EGENIX.COM

mxODBC Zope DA

The Zope ODBC Database Adapter

Version 1.0.10

Printed in Germany.

Contents

1. Introduction

This manual describes the *eGenix.com mxODBC Zope Database Adapter* (DA) for the Zope Application Server.

The mxODBC Zope Database Adapter (DA) gives you a professional quality Zope ODBC interface which allows you to easily connect to any ODBC data sources you have configured on your system via the Windows ODBC Manager on Windows or iODBC/unixODBC ODBC managers on Unix platforms.

The mxODBC Zope DA package includes everything you need to install the Zope DA: the latest `egenix-mx-base` and `egenix-mx-commercial` packages as well as the mxODBC Zope DA product itself.

1.1 Features

- Zope **Level 3** Database Adapter: the mxODBC Zope DA is fully multi-threaded and can handle multiple connections to multiple databases.
- **Drop-in replacement for** the less flexible and performant **ZODBC DA**. This is especially useful for heavily loaded sites.
- **Works on Windows and Unix** platforms without change.
- Fully **compatible to Z SQL Methods**.
- Fully **compatible to the Znolk SQL Wizard** Product **and other similar Zope Products** relying on the common database schema access methods .tables() and .columns().
- **Connection Pooling**: physical database connections are pooled and kept open, to reduce the connection overhead to a minimum. This is especially important for high latency database connections and ones like Oracle which take a considerable amount of time to setup
- **True Parallel Execution of Queries** on a single logical connection: the mxODBC Zope DA can manage any number of physical connections on a single logical connection. This enables running

truly parallel Z SQL Method queries — a feature not available in other Zope DAs which often serialize the queries.

- **Robust Mode of Operation**: connections which have timed out or go away due to network problems are automatically reconnected.

- **Cross-platform** Connection Objects: The Zope DA will automatically choose the right platform specific ODBC manager for you.

- **Per Connection Adjustable ODBC Interface**: mxODBC comes with many different subpackages to choose from on Unix. The Zope DA allows you to select these subpackages on a per-connection basis.

- **Per Connection Error Handling**: you can tell each connection whether it should report ODBC warnings or not; furthermore all warnings and errors are made available as list `.messages` on the `DatabaseConnection` object.

- **Built-in Schema Cache**: this results in improved performance under heavy load.

- **Direct Database Schema Access**: all ODBC catalog methods are made available for much better database schema inquiry. The catalog methods allow building generic database interrogation or manipulation tools and facilitates writing database independent Zope products.

- **Transaction Safe Automatic Reconnect:** when the DA finds that a connection has timed out, it automatically tries a reconnect and replays the transaction on the connection (unlike other DAs which break the transaction scheme by trying a reconnect without replay).

- **Customizable to many different ODBC drivers:** ODBC is a complicated standard and the levels supported by existing ODBC drivers vary greatly. The mxODBC Zope DA provides several connection options which can be used to enhance the compatibility of the interface to the used drivers.

1.2 Requirements

mxODBC Zope DA needs these environment on Windows and Unix for successful installation:

Windows

- All Windows platforms starting with Windows 98 are supported.
- Zope 2.3 or later needs to be installed and working.
- The Windows version of the mxODBC Zope DA uses the Windows ODBC manager as ODBC manager, so you have to configure your ODBC data sources using its GUI interface which is available through the system settings folder.
- You should setup at least one configured and running ODBC data source for testing purposes.

Unix

- SuSE and RedHat Linux distributions for x86 and x86_64 (AMD64/EM64T) processors, FreeBSD and Sun Solaris are supported Unix platforms. We can also provide custom builds for other Unix platforms on request. Please write to *sales@egenix.com* for details.
- Zope 2.3 or later needs to be installed and working.
- On Linux, FreeBSD and Solaris, the binary package includes support for the iODBC and the unixODBC managers. You must have at least one of these installed in order to be able to connect to ODBC data sources. Please use the ODBC manager GUI interfaces to configure the data sources. The Zope DA prefers iODBC over unixODBC if both are installed.
- You should setup at least one configured and running ODBC data source for testing purposes.

2. Installation

eGenix.com distributes the mxODBC Zope DA as bundle of the eGenix.com mx Base Package (`egenix-mx-base`), the eGenix.com mx Commercial Package (`egenix-mx-commercial`) and the mxODBC Zope DA Product in form of binary archive files.

2.1 Download the Software

You can download the binary archives for your combination of platform, Python version and Unicode variant from the eGenix.com web-site at *http://www.egenix.com/*.

> Please make sure that you download the right version for your Zope installation. If you get import errors, notices of failed initialization or Zope hangs, you likely have the wrong product version installed.

These parameters make a difference:

Platform (Windows, Linux, Solaris, FreeBSD)

All recent versions of these operating systems are supported.

Python Build Version (1.5.2, 2.1, 2.3, etc.)

To check which Python version your Zope installation is using, startup the Python interpreter[1] using the –V option:

```
python -V
```

This will print out the Python version number.

Python Build Architecture (32 bit or 64 bit)

On some platforms we support x86 32-bit and x86_64 (AMD64/EM64T) 64-bit versions of Python.

[1] Have a look at the ./bin/runzope or ./bin/runzope.bat startup file in your Zope directory to find the path to the Python interpreter.

To find out which version Zope is using, run the following command:

```
python -c 'import struct; print struct.calcsize("P")*8,"bit"'
```

This will print out “32 bit” or “64 bit”.

Unicode Variant (UCS2 or UCS4)

On Unix, Python can be built using two different Unicode variants: UCS2 and UCS4.

To find out which variant your Python version was compiled with, run the following command:

```
python -c 'print "UCS%s"%len(u"x".encode("unicode-internal"))'
```

This will either print out “UCS2” or “UCS4”.

2.2 Installation in Zope 2.7 and later

Starting with Zope 2.7, Zope is maintained in two separate directories:

- the *Zope Software directory* (Zope’s SOFTWARE_HOME) which we’ll call <Zope Software Directory>/ and
- the *Zope Instance directory* (Zope’s INSTANCE_HOME) which we’ll call <Zope Instance Directory>/.

You can run multiple Zope instances with just one installation of the Zope Software.

Since the mxODBC Zope DA is licensed per Zope Instance and because it is good practice to not install anything into the Zope Software directory (where the main Zope software resides), you are advised to install the mxODBC Zope DA in the **Zope Instance directory** (the directory from where you start your Zope instance by running ./bin/runzope on Unix or .\bin\runzope.bat on Windows).

2.2.1 Before You Start

The binary installation archives include everything you need to run the mxODBC Zope DA, including the necessary egenix-mx-base and egenix-mx-commercial packages for Zope.

If you already have a directory named "<Zope Instance Directory>/lib/python/mx" in your Zope Instance directory, be sure to rename this to some other name, e.g. "<Zope Instance Directory>/lib/python/**old-**mx" before installing the binary packages.

If you don't, the binaries that come with the mxODBC Zope DA will overwrite files from the existing installation of egenix-mx-base/egenix-mx-commercial which can result in a malfunctioning mxODBC Zope DA installation.

Upgrading

When upgrading from a previous version of the mxODBC Zope DA, you should rename the "<Zope Instance Directory>/Products/mxODBCZopeDA" product directory to a different name, e.g. "<Zope Instance Directory>/Products/**old-**mxODBCZopeDA". That way you can restore the previous installation in case something should go wrong.

License Files

In order to run the mxODBC Zope DA, you will need license files from eGenix.com.

If you want to test the product before buying it, you can request evaluation licenses via the eGenix.com web-site at *http://www.egenix.com/*.

When buying licenses from the eGenix.com online shop (*http://shop.egenix.com/*), you will receive the license files immediately after purchase.

In both cases, the license files are sent to the email address you specified during the purchase process or from which you wrote the evaluation license request in form of a ZIP license archive attached to the license email – usually named licenses.zip.

The license archive licenses.zip contains one subdirectory per Zope Instance license you bought. The directories are named after the license key for each Zope Instance license. A typical license archive will have these contents:

```
2100-8789-0322-0926-2568-6429/license.py
2100-8789-0322-0926-2568-6429/license.txt
2100-8089-0312-0926-2668-6529/license.py
2100-8089-0312-0926-2668-6529/license.txt
```

(in the above example, the license archive contains the files for two product licenses).

In order to install the license files, please unzip the license archive to a temporary directory.

During the installation process you will have to move the files to the right location in your Zope Instance directory for the mxODBC Zope DA product to pick them up during Zope startup time.

2.2.2 Step-by-step Installation Guide

Step 1

Unzip the binary distribution archive in the Zope Instance directory <Zope Instance Directory>/, i.e. where you find Zope's lib/, bin/, etc. directories. To unzip a binary package you can use common tools such as Winzip on Windows or Info-ZIP's Unzip on most other platforms.

> Depending on the permission settings on the target machine, you may have to use an *admin* or *root* account for the unzipping step.

Step 2

Move the mxODBCZopeDA/ directory from

<Zope Instance Directory>/lib/python/Products/mxODBCZopeDA

into the directory

<Zope Instance Directory>/Products/.

This is needed starting with Zope 2.7, since Zope no longer scans the lib/python/ directory for products.

You should now have two new directories available in your Zope Instance directory:

- <Zope Instance Directory>/Products/mxODBCZopeDA/ and
- <Zope Instance Directory>/lib/python/mx/

(note that on Windows, the forward slash has to be replaced by a backwards slash).

Step 3

Now that you have installed the product code, you need to install the proper licenses in order for the mxODBC Zope DA to startup correctly.

Go to the temporary directory where you unzipped the license archive and change to the license subdirectory which contains the license for the Zope Instance that you are currently working on.

Copy the two files license.py and license.txt from the license subdirectory to the directory <Zope Instance Directory>/lib/python/mx/ODBC/ in your Zope Instance installation.

Step 4

To complete the installation, restart the Zope instance. Zope will then automatically register the eGenix mxODBC Zope DA.

2.3 Installation in Zope 2.3 – 2.6

The standard installation installs into the <Zope Software Directory>/lib/python directory where <Zope Software Directory>/ is the directory where your Zope software is installed (Zope's SOFTWARE_HOME).

Unlike more recent Zope versions, Zope versions 2.3 through 2.6 do not make a distinction between a "Zope Instance" and the "Zope Installation", thus each instance you setup will require a complete Zope installation and everything resides in the same directory – the <Zope Software Directory>/.

2.3.1 Before You Start

The binary installation archives include everything you need to run the mxODBC Zope DA, including the necessary egenix-mx-base and egenix-mx-commercial packages for Zope.

If you already have a directory named "<Zope Software Directory>/lib/python/mx" in your Zope installation, be sure to rename this to some other name, e.g. "<Zope Software Directory>/lib/python/**old-**mx" before installing the binary packages.

If you don't, the binaries that come with the mxODBC Zope DA will overwrite files from the existing installation of egenix-mx-base/egenix-mx-commercial which can result in a malfunctioning mxODBC Zope DA installation.

Upgrading

When upgrading from a previous version of the mxODBC Zope DA, you should rename the "<Zope Software Directory>/lib/python/Products/mxODBCZopeDA" product directory to a different name, e.g. "<Zope Software Directory>/lib/python/Products/**old-**mxODBCZopeDA". That way you can restore the previous installation in case something should go wrong.

License Files

In order to run the mxODBC Zope DA, you will need license files from eGenix.com.

If you want to test the product before buying it, you can request evaluation licenses via the eGenix.com web-site at *http://www.egenix.com/*.

When buying licenses from the eGenix.com online shop (*http://shop.egenix.com/*), you will receive the license files immediately after purchase.

In both cases, the license files are sent to the email address you specified during the purchase process or from which you wrote the evaluation license request in form of a ZIP license archive attached to the license email – usually named licenses.zip.

The license archive licenses.zip contains one subdirectory per Zope Instance license you bought. The directories are named after the license key for each Zope Instance license. A typical license archive will have these contents:

```
2100-8789-0322-0926-2568-6429/license.py
2100-8789-0322-0926-2568-6429/license.txt
2100-8089-0312-0926-2668-6529/license.py
2100-8089-0312-0926-2668-6529/license.txt
```

(in the above example, the license archive contains the files for two product licenses).

In order to install the license files, please unzip the license archive to a temporary directory.

During the installation process you will have to move the files to the right location in your Zope Instance directory for the mxODBC Zope DA product to pick them up during Zope startup time.

2.3.2 Step-by-step Installation Guide

Step 1

Binary distributions of the package can simply be unzipped in the Zope Software's home directory (which we'll call <Zope Software Directory>/), i.e. where you find Zope's lib/, bin/, etc. directories.

To unzip a binary package you can use common tools such as Winzip on Windows or Info-ZIP's Unzip on most other platforms.

> Depending on the permission settings on the target machine, you may have to use an *admin* or *root* account for the unzipping step.

Step 2

Now that you have installed the product code, you need to install the proper licenses in order for the mxODBC Zope DA to startup correctly.

Go to the temporary directory where you unzipped the license archive and change to the license subdirectory which contains the license for the Zope Instance that you are currently working on.

Copy the two files license.py and license.txt from the license subdirectory to the directory <Zope Software Directory>/lib/python/mx/ODBC/ in your Zope Instance installation.

Step 3

To complete the installation, restart the Zope server. Zope will then automatically register the eGenix mxODBC Zope DA.

2.4 Installation in Plone 2 on Mac OS X

These installation instructions are for Plone 2.x on a Mac OS X system. We assume that you have installed Plone using the standard Plone installer for Mac OS X.

2.4.1 Choosing the Plone Target Site

You can run multiple Plone sites with Plone. Each of these sites is run using an instance of Zope, the underlying application server engine of Plone, a so-called **Zope Instance**.

Since the mxODBC Zope DA is licensed per Zope Instance and because it is good practice to install per-instance products in the Zope Instance directory, we will describe the process of installing the mxODBC Zope DA in the **Default Site** of Plone. This is usually located under the file path /Applications/Plone2/Sites/Default on your server.

If you wish to install the mx ODBC Zope DA for a different site, please change the path from …/Sites/Default to the path of the site you wish to target with the installation.

2.4.2 Before You Start

The binary installation archives include everything you need to run the mxODBC Zope DA, including the necessary egenix-mx-base and egenix-mx-commercial packages for Zope.

> If you already have a directory named "/Applications/Plone2/Sites/Default/lib/python/mx" in your Zope Instance directory, be sure to rename this to some other name, e.g. "/Applications/Plone2/Sites/Default/lib/python/**old-**mx" before installing the binary packages.

If you don't, the binaries that come with the mxODBC Zope DA will overwrite files from the existing installation of egenix-mx-base/egenix-mx-commercial which can result in a malfunctioning mxODBC Zope DA installation.

Upgrading

When upgrading from a previous version of the mxODBC Zope DA, you should rename the "/Applications/Plone2/Sites/Default/Products/mxODBCZopeDA" product directory to a different name, e.g. "/Applications/Plone2/Sites/Default/Products/**old-**mxODBCZopeDA". That way you can restore the previous installation in case something should go wrong.

License Files

In order to run the mxODBC Zope DA, you will need license files from eGenix.com.

If you want to test the product before buying it, you can request evaluation licenses via the eGenix.com web-site at *http://www.egenix.com/*.

When buying licenses from the eGenix.com online shop (*http://shop.egenix.com/*), you will receive the license files immediately after purchase.

In both cases, the license files are sent to the email address you specified during the purchase process or from which you wrote the evaluation license request in form of a ZIP license archive attached to the license email – usually named licenses.zip.

The license archive licenses.zip contains one subdirectory per Zope Instance license you bought. The directories are named after the license key for each Zope Instance license. A typical license archive will have these contents:

```
2100-8789-0322-0926-2568-6429/license.py
2100-8789-0322-0926-2568-6429/license.txt
2100-8089-0312-0926-2668-6529/license.py
2100-8089-0312-0926-2668-6529/license.txt
```

(in the above example, the license archive contains the files for two product licenses).

In order to install the license files, please unzip the license archive to a temporary directory.

During the installation process you will have to move the files to the right location in your Zope Instance directory for the mxODBC Zope DA product to pick them up during Zope startup time.

2.4.3 Step-by-step Installation Guide

Step 1

Download the distribution archive from the *http://www.egenix.com/* web-site. Be sure to select the Mac OS X download.

Unzip the archive to a temporary directory on the installation machine. We'll use the directory /tmp/egenix in our example.

This will create a new directory /tmp/egenix/lib with the software.

Step 2

Unzip the licenses.zip attachment from the license email to the same directory.

This will create at least one new directory with the license key as name, e.g. /tmp/egenix/1111-2222-3333-4444-5555-6666.

Step 3

Open a terminal shell and issue the following commands (please make sure that you issue the commands in the order given here):

```
# Change to the Plone instance home directory
cd /Applications/Plone2/Sites/Default

# Install the product
mv /tmp/egenix/lib/python/Products/mxODBCZopeDA Products/

# Install the product libs
cp -r /tmp/egenix/lib .

# Install the license files (use the license directory
# name of the licenses for this instance)
cp /tmp/egenix/1111-2222-3333-4444-5555-6666/licenses.* \
   lib/python/mx/ODBC/

# Make sure the permission are set correctly on the
# license files
chmod 644 lib/python/mx/ODBC/licenses.*
```

Step 4

Delete the temporary directory /tmp/egenix.

Step 5

Restart Plone. If your Plone installation start at boot time, it is easiest to reboot the machine.

Step 6

Test the new product. Please see section 3 for details on how to configure the product.

3. Configuration

The configuration of access to a database involves two steps:

- Configuration of the database as ODBC data source
- Creation of mxODBC Zope DA Connection objects

The next sections explain the details of these two steps.

3.1 ODBC Data Source Configuration

Before being able to connect to a database, you have to configure the database as data source in the Operating System's ODBC manager.

3.1.1 General Notes

These notes apply on all platforms.

Connection Pooling

The mxODBC Zope DA implements its own connection pooling. It is therefore not required to turn on connection pooling in the ODBC manager. In fact, this may even sometimes result in strange effects, very slow connections and errors due to the state stored on the pooled connections.

> Turning off ODBC manager connection pooling is especially important when using the ODBC driver for **MS SQL Server**. If you do not switch off connection pooling in the ODBC manager for SQL Server connections, you will see a much degraded performance of the mxODBC Zope DA. If you switch off connection pooling, be sure to reboot the client machine in order for the change to take effect.

3.1.2 Windows Platform

On Windows, you must configure the ODBC manager through the standard system settings dialogs (ODBC Data Sources).

Please consult the Windows help files and your database/ODBC driver documentation for details on how to setup data sources in the Windows ODBC Manager.

> Note that if you plan to run **Zope as Windows service**, it may be necessary to setup the ODBC data sources as System-DSN. Otherwise, the Zope process won't be able to see or access the ODBC data sources you setup in the Windows ODBC manager.

3.1.3 Unix Platform

On Unix (Linux or Solaris), it suffices to supply a standard ODBC INI file either as /etc/odbc.ini or in the Zope user home directory as ~/.odbc.ini (note the leading '.') file which uses the same syntax as the Windows file ODBC.INI.

Alternatively, you can use the iODBC/unixODBC management GUIs which allows setting up data sources in the same way as the Windows ODBC manager provides on Windows.

Details on the ODBC manager configuration on Unix can be found on the homepages of the iODBC/unixODBC managers:

iODBC - *http://www.iodbc.org/*

unixODBC - *http://www.unixodbc.org/*

Please consult your database / ODBC driver documentation for details on how to setup data sources using these ODBC managers.

Note that you only need to have one of these two ODBC managers installed on the installation machine for the mxODBC Zope DA to work.

The Zope DA will print out messages at Zope startup time which inform you about the successfully added interfaces for iODBC and/or unixODBC. It will issue a warning in case both interfaces fail to load. In that case, please make sure you have correctly installed the ODBC manager and that the Zope user account is setup to find the shared libraries provided by the ODBC manager of your choice.

3.2 ODBC Driver/Manager Troubleshooting

This section collects a few hints and tricks we have gathered during the beta testing and rollout phase which may be helpful in setting up a working ODBC connection.

Since ODBC drivers can sometimes vary in quality and features, care has to be taken when configuring the ODBC drivers so that you get the best performance and stability possible.

3.2.1 Windows ODBC Manager

The Windows ODBC manager implements a feature called *Connection Pooling* which allows faster connects to databases. In some cases we have observed failures and problems when using the connection pooling feature of the ODBC manager together with the mxODBC Zope DA.

If you are observing similar problems, we suggest that you turn off connection pooling in the Windows ODBC manager for those data sources that you wish to use the Zope DA for.

The mxODBC Zope DA implements its own connection pooling, so switching this feature off in the ODBC manager will not degrade performance.

> Turning off ODBC manager connection pooling is especially important when using the ODBC driver for **MS SQL Server**. If you do not switch off connection pooling in the ODBC manager for SQL Server connections, you will see a much degraded performance of the mxODBC Zope DA. If you switch off connection pooling, be sure to reboot the client machine in order for the change to take effect.

3.2.2 Unix iODBC / unixODBC Manager

On Unix the mxODBC Zope DA uses an already installed iODBC and/or unixODBC manager to communicate with the installed ODBC drivers.

At Zope startup time, the Zope DA tries to import the interfaces for the two ODBC managers and writes a notice to the Zope startup shell window. Zope can only use those interfaces which are successfully imported at this point.

If both interfaces fail to load, the mxODBC Zope DA will not be usable at all and a warning message is printed to the startup shell.

Typical problems which prevent the mxODBC Zope DA from correctly importing the underlying mxODBC interfaces to the ODBC managers are:

- missing ODBC manager installations,
- missing permissions of the Zope user account to access the shared libraries of the ODBC managers (these are typically called libiodbc.so and libodbc.so),
- incorrectly setup linker parameters: the dynamic linker cannot find the shared libraries; this can usually be remedied by setting the LD_LIBRARY_PATH environment variable,
- incompatible ODBC manager versions.

If you use recent versions of the iODBC and/or unixODBC managers, the last point is less likely, since eGenix always builds the binary distributions of the mxODBC Zope DA against the latest stable releases of these managers.

3.2.3 Microsoft Access ODBC Driver

The MS Access database uses the Jet Engine to access the database. ODBC drivers for the Jet Engine prior to version 4.0 are *not* thread-safe and can cause problems if used with mxODBC Zope DA.

Please make sure that you have the latest revision of the Jet Engine and corresponding ODBC drivers installed.

If you get an **error *HY024* mentioning an invalid option** value during connect, it is likely that the data source is an auto-commit-only data source (meaning that it doesn't support transactions), e.g. a file data source.

In such a case:

- create a connection object that is initially closed,
- go to the properties tab of the connection object,
- select "Use Auto-Commit" and "Open Connection"
- click "Save Changes"

The connection should now be opened in auto-commit mode. Note that the data source will not participate in the Zope transaction mechanism. You should only use such data sources for reading data, not writing data.

3.2.4 IBM DB2 ODBC Driver

The DB2 ODBC Driver for NT has an optimization option called "early cursor close" (or similar). This has to be switched off. Otherwise, you'll get lots of SQLSTATE 08001 or 080003 errors during connects and parallel execution of SQL Methods becomes impossible.

3.2.5 SAP DB ODBC Driver

Some versions of the SAP DB ODBC driver have a problem with reporting the correct scale of float columns. As a result, the mxODBC Zope DA returns float columns truncated to integers (this is the ZODBC DA default).

If you encounter this problem, you can configure the SAP DB connection object to not convert scale 0 numeric values to integers by selecting the "Leave scale 0 floats untouched" connection option.

3.2.6 FreeTDS ODBC Driver (access MS SQL Server from Linux)

The FreeTDS ODBC driver is a free ODBC driver for Unix which allows you to connect from Unix to Sybase and/or Microsoft's SQL Server running on different platforms such as Windows 2000.

Unfortunately, the driver's current versions (0.6x) are not yet suitable for production work because they lack support of many important ODBC features.

To work around the showstopper bugs in the driver, eGenix has added a set of compatibility features to the underlying mxODBC interface to at least make the setup mxODBC Zope DA + FreeTDS driver usable for simple queries to the supported databases. See the *mxODBC Documentation* for hints on how to setup FreeTDS to work together with the mxODBC Zope DA.

For production systems, we recommend deploying professional quality ODBC drivers to access MS SQL Server and/or Sybase, such as the ones available from EasySoft, OpenLink and DataDirect.

If you do intend to use the FreeTDS ODBC driver, these are some things to check:

- "*Leave scale 0 floats untouched*" may need to be enabled, since at least some versions of the FreeTDS ODBC driver always return scale 0 for floats which causes the mxODBC Zope DA to believe that the driver is sending an integer. As a result, float values are mistakenly truncated to integers.

3.2.7 PostgreSQL ODBC Driver

The PostgreSQL project has an ODBC driver which is available for Windows as binary and also compiles on Unix from source.

On Unix, the driver is typically included in unixODBC ODBC manager binary packages, so you may have the driver already installed if you're running Zope on Unix and have unixODBC installed (the ODBC driver file is called psqlodbc.so).

Connecting to PostgreSQL using e.g. unixODBC or the Windows ODBC manager works just like for all other databases.

The only known problem with the ODBC driver for PostgreSQL is the lack of support for BLOBs (binary long objects). Please refer to the ODBC driver

documentation for ways to work-around this caveat in the driver. Apart from that data type, all basic data types are supported.

3.2.8 Other ODBC Drivers and Manager Setups

More information about various ODBC driver and manager setups can be found in the *mxODBC Documentation: Interface – Subpackages - General Notes*.

3.2.9 Stored Procedures

If you are using stored procedures with the mxODBC Zope DA, you may get misleading results from the Z SQL Methods.

The reason is that the mxODBC Zope DA supports multiple result sets (unlike the ZODBC DA) and each SELECT used in the stored procedures can result in the ODBC driver generating a new result set.

Per default, the mxODBC Zope DA fetches and returns the first available result set from the ODBC driver, so this may not be the one you actually want from the stored procedure.

To work around this problem, the mxODBC Zope DA Connections provide a connection option "*Fetch last available result set ?*". Turning on this options will cause the Zope DA to fetch the last available result set, rather than the first. See section 3.4.1 *Connection Options* for details.

However, using this option can downgrade the performance of the Zope DA and you are advised to write your stored procedures in a way which does not generate multiple result sets or clear all but the final one prior to returning.

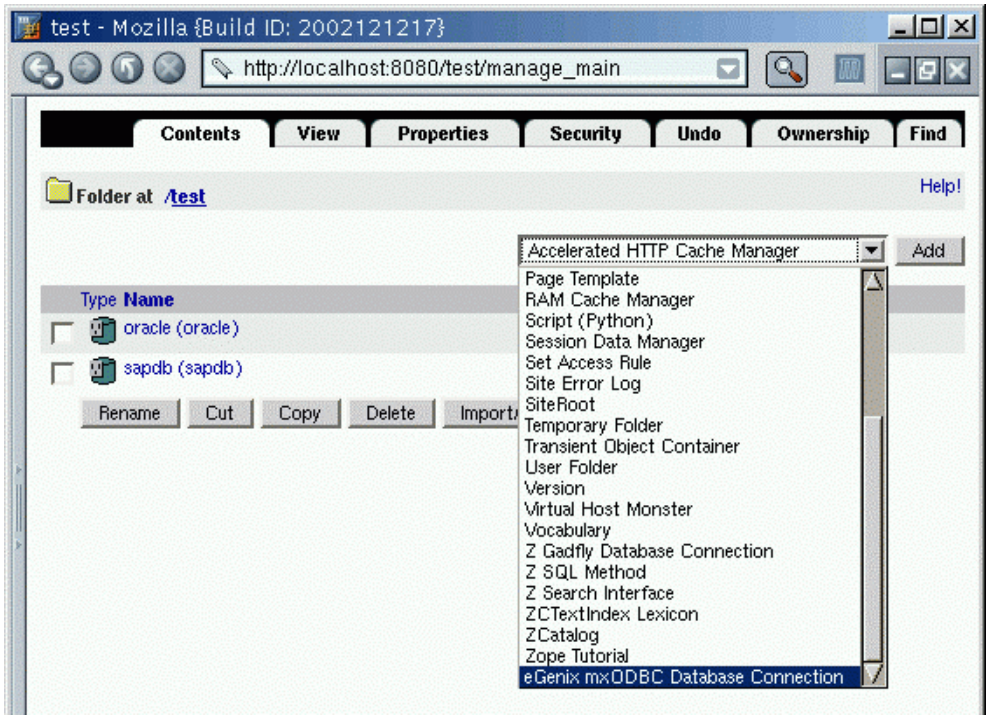
3.3 Creating mxODBC Zope DA Connections

The next few section explain how to create connection object using the mxODBC Zope DA.

Please note that most of the settings you make during the creation process can be adjusted after object creation in the *Properties* dialog.

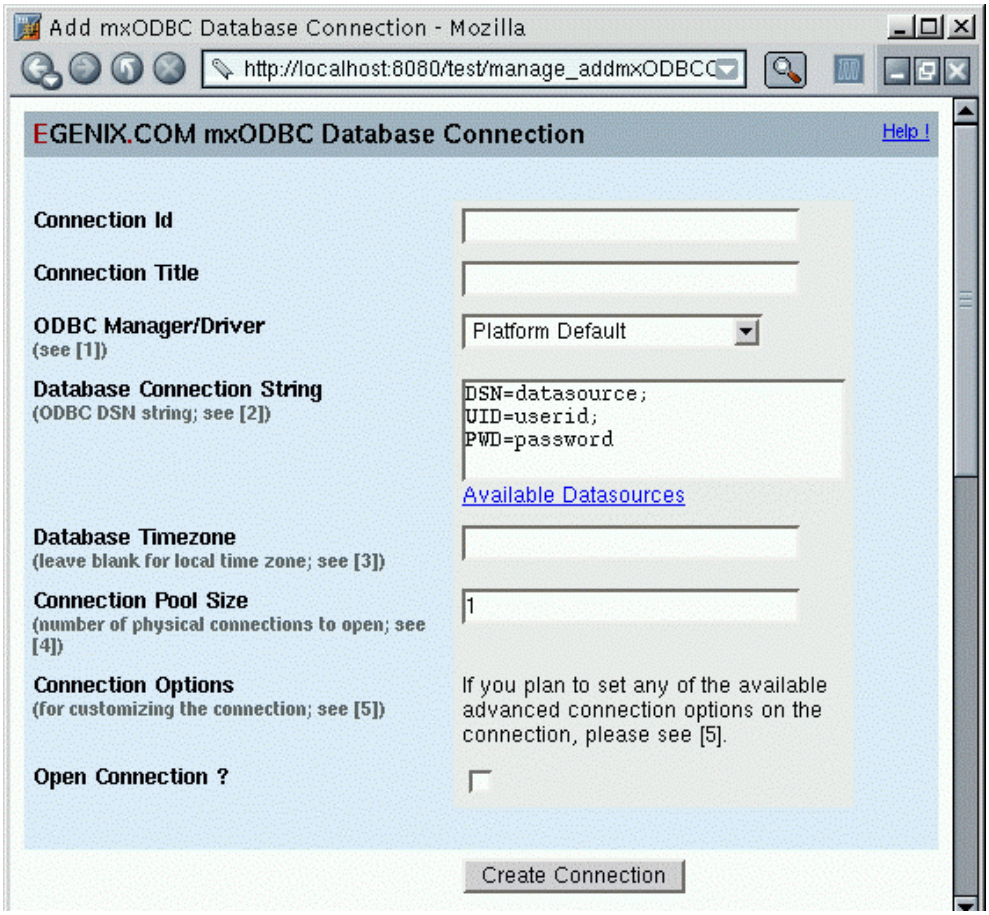
3.3.1 Adding mxODBC Zope DA Connection Objects

After installation and restart, the "Add Object" list box in the Zope management interface should show an entry "eGenix mxODBC Zope Database Connection".



To add a new connection, simply add an object of this type to a folder.

The object creation dialog will be shown, asking you for the parameters of the new object.



You will need to enter the object id, title and details about the connection.

The next sections explain the different connection parameters and how to configure them.

3.3.2 Choosing an ODBC Manager/Driver

Since mxODBC provides various interfaces to ODBC drivers and managers, the creation dialog lets you select the one you want to use for the connection object. Setting this to "Platform Default" will cause the mxODBC Zope DA to automatically choose an available ODBC manager interface for the platform Zope is currently running on.

Leaving this field set to “Platform Default” will make your connection objects portable between Windows and Unix platforms. Choosing a specific ODBC driver may increase performance.

3.3.3 Database Connection String

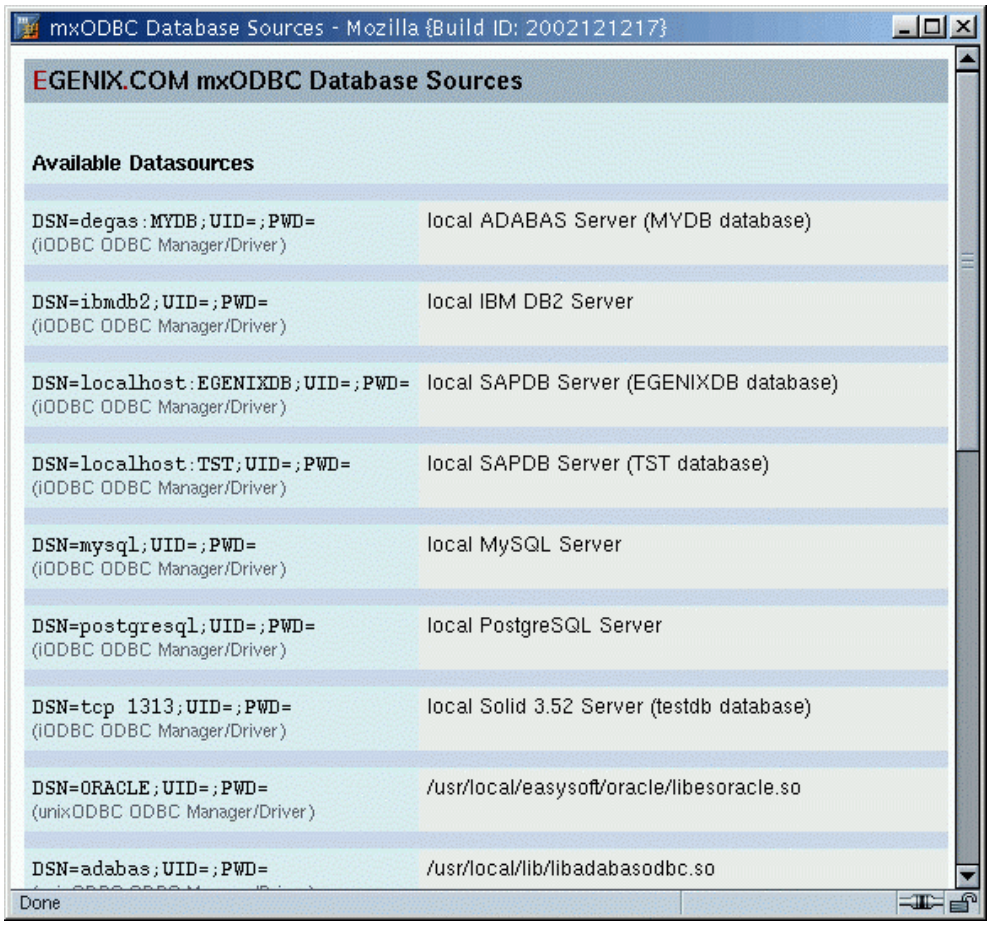
The connection string has to be formatted according to the ODBC DSN standard:

```
DSN=<datasource name as displayed in the ODBC manager>;
UID=<userid>;
PWD=<password>
```

e.g. `DSN=testsource;UID=test;PWD=test`

It is common for ODBC drivers to allow specifying additional parameters in the DSN string. Please consult your ODBC driver documentation for details.

You can click on the "Available Datasources" link to get a pop-up window displaying the currently configured ODBC data sources. The list is fetched from the available ODBC managers and allows for easy cut&paste of the connection string. You only have to fill in the user id and password.



3.3.4 Database Timezone

Zope stores date/time values with an associated time zone. Even though the SQL standard defines date/time values with timezones, not all databases support this and the ODBC standard defines no interface to obtain this information from the database. By entering a timezone in the entry field you can define the timezone to be used for all date/time values fetched from the database, e.g. if you know that the data in the database is GMT, then you can have the Zope DA create Zope DateTime instances which use the GMT timezone. The *Database Timezone* entry field can also be left blank to assume the local timezone.

Note that date/time values passed to the database are **not converted** from the Zope *DateTime* value's timezone to the database timezone, since there is no reasonable way to extract date/time values from the query string for manipulation. This is a limitation of Zope itself, not of the eGenix mxODBC Zope DA.

3.3.5 Connection Pool Size

The mxODBC Zope DA manages two types of connections: logical connections (the connection object you are creating is such a logical connection) and physical connections (these are actual connections to the database).

Unlike other database adapters, the mxODBC Zope DA can handle multiple physical connections per logical connection. This enables using a single logical connection in parallel on multiple threads, e.g. to have a Z SQL Method run on multiple requests simultaneously.

The *Connection Pool Size* defines the number of physical connections to manage on this logical connection object.

If supported by the ODBC driver, it is recommended to set the *Connection Pool Size* to the number of threads run by the Zope application server (usually at least 4).

> **Note:** Each Zope thread will only use one physical connection, so using a connection pool larger than the number of active Zope threads will not result in better performance.

3.3.6 Connection Options

Connection options can only be set on existing objects, so in case you plan to use any of these options, you will first have to create a closed connection object and then edit the connection options in the "*Properties*" tab of the connection object before opening the connection.

See section 3.4 *Configuring mxODBC Zope DA Connections* for details.

3.3.7 Open Connection

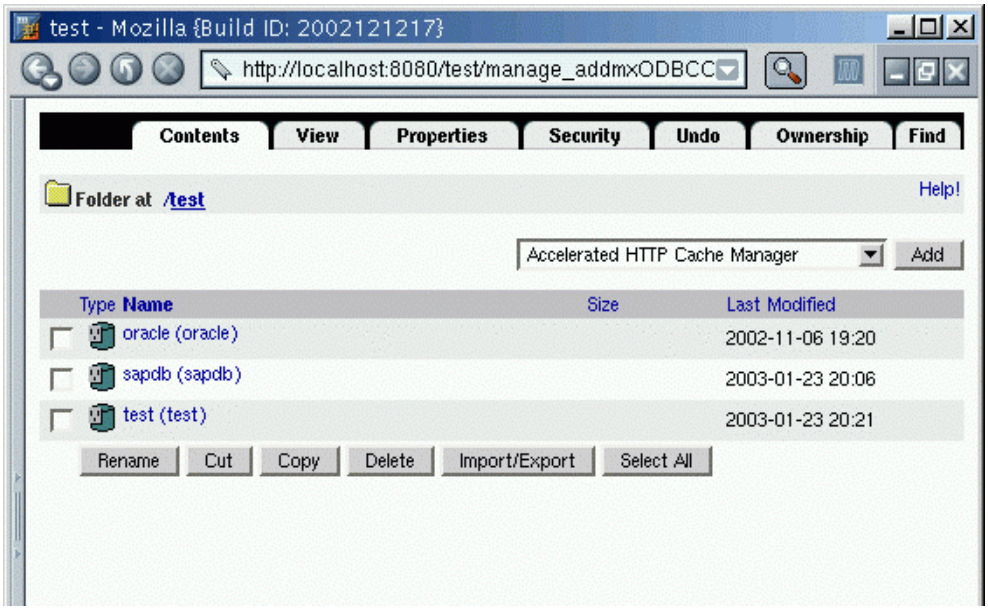
You can either create a closed connection or an open one. Creating a closed connection is sometimes useful in case you want to setup a connection object which is meant to run on a different system. The default is to create a closed connection.

Unlike some other Zope database adapters, the mxODBC Zope DA maintains the state you define here across Zope restarts. If you create an open connection, the Zope DA will try to reopen the connection after a Zope restart. A closed connection will not be opened after a restart.

3.3.8 Create Connection

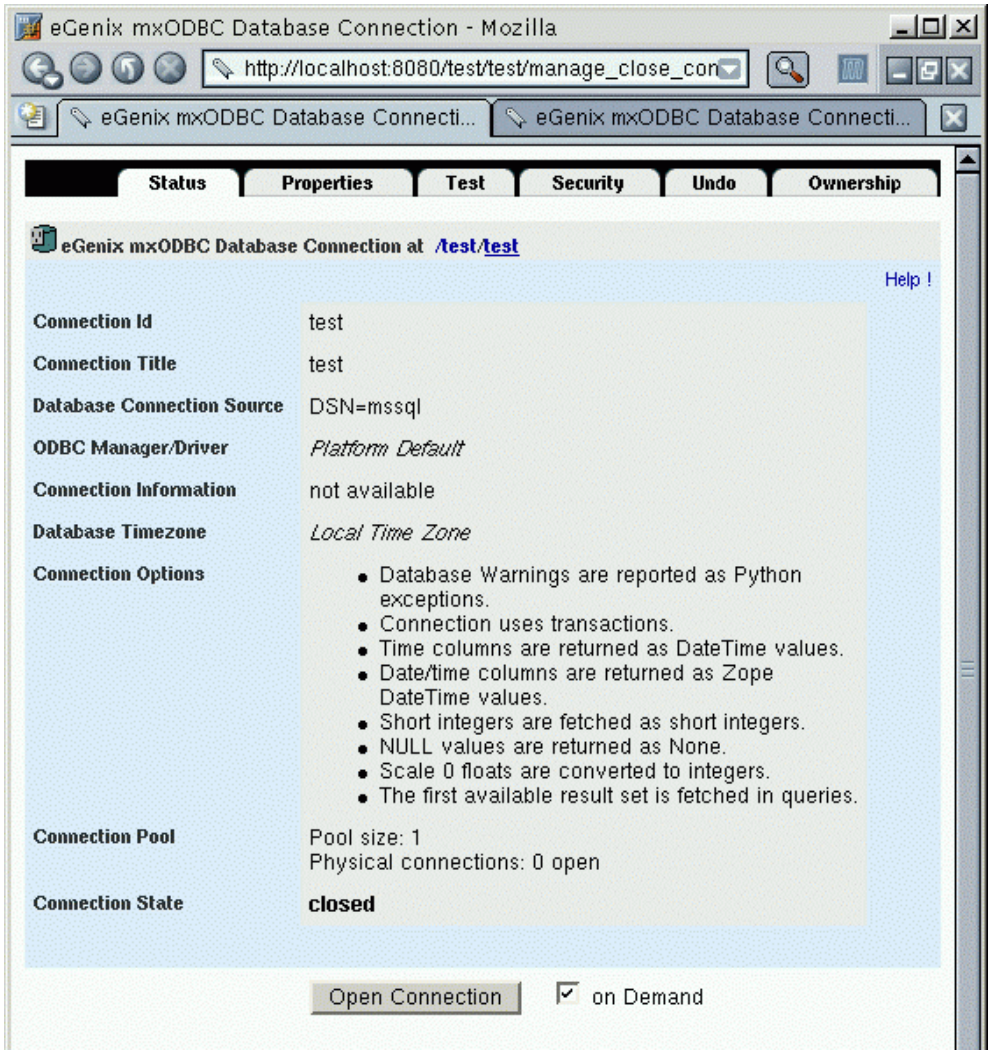
Hitting the Create Connection button will create the mxODBC Zope DA connection object, possibly trying to open the physical connections provided that you have chosen to create an open connection.

You should then see a new connection object in the folder.



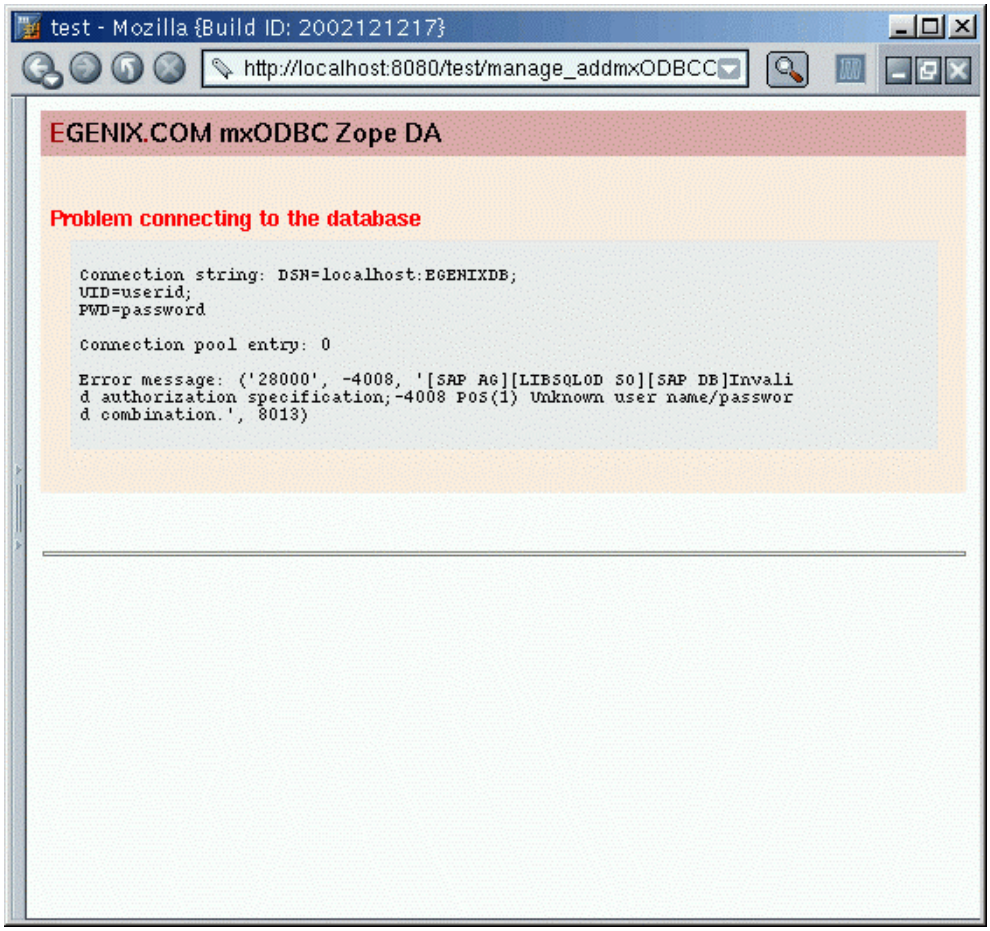
3.3.9 Connection Status

Clicking on the connection object in the folder, brings up the connection status page of the connection management interface.



3.3.10 Error messages

If there is an error during the connection process, the mxODBC Zope DA will display all available information for further inspection.

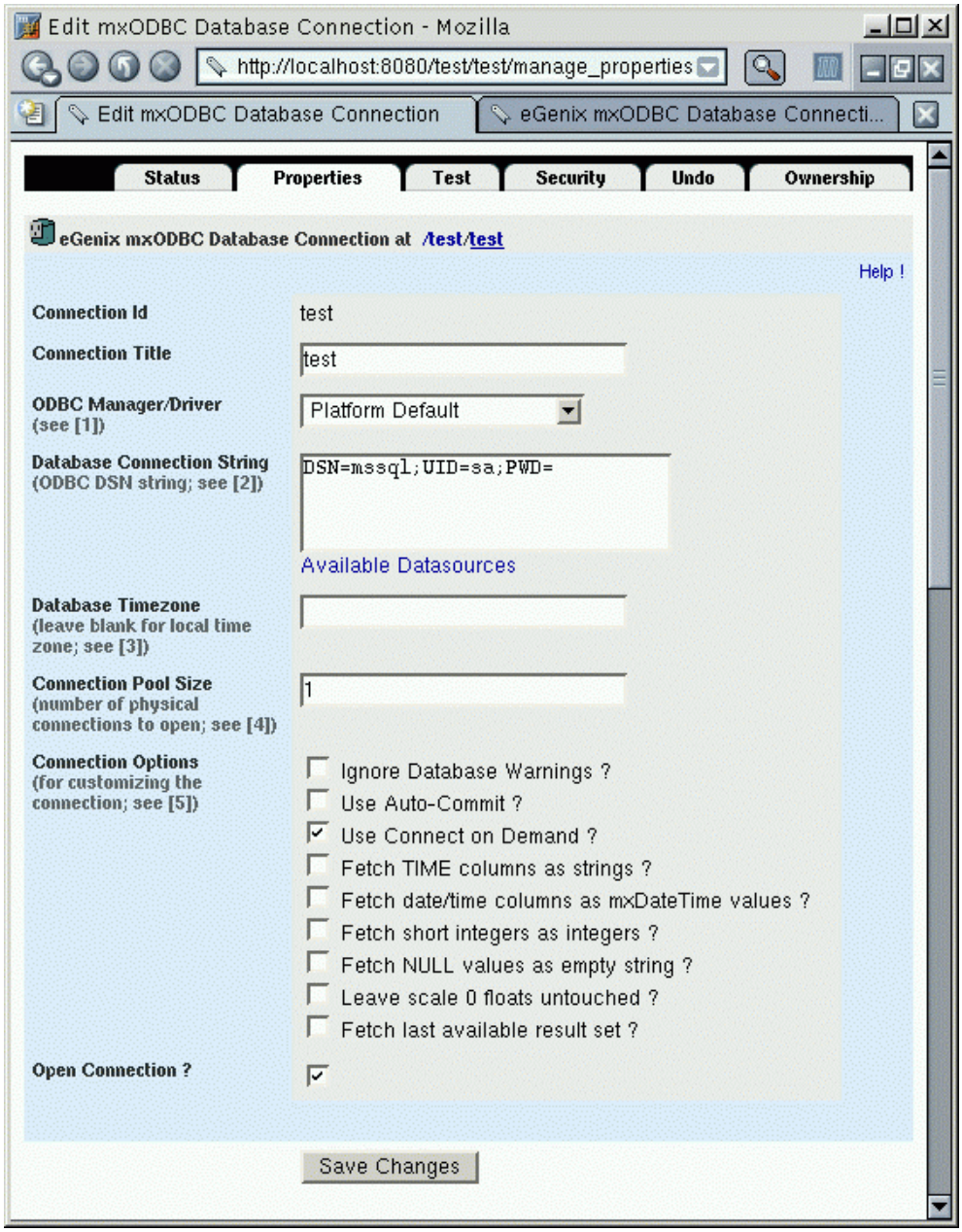


3.4 Configuring mxODBC Zope DA Connections

You can change most mxODBC Zope DA Connection parameters after having created them.

To change any of the options, go to the "*Properties*" tab of the connection object management interface. This will display the full set of available connection options.

You can then make your changes and save them by clicking on the "*Save Changes*" button near the bottom of the screen.



3.4.1 Choosing an ODBC Manager/Driver

Since mxODBC provides various interfaces to ODBC drivers and managers, the creation dialog lets you select the one you want to use for the connection object. Setting this to "Platform Default" will cause the mxODBC

Zope DA to automatically choose an available ODBC manager interface for the platform Zope is currently running on.

Leaving this field set to "Platform Default" will make your connection objects portable between Windows and Unix platforms. Choosing a specific ODBC driver may increase performance.

3.4.2 Database Connection String

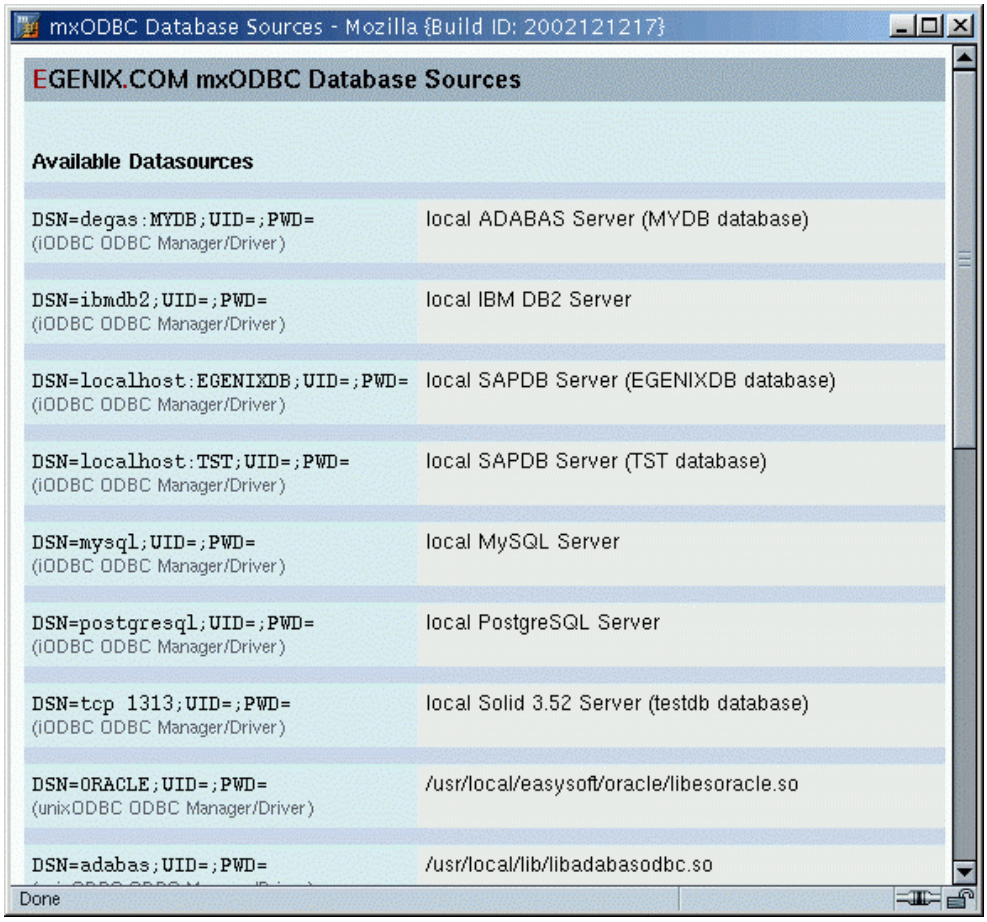
The connection string has to be formatted according to the ODBC DSN standard:

```
DSN=<datasource name as displayed in the ODBC manager>;
UID=<userid>;
PWD=<password>
```

e.g. `DSN=testsource;UID=test;PWD=test`

It is common for ODBC drivers to allow specifying additional parameters in the DSN string. Please consult your ODBC driver documentation for details.

You can click on the "Available Datasources" link to get a pop-up window displaying the currently configured ODBC data sources. The list is fetched from the available ODBC managers and allows for easy cut&paste of the connection string. You only have to fill in the user id and password.



3.4.3 Database Timezone

Zope stores date/time values with an associated time zone. Even though the SQL standard defines date/time values with timezones, not all databases support this and the ODBC standard defines no interface to obtain this information from the database. By entering a timezone in the entry field you can define the timezone to be used for all date/time values fetched from the database, e.g. if you know that the data in the database is GMT, then you can have the Zope DA create Zope DateTime instances which use the GMT timezone. The *Database Timezone* entry field can also be left blank to assume the local timezone.

Note that date/time values passed to the database are **not converted** from the Zope *DateTime* value's timezone to the database timezone, since there is no reasonable way to extract date/time values from the query string for manipulation. This is a limitation of Zope itself, not of the eGenix mxODBC Zope DA.

3.4.4 Connection Pool Size

The mxODBC Zope DA manages two types of connections: logical connections (the connection object you are creating is such a logical connection) and physical connections (these are actual connections to the database).

Unlike other database adapters, the mxODBC Zope DA can handle multiple physical connections per logical connection. This enables using a single logical connection in parallel on multiple threads, e.g. to have a Z SQL Method run on multiple requests simultaneously.

The *Connection Pool Size* defines the number of physical connections to manage on this logical connection object.

If supported by the ODBC driver, it is recommended to set the *Connection Pool Size* to the number of threads run by the Zope application server (usually at least 4).

Note: Each Zope thread will only use one physical connection, so using a connection pool larger than the number of active Zope threads will not result in better performance.

3.4.5 Connection Options

For some data sources or ODBC drivers accessing these data sources, which don't implement the full ODBC standard or have bugs, it is necessary to enable some connection options provided by the mxODBC Zope DA to make it more compatible to these drivers/data sources.

Ignore Database Warnings

Some ODBC drivers tend to raise warnings in situations where extra information needs to be passed to the user such as truncation of data, changing of context, implicit conversions, etc. This means that e.g. an INSERT can cause a Warning exception informing about some special event even though the execution of the statement succeeded. Note that Warning exceptions cause result sets of SELECT statements to be cleared.

mxODBC Zope DA defaults to reporting these warnings to the user, but in some situations it may be required that this is not done, e.g. for deploying Zope on production systems.

You can turn off the reporting of warnings, by selecting the "*Ignore Database Warnings*" checkbox in the connection's properties dialog.

Use Auto-Commit

The mxODBC Zope DA operates in transactional mode per default. This assures data consistency and should always be used in production environments to prevent loss of data.

However, during testing or in read-only situations, it may be desirable to operate some data sources in non-transactional mode. Some data sources also lack transaction support, such as MySQL3, MS Excel flat files, etc.

If you enable "*Use Auto-Commit*", all modifications on the connection will directly change the database, even if Zope rolls back the transaction due to an error.

USE WITH CARE: Failing Zope transactions can cause you data source to become inconsistent or even corrupted. You are safe if the connection is only used for reading data from the data source.

Use Connect on Demand

To save resources during Zope operation, the mxODBC Zope DA can open connections on demand. If you enable "*Use Connect on Demand*", connections will be put in the open state, but the actual data source connection will be deferred to connection usage time.

A connection which is configured to use connect-on-demand will display the "*on Demand*" option in the status page of the connection management screen. This let's you open the connection in the "*Connect on Demand*" state.

Fetch TIME columns as strings

Zope only supports date/time value which have both a date and a time component.

Since database columns of the TIME type do not carry any date information, Zope's DateTime implementation automatically adds the current date to the value.

If you enable "*Fetch TIME columns as strings*", the mxODBC Zope DA will convert these values to time strings rather than Zope DateTime values.

Fetch date/time columns as mxDateTime values

The mxODBC Zope DA converts all mxDateTime instances fetched from the underlying mxODBC interface to Zope DateTime instances to maintain

better compatibility to other Zope database adapters and to make migration to the mxODBC Zope DA easier.

However, the Zope DA can also be configured to return mxDateTime instances directly. This also gives a small performance gain, since the Zope DA does not have to convert the mxDateTime instance it receives from the underlying mxODBC interface into Zope DateTime instance for every query.

To have a connection return mxDateTime instances without conversion, enable the option "*Fetch date/time columns as mxDateTime values*".

Fetch short integers as integers

Some ODBC drivers have problems fetching small integers. As a result the drivers issue database warnings or errors due to overflow problems.

Setting the "*Fetch short integers as integers*" option, will cause mxODBC to fetch these as integers.

Fetch NULL values as empty string

The mxODBC Zope DA will normally fetch SQL NULL values as Python None object. Some other Zope adapters return empty strings instead.

Setting the "*Fetch NULL values as empty strings*" option, will cause mxODBC to return empty strings instead of Python None objects for SQL NULL values.

Leave scale 0 floats untouched

For compatibility with the ZODBC DA, the mxODBC Zope DA will automatically convert scale 0 floating point values (e.g. DECIMAL(10,0)) to integers.

Unfortunately, some ODBC drivers report a wrong scale value for floating point columns. The option "*Leave scale 0 floats untouched*" let's you switch off this behavior, so that floating point numbers will always be returned as floats.

Fetch last available result set

Unlike the ZODBC DA, the mxODBC Zope DA supports fetching multiple result sets from the data source, e.g. in case a stored procedure generated multiple result sets.

mxODBC Zope DA returns the first available result set per default. This can sometimes cause the result set returned by Z SQL Methods to return an intermediate result set from a stored procedure call, e.g. if the stored procedure called a subroutine which then generated a result set.

The option "*Fetch last available result set*" allows configuring the mxODBC Zope DA Connection to work-around this problem by always fetching the last available result set.

Note: It is always advised to code the stored procedure to remove any unneeded result sets before returning, since using this option can introduce a performance penalty.

Always fetch the complete available result set

When using the mxODBC Zope DA with Z SQL Methods, you can specify a value for the maximum number of rows to fetch from a result set. Unfortunately, Z SQL Methods default to a value of 1000 for this `max_rows` limit attribute, instead of defaulting to no limit at all.

If you want to remove that limit, you could set the `max_rows` Z SQL Method attribute to 0 (no limit). Another possibility is using a very high value (e.g. 100000), but this is likely to cause problems with the driver and will significantly increase the processing time of the query.

Enabling the "*Always fetch the complete available result set*" option will let the mxODBC Zope DA always ignore the `max_rows` parameter for queries on the connection object. This will effectively remove the limit and its implications for the connection object and can be used to work-around situations where setting the `max_rows` default value is not possible or not desired.

The `max_rows` query parameter has the following interpretation in the mxODBC Zope DA:

- `0, None`: fetch all rows in the result set
- `< 0`: don't fetch any rows from the result set
- `n`: fetch at most n rows from the result set

If you are using Z SQL Methods to do queries on the connection, you can set the `max_rows` parameter passed to the mxODBC Zope DA by adjusting the Z SQL Method attribute "*Maximum rows to retrieve*" on the objects "Advanced" tab.

3.5 Migration from other Zope Database Adapters

This section explains a few things to consider when migrating from other Zope Database Adapters to the mxODBC Zope DA.

3.5.1 General Notes

The mxODBC Zope DA offers quite a few connection options which allow customizing the connection objects to various needs. If you run into a problem during the migration phase, please check whether you can use these options to work-around the problem without having to change your Zope code.

3.5.2 Migration from the ZODBC DA

For many years, the ZODBC DA has been the only available ODBC database adapter for Zope. If you use the ZODBC DA in production, you will find that it is far slower than the mxODBC Zope DA and it doesn't provide nearly as many professional features as the latter. Furthermore, the Z ODBC DA is only available for the Windows platform, whereas the mxODBC Zope DA runs on most commonly used Zope platforms. such as Windows, Linux, FreeBSD and Solaris, providing the same functionality and interfaces on all supported platforms.

Migration from the ZODBC DA to mxODBC Zope DA can be done by simply recreating the database connection objects using the mxODBC Zope DA as basis.

In some cases, you will find that the mxODBC Zope DA returns data differently than the ZODBC DA. This is due to the more advanced way of fetching data in the mxODBC Zope DA, e.g. it supports multiple result sets and many more data types such as native Unicode whereas the ZODBC DA only supports strings.

To make sure that the new connection behaves in the same way as the old ZODBC DA one, please create a table in your database which uses all column types you are using in your Zope application and compare the results from mxODBC Zope DA and ZODBC DA.

Typical cases to check are:

- Format of date/time values. If these are different, try enabling the "*Fetch TIME columns as strings*" connection option.

- Format of float values. If these come out truncated to integers, try enabling the "*Leave scale 0 floats untouched*" connection option.

- Stored procedures don't return any result set or a wrong one. If this is the case, try enabling the connection option "*Fetch last available result set*".

- Output of NULL values has changed. If your application displays NULL values as "None" on the output screens after you have switched to the mxODBC Zope DA, the Zope adapter you previously used, fetched SQL NULL values as empty string. To have the mxODBC Zope DA expose the same behavior, enable the "*Fetch NULL values as empty string*" option.

3.5.3 Migration from the unofficial mxODBC Zope DA

Some years ago, there was an unofficial Zope Database Adapter available for mxODBC called ZmxODBCDA. This Zope DA was shipped together with mxODBC binaries, but without a proper redistribution license and was thus illegal to use.

However, since the unofficial DA was in already in wide-spread use, eGenix decided to make the official mxODBC Zope DA compatible to this DA.

Migration from the no longer available ZmxODBCDA to mxODBC Zope DA can be done by simply recreating the database connection objects using the official DA.

There are no known incompatibilities between ZmxODBCDA and the mxODBC Zope DA. The latter is much more robust and provides many more advanced features not available in ZmxODBCDA.

3.5.4 Migration from ZMySQLDA

MySQL ships with ODBC drivers for the database which can be used together with the mxODBC Zope DA. Even older MySQL databases which did not have transaction management can be used together with the mxODBC Zope DA, by disabling the transaction management in mxODBC Zope DA[2]. This is done by enabling the connection option "*Use Auto-Commit*" in the connection object management interface.

[2] You should note that operating a database without transactions in read/write-mode can result in severe data corruption as a result of a Zope request not being completed

If you want to run ZMySQLDA and the mxODBC Zope DA in parallel, you should be aware that ZMySQLDA will start to use the mxDateTime product which is part of the mxODBC Zope DA for fetching date/time values.

This can confuse Zope and cause your MySQL Zope code to fail, e.g. passing a date value as mxDateTime DateTime instance to the Zope DateTime constructor can fail. On the other hand, programming against the mxDateTime instances can be easier in some cases and also provides more features.

Here's an example:

```
<span class="text"
tal:content="python:DateTime(items.date_submitted).strftime('%d/%m/
%Y')">21/01/03</span>
```

This code can be written differently with the mxDateTime package installed:

```
<span class="text"
tal:content="python:items.date_submitted.strftime('%d/%m/%Y')">21/0
1/03</span>
```

The reason is that `date_submitted` will be an mxDateTime DateTime instance which provides the requested `.strftime()` method directly.

Configuring mxODBC Zope DA to return mxDateTime Instances

Even though the mxODBC Zope DA converts all mxDateTime instances fetched from the underlying mxODBC interface to Zope DateTime instances per default, it can also be configured to return mxDateTime instances directly, just as in the example above.

To have a connection return mxDateTime instances directly, please enable the option "*Fetch date/time columns as mxDateTime instances*" in the connection object management interface.

---

properly. We strongly suggest to either only access MySQL non-transactional databases read-only or to upgrade to a transactional MySQL database.

4. Usage

The mxODBC Zope DA Connection objects are designed to be usage compatible to most other available Zope database connections, so using them should be straight forward if you already know how to work with relational databases under Zope.

Special care was taken to make the mxODBC Zope DA connection a drop-in replacement for Zope's own simplistic ZODBC database adapter which is much less performant and robust.

4.1 Opening and Closing the Connection

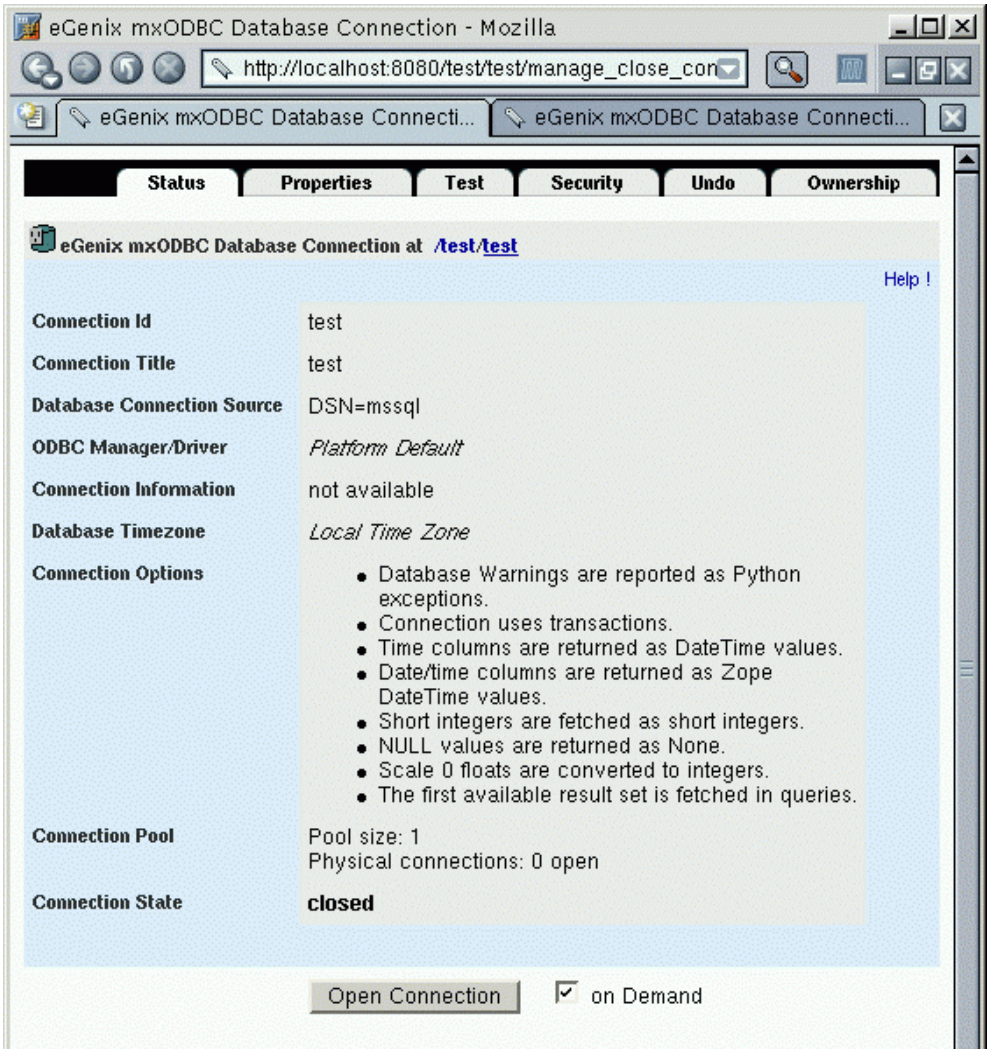
The mxODBC Zope DA Connection object keep persistent state about the connectivity information, i.e. the information whether a connection was put in open or closed state is maintained across Zope restarts (unlike with other Zope DAs).

The mxODBC Zope DA maintains the state at three levels:

- **Closed**: the connection is closed, no resources are used
- **Connect on demand**: the connection to the data source is not open, but will be opened when Zope needs a data source connection.
- **Connected**: the connection to the data source is open.

You can determine the current state of a Connection by checking the connection management status screen.

4.1.1 Opening a Connection



To open the connection, simply press the button Open Connection. Note that the button is only visible in case the connection is closed.

If the connection object was configured to "*Use Connect on Demand*", you may also see a checkbox next to the button "*on Demand*".

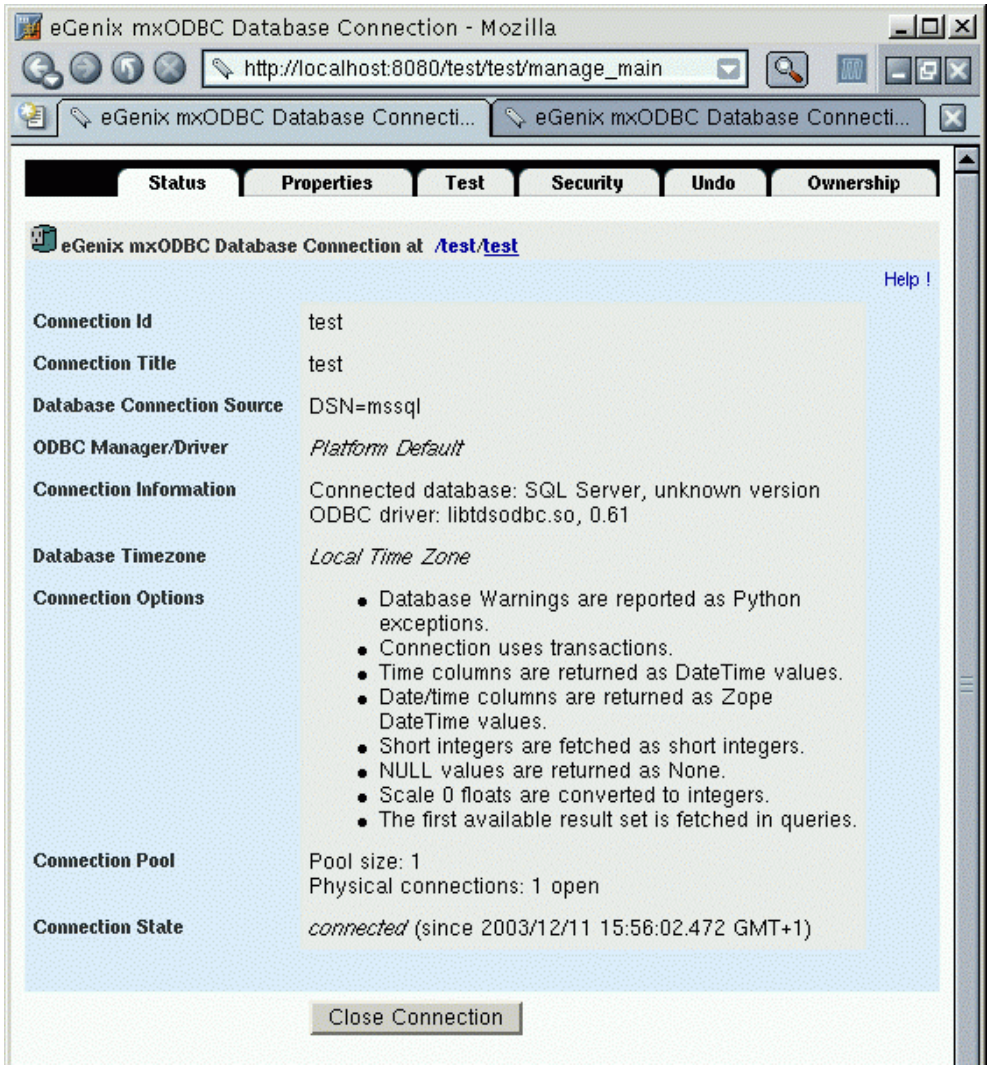
If you select the "*on Demand*" option, the connection will be put into the "*Connect on Demand*" state. This can help save resources, since the connection will only be connecting to the database in case there is demand for it.

When testing the configuration you should uncheck the "*on Demand*" checkbox. This will cause the physical connection to be opened and you

will be able to see any error messages this might cause for the purpose of debugging the database setup.

4.1.2 Closing a Connection

To close a connected connection object, simply press the Close Connection button on the status screen:

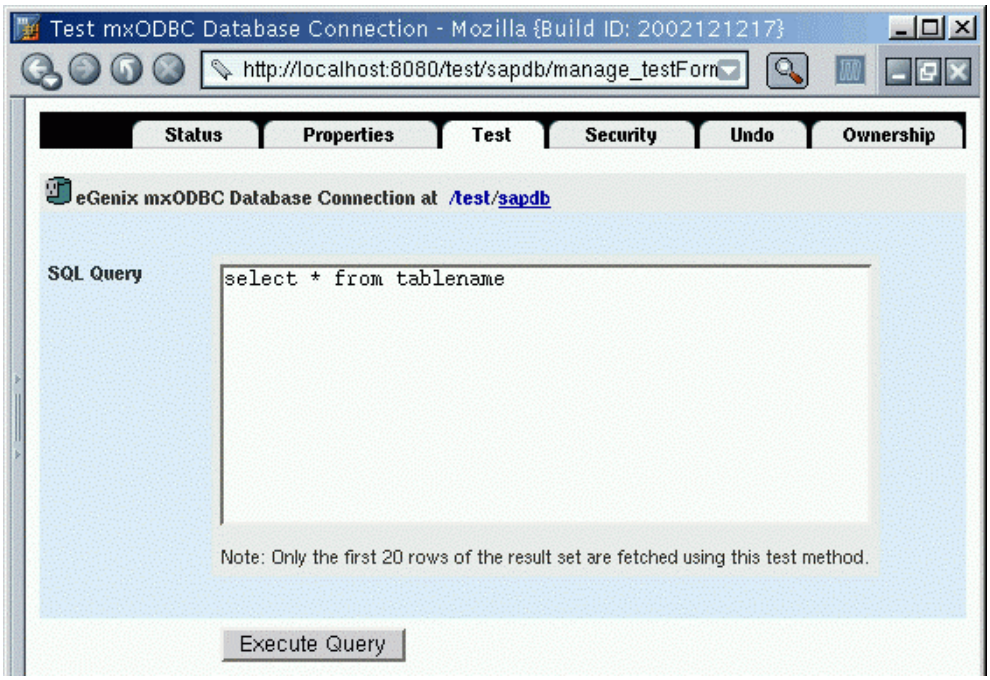


Note that under heavy load, the physical connections may not be freed immediately. This is due to the multi-threaded nature of Zope. Opening and closing the connection again usually helps in these rare cases.

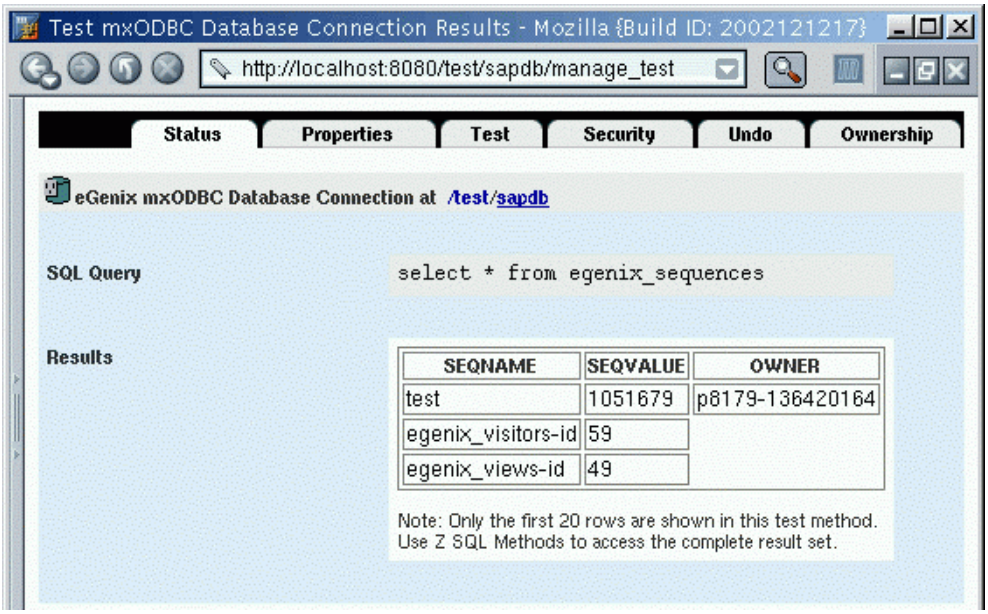
4.2 Testing the Connection

After you have created a connection object, you can test the connection by clicking on the "*Test*" tab of the connection object's management page and entering an SQL query.

The connection has to be open for the test query tab to work.



The results should be displayed as table after pressing the execute button. Note that only the first 20 rows are displayed.



4.3 Copying and Moving Connection Objects

Moving connections is possible for all connections (open and closed) and can be done in the usual way provided by the Zope Management Interface.

Copying is only possible for closed connection objects. This precaution is taken to avoid problems with two connection objects using the same physical connection pool.

4.4 Z SQL Methods

The mxODBC Zope DA is fully compatible with Zope SQL methods.

The creation dialog of Z SQL Methods will automatically pick up the available connections in the current folder and the parent folders.

4.5 Limitations

These limitations are known:

- Depending on you ODBC driver, it may not be possible to safely use more than one physical connection for a logical connection.

  Please consult your ODBC driver documentation for details on executing statements in parallel and its multi-threaded capabilities.

  Modern ODBC drivers usually don't have a problem with this at all, but some of the older drivers or ones which are not implemented in a thread-safe way, may have problems with this.

  The mxODBC Zope DA does protect physical connections using thread locks to make sure that not 100% thread-safe ODBC drivers are still usable, but it is always recommended to use at least ODBC 3.0 compliant ODBC drivers (which have to be implemented thread-safe on platforms supporting threads).

- Limitations of previous mxODBC Zope DA versions have been lifted.

5. Python Interface

The mxODBC Zope DA does not only interface to Z SQL Methods, it also provides an extensive Python API which you can use directly in your Zope applications.

5.1 Object Classes

These APIs are available through the `Products.mxODBCZopeDA.ZopeDA` module.

5.1.1 Class "Products.mxODBCZopeDA.ZopeDA.DatabaseConnection"

mxODBC handled connection to an ODBC data source under Zope Transaction Manager control.

These are the physical connections to the data sources. There can be more than one connection for a logical Zope connection, since these allow pooling connections.

Base class(es): Shared.DC.ZRDB.TM.TM

Attributes:

`.DatabasePackage`

mxODBC database subpackage module to use. Defaults to the platform specific default subpackage.

`.SQL = None`

SQL codes defined by the mxODBC subpackage; this is initialized to `.DatabasePackage.SQL`.

`.connection = None`

mxODBC connection object.

`.connection_string = ''`

Connection string defining the connection.

`.connection_timezone = ''`

Connection timezone.

`.datetime_as_mxdatetime = 0`

Fetch date/time columns as mxDateTime DateTime/DateTimeDelta instances.

`.dont_fix_floats = 0`

Automatically convert scale 0 floats to integers; some drivers like e.g. SAP DB can return scale 0 for floats even though the value does have scale.

`.fetch_last_result_set = 0`

Always fetch the last available result set (rather than the first) ?

`.ignore_max_rows = 0`

Flag to ignore the `max_rows` parameter in all queries.

`.ignore_warnings = 0`

Flag to ignore mxODBC warnings.

`.messages = None`

List of error and warning messages generated by mxODBC.

`.null_as_empty_string = 0`

Convert fetched NULL values to empty strings.

`.record_messages_only = 0`

Flag to tell the error handler to not raise any exceptions for errors and warnings.

`.shortint_as_int = 0`

Fetch short integers (e.g. SMALLINT) as INTEGER. This helps workaround bugs in some drivers which have problems with fetching unsigned shorts.

`.time_as_string = 0`

Fetch TIME values as string instead of as DateTime value (with the date part set to the current day).

`.use_auto_commit = 0`

Run the connection in auto-commit mode (without transactions) ?

`.use_lazy_connect = 0`

Use connect on demand for this connection object ? If turned on, the connection will not automatically reconnect when being loaded from the ZODB.

`.zopetype = None`

Dictionary mapping mxODBC column type codes to Zope ones.

Methods:

`.__init__(connection_string='', subpackage=None, **options)`

Create a DatabaseConnection object for the given connection_string.

subpackage defines the mxODBC subpackage to use for the connection (as string, e.g. 'Windows' maps to mx.ODBC.Windows). If not given, the platform default is used.

Additional keyword parameters may be given to enable work-arounds for specific problems. The following keywords are currently supported:

`ignore_warnings = boolean_flag`

Don't raise exceptions for database warnings, just append them to the .messages list.

`ignore_max_rows = boolean_flag`

Ignore the `max_rows` parameter in all queries. This will cause the queries to always return the complete result set.

`connection_timezone = timezone_string`

Sets the timezone to assume when for fetching date/time data from the connection.

`time_as_string = boolean_flag`

Fetch SQL.TIME values are strings (instead of DateTime values with the date part set to the current day)

`datetime_as_mxdatetime = boolean_flag`

Fetch date/time values as mxDateTime instances.

`shortint_as_int = boolean_flag`

Fetch short integers as integers (instead of as short integers).

`null_as_empty_string = boolean_flag`

Fetch NULL values as empty string.

`dont_fix_floats = boolean_flag`

Default is to automatically convert float values which the database says have scale 0 to integers. Enabling this flag causes floats to be left untouched.

`use_auto_commit = boolean_flag`

Run the connection in auto-commit mode (without transactions). Default is to use transactional mode, but not all ODBC data sources support this.

`use_lazy_connect = boolean_flag`

Use lazy connect for this connection object ? Enabling this option results in the connection being established on demand rather than at object load time. Default is not to use lazy connects.

`fetch_last_result_set = boolean_flag`

When fetching results, try to fetch the last result set rather then the first (default). This can become important when working with stored procedure which often generate multiple result sets.

`.alive()`

Check the connection state.

`.build_query_result(rowset, description)`

Takes a rowset and a description tuple as returned from mxODBC cursors and builds a result tuple as needed by Zope.

description may be None to signal the non-availability of a result set.

This method is used internally by the mxODBC Zope DA, but may also be of use to subclasses you create.

`.callproc(procname, *args)`

Calls the database procedure procname using the given arguments and return a tuple (args, resultsets).

args is either None or the possibly modified list of arguments. resultsets is a list of Zope Results instances holding all available result sets generated by the procedure.

Note that the Zope DA currently does not support multiple result sets, so the list will always just have length 1.

`.close()`

Close the connection.

Errors are ignored.

`.columninfos(qualifier=None, owner=None, table=None, column=None)`

Returns a list of dictionaries describing the columns of the database table that matches the given parameters.

The information returned by this method is more complete than what you get from the .columns() method.

For an explanation of the result set, please see the documentation for mxODBC's cursor.columns() method.

`.columnprivileges(qualifier=None, owner=None, table=None, column=None)`

Returns a list of dictionaries describing the privileges assigned to the database column that match the given parameters.

For an explanation of the result set, please see the documentation for mxODBC's cursor.columnprivileges() method.

`.columns(table_name)`

Returns a list of dictionaries with entries 'Name', 'Type', 'Precision', 'Scale', 'Nullable' ('with Null' or '') for each column in the table table_name.

`.connect()`

Connect to the database.

This method may also be used to reconnect to the database.

`.connected()`

Is the DatabaseConnection currently connected ?

`.errorhandler(connection, cursor, errorclass, errorvalue)`

Default mxODBC error handler.

Note: connection and cursor refer to the mxODBC objects, not the Zope database adapter ones.

The default error handler reports all errors and warnings using exceptions and also records these in self.messages as list of tuples (errorclass, errorvalue).

`.execute(sql, params=(), max_rows=None)`

Process an SQL query sql using the parameters given in the sequence params and return at max max_rows (defaults to all rows) of results back to Zope.

sql has to use the '?' marker as positional parameter marker, e.g. "select * from table where col1=?, col2=?". The first parameter will get bound to the first '?' marker (col1), the second parameter to the second '?' (col2) and so on.

The return value is a tuple (columns, rowset) with columns being a Zope specific list of dictionaries describing the column types and rowset a list of row tuples (in the order given in the columns definition).

Using this form of query interface has the advantage of allowing the ODBC driver to do the escaping of the data passed to the database. The standard .query() interface only works with SQL statements which have the data included verbatim and properly quoted.

Possible values for `max_rows`:

- `0, None`: fetch all rows in the result set
- `< 0`: don't fetch any rows from the result set
- `n`: fetch at most n rows from the result set

Note that `max_rows` is ignored if the option `.ignore_max_rows` is true.

`.executemany(sql, paramsbatch=(), max_rows=None)`

Process an SQL query sql using the parameter batch given in the sequence of sequences paramsbatch and return at max max_rows (defaults to all rows) of results back to Zope.

Each sequence of parameters in paramsbatch is executed against the sql query in an optimized way. This makes it possible to e.g. insert a few hundred rows of data with one call to the database interface.

sql has to use the '?' marker as positional parameter marker, e.g. "select * from table where col1=?, col2=?". The first parameter will get bound to the first '?' marker (col1), the second parameter to the second '?' (col2) and so on.

If there is a result set, the return value is a tuple (columns, rowset) with columns being a Zope specific list of dictionaries describing the column types and rowset a list of row tuples (in the order given in the columns definition).

Using this form of query interface has the advantage of allowing the ODBC driver to do the escaping of the data passed to the database. The standard .query() interface only works with SQL statements which have the data included verbatim and properly quoted.

Possible values for `max_rows`:

- `0, None`: fetch all rows in the result set
- `< 0`: don't fetch any rows from the result set
- `n`: fetch at most n rows from the result set

Note that `max_rows` is ignored if the option `.ignore_max_rows` is true.

```
.foreignkeys(primary_qualifier=None, primary_owner=None,
pimary_table=None, foreign_qualifier=None, foreign_owner=None,
foreign_table=None)
```

Returns a list of dictionaries describing the foreign keys of the database table(s) that match the given parameters.

For an explanation of the result set, please see the documentation for mxODBC's cursor.foreignkeys() method.

```
.gettypeinfo(sqltypecode)
```

Returns a list of dictionaries describing the SQL type code given in sqltypecode.

For an explanation of the result set, please see the documentation for mxODBC's cursor.gettypeinfo() method.

```
.primarykeys(qualifier=None, owner=None, table=None)
```

Returns a list of dictionaries describing the primary keys of the database tables that match the given parameters.

For an explanation of the result set, please see the documentation for mxODBC's cursor.primarykeys() method.

```
.procedurecolumns(qualifier=None, owner=None, procedure=None,
column=None)
```

Returns a list of dictionaries describing the database procedure parameter columns of the database procedures that match the given parameters.

For an explanation of the result set, please see the documentation for mxODBC's cursor.procedurecolumns() method.

```
.procedures(qualifier=None, owner=None, procedure=None)
```

Returns a list of dictionaries describing the database procedures stored in the database which match the given parameters.

For an explanation of the result set, please see the documentation for mxODBC's cursor.procedures() method.

```
.query(sql, max_rows=None)
```

Process an SQL query sql and return at max max_rows (defaults to all rows) of results back to Zope.

The return value is a tuple (columns, rowset) with columns being a Zope specific list of dictionaries describing the column types and rowset a list of row tuples (in the order given in the columns definition).

Multiple SQL statements may be concatenated using '\0' (<dtml-var sql_delimiter> is mapped to '\0' in Z SQL Methods). These will be executed one at a time. Note that only the last statement may generate a result set.

Possible values for `max_rows`:

- `0, None`: fetch all rows in the result set
- `< 0`: don't fetch any rows from the result set
- `n`: fetch at most n rows from the result set

```
.run_cursor_callback(callback, max_rows=None, **kws)
```

Process a cursor callback and return at max. max_rows (defaults to all rows) of the result set and the description list as returned from mxODBC.

The callback is called with the mxODBC cursor as first argument and any additional keyword arguments passed to this method. It should implement the query call, e.g. use .execute() or one of the catalog methods to generate a result set on the cursor.

This method is used internally by the mxODBC Zope DA, but may also be of use to subclasses you create.

Possible values for `max_rows`:

- `0, None`: fetch all rows in the result set
- `< 0`: don't fetch any rows from the result set
- `n`: fetch at most n rows from the result set

Note that `max_rows` is ignored if the option `.ignore_max_rows` is true.

```
.set_errorhandler(errorhandler=None)
```

Set mxODBC error handler to errorhandler.

The errorhandler must be a function accepting the following arguments: connection, cursor, errorclass, errorvalue. connection and cursor refer to the mxODBC objects, not the Zope database adapter ones.

Without argument, the method resets the error handler to the default one defined in the class definition.

The default error handler reports all errors and warnings using exceptions and also records these in self.messages as list of tuples (errorclass, errorvalue).

```
.specialcolumns(qualifier=None, owner=None, table=None,
coltype=None, scope=None, nullable=None)
```

Returns a list of dictionaries describing special columns of the database tables that match the given parameters.

Special columns are e.g. those that qualify as primary keys or row identifier.

For an explanation of the result set, please see the documentation for mxODBC's cursor.specialcolumns() method.

```
.statistics(qualifier=None, owner=None, table=None, unique=None,
accuracy=None)
```

Returns a list of dictionaries providing statistics about the database tables that match the given parameters.

For an explanation of the result set, please see the documentation for mxODBC's cursor.statistics() method.

```
.tableprivileges(qualifier=None, owner=None, table=None)
```

Returns a list of dictionaries describing the privileges assigned to the database tables that match the given parameters.

For an explanation of the result set, please see the documentation for mxODBC's cursor.tableprivileges() method.

```
.tables(qualifier=None, owner=None, name=None, type=None)
```

Returns a list of dictionaries describing the database tables that match the given parameters.

For an explanation of the result set, please see the documentation for mxODBC's cursor.tables() method.

5.1.2 ExtensionClass "Products.mxODBCZopeDA.ZopeDA.ZopeConnection"

mxODBC Zope DA Connection.

Logical Zope connection object. This object manages a configurable number of physical database connections.

Base class(es): Shared.DC.ZRDB.Connection.Connection

Attributes:

```
.DatabaseConnection = Class
"Products.mxODBCZopeDA.ZopeDA.DatabaseConnection"
```

Class to use for the physical database connections. Note that only subclasses of the default class are allowed here, since they are managed in an internal connection pool.

```
._isAnSQLConnection = 1
```

Indicator needed for Z SQL Method compatibility.

```
._v_database_connection = None
```

.DatabaseConnection object.

```
.closed = 1
```

Is this logical connection in a closed state ?

`.connection_string = ''`

Connection string to use.

`.connection_timezone = ''`

Connection timezone to use.

`.database_type = 'mxODBC'`

Database type.

`.icon = 'misc_/mxODBCZopeDA/connection_icon'`

Icon.

`.id = 'eGenix_mxODBC_Database_Connection'`

Default Object ID.

`.ignore_warnings = 0`

Ignore database warnings ?

`.ignore_max_rows = 0`

Ignore `max_rows` query parameter ?

`.implementation_version = ''`

ZopeConnection implementation version. This is used for automatic upgrade of existing mxODBC Zope DA Connections after a software upgrade.

`.license = 'EVALUATION USE CPU License for Evaluation User [#1]'`

License string. This is read directly from the mxODBC license module.

`.manage_close_connection__roles__ = ('Manager',)`

Close connection roles.

`.manage_edit__roles__ = ('Manager',)`

Edit connection roles.

`.manage_main = <DTMLFile instance at 82ef4d8>`

DTMLFile for the management dialog.

`.manage_main__roles__ = ('Manager',)`

Management dialog roles.

`.manage_properties = <DTMLFile instance at 82f7800>`

DTMLFile for the properties dialog.

```
.manage_properties__roles__ = ('Manager',)
```

Properties dialog roles.

```
.manage_testForm = <DTMLFile instance at 8270b58>
```

DTMLFile for the test dialog.

```
.manage_testForm__roles__ = ('Manager',)
```

Test form roles.

```
.manage_test__roles__ = ('Manager',)
```

Test dialog roles.

```
.meta_type = 'eGenix mxODBC Database Connection'
```

Name of the Product object type.

```
.pool_size = 1
```

Pool size (number of physical DatabaseConnections to set up in the pool).

```
.subpackage = None
```

mxODBC subpackage to use for this connection.

```
.title = 'eGenix mxODBC Database Connection'
```

Default title of the object.

Methods:

```
.__init__(id, title, connection_string, connection_check=None,
pool_size=1, subpackage=None, **options)
```

Create a ZopeConnection object managing access to the data source connection_string having the given id and title.

If connection_check is true, the connection is given the open state and kept open by Zope—even across restarts.

pool_size may be given as integer and defines how many physical DatabaseConnection objects should be managed and opened by this ZopeConnection object.

subpackage defines the mxODBC subpackage to use for the connection (as string, e.g. 'Windows' maps to mx.ODBC.Windows). If not given, the platform default is used.

Additional keyword parameters may be given to enable work-arounds for specific problems on the underlying DatabaseConnection object. Please see the documentation for DatabaseConnection for details.

`.__call__(*ignore)`

Return a physical DatabaseConnection object, possibly connecting first if the connection is not marked as closed.

`._canCopy(operation=0)`

Do we allow copying ?

operation is 0 for copying, 1 for moving.

Only non-connected connections are copyable. All connections are movable.

`.close()`

Close the DatabaseConnection(s).

Errors are ignored.

`.connect(connection_string=None)`

Connect to the database using the given connection_string.

This first closes any open physical connections and then opens .pool_size new ones.

`.connected(allow_lazy_connects=1)`

Is the object connected ?

Returns the following states:

- 0 - connection is closed
- 1 - connection to the data source is established
- 2 - connection to the data source is currently not established, but a connection will be made on demand (lazy state)

State 2 is only returned in case allow_lazy_connects is true (default). If allow_lazy_connects is false, then on demand connections are reported as being closed (state 0).

`.connection_dsn()`

Return the DSN name of the current connection.

`.connection_info()`

Return a string giving some details about the connected DBMS and ODBC driver.

`.connection_state()`

Returns the connection state as string:

- 'closed' - connection is closed

- 'connected' - connection to the data source is established
- 'on demand' - connection to the data source is currently not established, but a connection will be made on demand (lazy state)

This method is useful for GUI style status reports.

```
.connection_time()
```

Returns the connection time as Zope DateTime instance or None in case the connection is not established.

```
.current_pool_size()
```

Return the current DatabaseConnection pool size for this kind of ZopeConnection.

```
.database_connection(*ignore)
```

Return a physical DatabaseConnection object, possibly connecting first if the connection is not marked as closed.

```
.edit(title, connection_string, check_connection=1,
connection_timezone='', pool_size=1, subpackage=None,
ignore_warnings=0, time_as_string=0, shortint_as_int=0,
dont_fix_floats=0, use_auto_commit=0, fetch_last_result_set=0,
null_as_empty_string=0, ignore_max_rows=0)
```

(Re)Init the object with the given attributes.

See .__init__() for documentation of the attributes.

Note: Always use keyword parameters for this function, since the API may change between Zope DA versions !

```
.get_connection()
```

Get a working, preferably unused DatabaseConnection object.

If no such connections exist in the pool, the method will try to restablish the connections or open new ones.

```
.lazy_connect()
```

Puts the object into the lazy connected state.

Lazy connected objects look like connected ones, but only establish a real database connection on demand, i.e. when the object is asked for a DatabaseConnection object.

If the object is already connected to the data source, no further action is taken and the state is not changed.

```
.manage_close_connection(REQUEST)
```

Management interface to close the connection.

Note: Always use keyword parameters for this function, since the API may change between Zope DA versions !

```
.manage_edit(title, connection_string, check_connection=0,
connection_timezone='', pool_size=1, subpackage=None,
ignore_warnings=0, time_as_string=0, shortint_as_int=0,
dont_fix_floats=0, use_auto_commit=0, fetch_last_result_set=0,
datetime_as_mxdatetime=0, use_lazy_connect=0,
null_as_empty_string=0, ignore_max_rows=0, REQUEST=None)
```

Edit connection settings.

Note: Always use keyword parameters for this function, since the API may change between Zope DA versions !

```
.manage_open_connection(lazy_connect=0, REQUEST=None)
```

Management interface to open the connection.

Note: Always use keyword parameters for this function, since the API may change between Zope DA versions !

```
.manage_test(query, query_start=None, REQUEST=None)
```

Executes an SQL query on the ZopeConnection and returns the results.

Only the first 20 rows are shown.

Note: Always use keyword parameters for this function, since the API may change between Zope DA versions !

```
.upgrade_object()
```

Upgrade object from a previous mxODBC Zope DA version to the installed one.

5.2 Errors

The interface defines and uses these Python exception objects.

5.2.1 Class "Products.mxODBCZopeDA.ZopeDA.DatabaseConnectionError"

Database connection error.

This exception is raised in case a database connection cannot be established.

Base class(es): exceptions.StandardError

5.2.2 Class "Products.mxODBCZopeDA.ZopeDA.ReplayTransaction"

Replay the current transaction after having rolled back any changes.

Raising this exception will cause Zope to retry the complete transaction. This can be useful in case a database connection was lost.

Base class(es): ZODB.POSException.ConflictError

5.2.3 Class "Products.mxODBCZopeDA.ZopeDA.BadRequest"

Bad Request Error used to signal errors to the user.

Base class(es): exceptions.StandardError

Attributes:

```
.maxlinelength = 68
```

Wrap long lines exceeding this length.

```
.template = '\n    <table cellspacing="0" cellpadding="5"
width="100%%">\n      <th align="left" valign="top"
bgcolor="#DDAAAA" colspan="3">\n        <b><font size="+1"
color="#000000"><font color="#990000">E</font>GENIX<font
color="#990000">.</font>COM\n        mxOD ...
```

HTML template snippet.

Methods:

```
.__init__(title, message, *args)
```

Create and initialize the object

6. Copyrights & Licenses

Please see the LICENSE/COPYRIGHT files in each package's subdirectory for information on copyright, licensing conditions and authorized use of the respective package.

The egenix-mx-base part of the package (e.g. mxTextTools and mxDateTime) is distributed under the *eGenix.com Public License* and the egenix-mx-commercial (e.g. mxODBC) while the mxODBC Zope Database Adapter is distributed under the *eGenix.com Commercial License*.

If in doubt, please check the web-site at http://www.egenix.com or contact licenses@egenix.com for more information on copyright, licensing conditions and authorized use.

eGenix.com Software, Skills and Services GmbH
Pastor-Loeh-Str. 48
D-40764 Langenfeld
Germany

6.1 eGenix.com Commercial License

This is the eGenix.com Commercial License Agreement which covers the mxODBC and mxODBC Zope DA software and documentation.

EGENIX.COM COMMERCIAL LICENSE AGREEMENT

Version 1.1.0

1. Introduction

This "License Agreement" is between eGenix.com Software, Skills and Services GmbH ("eGenix.com"), having an office at Pastor-Loeh-Str. 48, D-40764 Langenfeld, Germany, and the Individual or Organization ("Licensee") accessing and otherwise using this software in source or binary form and its associated documentation ("the Software").

2. Terms and Definitions

The "Software" covered under this License Agreement includes without limitation, all object code, source code, help files, publications, documentation and other programs, products or tools that are included in the official "Software Distribution" available from eGenix.com.

The "Proof of Authorization" for the Software is a written and signed notice from eGenix.com providing evidence of the extent of authorizations the Licensee has acquired to use the Software and of Licensee's eligibility for future upgrade program prices (if announced) and potential special or promotional opportunities. As such, the Proof of Authorization becomes part of this License Agreement.

Installation of the Software ("Installation") refers to the process of unpacking or copying the files included in the Software Distribution to an Installation Target.

"Installation Target" refers to the target of an installation operation. Targets are defined as follows:

1) "CPU" refers to a central processing unit which is able to store and/or execute the Software (a server, personal computer, or other computer-like device) using at most two (2) processors,
2) "Site" refers to a single site of a company,
3) "Corporate" refers to an unlimited number of sites of the company,
4) "Developer CPU" refers to a single CPU used by at most one (1) developer.

When installing the Software on a server CPU for use by other CPUs in a network, Licensee must obtain a License for the server CPU and for all client CPUs attached to the network which will make use of the Software by copying the Software in binary or source form from the server into their CPU memory. If a CPU makes use of more than two (2) processors, Licensee must obtain additional CPU licenses to cover the total number of installed processors. Likewise, if a Developer CPU is used by more than one developer, Licensee must obtain additional Developer CPU licenses to cover the total number of developers using the CPU.

"Commercial Environment" refers to any application environment which is aimed at directly or indirectly generating profit. This includes, without limitation, for-profit organizations, private educational institutions, work as independent contractor, consultant and other profit generating relationships with organizations or individuals. Governments and related agencies or organizations are also regarded as being Commercial Environments.

"Non-Commercial Environments" are all those application environments which do not directly or indirectly generate profit. Public educational

institutions and officially acknowledged private non-profit organizations are regarded as being Non-Commercial Environments in the aforementioned sense.

“Educational Environments“ are all those application environments which directly aim at educating children, pupils or students. This includes, without limitation, class room installations and student server installations which are intended to be used by students for educational purposes. Installations aimed at administrational or organizational purposes are not regarded as Educational Environment.

3. License Grant

Subject to the terms and conditions of this License Agreement, eGenix.com hereby grants Licensee a non-exclusive, world-wide license to

1) use the Software to the extent of authorizations Licensee has acquired and
2) distribute, make and install copies to support the level of use authorized, providing Licensee reproduces this License Agreement and any other legends of ownership on each copy, or partial copy, of the Software.

If Licensee acquires this Software as a program upgrade, Licensee’s authorization to use the Software from which Licensee upgraded is terminated.

Licensee will ensure that anyone who uses the Software does so only in compliance with the terms of this License Agreement.

Licensee may not

1) use, copy, install, compile, modify, or distribute the Software except as provided in this License Agreement;
2) reverse assemble, reverse engineer, reverse compile, or otherwise translate the Software except as specifically permitted by law without the possibility of contractual waiver; or
3) rent, sublicense or lease the Software.

4. Authorizations

The extent of authorization depends on the ownership of a Proof of Authorization for the Software.

Usage of the Software for any other purpose not explicitly covered by this License Agreement or granted by the Proof of Authorization is not permitted and requires the written prior permission from eGenix.com.

5. Modifications

Software modifications may only be distributed in form of patches to the original files contained in the Software Distribution.

The patches must be accompanied by a legend of origin and ownership and a visible message stating that the patches are not original Software delivered by eGenix.com, nor that eGenix.com can be held liable for possible damages related directly or indirectly to the patches if they are applied to the Software.

6. Experimental Code or Features

The Software may include components containing experimental code or features which may be modified substantially before becoming generally available.

These experimental components or features may not be at the level of performance or compatibility of generally available eGenix.com products. eGenix.com does not guarantee that any of the experimental components or features contained in the eGenix.com will ever be made generally available.

7. Expiration and License Control Devices

Components of the Software may contain disabling or license control devices that will prevent them from being used after the expiration of a period of time or on Installation Targets for which no license was obtained.

Licensee will not tamper with these disabling devices or the components. Licensee will take precautions to avoid any loss of data that might result when the components can no longer be used.

8. NO WARRANTY

eGenix.com is making the Software available to Licensee on an “AS IS” basis. SUBJECT TO ANY STATUTORY WARRANTIES WHICH CAN NOT BE EXCLUDED, EGENIX.COM MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, EGENIX.COM MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

9. LIMITATION OF LIABILITY

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL EGENIX.COM BE LIABLE TO LICENSEE OR ANY OTHER USERS OF THE SOFTWARE FOR (I) ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER PECUNIARY LOSS) AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF; OR (II) ANY AMOUNTS IN EXCESS OF THE AGGREGATE AMOUNTS PAID TO EGENIX.COM UNDER THIS LICENSE AGREEMENT DURING THE TWELVE (12) MONTH PERIOD PRECEEDING THE DATE THE CAUSE OF ACTION AROSE.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE EXCLUSION OR LIMITATION MAY NOT APPLY TO LICENSEE.

10. Termination

This License Agreement will automatically terminate upon a material breach of its terms and conditions if not cured within thirty (30) days of written notice by eGenix.com. Upon termination, Licensee shall discontinue use and remove all installed copies of the Software.

11. Indemnification

Licensee hereby agrees to indemnify eGenix.com against and hold harmless eGenix.com from any claims, lawsuits or other losses that arise out of Licensee's breach of any provision of this License Agreement.

12. Third Party Rights

Any software or documentation in source or binary form provided along with the Software that is associated with a separate license agreement is licensed to Licensee under the terms of that license agreement. This License Agreement does not apply to those portions of the Software. Copies of the third party licenses are included in the Software Distribution.

13. High Risk Activities

The Software is not fault-tolerant and is not designed, manufactured or intended for use or resale as on-line control equipment in hazardous

environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines, or weapons systems, in which the failure of the Software, or any software, tool, process, or service that was developed using the Software, could lead directly to death, personal injury, or severe physical or environmental damage ("High Risk Activities").

Accordingly, eGenix.com specifically disclaims any express or implied warranty of fitness for High Risk Activities.

Licensee agree that eGenix.com will not be liable for any claims or damages arising from the use of the Software, or any software, tool, process, or service that was developed using the Software, in such applications.

14. General

Nothing in this License Agreement affects any statutory rights of consumers that cannot be waived or limited by contract.

Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between eGenix.com and Licensee.

If any provision of this License Agreement shall be unlawful, void, or for any reason unenforceable, such provision shall be modified to the extent necessary to render it enforceable without losing its intent, or, if no such modification is possible, be severed from this License Agreement and shall not affect the validity and enforceability of the remaining provisions of this License Agreement.

This License Agreement shall be governed by and interpreted in all respects by the law of Germany, excluding conflict of law provisions. It shall not be governed by the United Nations Convention on Contracts for International Sale of Goods.

This License Agreement does not grant permission to use eGenix.com trademarks or trade names in a trademark sense to endorse or promote products or services of Licensee, or any third party.

The controlling language of this License Agreement is English. If Licensee has received a translation into another language, it has been provided for Licensee's convenience only.

15. Agreement

By downloading, copying, installing or otherwise using the Software, Licensee agrees to be bound by the terms and conditions of this License Agreement.

For question regarding this License Agreement, please write to:

eGenix.com Software, Skills and Services GmbH

Pastor-Loeh-Str. 48

D-40764 Langenfeld

Germany

EGENIX.COM PROOF OF AUTHORIZATION

1 CPU License (Example)

This is an example of a "Proof of Authorization" for a 1 CPU License. These proofs are either wet-signed by the eGenix.com staff or digitally PGP-signed using an official eGenix.com PGP-key.

1. License Grant

eGenix.com Software, Skills and Services GmbH (“eGenix.com”), having an office at Pastor-Loeh-Str. 48, D-40764 Langenfeld, Germany, hereby grants the Individual or Organization (“Licensee”)

Licensee: <name of the licensee>

a non-exclusive, world-wide license to use the software listed below in source or binary form and its associated documentation (“the Software”) under the terms and conditions of this License Agreement and to the extent authorized by this Proof of Authorization.

2. Covered Software

Software Name: <product name>

Software Version: <product version>

(including all patch level releases)

Software Distribution: As officially made available by

eGenix.com on *http://www.egenix.com/*

Operating System: any compatible operating system

3. Authorizations

eGenix.com hereby authorizes Licensee to copy, install, compile, modify and use the Software on the following Installation Targets under the terms of this License Agreement.

Installation Targets: 1 CPU

Use of the Software for any other purpose or redistribution IS NOT PERMITTED BY THIS PROOF OF AUTHORIZATION.

4. Proof

This Proof of Authorization was issued by

<name>, <title>

Langenfeld, <date>

Proof of Authorization Key:

<license key>

7. The mxODBC Interface

The mxODBC Zope DA is built on top of the well-known *mxODBC Python ODBC Extension Interface* which is available for Windows and Unix platforms.

After mxODBC Zope DA installation, the low-level mxODBC interface is available in Zope as package `mx.ODBC` with the platform specific subpackages explained in the next sections.

When using mxODBC directly you should be aware that the transaction mechanism of the low-level mxODBC connections is not tied into the Zope transaction mechanism. The mxODBC Zope DA implements this feature, so you should always access mxODBC through the DA rather than directly when you need transaction safety for Zope extensions.

Please see the mxODBC documentation for details on the low-level aspects of ODBC programming.

7.1 Windows Platform

mxODBC Zope DA for Windows only ships with the `mx.ODBC.Windows` subpackage of mxODBC.

Other database subpackages supported by mxODBC on Windows are not available in the mxODBC Zope DA.

7.2 Unix Platform

mxODBC Zope DA binary packages for Linux and Solaris only ship with the `mx.ODBC.iODBC` and `mx.ODBC.unixODBC` subpackages of mxODBC.

The other database subpackages supported by mxODBC are not available in the mxODBC Zope DA.