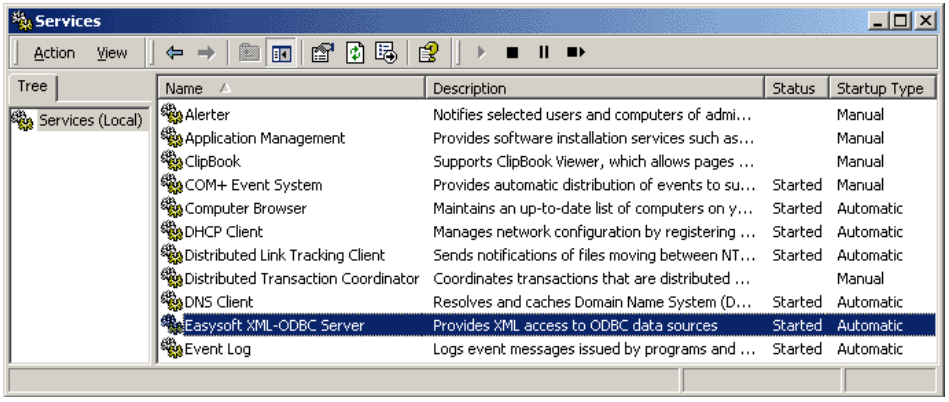

Figure 12: The Easysoft XML-ODBC Server Services entry

2. If the `Startup` field says `Disabled` then click **Startup...**.
3. In the resulting dialog box, select **Automatic** and then click **OK**.
4. If the `Status` field does NOT say `Started` then click **Start** to bring the Server on-line.
5. Click **Close**.
6. Close the **Control Panel**.

SERVICES FILE ENTRIES

The Easysoft XML-ODBC Server does not require an entry in the services file to function correctly, but if you want to see the service name rather than the port number displayed by the `netstat` command, you can edit the `services` file.

Adding this entry also reserves the port for the Easysoft XML-ODBC Server.

On Windows, the `services` file is usually located in

```
<install directory>\system32\drivers\etc
```

Edit the `services` file with any Windows text editor, such as Notepad. The Easysoft XML-ODBC Server service should be defined as "`8895/TCP`" assuming you accepted the default port of 8895 during installation. The entry should look like:

```
esxmlodbcserver              8895/tcp
```

If you have enabled the control interface for the Easysoft XML-ODBC Server (it is enabled by default), then you may also add:

```
esxmlodbcserver_c            8896/tcp
```

TO START OR STOP THE SERVICE

The Easysoft XML-ODBC Server can be configured 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:

1. Choose:

   Start > Settings > Control Panel > Administrative Tools (in Windows 2000)

   – OR –

   Start > Settings > Control Panel (in Windows NT)

2. Open the `Services` icon.

   The Service Manager displays a list of services, along with their status (see **"To start or stop the service" on page 56**) and their startup configuration (see **"Startup configuration" on page 57**).

3. In the **Services** dialog box, ensure that the `Easysoft XML-ODBC Server` entry is selected.

4. 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** when the Service Manager asks you to confirm your action.

STARTUP CONFIGURATION

When installed, the Easysoft XML-ODBC Server is configured to run automatically when Windows starts.

Windows offers two other options for services:

- `Manual` startup mode dictates that the service should start only when requests come in or the user explicitly runs the service. This can save resources if the service in question is rarely used.
- `Disabled` allows the service to be made unavailable from time to time if, for example, you need to restructure your data source or you have a security policy stating that services should not be available outside allotted times.

1. In the **Services** dialog box, ensure that the `Easysoft XML-ODBC Server` entry is selected.
2. Click **Startup...** (*not* **Start**!) to open the Service configuration dialog box.

3. In the **Startup Type** section, choose the relevant radio button:
   - 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**
4. Click **OK** to close the dialog box.

USER ACCOUNTS

Windows distinguishes two types of service with respect to their privileges:

- services running in the local System Account, which have highly privileged access to resources on the local machine whilst idle.
- services set to run as a specific user, which are given the set of privileges assigned to that user.

The Easysoft XML-ODBC Server is designed to be run in the System Account, with administrative privileges, which allow it access to any System data source on the system.

This is the default configuration of the Easysoft XML-ODBC Server and it provides the greatest general flexibility and security.

The other option is to set up the server to always run with the permissions of a specific user.

This is not recommended, but if you do want to do this the procedure is as follows:

1. 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 Easysoft XML-ODBC Server.

Caution!

For the Easysoft XML-ODBC 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 Easysoft XML-ODBC Server.

2. In the **Services** dialog box, ensure that the `Easysoft XML-ODBC Server` entry is selected.
3. Click **Startup...** (*not* **Start**) to open the Service configuration dialog box.
4. In the **Log On As** section, select the **This Account** radio button.
5. In the **This Account** text box, type the user name or click **...** to browse for a user name.
6. In the **Password** box, type the user account password.
7. Type the same password in the **Confirm Password** box.
8. 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 Easysoft XML-ODBC Server.

Running under Unix

When the Easysoft XML-ODBC 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`:

```
esxmlodbcserver stream tcp nowait root /bin/sh
/bin/sh /usr/local/easysoft/xmlodbc/server/server
```

- and in `services`:

```
esxmlodbcserver 8888/tcp # Easysoft XML-ODBC Server
```

These are just examples and may differ if you picked a different port, service name or installation path.

THE MECHANICS OF `inetd`

`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 listens on each port used by those services that have been configured in the `/etc/services` file.

When a new connection is requested from outside, `inetd` starts the relevant program and hands over control.

The following example applies to the default installation, when a client connects to port 8895 and `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 Easysoft XML-ODBC Server startup script `<InstallDir>/easysoft/xml-odbc/server/server`.

The script then sets any necessary environment variables required by the dynamic linker and runs `esxmlodbcserver`.

The script passes an argument of `inetd` to the server executable. This notifies the server that it is running under `inetd`, rather than at the command line.

The Easysoft XML-ODBC Server inherits the sockets from `inetd` and begins communicating with the client application.

`inetd` returns to listening on port 8895 for any new connections.

CHANGING THE 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 Easysoft XML-ODBC Server, then you will need to edit the configuration files manually.

The port number appears only in `/etc/services`, but the service name appears in both `/etc/inetd.conf` and `/etc/services`.

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

On Unix, the Easysoft XML-ODBC Server is installed as a service running under `{x}inetd` by default, which prevents any meaningful statistics information about the program being retrieved (see **"Example Statistics XML" on page 139**).

However, the Easysoft XML-ODBC Server will also run in standalone mode (i.e. without `{x}inetd`), which may be preferable if you are making a lot of small connections (as these will then be accepted much faster) and will also allow you to retrieve statistics.

Running standalone means that `{x}inetd` does not know about the Easysoft XML-ODBC Server service, which must therefore be run either manually or by some other process.

Note that as the Easysoft XML-ODBC Server no longer uses `{x}inetd` to listen on the socket when running as standalone, you cannot run `tcp` wrappers.

To start the Easysoft XML-ODBC Server in standalone mode:

1. Comment out the `esxmlodbcserver` entries in the`{x}inetd` and `/etc/services` files.
2. Force the configuration files to be re-read using:
   - for `inetd`:

   ```
   kill -HUP pid
   ```

   where *pid* is the process ID of `inetd`
   - for `{x}inetd` :

Use the `ps` command to find the Process ID (PID) of the `xinetd` process and send either a SIGUSR1 or SIGUSR2 signal (consult the `xinetd(8)` man page for the correct option):

`kill -USR1` *pid*

– OR –

`kill -USR2` *pid*

3. Start the server with:

   path`/esxmlodbcserver 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 Easysoft XML-ODBC Server reads the `<Port>` and `<ControlPort>` settings in *path*`/config.xml` for its configuration (see **"Example Configuration XML" on page 133**).

NB	Start the Easysoft XML-ODBC Server Server as the `root` user.

   It is possible to add a line to one of your `rc` scripts to have the Easysoft XML-ODBC 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.

CREATING A DATA SOURCE

4

Creating an ODBC data source for the Easysoft XML-ODBC Server

This section explains how to create data sources for the Easysoft XML-ODBC Server.

Chapter Guide

Introduction

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 need to create a data source on the server machine to make the database available to client applications.

For details of the Easysoft XML-ODBC Server implementation, go to the section appropriate to your server platform:

- **"Windows data sources" on page 68**
- **"Unix data sources" on page 74**

ODBC data sources

An ODBC application (the Easysoft XML-ODBC Server in this case) connects to a database by using a description of that data source, the content of which depends on the ODBC driver being used to access the database and consists of a set of attribute/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/system files under Unix.

It therefore follows that before the Easysoft XML-ODBC Server 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).

With the Easysoft XML-ODBC Server you need a data source on the server to describe the target database in whatever terms the ODBC driver requires (e.g. in Windows you use a dialog box provided by the ODBC driver and accessed from the ODBC Administrator).

Connecting to data sources

In general, ODBC applications must be linked with either an ODBC driver (although this is very rare) or a driver manager.

The Easysoft XML-ODBC Server takes a connection string from a client application and uses it in a `SQLDriverConnectW` call to the ODBC driver manager.

The driver manager examines the connection attributes and loads the required driver, which is either named in the connection string `DRIVER` attribute or is referenced from either a registry key (on Windows) or a `.ini` file (on Unix).

The connection string contains a list of connection attributes, normally including one of the following:

- a `DSN=` (Data Source Name) attribute which names the data source. Specifying a DSN explicitly tells the driver manager which driver to load because each DSN will contain a `DRIVER` attribute.
- a `DRIVER=` attribute which names the driver, which allows you to choose any database to which the driver has access.
- a `FILEDSN=` attribute which allows the connection string 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 user name with which to connect to the database, and `easysoft` is the password for the `demo` user.

Windows data sources

The Easysoft XML-ODBC Server can connect to any system data source configured on a Windows machine, given the necessary information. It can also connect to any database if all necessary ODBC attributes the driver needs have been specified by using a connection string.

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

The instructions in this section show you how to create a data source for Microsoft's Northwind database, which is shipped with Microsoft Access. You should follow the same procedure to connect to your own database on your server machine, using the Microsoft Data Source Administrator.

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 Microsoft's `northwind.mdb`.

USING THE MICROSOFT DATA SOURCE ADMINISTRATOR

Select **Start >Settings > Control Panel** and open the ODBC icon.

XP	Select **Start >Settings > Control Panel > Administrative Tools > Data Sources (ODBC)**.

2000	To find the ODBC icon in Windows 2000, open `Administrative Tools` in Control Panel. The ODBC icon is called `Data Sources (ODBC)`.

The ODBC Data Source Administrator opens.

4. Click the **System DSN** tab:

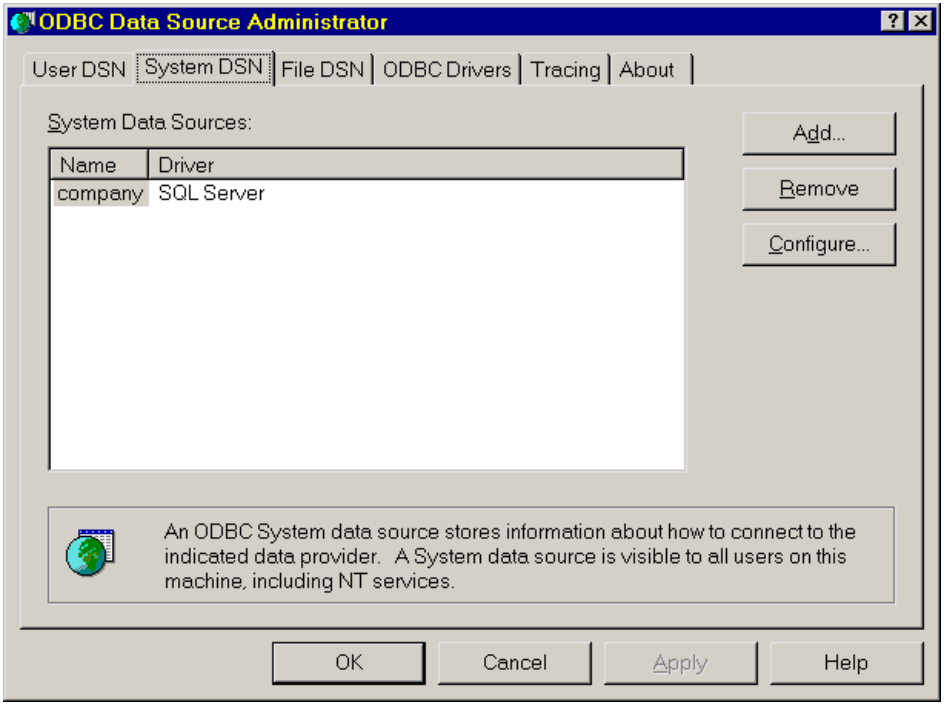

Figure 13: The ODBC Data Source Administrator System DSN tab

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 Easysoft XML-ODBC Server runs as a service, User DSNs are not available to it.

5. Click the **Add...** button to add a new data source. The **Create New Data Source** dialog box displays a list of drivers:

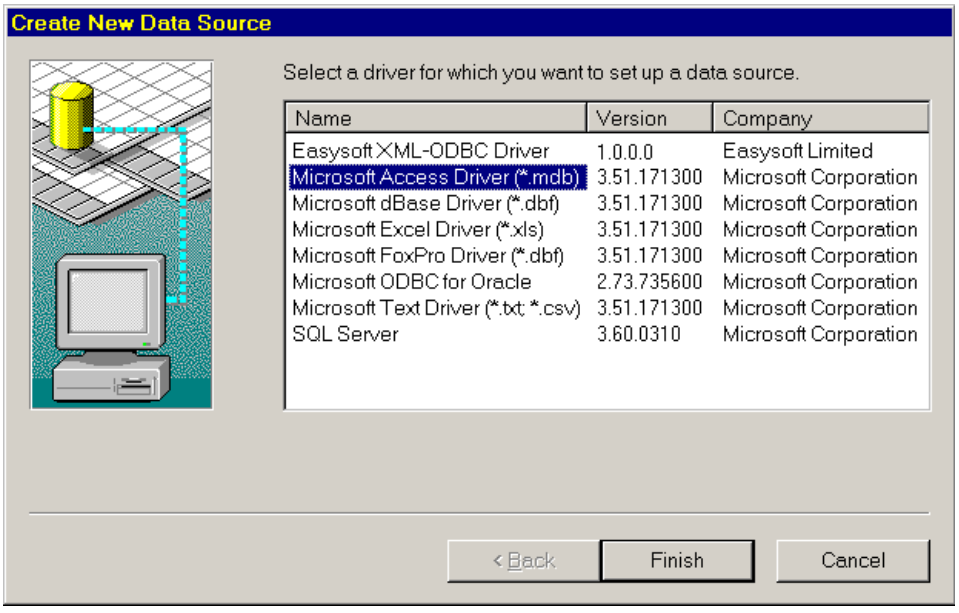

Figure 14: The Create New Data Source dialog box

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

   The ODBC driver for Microsoft Access presents 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 Easysoft XML-ODBC Server to be configured to run in multi-process mode, rather than the default multi-threaded mode.

7. Enter your chosen name for this data source in the **Data Source Name** box (e.g. "`XML-ODBC-Demo`").
8. In the **Description** field, enter something that would help a user faced with a choice of data sources (e.g. "`For demonstrating XML-ODBC`"):

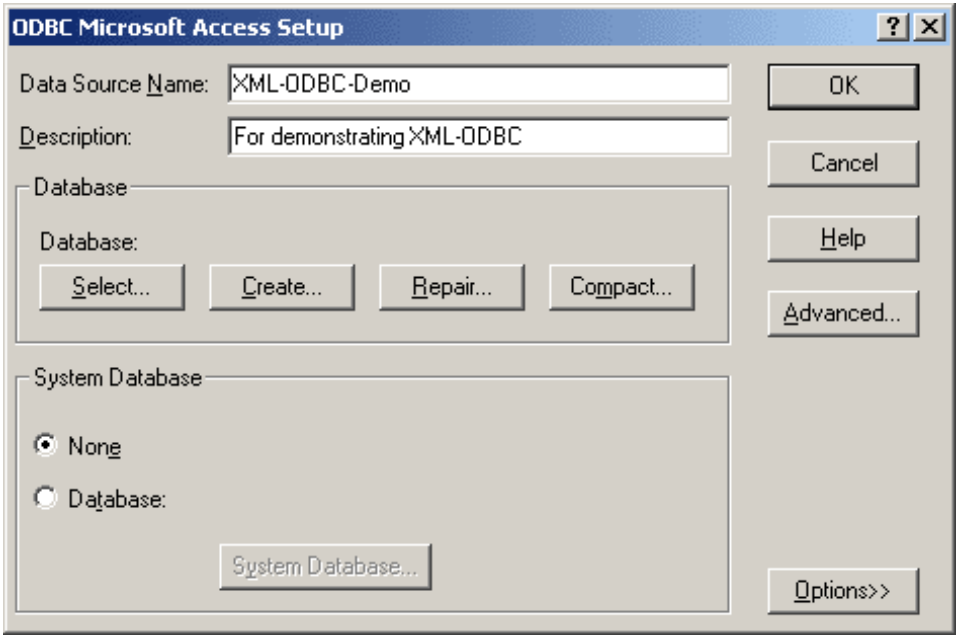

Figure 15: The ODBC Microsoft Access Setup dialog box

9. Click **Select...** to browse for the target database. Select your chosen database and click **OK**.

   Although the example used the `Microsoft Office\Office\Samples\Northwind.mdb` database, this database may not exist or may be in a different location on your system, in which case you may use any database you have to hand.

NB	Note the data source name because you will need to specify it when you create a data source on the client machine.

10. Click **OK**.

    You are returned to the **ODBC Data Source Administrator** window.

    Note:

    - the window now contains a line with your new data source in it
    - the System DSN tab should be selected.

      If it is not, then you must remove the new data source, select the **System DSN** tab, and return to **step 5 on page 70**.

11. Click **OK**.

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

TESTING THE DATA SOURCE

You now have the Easysoft XML-ODBC Server running on your Windows machine and a data source connecting to the database on the server.

To test that this data source is working, so that you can verify that the server side is functioning correctly, you can run 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.

Unix data sources

The Easysoft XML-ODBC Server can connect to any system data source configured on a Unix machine, given the necessary information.

You MUST use the unixODBC driver manager on Unix.

A version of unixODBC is supplied with the Easysoft XML-ODBC Server, which requires version 2.2.2 or greater to be installed in order to function correctly (see **"The unixODBC driver manager and Unicode" on page 159**).

With unixODBC, you can create a data source by either:

- 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`).

EDITING A CONFIGURATION FILE

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

If you accepted the default Easysoft XML-ODBC Server installation, system data sources will be stored in `/etc/odbc.ini`.

However, if you have built unixODBC yourself, then it will be whatever path you specified in the `sysconfdir=`*directory* configure option. If `sysconfdir` has not been specified then the path will default to `/usr/local/etc`.

NB	You usually need to be logged in as `root` to edit a system data source defined in `/etc/odbc.ini`, but user data sources may be created and stored in an `.odbc.ini` file within that user's home directory.

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 shown below:

```
[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 =
```

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.

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

USING THE ODBC DATA SOURCE ADMINISTRATOR

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

1. Run an X session connecting to your Unix machine, ensuring that you log in as `root`.
2. 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.
5. Click the **Add** button to create a new data source.

   The **Adding a New Data Source** dialog box appears, listing the drivers available.
6. Select the driver that you want to use to connect to the database and then click **OK**.

   A configuration dialog box specific to that driver is now displayed.

The dialog box for configuring a PostgreSQL is illustrated below:

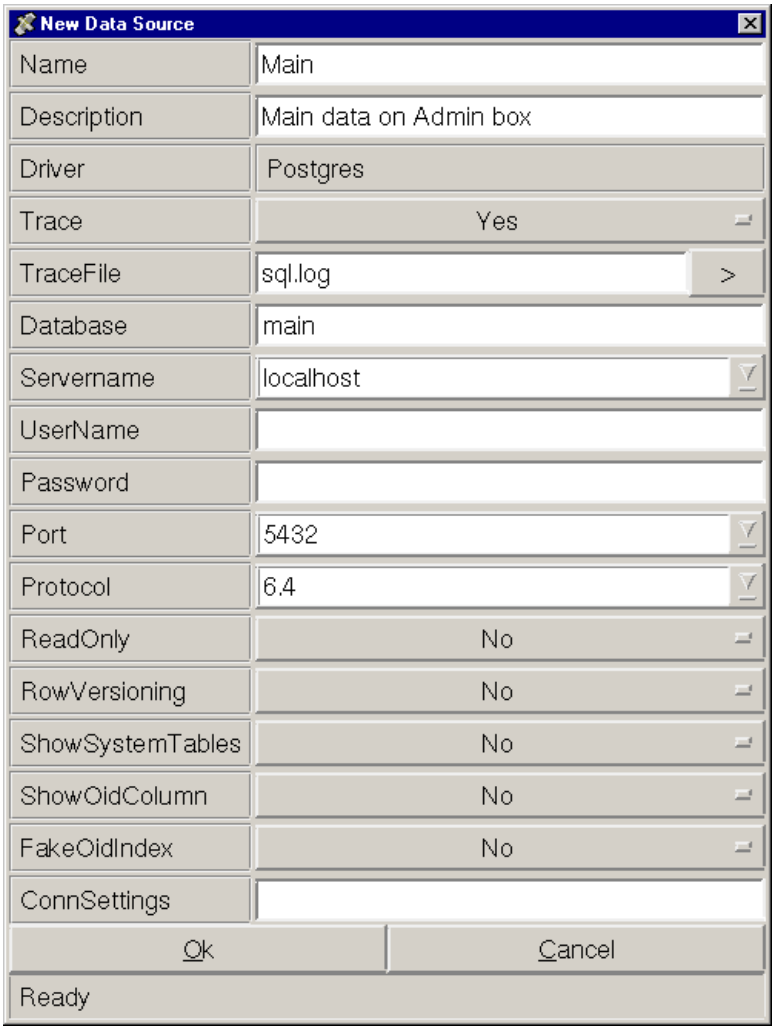

Figure 16: The Configuration dialog box for a PostgreSQL data source

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

7. **OK** this dialog box when you have specified all the data source attributes and then close the ODBC Data Source Administrator.

TESTING THE DATA SOURCE

You should now verify that the data source you have created and the Easysoft XML-ODBC Server setup are functioning correctly by using any ODBC application available on your Unix machine or by using the unixODBC `isql` utility.

To use `isql` to test the data source:

1. Change into the *<InstallDir>*`easysoft/unixODBC/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 `isql "-v"` returns ODBC diagnostic messages, which are useful if you have problems connecting to your data source using `isql`.

   If your server database requires authentication, you should include user name and password arguments in the `isql` command.

   e.g.

   ```
   ./isql - v data_source_name dbuser dbpassword
   ```

3. Once connected, type an SQL statement to query the data. e.g.

   ```
   select * from tablename
   ```

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

   Or simply type `help` to get a list of tables in the database.

4. To leave `isql` and return to the system prompt, press *<Enter>*.

CLIENT APPLICATIONS

5

Developing Client Applications for the Easysoft XML-ODBC Server

This section includes sample application code for connecting to the Easysoft XML-ODBC Server.

Chapter Guide

Introduction

In order to communicate with the Easysoft XML-ODBC Server, a client application needs to perform the following actions:

1. connect a socket to port 8895 (the default) on the remote server.
2. send an XML request to the server.
3. read an XML response from the server (where "`</Result>`" signals the last line of XML).
4. go back to **step 2** or continue on to **step 5**.
5. close the socket.

Note that any number of requests and responses may be transmitted once a socket has been opened. It is only necessary to close the socket once all operations have been completed. See **"Sample client applications" on page 91** for details of how this scheme has been implemented.

A streaming XML text protocol is used between the client and server, allowing client applications to be easily written in any language.

The examples provided here are all fully functional programs, the source code for which is distributed with the Easysoft XML-ODBC Server in a subdirectory of the `Client` directory for each language.

Example SQL XML

Sample XML code is supplied with the Easysoft XML-ODBC Server distribution for the various SQL requests and responses sent to and from the Easysoft XML-ODBC Server.

There are three types of XML SQL request which can be made to the Easysoft XML-ODBC Server, each of which uses the `<Format>` element of the XML request to defines the type of output required:

- **SQL requests**

There are three corresponding XML formats for responses from the Easysoft XML-ODBC Server to SQL requests which are successful:

- **Long SQL response**
- **Short SQL response**
- **TableTags SQL response**

There is an additional XML error response format for failed SQL requests to the Easysoft XML-ODBC Server:

- **SQL Error response**

SQL REQUESTS

See **"sql_request.dtd" on page 148** for the SQL request DTD layout and detailed information of all the possible fields and elements which XML SQL request code may include.

A SQL request to return XML in Long, Short, Standard or TableTags format:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Request>
<ConnectionString>
  DSN=demo;UID=demo;PWD=easysoft;
</ConnectionString>
<Sql>select * from oobdist_contact</Sql>
<Format>
```

```
  <Long/> OR <Short/> OR <Standard/> OR
<TableTags/>

</Format>

</Request>
```

<ConnectionString> is ODBC defined, containing attributes and values, separated by semicolons, which are to be used on the Easysoft XML-ODBC Server to connect to your database. This example connects to DSN called "demo" using database authentication consisting of a user name of "demo" and a password of "easysoft".

<Sql> is any SQL statement that your database accepts and which you want to execute on the Easysoft XML-ODBC Server. This example is asking for all the rows to be returned from a table called "oobdist_contact".

<Format> must contain ONE of <Long>, <Short>, <Standard> and <TableTags>, where <Standard> is currently the same as <Long>. The output format defaults to <Long> if <Format> is not defined.

NB	This code is available in *<InstallDir>*/easysoft/xml-odbc/documentation/dtds under Unix, or *<InstallDir>*\Easysoft\XML-ODBC Server\Documentation\dtds under Windows as sql_request_long.xml, sql_request_short.xml or sql_request_tabletags.xml

LONG SQL RESPONSE

This is the default output format, producing a larger output for a given result set, because each <Column> in a <Row> contains a "Name" attribute to which is assigned the column name.

All the sample client applications distributed with the Easysoft XML-ODBC Server issue requests for the default format and expect <Long> format output (see **"Sample client applications" on page 91**) .

```
<Result State="00000">
  <Row>
    <Column Name="column1name">column 1 data</Column>
    <Column Name="column2name">column 2 data</Column>
  </Row>
</Result>
```

See **"sql_response_long.dtd" on page 150** for the Long SQL response DTD layout.

SHORT SQL RESPONSE

This format firstly defines the column names in the <Columns> element.

Each <Row> element then contains multiple <Column> elements (one for each column in a result set row), each containing data for the columns which have been defined.

```
<Result State="00000">
  <Columns>
    <Column>column 1 name</Column>
```

```
    <Column>column 2 name</Column>
  </Columns>
  <Row>
    <Column>column 1 data</Column>
    <Column>column 2 data</Column>
  </Row>
</Result>
```

See **"sql_response_short.dtd" on page 151** for the Short SQL response DTD layout.

TABLETAGS SQL RESPONSE

This is the shortest output format, where the elements used to describe the result set are based on the ODBC driver cursor name and the column names.

`<CursorName>` will be the result of calling `SQLGetCursorNameW()`.

You can use this output format to import data directly into Microsoft Office applications.

```
<Result>
  <CursorName>
    <Column1Name>column 1 data</Column1Name>
    <Column2Name>column 2 data</Column2Name>
  </CursorName>
</Result>
```

See **"sql_response_tabletags.dtd" on page 152** for the TableTags SQL response DTD layout.

SQL Error response

If a SQL request in any format is unsuccessful, then a response is returned containing diagnostic information rather than data (the diagnostic format is the same, irrespective of the request format).

An error can be distinguished by examining the `<State>` attribute in the `<Request>` element, which will contain either "`00000`" for successful requests, or some other state (defined by ODBC) for unsuccessful requests.

This will then be followed in the XML output by one or more `<Diagnostic>` elements.

e.g.

```
<Result State="S0002">
<Diagnostic Number="1">
    <State>S0002</State>
    <Native>208</Native>
    <Text>[Microsoft][ODBC SQL Server Driver][SQL Server]
          Invalid object name 'tabledoesnotexist'.
    </Text>
</Diagnostic>
```

See **"SQL" on page 148** for response DTD layouts, showing the `<Diagnostic>` field and its elements.

Binary columns

The Easysoft XML-ODBC Server currently handles binary columns in one of three ways specified by the `<Binary>` element `"Output"` attribute (see **"sql_request.dtd" on page 148**):

- `<Binary Output="sql">`

This is the default (when no `<Binary>` element is specified in the request).

The Easysoft XML-ODBC Server requests binary columns from the ODBC driver as `SQL_C_CHAR`, producing two digit hex output (i.e. each 8 bit character in the binary column is output as two hex digits).

- `<Binary Output="none">`

Here binary columns are ignored and are omitted from the result set.

NB	This means that if you issue a query which includes a binary column, it will be omitted from the XML result. This is the one case where the XML result does not match your SQL query.

- `<Binary Output="base64">`

Binary columns are output as base64.

Column data output formats

Column data may be output as either PCDATA or CDATA (see **"sql_request.dtd" on page 148**).

By default the Easysoft XML-ODBC Server outputs column data as PCDATA.

This is selected with:

```
<Data Type="pcdata">
```

Column data output as PCDATA follows these rules:

1. Characters with the eighth bit clear are output without change (for UTF-8).
2. Characters with the 8th bit set are output as character references

   e.g.

   `&#xXX;` (where "XX" is the value in hex)
3. Characters with values greater than "`0xff`" are output as two byte character references

   e.g.

   `&#xXXXX;`
4. Entity references are output for "<", ">", "&", "'" and """, as follows:

   `> >`

   `< <`

   `' &apos;`

   `" &quot;`

   `& &`

   `<Data Type="cdata">`

   If you request column data as CDATA then column data follows these rules:

1. Column data in enclosed in a CDATA section:

   `<![CDATA[column data here]]>`
2. Character references and character entities are not used.
3. The UTF-8 output is pure UTF-8.

Truncating character columns

Fixed length SQL_CHAR columns are returned by an ODBC driver with all the trailing spaces.

You can override this behavior in the Easysoft XML-ODBC Server by using the <TruncateTrailingBlanks> element in the <Format> element of the request (see **"sql_request.dtd" on page 148**).

e.g.

```
<Format>
  <Long/>
  <TruncateTrailingBlanks/>
</Format>
```

Column names

Column names are used as elements and element attributes in the various output formats.

Some queries you may issue might produce a result set where the ODBC driver will not name some of the columns.

e.g.

```
select 1*5,column from table"
```

If the ODBC driver does not produce a name for the column the Easysoft XML-ODBC Server will use <ColumnN> (where "N" is the column number) as the column name.

You can overrride this behaviour in you SQL by using the "as" clause.

e.g

```
"select 1*5 as number,column from table"
```

Adding Style Sheet references

By default the Easysoft XML-ODBC Server adds an XML version definition and an encoding definition, but no style sheet reference, to the start of the output XML.

You can specify a style sheet name and type in the request using the `<StyleSheet>` element:

```
<StyleSheet>
  <Name>stylesheet name</Name>
  <Type>stylesheet type</Name>
</StyleSheet>
```

e.g.

```
<StyleSheet>
  <Name>test.xsl</Name>
  <Type>text/xsl</Name>
</StyleSheet>
```

This results in the following XML Processing Instruction at the top of the XML output:

```
<?xml-stylesheet href="test.xsl" type="text/xsl"?>
```

This is useful if you define a style sheet template and want your browser to apply the template to the XML.

Style Sheets

Microsoft Internet Explorer uses a default style sheet if one is not specified in an XML file with an XML Processing Instruction.

Mozilla by default removes the XML tags.

You can add a style sheet reference to the output XML using the `<StyleSheet>` element (see **"Adding Style Sheet references" on page 89**).

Transaction support

The Easysoft XML-ODBC Server can accept requests containing multiple SQL statements to be executed as a single transaction.

You may want to use transactions if you are performing multiple updates, inserts or deletes as part of a single operation.

For example, suppose you allow users to register on your web site and have a checkbox which they can check to receive a newsletter.

When they register you want to insert an entry into a registered users table and then insert another entry into the table listing users who want to receive the newsletter. There is no point in adding a entry to the newsletter table if the insertion into the registered users table failed so you want both inserts to work or neither.

You can achieve this using transactions, which can be used with the Easysoft XML-ODBC Server by enclosing multiple `<Sql>` elements within a `<Transaction>` element.

e.g.

```
<Request>

 <ConnectionString>xxx</ConnectionString>
```

```
 <Transaction>
   <Sql>first SQL statement in the transaction</Sql>
   <Sql>second SQL statement in the transaction</Sql>
   .
   .
 </Transaction>
</Request>
```

When the Easysoft XML-ODBC Server sees a transaction, it turns off AUTOCOMMIT in the ODBC driver and starts issuing the SQL statements one at a time.

If any of the SQL statements fail, an `SQLEndTrans` (`SQL_ROLLBACK`) statement is issued to roll back the transaction.

If all of the SQL statements succeed it issues a `SQLCommit()` statement to commit the transaction.

Sample client applications

The Easysoft XML-ODBC Server includes information and examples for following languages and programs:

- **"C" on page 92**
- **"Perl" on page 98**
- **"PHP" on page 112**
- **"Java" on page 118**
- **"Visual Basic" on page 125**
- **"netpipes" on page 129**

All the files listed are distributed with the Easysoft XML-ODBC Server product and are located in the corresponding directories within the `Clients` subdirectory of your chosen install path.

C

File List

- `Makefile`

A generic `make` file for building the sample C client.

- `Makefile.xxx`

Various other versions of the above `make` file for different platforms. If a `make` file is not supplied for your platform, then use the generic version and edit it to reflect your required compiler and compiler options.

- `xmlodbc_client.c`

The `xmlodbc_client` program C source code.

- `config.h`

A header file for `xmlodbc_client.c` containing various macros (this file may need to be edited if `xmlodbc_client.c` does not compile successfully on your platform).

- `xmlodbc_client`

The example `xmlodbc_client` program prebuilt for this platform distribution.

Building the example Client

You will need an ANSI C compiler and `make` (although it is possible to compile the client without `make`).

The distribution includes a generic `make` file (`Makefile`), which should suffice for most platforms, and also additional platform-specific `make` files named `Makefile.<platform>`.

If you see a `make` file for your platform use:

`make -f Makefile.`*`xxx`*

where *xxx* is the platform.

Use the generic make file if there is no `make` file for your platform:

```
make -f Makefile
```

If the code compiles, but fails to link then you may need to specify additional libraries on the link line, of which the most likely to be missing are the socket libraries (see the `Makefile.sunos` file for an example of a platform requiring additional link libraries).

If the code fails to compile then ensure that:

- you are using an ANSI compiler

  – OR –

- you have specified the options required to compile in an ANSI mode (on HP-UX, for example, you might need the "`-Ae`" option to force ANSI compilation).

Using the example Client

Prerequisites:

- the Easysoft XML-ODBC Server is installed somewhere on your network accessible from the machine running the client application
- you know the name of the machine where the Easysoft XML-ODBC Server is installed

- you know the name of a System DSN on the machine where the Easysoft XML-ODBC Server is installed
- you know the name of a table in the database pointed to by the relevant System DSN on the machine where the Easysoft XML-ODBC Server is installed

The `xmlodbc_client` program supports the following command line arguments:

`-h hostname`

The name of the server machine where the Easysoft XML-ODBC Server is running. This may be an IP address, host name or a fully qualified domain name, but in each case it is passed to the `gethostbyname()` function, so your client machine needs to be able to resolve this argument value into an IP address if a host name is specified.

`-p port`

This is the port number on which the Easysoft XML-ODBC Server listens for database requests. This setting is optional and the default port of 8895 is used if it is not specified. Two ports can be configured for the Easysoft XML-ODBC Server, one for database requests and one for configuration requests (usually ports 8895 and 8896). You should send database queries to the database port (8895) and configuration requests/operations to the control port (8896).

`-c connection_string`

The connection string to use on the Easysoft XML-ODBC Server to connect to your database. This is an ODBC connection string i.e. what is passed to the ODBC API `SQLDriverconnect()` function.

In its simplest form this requires a DSN name only.

e.g.

```
"DSN=mydsn;"
```

For databases with built in authentication you may need to add UID and PWD attributes for the database user name and password.

e.g.

```
"DSN=mydsn;UID=database_username;PWD=database_password;"
```

Data sources may require other attributes too. You should specify all these attributes in the connection string separating each one with a ';'.

e.g.

```
"ATTRIBUTE1=value;ATTRIBUTE2=value;"
```

NB	If you are still getting ODBC driver connection errors at this point, then consult your ODBC driver documentation.

```
-s query_string
```

This is the database query you want to execute on the server machine. This is usually a request for elements of a table.

e.g.

```
"select * from mytable"
```

NB

If you need to issue SQL containing XML reserved characters (e.g. "<"), then the C client will wrap your SQL in a CDATA section before sending it to the server. If you would rather do this yourself, you need to specify your SQL as follows:

```
-s '<![CDATA[your SQL statement]]>'
```

e.g.

```
-s '<![CDATA[select * from table where
column < 2]]>'
```

The result of this query (the result set) will be returned in XML.

If, for example:

- you have installed the Easysoft XML-ODBC Server on "`myserver`"
- the Easysoft XML-ODBC Server is running in its default configuration (i.e. listening on ports 8895 for database requests and 8896 for configuration/control requests)
- you have a Microsoft SQL Server database on "`myserver`" with user authentication (user name and password `dbuser`/`dbpass`) and a data source called "`mydsn`" pointing to it
- the database contains a table called "`mytable`".

then the command line would be either:

```
./xmlodbc_client
-h myserver -p 8895
-c "DSN=mydsn;UID=dbuser;PWD=dbpass;"
-s "select * from mytable"
```

or, alternatively,

```
./xmlodbc_client
-h myserver -p 8895
-f file.xml
```

where `file.xml` contains:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Request>
<ConnectionString>
  DSN=mydsn;UID=dbuser;PWD=dbpass;
</ConnectionString>
<Sql>select * from mytable</Sql>
</Request>
```

The output should be the contents of the "`mytable`" table in XML. Any error that occurs is also output in XML, rather than as a result set, and both are described in the the "`Documentation`" subdirectory of the Easysoft XML-ODBC Server distribution.

NB	If you need to issue SQL containing XML escape characters (e.g. "<"), then the example Perl client will escape these for you. If you would rather control this yourself you can enter the SQL as XML CDATA: `e.g. -s '<![CDATA[select * from table where column < 2]]>'`

Client Return Status and output

The return status of the `xmlodbc_client` program is "`0`" for success and any other value for an error (see the code for all the possible return codes).

The XML output (including XML output generated for an error occurring on the server) is always directed to `stdout` and errors occurring at the client end are written to `stderr` (see the code for examples).

As a result you can redirect the XML output to a file in most shells with something like:

```
./xmlodbc_client args 1> file.xml
```

Parsing the XML output

There are many XML parsers available on the net which have C APIs you could use with this example C client, such as `Expat` (**http://expat.sourceforge.net**), which requires very straight forward code to parse the XML output of this client, as illustrated in the accompanying examples.

PERL

File List

- `xmlodbc_client.pl`

This is a small Perl script which allows a query to be executed on the specified Easysoft XML-ODBC Server with the results returned exactly as the server returns them (i.e. in XML).

You can also issue configuration requests or ask for statistics.

This script uses `ESXMLODBC::XMLREQUEST` and its methods, so the directory `ESXMLODBC` needs to be on your Perl INC path.

Usage:

```
xmlodbc_client.pl
  [-V]
  -h server [-p port]
  [-t]
  [-r]
  [-f file-request] |
  [-c connection_string
  [-s sql_query |
SQL_statements_in_a_transaction]]
```

where:

`-h` is the name of machine where the server is running

`-p` is the server port to connect to (usually 8895)

`-c` is the ODBC connection string

`-s` is the SQL to run

`-t` truncates trailing characters in SQL_CHAR columns (cannot be used with the `-f` option)

`-r` displays the XML request (cannot be used with the `-f` option)

`-V` prints the `ESXMLODBC::XMLREQUEST` version being used

e.g.

```
xmlodbc_client.pl
  -h myserver.company.co.uk
  -p 8895
  -c "DSN=test;UID=dbuser;PWD=dbpassword;"
```

```
  -s "select * from mytable"
```

where the Easysoft XML-ODBC Server on `myserver.company.co.uk` listens on port 8895 to connect to the `test` data source with a database user name and password of `dbuser` and `dbpassword`, executes the select statement and returns the results in XML.

e.g.

```
xmlodbc_client.pl
  -h myserver.company.co.uk
  -p 8895
  -c "DSN=test;UID=dbuser;PWD=dbpassword;"
          "insert into users values (1,'user@domain')"
         "insert into mailing values ('user@domain', 'Yes')"
```

where the Easysoft XML-ODBC Server on `myserver.company.co.uk` listens on port 8895 to connect to the `test` data source with a database user name and password of `dbuser` and `dbpassword`, and executes the two insert statements. If both inserts work they are committed. If either insert fails then the transaction is rolled back.

e.g.

```
xmlodbc_client.pl
  -h myserver.company.co.uk
  -p 8895
  -f file.xml
```

This sends the SQL request in `file.xml` to the Easysoft XML-ODBC Server on `myserver.company.co.uk` listening on port

8895 and returns the result in XML. You can also use this method to send configuration requests or instructions (see **"Example Configuration XML" on page 133**) and statistics requests (see **"Example Statistics XML" on page 139**) to the Easysoft XML-ODBC Server control port.

- `ESXMLODBC`

A directory containing the Perl modules for the Easysoft XML-ODBC Server.

- `XMLREQUEST.pm`

A Perl module containing the methods which query a particular Easysoft XML-ODBC Server (for examples of how to use this module see `xmlodbc_client.pl` and `xmlodbc_example.pl`):

XMLREQUEST methods and members

new

You must create a new `$h` before using any of the other methods.

```
$h = ESXMLODBC::XMLREQUEST->new();
```

Connect

Creates a connection between the client and server and requires two attributes to achieve this:

Server

The name or IP address of the machine where the Easysoft XML-ODBC Server is listening.

Port

The port on which the Easysoft XML-ODBC Server is listening. **Port** is optional and defaults to 8895.

You may independently set **Port** and **Server** before using **Connect** or pass a hash reference of connection attributes to **Connect**.

e.g.

```
$h->{Server} = "demo.easysoft.com";

$h->{Port} = "8895";

$h->Connect();
```

or

```
$h->Connect({Server=>"demo.easysoft.com",
Port=>"8895"});
```

Connect returns true on successful connection.

Disconnect

Closes the connection between the client and the Easysoft XML-ODBC Server.

ExecuteFile

```
$h->ExecuteFile($filespec);
```

Allows you to pass an XML request residing on the specified file directly to the Easysoft XML-ODBC Server you are connected to.

For more details of the format of these XML requests, see **"sql_request.dtd" on page 148**.

e.g.

a file called "`/tmp/request.xml`" containing the following data:

```
<Request>

<ConnectionString>DSN=demo;UID=demo;PWD=easysoft;

</ConnectionString>
```

```
<Sql>select * from oobdist_contact</Sql>
</Request>
```

can then be executed as follows:

```
$h->Execute(/tmp/request.xml);
```

Execute

```
$h->Execute();
$h->Execute($sql);
$h->Execute($sql, \%attr);
```

Allows you to send an SQL request to the Easysoft XML-ODBC Server and requires two attributes:

ConnectionString - an ODBC connection string defining the ODBC driver and database to which you want to connect.

Sql - passed either as the first argument to **Execute** or set as a reference to array of SQL statements you would like executed as part of a transaction (see **Transactions**).

You can define **ConnectionString** and **Sql** before calling **Execute** by setting them in the hash reference.

e.g.

```
$h->{ConnectionString} = "DSN=demo.easysoft.com";
$h->{Sql} =
  ["insert into table values (1)",
    "update table set columna = 2 where columnb =
1"];
```

Do not set **Sql** as above unless you are wanting to use transactions. Instead pass the SQL as the first argument to **Execute**:

```
$h->Execute("select * from table");
```

There are various other execution attributes which may be set in the **Execute** call by providing a hash reference:

e.g.

```
$h->Execute("select * from table", {Parse=>"1"});
```

See **"Execution attributes" on page 106**.

First Next and Last

These methods may be used after using **Execute** or **ExecuteFile**.

All these methods return a reference to a hash containing the column information (with a key consisting of the column name) or `undef`.

```
$h->First - return the first row in the result set
```

```
$h->Next - return the next row in the result set
```

```
$h->Last - return the last row in the result set
```

Note that these methods may only be used if the **Execution** attribute `<Parse>` is set to 1 before the **Execute** method is used.

Note that you also need Expat and XML::Parser installed (see the `XML_Parser.txt` file included with the distribution in the `Perl section` of the `Client` subdirectory).

e.g.

```
$h->Execute("select * from table", {Parse=>"1"});

$hr = $xoh->First() || die "cannot get first row";

print join ("," keys %$hr), "\n"; # print column names

while($hr) {

  my $col;
```

```
  foreach $col (keys %$hr) {
    print $hr->{$col}, " ";
  }
  print "\n";
  $hr = $xoh->Next();
}
```

Connection attributes

These attributes may be set directly in the hash off $h or by passing a hash reference to the **Connect** method.

`<ConnectionString> (string, read/write)`

The ODBC connection string to be passed to the ODBC API `SQLDriverConnectW` function. It consists of a number of `attribute=value` pairs separated by semicolons.

At its simplest a `<ConnectionString>` is:

`DSN=system_dsn_name;`

where the DSN attribute names the System data source name known to the ODBC driver manager.

However, most ODBC drivers define a number of additional attributes

e.g.

UID

The user name for accessing the database.

PWD

The user name password for accessing the database.

The ODBC driver manager also accepts some specific attibutes like "`Driver`" and "`FileDSN`", which can be used to specify the ODBC driver or a data source defined in a file.

Consult a good ODBC reference and your ODBC driver's documentation for a list of all the possibilities.

Execution attributes

These attributes may be set directly in the hash off $h or by passing a hash reference to the **Execute** method.

Parse (boolean, read/write)

If set to true, causes the resulting XML to be parsed so the **First**, **Next** and **Last** methods may be used. It also sets **NumRows**, **NumColumns** and **Rows** (see **"Execution results" on page 106**).

Truncate

If set to true, causes the **Truncate** Format element to be sent to the Easysoft XML-ODBC Server, requesting the removal of trailing spaces from the end of SQL_CHAR columns.

Execution results

There are a number of execution result attributes which may be accessed off $h.

XML

The XML result returned from the Easysoft XML-ODBC Server.

NumRows

The number of rows in the result set. This is undefined unless **Parse** is set to 1 before execution.

Request

The XML request sent to the Easysoft XML-ODBC Server. This attribute is only defined if the **Execute** method is used.

NumColumns

The number of columns in a row in the result set. This attribute is undefined unless **Parse** is set to true before execution.

Rows

A reference to an array of hash references. There is one hash reference for each row in the result set. The keys to the hash reference are the column names.

```
Rows->[0]->{Column1Name}
Rows->[1]->{Column1Name}
Rows->[2]->{Column1Name}
.
.
Rows[n]->{ColumnnName}
```

This attribute is only defined if the **Execute** method is used.

Error Handling

In this beta of the Easysoft XML-ODBC Server the error handling in this module has not been finalised.

Currently, all methods return `undef` for error and set **ErrorState** and **ErrorText** when an error occurs.

ErrorState

A five character string identifying the error. This is similar to ODBC states. The string "`00000`" represents success. Any other error state string is an error.

ErrorText

A description of the error.

If **Parse** is set to false then `<ErrorState>` will always be "`00000`" if the request was sent to the server and a response was received, even though the XML response suggests the requested action could not be completed successfully.

In this case you need to actually examine the returned XML to identify the problem.

e.g.

With **Parse** set to false and executing this request:

```
<Request>
  <ConnectionString>DSN=test;</ConnectionString>
  <Sql><![CDATA[select * from tabledoesnotexist]]></Sql>
</Request>
```

The returned XML is probably something like this:

```
<Result State="S0002">
<Diagnostic Number="1">
    <State>S0002</State>
    <Native>208</Native>
    <Text>[Microsoft][ODBC SQL Server Driver][SQL Server]
          Invalid object name 'tabledoesnotexist'.
    </Text>
</Diagnostic>
```

indicating the table did not exist, but the **Execute** method will return successfully and **ErrorState** will be "`00000`".

To get full **ErrorState** and **ErrorText** strings from the returned XML you need to set Parse to true. In this case **ErrorState** is set to either a client end error state (e.g. socket could not be created) or to the "`State`" attribute in the `<Result>` element of the result (this will always be "`00000`" for success or the State in the first ODBC diagnostic). In this way you can tell if your SQL execution was successful by testing if **ErrorState** is equal to "`00000`".

- `xmlodbc_example.pl`

A small Perl script serving as an example of how to use ESXMLODBC::XMLREQUEST.

This is a working example, but only if the constants defined for server, port, connection string and SQL are modified.

Using the example Client

You will need to obtain and install Perl (**http://www.perl.com**).

If you want to parse the output XML you will also need

- `Expat` (**http://expat.sourceforge.net**)

and

- the Perl module `XML::Parser` (**http://www.cpan.org**).

If the Perl interpreter is not installed in `/usr/bin` you will need to either:

- edit the first line of each of the Perl sample clients to put the correct path to your Perl interpreter

  – OR –

- prefix all the commands with "`perl`".

On most types of Unix you can find out where your Perl interpreter is installed is by typing:

```
$ which perl
```

If, for example, the install directory is returned as

```
/usr/local/bin/perl
```

then the first line of each of the Perl sample clients would require amending to read

```
#!/usr/local/bin/perl
```

rather than the default

```
#!/usr/bin/perl
```

OR

add the prefix "`perl`" when running a sample client:

```
perl sample_client.pl arguments
```

Install the Easysoft XML-ODBC Server on a machine which has an ODBC driver for your database and create a System data source for your database (User data sources are not seen by the Easysoft XML-ODBC Server, as it runs as a service).

All the example Perl clients are similar in that they need to be passed a server host (where the Easysoft XML-ODBC Server is running), an ODBC connection string (describing the data source on the server you want to connect to), some SQL to execute and a port on which the Easysoft XML-ODBC Server is listening.

For example, if the Easysoft XML-ODBC Server is installed on a Windows machine called "`mywinserver.company.com`" which has an ODBC driver for Microsoft SQL Server and a SQL Server database called "`mydatabase`":

1. Create a System DSN to describe the database:
   - Go into the **Control Panel** (and maybe **Administrative Tools**) and select **ODBC Data Sources**.
   - Click on the **System DSN** tab, then click **Add** and select the **SQL Server** ODBC driver.
   - Give the data source a name (e.g. "`mydatasource`") and use the dialog boxes to select the Microsoft SQL Server machine, the database ("`mydatabase`") and your authentication mechanism.
2. Once the data source has been created, test it (there is a test button in the Microsoft SQL Server ODBC driver dialog boxes and you can also test it, once created, with an ODBC application).
3. On your client machine (where you want to issue the SQL and retrieve the results in XML) you need to install the sample Perl clients and (optionally) `Expat` and `XML::Parser`.
4. Issue a SQL query using the `xmlodbc_client.pl`, a database user name and password of "`dbuser`" and "`dbpass`" for SQL Server authentication and a table name of "`table`".

   e.g.

```
./xmlodbc_client.pl mywinserver.company.com 8895
 "DSN=mydatasource;UID=dbuser;PWD=dbpass;"
'select * from table'
```

NB	If you need to issue SQL containing XML escape characters (e.g. "<"), then the example Perl client will escape these for you. If you would rather control this yourself you can enter the SQL as XML CDATA: `e.g. -s '<![CDATA[select * from table where column < 2]]>'`

The result set generated by your query will be returned in XML to `stdout`, but you can use standard shell redirection to direct `stdout` to a file if you prefer.

The sample clients do not allow you to choose the format of the returned XML, because some of them are dependent on the default Easysoft XML-ODBC Server XML output format (see the "`/Documentation`" subdirectory for a description of the different output formats).

PHP

File List

- `xmlodbc.phtml`

Sample PHP code using the XMLODBC class with `Expat` to parse the XML result set, outputting it as HTML.

- `XMLODBC.inc`

A PHP include file containing the XMLODBC class for use with PHP.

The XMLODBC class contains the following methods:

- **new XMLODBC;**
- **Connect(server, port);**

  Connects to the Easysoft XML-ODBC Server.

  – OR –

  `Hostname = "server";`

  The machine where the Easysoft XML-ODBC Server is located.

  `Port = 8895;`

The port on which the Easysoft XML-ODBC Server is listening.

```
$x->Connect();
```

- **Execute(connection_string, sql);**

  Executes either a single SQL statement or a transaction containing one or more SQL statements.

  – OR –

  ```
  ConnectionString = "connection string";
  ```

  ```
  SQL = "SQL statement";
  ```

  – OR –

  ```
  SQL[0] = "insert/update/delete statement 1
  of transaction";
  ```

  ```
  SQL[1] = "insert/update/delete statement 2
  of transaction";
  ```

  Executes statements 1 and 2 together as one transaction.

- **Disconnect();**

  Disconnects from the Easysoft XML-ODBC Server.

- **TruncateTrailingBlanks**

  Defaults to false, but if set to true this modifies the request to the Easysoft XML-ODBC Server to request the truncation of all trailing blanks from `SQL_CHAR` columns.

- **Request**

  The XML request issued by `XMLODBC` to the Easysoft XML-ODBC Server.

- **XML**

  The XML returned from the Easysoft XML-ODBC Server.

- **Parse**

  If set to true causes the returned XML to be parsed with Expat (see **http://expat.sourceforge.net**).

  This makes available the **NumRows**, **NumColumns**, **First**, **Next**, **Last** and **Rows** methods.

- **NumRows**

  If the **Parse** method is set to true and in an executed state, then this is the number of rows in the result set.

- **NumColumns**

  If the **Parse** method is set to true and in an executed state, then this is the number of columns in each row in the result set.

- **Rows**

  If the **Parse** method is set to true and in an executed state, then this is an array of associated arrays. The first dimension is the rows in the result set.

The second dimension contains associative arrays when the column names are the keys.

e.g.

```
$p = $x->Row[1];
```

row 2 - this is an associative array

```
print_r(array_keys ($p));
```

prints the keys which are column names

- **First**

  If the **Parse** method is set to true and in an executed state, then this returns the first row of the result set (see **"Rows" on page 114**).

- Next

  If the **Parse** method is set to true and in an executed state, then this returns the next row of the result set (see **"Rows" on page 114**).

- Last

  If the **Parse** method is set to true and in an executed state, then this returns the last row of the result set (see **"Rows" on page 114**).

Using the Example Client

You need PHP and optionally `Expat` and XML support built in to PHP.

1. Copy `XMLODBC.inc` and include that file into any PHP in which you intend using the class (the example `xmlodbc.phtml` file contains some sample code using the class).
2. Install the Easysoft XML-ODBC Server on a machine which has an ODBC driver for your database.
3. Create a System data source for your database (User data sources are not seen by the Easysoft XML-ODBC Server, as it runs as a service).

   For example, if the Easysoft XML-ODBC Server is installed on a Windows machine called "`mywinserver.company.com`" which has an ODBC driver for Microsoft SQL Server and a SQL Server database called "`mydatabase`":

4. Create a System DSN to describe the database:
   - Go into the **Control Panel** (and maybe **Administrative Tools**) and select **ODBC Data Sources**.
   - Click on the **System DSN** tab, then click **Add** and select the **SQL Server** ODBC driver.
   - Give the data source a name (e.g. "`mydatasource`") and use the dialog boxes to select the Microsoft SQL Server machine, the database ("`mydatabase`") and your authentication mechanism.
5. Once the data source has been created, test it (there is a test button in the Microsoft SQL Server ODBC driver dialog boxes and you can also test it, once created, with an ODBC application).
6. On your client machine (where you want to issue the SQL and retrieve the results in XML):
   - install the sample PHP in your web server `htdocs` tree
   - make the sample PHP readable and executable by the user name under which the web server is running
   - make sure your web server knows to pass `.phtml` files to the PHP parser (if you use `.php` or some other extension to identify PHP files to your web server you can change the extension on the `xmlodbc.phtml` file)

   To parse the returned XML you will also need `Expat` and XML built in to PHP.

NB	If you are running PHP from the command line then little of the above applies.

7. As an example, you may now use `xmlodbc.phtml` to issue a SQL query with the Easysoft XML-ODBC Server on a machine called "`mywinserver.company.com`", an ODBC data source called "`mydatasource`", a database user name and password of "`dbuser`" and "`dbpass`" (assuming SQL Server authentication is in use) and a table called "`table`":

```
<?

$Test = new XMLODBC;

$Test->Connect("mywinserver.company.com", "8895");

$Test->Execute("DSN=mydatasource;UID=dbuser;PWD=dbpass;",

               "select * from table");

print $Test->XML;

// see xmlodbc.phtml for other methods.

$Test->Disconnect();

?>
```

> **NB**
>
> If you need to issue SQL containing XML escape characters (e.g. "<"), then the example PHP client will escape these for you.
>
> If you would rather control this yourself you can enter the SQL as XML CDATA:
>
> `e.g. -s '<![CDATA[select * from table where column < 2]]>'`

The result set generated by your query will be returned in XML in `$Result`.

The sample clients do not allow you to choose the format of the returned XML

This is because some of them are dependent on the default Easysoft XML-ODBC Server XML output format (see **"Example SQL XML" on page 80**).

Useful references

- Parsing XML with PHP:

http://www.zend.com/zend/art/parsing.php

- Parsing XML with PHP:

http://www.wirelessdevnet.com/channels/wap/features/xmlcast_php.html

- PHP XML Parsing Basics -- A Tutorial:

http://www.analysisandsolutions.com/code/phpxml.htm

JAVA

File List

- `XmlOdbc.java`

The Java source for the client class

- `XmlOdbc.class`

The compiled `XmlOdbc.java` file

- `XmlOdbcDemo.java`

The Java source for the example demo

- `XmlOdbcDemo.class`

The compiled `XmlOdbcDemo.java` file

Building the example Client

Running:

```
javac XmlOdbc.java
```

will produce `XmlOdbc.class`.

Running:

```
javac XmlOdbcDemo.java
```

will produce `XmlOdbcDemo.class`.

XmlOdbc class

`XmlOdbc.class` is a basic client which can be used to communicate with the Easysoft XML-ODBC Server.

It allows you to send SQL statements to an ODBC driver on the same machine as the Easysoft XML-ODBC Server and retrieve the results in XML.

The protocol used is simple and documented by the Document Type Definitions distributed with the Easysoft XML-ODBC Server (see **"SQL" on page 148**).

It has a basic constructor and six methods as follows:

Constructor:

- `public XmlOdbc()`

Methods:

- **connect**

```
public void connect(java.lang.String server)
            throws java.net.UnknownHostException,
              java.io.IOException
```

Establishes a socket connection to the Easysoft XML-ODBC Server using a default port of 8895.

Parameters:

`server` - the Easysoft XML-ODBC Server machine.

- **connect**

```
public void connect(java.lang.String server,
               int port)
            throws java.net.UnknownHostException,
              java.io.IOException
```

Establishes a socket connection to the Easysoft XML-ODBC Server.

Parameters:

`server` - the Easysoft XML-ODBC Server machine.

`port` - the port on which the Easysoft XML-ODBC Server is listening.

- **sendRequest**

```
public void sendRequest(java.lang.String sql,
                java.lang.String odbcConnectionString)
                    throws java.lang.Exception
```

  Constructs a request in XML and sends it to the Easysoft XML-ODBC Server.

  Establishes a connection to an ODBC data source on the machine on which the Easysoft XML-ODBC Server is running using the connection string supplied and executes the given query.

  The result set generated can be retrieved as a string using `getStringResponse()` or as an input stream using `getStreamResponse()`.

  Parameters:

  `sql` - the SQL query to be sent to the Easysoft XML-ODBC Server.

  `odbcConnectionString` - the connection string.

- **getStringResponse**

```
public java.lang.String getStringResponse()
                                throws java.lang.Exception
```

  Gets an XML response from the Easysoft XML-ODBC Server as a string.

- **getStreamResponse**

  ```
  public java.io.DataInputStream getStreamResponse()
                                  throws java.lang.Exception
  ```

  Gets an XML response from the Easysoft XML-ODBC Server as an input stream that can be passed to a parser.

- **sendXml**

  ```
  public void sendXml(java.lang.String xmlrequest)
                                  throws java.lang.Exception
  ```

  Sends an XML request to the Easysoft XML-ODBC Server given as a string parameter

  Parameters:

  `xmlrequest` - the XML request (see **"sql_request.dtd" on page 148**).

Using the example Java Client

Prerequisites:

1. The Easysoft XML-ODBC Server must have been installed in a location which is accessible from this machine.
2. The name of the installation machine used in [1].
3. The name of a System data source on the installation machine used in [1].
4. The name of a table in the database referenced by the System data source on the installation machine used in [1].

   The `XmlOdbcDemo` program supports the following command line arguments:

   - `-h hostname`

The name of the server machine on which the Easysoft XML-ODBC Server is running. This may be an IP address, a host name or a fully qualified domain name. In each case it is passed to `gethostbyname()`, so your client machine needs to be able to resolve this argument value into an IP address.

- `-p port`

Optional. If not specified port 8895 is used. This is the port the XML-ODBC Server is listening on. The Easysoft XML-ODBC Server can be configured to listen on two ports, one for database requests and one for configuration requests (these are usually ports `8895` and `8896`). You should obviously send database queries to the database port (`8895`) and configuration requests/operations to the control port (`8896`).

- `-c connection_string`

The connection string to use on the Easysoft XML-ODBC Server to connect to your database. This is an ODBC connection string i.e. the data passed to the ODBC API `SQLDriverconnect()` function.

In its simplest form this is a simple DSN name

e.g. "`DSN=mydsn;`".

For databases with built in authentication you may need to add `UID` and `PWD` attributes for the database user name and password

e.g.

```
"DSN=mydsn;UID=database_username;PWD=database_password;"
```

Data sources may require other attributes too, which should be specified in the connection string, separating each one with a `';'`

e.g.

"ATTRIBUTE1=value;ATTRIBUTE2=value;"

- -s query_string

The database query to be executed on the server machine.

This is usually a request for elements of a table.

e.g.

"select * from mytable"

The result of this query (the result set) will be returned in XML.

e.g.

Suppose you have installed the Easysoft XML-ODBC Server on "myserver", running in its default configuration (i.e. listening on ports 8895 for database requests and 8896 for configuration and control requests).

Suppose you have a Microsoft SQL Server database on "myserver" with user authentication (a user name of "dbuser" and a password of "dbpass") and a data source called "mydsn" pointing to it. The database contains a table called "mytable".

```
java XmlOdbcDemo -h myserver -p 8895
                 -c "DSN=mydsn;UID=dbuser;PWD=dbpass;"
                 -s "select * from mytable"
```

The output should be the contents of the "mytable" table in XML and if an error occurs then the error is output in XML instead of as a result set.

Both are described in the "/Documentation" subdirectory of this distribution.

Client Return Status And Output

The return status of the `XmlOdbc` program is "`0`" for success and any other value for an error (see the code for all the possible return codes).

The XML output (including XML output generated for an error occurring on the server) is always to stdout. Errors occurring at the client end are written to stderr (see the code for examples). As a result you can redirect the XML output to a file in most shells with something like:

```
java XmlOdbcDemo args 1> file.xml
```

Parsing The XML Output

There are many XML parsers available on the net for Java that you could use with this example client.

A good parser is Expat (**http://expat.sourceforge.net**) and James Clark has XP for Java at **http://www.jclark.com/xml/xp**.

The code required to parse the XML output of this client with XP is very straight forward, as can be seen by the many examples that come with XP.

VISUAL BASIC

File List

- `xmlodbc.dll`

An Active X DLL.

- `example.vba`

An example using `xmlodbc.dll` in Microsoft Excel.

xmlodbc.dll

This Active X DLL contains the following properties, collections and methods:

Properties and Collections:

- **ColumnCount** (read-only)

The number of columns in a row of the result set. Only meaningful if the **Parse** method is used.

- **ColumName** (read-only)

A collection containing Count (the number of columns in a row of the result set) and Name (the name of nth column). Only meaningful if the **Parse** method is used.

- **Connected** (read-only)

A boolean field stating whether the **Connect** method succeeded and you are connected.

- **ConnectionString** (read/write)

Set this to the ODBC connection string to be passed to the Easysoft XML-ODBC Server. Alternatively, this may be passed to the **Connect** method directly.

- **CurrentRow** (read-only)

The index of the current row in the result set. Only meaningful if the **Parse** method is used.

- **Data**(**Column** as Long, [**Row** as Long]) (read-only)

Return the data for the specified column in the specified row. If **Row** if omitted it defaults to **CurrentRow**. Only meaningful if the **Parse** method is used.

- **Eof** (read-only)

A boolean field indicating whether **CurrentRow** now points past the end of the result set i.e. the **Next** method has been called repeatedly to get to the end of **Data**. Only meaningful if the **Parse** method is used.

- **ErrorState** (read-only)

A string containing the state of the last **Execute** method. This mirrors ODBC status codes and is always "`00000`" for success and some other state (e.g. "`HY000`") for errors. You should check this after calling **Execute** to make sure the SQL you specified was executed correctly.

- **ErrorText** (read-only)

If **ErrorState** is not "`00000`", then this contains the error text. This may be the error text for a winsock error, an internal Easysoft XML-ODBC Server error or the error returned from the ODBC driver.

- **Port** (read/write)

This is the port number the Easysoft XML-ODBC Server is listening on. You can set this before calling the **Connect** method or pass the port number to **Connect**.

- **RawData** (read-only)

This is the XML returned by the Easysoft XML-ODBC Server in response to the last **Execute**.

If you do not want the XML parsed, so that you can access some of the above properties, you can use this yourself.

See the Document Type Definitions (DTDs) distributed with the Easysoft XML-ODBC Server for descriptions of the returned XML (note that this Active X DLL requests all result sets using the Easysoft XML-ODBC Server `Long` format).

- **RowCount** (read-only)

This is a count of the number of rows in the result set. Only meaningful if the **Parse** method is used.

- **Server** (read/write)

This should be the name of the server machine where the Easysoft XML-ODBC Server is running. You can set this before calling **Connect** or pass it directly to **Connect**.

Methods:

- **Connect**([**ServerName** As String, [**Port** as Integer]) As Boolean

Use this method to connect to an Easysoft XML-ODBC Server. The **ServerName** and **Port** are optional as they may be set using the **Server**/**Port** properties.

This method returns true if successful. If an error occurs use **ErrorState** or **ErrorText** to identify the problem.

Once you have finished issuing requests you should use the **Disconnect** method.

- **Disconnect**

This method disconnects you from the Easysoft XML-ODBC Server by closing the connected socket.

- **Execute**(**SQLStatement** As String, [**ConnectionStr** As String])

This method requests the Easysoft XML-ODBC Server to which you are connected to access the ODBC data source defined in **ConnectionStr** and issue the SQL in **SQLStatement**.

If you wish, you can set **ConnectionString** separately before calling **Execute**, rather than passing it to the **Execute** method.

The Easysoft XML-ODBC Server will return either a result set or a status in XML to this client. If you use the **Parse** method you can examine **ErrorState** and **ErrorText** to see if the request was successful.

- **Parse**

This method causes the XML returned from the Easysoft XML-ODBC Server to be parsed and initialises most of the above properties.

- **FirstRecord**

This sets the **CurrentRow** property to point to the first row in the result set.

- **NextRecord**

This increments the **CurrentRow** property.

- **PreviousRecord**

This decrements the **CurrentRow** property.

- **LastRecord**

This sets the **CurrentRow** property to point to the last row in the result set.

NETPIPES

`netpipes` (**http://web.purplefrog.com/~thoth/netpipes/ftp**) is a useful program for connecting to the Easysoft XML-ODBC Server, issuing requests and obtaining the results.

The principle benefit of `netpipes` is the ability to make requests from a shell script.

Once built, the `hose` command can be used to send a request to the Easysoft XML-ODBC Server.

e.g.

```
hose server port -in -out sh -c
    "(cat ./request.xml; sockdown) & cat > out.xml"
```

where:

`server` is the server machine where the Easysoft XML-ODBC Server is running

`port` is the port on `server` where the Easysoft XML-ODBC Server is running

`request.xml` is a file containing an XML request

`out.xml` is the output file

NB	There is another tool similar to `netpipes` which you may wish to investigate, known as socket (**http://www.jnickelsen.de/socket/index.html**).

SERVER CONFIGURATION

6

Configuring the Easysoft XML-ODBC Server

This section explains how to configure the Easysoft XML-ODBC Server.

Chapter Guide

Introduction

To describe and control the configuration of the software, the Easysoft XML-ODBC Server uses an XML file called `config.xml`, which is included with the distribution.

`config.xml` is read and parsed at startup to retrieve all the configurable parameters, but after the server has started any client application is able to use an XML request to either request the current running configuration of the Easysoft XML-ODBC Server or to update it.

The Document Type Definitions required for defining configuration requests, responses and updates are defined in the following sections:

- **"config_request.dtd" on page 153**
- **"configuration.dtd" on page 153**
- **"config_response.dtd" on page 155**

If you want to **query** the current configuration of a running Easysoft XML-ODBC Server, then send an XML configuration request to the control port, using the layout in **"config_request.dtd" on page 153**.

Here is a simple request asking for the current configuration:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Request>
  <Configuration/>
</Request>
```

The Easysoft XML-ODBC Server responds to this request with an XML file defining its current configuration using the layout in **"configuration.dtd" on page 153**.

If you want to **update** the current configuration of a running Easysoft XML-ODBC Server, then send an XML configuration update request to the control port using the layout in **"configuration.dtd" on page 153**.

The Easysoft XML-ODBC Server responds to this request with an XML file describing the success or failure of the update using the layout in **"config_response.dtd" on page 155**.

You can use the `xmlodbc_client` compiled C example to issue these requests.

Example Configuration XML

There is an XML format for a request for Easysoft XML-ODBC Server configuration details and for either a response to that request or a request to update those details:

- **Configuration request**
- **Configuration update**

CONFIGURATION REQUEST

An example XML request to return server configuration details:

```
<?xml version="1.0" encoding="UTF-8"?>
<Request>
        <Configuration/>
```

```
</Request>
```

> **NB** This code is available as `config_request.xml` in *<InstallDir>*`/easysoft/xml-odbc/documentation/dtds` under Unix or *<InstallDir>*`\Easysoft\XML-ODBC Server\Documentation\dtds` under Windows.

See **"config_request.dtd" on page 153** for the Document Type Definition for this file.

CONFIGURATION UPDATE

Example XML for either receiving or updating Easysoft XML-ODBC Server configuration details:

```
<?xml version="1.0" encoding="UTF-8" ?>

<Configuration>

    <Port>8895</Port>

    <ControlPort>8896</ControlPort>

    <MaxConcurrentConnect>0</MaxConcurrentConnect>

    <MaxClientConnect>0</MaxClientConnect>

    <Logging>0x0</Logging>

    <LogDir>/tmp</LogDir>

    <Flags>

        <MultiProcess/>

        <EnableControlInterface/>

         <WaitForChildrenTerminationOnClosedown/>

        <KillChildrenOnClosedown/>
```

```
        <ReverseLookup/>
    </Flags>
    <!-- Allow a specific client to issue requests on the
normal request port -->
    <Client
Allow="Yes">194.131.236.64/255.255.255.255</Client>
    <!-- Disallow all other clients from using the normal
request port -->
    <Client Allow="No">0.0.0.0/0.0.0.0</Client>
    <!-- Allow a specific client to issue control requests -
->
    <ControlClient
Allow="Yes">194.131.236.64/255.255.255.255</ControlClient>
    <!-- Disallow all other clients from using the control
port -->
    <ControlClient
Allow="No">0.0.0.0/0.0.0.0</ControlClient>
</Configuration>
```

NB This code is available as `config_change.xml` in *<InstallDir>*`/easysoft/xml-odbc/documentation/dtds` under Unix or *<InstallDir>*`\Easysoft\XML-ODBC Server\Documentation\dtds` under Windows.

The format elements of this file are as follows:

- `Port`

  The port on which the Easysoft XML-ODBC Server listens for connections from clients for SQL requests.

  This is set to `8895` in the initial Easysoft XML-ODBC Server distribution.

- `ControlPort`

  The port on which the Easysoft XML-ODBC Server listens for connections from clients for configuration requests.

  This is set to `8896` in the initial Easysoft XML-ODBC Server distribution.

- `MaxConcurrentConnect`

  The maximum number of threads or processes that the Easysoft XML-ODBC Server will allow at any one time (there will be one thread or process for every ODBC connection). If `MaxConcurrentConnect` is set to 0 (the default) then there is no limit, but you can use this parameter to prevent too many simultaneous connections from swamping your server.

- `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). There is no limit if `MaxClientConnect` is set to 0 (the default). You can

use this parameter to prevent people from swamping your machine with ODBC connections.

- `Logging`

  A bitmask specifying the events to be logged by the Easysoft XML-ODBC Server.

  `Logging` slows the server 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 to which log files are written if logging is turned on (see **"Logging" on page 137**).

  The default values are `/tmp` in Unix (used in this example) and `c:\temp` in Windows, but logging is turned off by default.

- `MultiThreaded`/`MultiProcess`

  Set `MultiThreaded` to start a new child thread for each connection to the Easysoft XML-ODBC Server or

`MultiProcess` to start a new child process for each incoming connection (the default).

This is set to `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 has no effect.

- `EnableControlInterface/DisableControlInterface`

  Set `EnableControlInterface` to enable the control port or `DisableControlInterface` to disable it.

  This is set to `EnableControlInterface` in the initial Easysoft XML-ODBC Server distribution.

- `WaitForChildrenTerminationOnClosedown/IgnoreChildrenOnClosedown`

  Specifies whether the server should wait for its child processes or threads handling connections to terminate before closing down the server.

  This is set to `WaitForChildrenTerminationOnClosedown` in the initial Easysoft XML-ODBC Server distribution.

- `KillChildrenOnClosedown/LeaveChildrenOnClosedown`

  Specifies whether child processes or threads created by the server to handle incoming connections should be terminated when the server is closed down (i.e. if `KillChildrenOnClosedown` is set and a client is connected when the Easysoft XML-ODBC Server service

is closed down, then the processes or threads handling the client connections will be killed, causing those clients to lose their connection to the server).

This is set to `KillChildrenOnClosedown` in the initial Easysoft XML-ODBC Server distribution.

- `ReverseLookup`/`NoReverseLookup`

  Specifies whether the Easysoft XML-ODBC Server should perform a reverse lookup on the connecting client's IP address to get the client's name.

  The name can be used in the statistics and is set to `ReverseLookup` in the initial Easysoft XML-ODBC Server distribution.

- `Client`

  See **"Access control" on page 144**.

- `ControlClient`

  See **"Access control" on page 144**.

See **"configuration.dtd" on page 153** for the Document Type Definition for this file.

Example Statistics XML

There is a request and a response XML format for Easysoft XML-ODBC Server statistics:

- **Statistics request**
- **Statistics response**

STATISTICS REQUEST

An example XML request for server statistics:

```
<?xml version="1.0" encoding="UTF-8"?>
<Request>
        <Statistics/>
</Request>
```

> **NB** This code is available as `statistics_request.xml` in *<InstallDir>*`/easysoft/xml-odbc/documentation/dtds` under Unix or *<InstallDir>*`\Easysoft\XML-ODBC Server\Documentation\dtds` under Windows.

See **"stats_request.dtd" on page 156** for the Document Type Definition for this file.

STATISTICS RESPONSE

An example of an XML response to a request for server statistics:

```
<?xml version="1.0" encoding="UTF-8"?>
<Result>
<Result State="00000">
  <UpTime>0 days, 1 hour, 20 minutes and 24 seconds</UpTime>
  <CPUTime>
    <Kernel>0.01</Kernel>
    <User>0.01</User>
  </CPUTime>
```

```
  <Connections>
    <Total>3</Total>
    <Refused>0</Refused>
    <Active>0</Active>
    <MaxConcurrent>1</MaxConcurrent>
    <ChildrenCreated>3</ChildrenCreated>
    <LastConnect>Fri Jul 05 09:24:26 2002</LastConnect>
   <LastDisconnect>Fri Jul 05 09:24:31 2002</LastDisconnect>
  </Connections>
  <Clients>
    <Client Name="test.easysoft.com">192.168.0.0</Client>
  </Clients>
  <Control>
    <Connections>
      <Total>2</Total>
    </Connections>
  </Control>
</Result>
```

The elements of this file are as follows:

- `Result State`

  The result of a request, containing `State="xxxxx"`, where a successful result is indicated by

`State="00000"` (as in ODBC) and any other value indicates an error.

- `UpTime`

  The time in days, hours, minutes and seconds since the Easysoft XML-ODBC Server was started.

- `Kernel`

  The kernel CPU time consumed by the Easysoft XML-ODBC Server process, including the ODBC Driver Manager, any ODBC drivers and any child processes.

- `User`

  The user CPU time consumed by the Easysoft XML-ODBC Server process, including the ODBC Driver Manager, any ODBC drivers and any child processes.

- `Total`

  The total number of connections (or attempted connections) to the Easysoft XML-ODBC Server, including refused connections, port scanners or anyone

using `telnet` to access the Easysoft XML-ODBC Server ODBC port.

- `Refused`

  The total number of connections refused by the Easysoft XML-ODBC Server.

- `Active`

  The total number of active threads or processes the Easysoft XML-ODBC Server has created to handle ODBC connections.

- `MaxConcurrent`

  The maximum value recorded in the `Active` field.

- `ChildrenCreated`

  The total number of threads or processes that the Easysoft XML-ODBC Server has created during its execution. Connections denied access do not count, because the Easysoft XML-ODBC Server does not start a thread or process for these.

- `LastConnect`

  The time the last Easysoft XML-ODBC Server client connected.

- `LastDisconnect`

  The time the last Easysoft XML-ODBC Server client was disconnected.

- `Clients`

  The number of different client machines which have connected to the Easysoft XML-ODBC Server (where a

client machine is identified by its IP address). "Name" is only displayed if you have ReverseLookup enabled (see **"ReverseLookup/NoReverseLookup" on page 139**).

- Control

  This field contains the same list of elements as the Connections element (<Total>, <Refused>, <Active>, <MaxConcurrent>, <ChildrenCreated>, <LastConnect> and <LastDisconnect>), but for the Easysoft XML-ODBC Server Control Port.

See **"stats_response.dtd" on page 156** for the Document Type Definition for this file.

Access control

On startup, the Easysoft XML-ODBC Server opens the ports specified in config.xml (see **"Example Configuration XML" on page 133**) in order to accept connections with SQL requests (as specified by the Port element) and with requests or updates to the server configuration (as specified by the ControlPort element).

Access to these ports is controlled by access control rules which are defined by using the Client and ControlClient elements (see **"configuration.dtd" on page 153**).

Clients allowed or denied access to the standard SQL request port are defined using the <Client> element.

Clients allowed or denied access to the configuration request port are defined using the <ControlClient> element.

DEFINING CLIENT ACCESS

The order of defined clients is important as the server stops when it finds the first match for a client. If the server finds no match for a client it is allowed access.

Here is an example:

```
<Client Allow="Yes">194.131.236.64/255.255.255.255</Client>
```

The `Allow` attribute may be "`Yes`" or "`No`" specifying whether the named client or network is permitted access to the Easysoft XML-ODBC Server or not.

The Client element data should be of the form:

```
{IP_Address}{/Network_Mask}
```

Both the IP address and the network mask are optional but you need to specify at least one of them. If the network mask is omitted it implies an exact IP address match (i.e. as if network mask is `255.255.255.255`).

If the network mask is specified it is applied to the IP address and the client IP address before comparison.

In the above example, client `194.131.236.64` is allowed access to the Easysoft XML-ODBC Server (i.e. it may issue SQL requests to the main port).

This example rule applies ONLY to client `194.131.236.64` because the network mask is `255.255.255.255`. In this particular case the network mask is not required. However, the network mask is useful when defining networks of computers.

e.g.

```
<Client Allow="Yes">194.131.236.0/255.255.255.0</Client>
```

For this class B address any client in that network is allowed access.

A useful way of denying everyone except the clients you specify as allowed is to terminate the list of clients with a rule like this:

```
<Client Allow="No">0.0.0.0/0.0.0.0</Client>
```

This works because the mask `0.0.0.0` is applied to the IP address `0.0.0.0` to get `0.0.0.0`. The same mask is applied to the client IP address, which also produces `0.0.0.0` and hence the client matches when a comparison is made.

You may define as many or as few clients as you like, but you are recommended to use this last mechanism as a method of ensuring that only the required clients can gain access to the Easysoft XML-ODBC Server.

DOCUMENT TYPE DEFINITIONS

A

Document Type Definitions for the Easysoft XML-ODBC Server

This section includes the Document Type Definitions (DTDs) for the various XML requests and responses sent to and from the Easysoft XML-ODBC Server.

Chapter Guide

Introduction

Document Type Definitions (DTDs) are supplied for the various requests, responses and updates sent to and from the Easysoft XML-ODBC Server.

SQL

The following DTDs are used for SQL requests and responses:

- **sql_request.dtd**
- **sql_response_short.dtd**
- **sql_response_short.dtd**
- **sql_response_tabletags.dtd**

SQL_REQUEST.DTD

Defines a SQL request to be sent to the Easysoft XML-ODBC Server:

```
<!ELEMENT Request (Environment*,
                   ConnectionString,
                   (Sql|Transaction),
                   Format?,
                   StyleSheet?)>
<!ELEMENT Environment (#PCDATA)>
<!ELEMENT ConnectionString (#PCDATA)>
<!ELEMENT Transaction (Sql+)>
<!ELEMENT Sql (#PCDATA)>
```

```
<!ELEMENT StyleSheet (Name, Type)>

<!ELEMENT Name (#PCDATA)>

<!ELEMENT Type (#PCDATA)>

<!ELEMENT Format ((Short | Long | Standard | TableTags),
Data?, Binary?, TruncateTrailingBlanks?)>

<!ELEMENT Short EMPTY>

<!ELEMENT Long EMPTY>

<!ELEMENT Standard EMPTY>

<!ELEMENT TableTags EMPTY>

<!ELEMENT Data EMPTY>

<!ATTLIST Data Type (pcdata | cdata) "pcdata">

<!ELEMENT Binary EMPTY>

<!ATTLIST Binary Output (none | sql | base64) "sql">

<!ELEMENT TruncateTrailingBlanks EMPTY>
```

`Environment` contains zero or more strings of the form `ENV_VAR=VALUE`, one for for each operating system environment variable which is required to be set. `ENV_VAR` is the name of the variable and `VALUE` is the required value. This field may be useful when accessing certain drivers (Oracle, for example, which requires ORACLE_HOME to be defined).

`ConnectionString` is ODBC defined, containing attributes and values separated by semicolons.

`<Transaction>` contains one or more SQL statements that your database accepts.

`<Sql>` is any SQL statement that your database accepts.

If defined, <StyleSheet> must contain "Name" and "Type" elements, which add an xml-stylesheet Processing Instruction to the start of the output, setting href to the value in "Name" and type to the value in "Type".

If defined, <Format> must contain ONE of <Long>, <Short>, <Standard> and <TableTags>, where <Standard> is currently the same as <Long> (see **"Example SQL XML" on page 80**). <Data>, <Binary> and <TruncateTrailingBlanks> are optional.

<Data>:

<Data Type = "pcdata"> (the default) specifies that column data be output as PCDATA and non 7-bit ASCII characters be output as character references.

<Data Type = "cdata"> specifies that column data be output in CDATA sections and non-ASCII data be output as pure UTF-8.

<Binary>:

Output="none" specifies that binary columns be ignored.

Output="sql" specifies that binary columns be retrieved in ODBC SQL_C_CHAR format and output as two digit hex strings.

Output="base64" specifies that binary columns be output in base 64.

<TruncateTrailingBlanks> removes the trailing spaces at the end of SQL_CHAR columns.

SQL_RESPONSE_LONG.DTD

Specifies the result of a SQL request sent to the Easysoft XML-ODBC Server where the <Long> output Format was requested:

```
<!ELEMENT Result (Row* | Diagnostic*)>
```

```
<!ATTLIST Result State CDATA #REQUIRED>
<!-- Note that State="00000" for all success cases -->
<!ELEMENT Row (Column*)>
<!ELEMENT Column (#PCDATA)>
<!ATTLIST Column Name CDATA #REQUIRED>
<!ELEMENT Diagnostic (State, Native, Text)>
<!ATTLIST Diagnostic Number CDATA #REQUIRED>
<!ELEMENT State (#PCDATA)>
<!ELEMENT Native (#PCDATA)>
<!ELEMENT Text (#PCDATA)>
```

SQL_RESPONSE_SHORT.DTD

Specifies the result of a SQL request sent to the Easysoft XML-ODBC Server where the <Short> output Format was requested:

```
<!ELEMENT Result ((Columns, Rows) | Diagnostic*)>
<!ATTLIST Result State CDATA #REQUIRED>
<!-- Note that State="00000" for all success cases -->
<!ELEMENT Columns (Column*)>
<!ELEMENT Rows (Row*)>
<!ELEMENT Row (Column*)>
<!ELEMENT Column (#PCDATA)>
<!ELEMENT Diagnostic (State, Native, Text)>
<!ATTLIST Diagnostic Number CDATA #REQUIRED>
<!ELEMENT State (#PCDATA)>
```

```
<!ELEMENT Native (#PCDATA)>

<!ELEMENT Text (#PCDATA)>
```

SQL_RESPONSE_TABLETAGS.DTD

Specifies the result of a SQL request sent to the Easysoft XML-ODBC Server where the <TableTags> output Format was requested:

CursorName is the result of calling SQLGetCursorName in the ODBC driver. <Column1Name>, <Column2Name>... etc. are the actual names of the columns in the result set.

```
<!ELEMENT Result (CursorName* | Diagnostic*)>

<!ATTLIST Result State CDATA #REQUIRED>

<!-- Note that State="00000" for all success cases -->

<!ELEMENT CursorName (Column1Name, Column2Name,
Column3Name)>

<!ELEMENT Column1Name (#PCDATA)>

<!ELEMENT Column2Name (#PCDATA)>

<!ELEMENT Column3Name (#PCDATA)>

<!ELEMENT Diagnostic (State, Native, Text)>

<!ATTLIST Diagnostic Number CDATA #REQUIRED>

<!ELEMENT State (#PCDATA)>

<!ELEMENT Native (#PCDATA)>

<!ELEMENT Text (#PCDATA)>
```

NB	An error response in any of `sql_response_long`, `sql_response_short` or `sql_response_tabletags` can be distinguished by examining the `<State>` attribute in the `<Request>` element, which will be either "`00000`" for successful requests, or some other state (defined by ODBC) for unsuccessful requests (see **"SQL Error response" on page 85**).

Configuration

The following DTDs are used to request, receive and update Easysoft XML-ODBC Server configuration details:

- **config_request.dtd**
- **configuration.dtd**
- **config_response.dtd**

CONFIG_REQUEST.DTD

Defines a request to return Easysoft XML-ODBC Server configuration details:

```
<!ELEMENT Request (Configuration)>
```

```
<!ELEMENT Configuration EMPTY>
```

CONFIGURATION.DTD

Defines BOTH the response to a request for Easysoft XML-ODBC Server configuration details (see **"config_request.dtd" on page 153**) AND a request to update Easysoft XML-ODBC Server configuration details:

```
<!ELEMENT Configuration (Port,
                         ControlPort,
                            MaxConcurrentConnect,
                         MaxClientConnect,
                         Logging,
                         LogDir,
                         Flags,
                         Client*,
                         ControlClient*)>
<!ELEMENT Port (#PCDATA)>
<!ELEMENT ControlPort (#PCDATA)>
<!ELEMENT MaxConcurrentConnect (#PCDATA)>
<!ELEMENT MaxClientConnect (#PCDATA)>
<!ELEMENT Logging (#PCDATA)>
<!ELEMENT LogDir (#PCDATA)>
<!ELEMENT Flags ((MultiThreaded | MultiProcess),
                 (EnableControlInterface |
DisableControlInterface),
                 (WaitForChildrenTerminationOnClosedown |
IgnoreChildrenOnClosedown),
                 (KillChildrenOnClosedown |
LeaveChildrenOnClosedown),
                 (ReverseLookup | NoReverseLookup))>
<!ELEMENT MultiThreaded EMPTY>
```

```
<!ELEMENT MultiProcess EMPTY>

<!ELEMENT EnableControlInterface EMPTY>

<!ELEMENT DisableControlInterface EMPTY>

<!ELEMENT WaitForChildrenTerminationOnClosedown EMPTY>

<!ELEMENT IgnoreChildrenOnClosedown EMPTY>

<!ELEMENT KillChildrenOnClosedown EMPTY>

<!ELEMENT LeaveChildrenOnClosedown EMPTY>

<!ELEMENT ReverseLookup EMPTY>

<!ELEMENT NoReverseLookup EMPTY>

<!ELEMENT Client (#PCDATA)>

<!ATTLIST Client Allow (Yes | No)  #REQUIRED>

<!ELEMENT ControlClient (#PCDATA)>

<!ATTLIST ControlClient Allow (Yes | No)  #REQUIRED>
```

CONFIG_RESPONSE.DTD

Defines the response to a request to update Easysoft XML-ODBC Server configuration details (see **"configuration.dtd" on page 153**):

```
<!ELEMENT Result (Diagnostic*) >

<!ATTLIST Result State CDATA #REQUIRED>

<!-- Note that State="00000" for all success cases -->

<!ELEMENT Diagnostic (Text)>

<!ATTLIST Diagnostic Number CDATA #REQUIRED>

<!ELEMENT Text (#PCDATA)>
```

Statistics

The following DTDs are used to request and receive Easysoft XML-ODBC Server statistics:

- **stats_request.dtd**
- **stats_response.dtd**

STATS_REQUEST.DTD

Defines a request to return Easysoft XML-ODBC Server statistics:

```
<!ELEMENT Request (Result)>

<!ELEMENT Result EMPTY>
```

STATS_RESPONSE.DTD

Defines the response to a request for Easysoft XML-ODBC Server statistics (see **"stats_request.dtd" on page 156**):

```
<!ELEMENT Result (UpTime,
                  CPUTime,
                  Connections,
                  Clients+,
                  Control)>

<!ATTLIST Result State CDATA #REQUIRED>

<!-- Note that State="00000" for all success cases -->

<!ELEMENT UpTime (#PCDATA)>

<!ELEMENT CPUTime (Kernel, User)>

<!ELEMENT Kernel (#PCDATA)>
```

```
<!ELEMENT User (#PCDATA)>
<!ELEMENT Connections (Total,
                       Refused*,
                       Active*,
                       MaxConcurrent*,
                       ChildrenCreated*,
                       LastConnect*,
                       LastDisconnect*)>
<!ELEMENT Total (#PCDATA)>
<!ELEMENT Refused (#PCDATA)>
<!ELEMENT Active (#PCDATA)>
<!ELEMENT MaxConcurrent (#PCDATA)>
<!ELEMENT ChildrenCreated (#PCDATA)>
<!ELEMENT LastConnect (#PCDATA)>
<!ELEMENT LastDisconnect (#PCDATA)>
<!ELEMENT Clients (#PCDATA)>
<!ELEMENT Control (Connections)>
```

TECHNICAL REFERENCE

B

Technical Reference for the Easysoft XML-ODBC Server

This section contains additional technical information relating to the deployment of the Easysoft XML-ODBC Server.

Appendix Guide

The unixODBC driver manager and Unicode

If invalid characters are found in your ODBC diagnostic text or XML result set output, then this is usually the result of using a version of the unixODBC driver manager prior to 2.2.2 which had some Unicode problems.

These older UnixODBC releases expected big endian input data and converted data to big endian rather than native endian format, a restriction which has now been removed.

All versions of the Easysoft XML-ODBC Server contain at least unixODBC 2.2.2 and should not have this problem, but it is possible that you may have elected to use an existing installation of unixODBC when the Easysoft XML-ODBC Server was installed.

If this is the case then re-run the the Easysoft XML-ODBC Server installation, select to install the included unixODBC and then make sure your dynamic linker picks up this new version of the driver manager, rather than your previous version.

GLOSSARY

C

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 (see **“Flags” on page 163**) in a small amount of memory and can be viewed in decimal as a single number resulting from adding up the values of the individual bits, worth 1, 2, 4, 8, 16, 32 and so on.

C

A programming language developed in the mid 1970s and used originally to write the Unix operating system. It is also used for many business and engineering applications, due to its compactness and low demand for memory.

CGI

Common Gateway Interface. A specification for transferring information between a Web server and a program designed to accept and return data that conforms to the CGI specification, and where the processing occurs on the Web server, rather than on the client machine.

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 (see **“Host” on page 163**) 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 (see **“Driver Manager” on page 162**) 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 162**) to identify it to the ODBC Driver Manager.

DLL

Dynamic Link Library. Windows’ mechanism for shared object code. See also **“Shared Object” on page 165**.

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 HTTP (see **“HTTP” on page 163**).

Driver

See **“ODBC driver” on page 164**.

Driver Manager

Software whose main function is to load ODBC drivers. ODBC applications connect to the Driver Manager and request a DSN (see **“DSN” on page 162**). 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 (see **“Bitmask” on page 160**).

Host

A computer visible on the network.

HTTP

HyperText Transfer Protocol. The means of transferring web pages.

Java

An object-oriented language similar to C++ developed by Sun, which can either be compiled or run via interpreters and runtime environments (called Java Virtual Machines, or JVMs). Small Java applications known as applets can be downloaded from Web Servers and run on your computer by a Java-compatible Web browser.

Middleware

Software that is placed between the client and the server (see **“Server” on page 165**) to improve or expand functionality.

ODBC (Open DataBase Connectivity)

A standard API (see **“API” on page 160**) for connecting application programs to relational database systems through a suitable driver (see **“ODBC driver” on page 164**).

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.

Perl

Practical Extraction and Report Language. An interpretive programming language especially designed for processing text, which has become one of the most popular languages for writing CGI scripts (see **“CGI” on page 161**).

Platform

A description of 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.

PHP

PHP Hypertext Preprocessor. A server-side scripting language which can be embedded within HTML to create dynamic Web pages. The strength of PHP lies in its compatibility with many types of databases and network protocols and it is now shipped as standard with a number of Web servers, including RedHat Linux.

Processing Instruction

A mechanism provided by XML to allow rules to be passed to an application as to how to handle a document element (or its contents) along with the document itself.

SAX

Simple API for XML. An API (see **“API” on page 160**) that allows a Web file that uses XML (see **“XML” on page 167**) to describe a collection of data to be interpreted. SAX is used to specify and control events via an XML parser.

Server

A computer on a 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 161**.

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 166**.

Table

A data set in a relational database, composed of rows and columns. For example:

software	
vendor	name
Easysoft	Easysoft XML-ODBC Server
MySoft	My ODBC Client Application

This table has two columns; `vendor`, and `name`. It has two rows: one corresponding to Easysoft XML-ODBC, and the other corresponding to MySoft's ODBC client software. The term "`table`" can also apply to just the definition of the table, without its data.

Tag

A formatting command inserted in a document that specifies how all or part of a document, should be interpreted.

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 165**.

Visual Basic

A programming language and environment launched by Microsoft in 1990, which was one of the first products to provide a graphical environment for developing user interfaces. Visual Basic has an object-oriented philosophy and is sometimes referred to as an event-driven language because each object can be programmed to react to a user event, such as a mouse click.

XML

Extensible Markup Language. A specification for documents which allows users to define, transmit and receive data using customized tags.

XSL

Extensible Style Language. A specification for separating style from content when creating XML Web pages, which works like a template to allow a single style document to be applied to multiple pages.

XSLT

Extensible Style Language Transformation. The language used in XSL style sheets to transform all or part of an XML document (see **“XML” on page 167**) stored in one format into a document with a different definition.

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X