eGenix.com™

mxODBC Zope DA

ODBC Database Interface
for Plone and Zope

Version 2.2

Contents

1. Introduction

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

The mxODBC Plone/Zope Database Adapter (Zope DA) gives you a professional quality Plone/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, the Mac OS X ODBC Manager on Mac OS X, or iODBC/unixODBC/DataDirect ODBC managers on Unix platforms.

The mxODBC Zope DA package includes everything you need to install the Plone/Zope database adapter: the latest `egenix-mx-base` and `egenix-mxodbc` 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.

- Fully compatible to **standard Z SQL Methods**.

- **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

- **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.

- **Zero Maintenance Mode of Operation**: connections which are having temporary network problems are automatically reconnected as they become available again.

- **Full Unicode Support**: The Zope DA can communicate with the database using Unicode, 8-bit strings or a mix of the two. It provides automatic conversion as necessary to make it possible to adapt the database or the Zope application to the required needs.

- **Full 64-bit Support**: The underlying mxODBC 3.3 library fully supports 64-bit platforms such as Mac OS X 10.6 (Snow Leopard) and 64-bit Linux systems that use iODBC or unixODBC.

- **Compatible with all recent Zope and Python releases**: Support for Zope 2.12 and later, Python 2.6 and 2.7. If you need support for older Plone/Zope releases, please use version 2.0 or 1.0 of our mxODBC Zope DA.

- **Compatible with Plone 4.0-4.3 and Plone 5.0**: The zc.buildout compatible installation process allows easy integration with your existing Plone 4.0-4.3 and Plone 5.0 installations.

- **Compatible with all major ODBC managers:** mxODBC 3.3 supports the Windows ODBC manager, the Mac OS X ODBC manager and the three most popular ODBC managers on Linux: unixODBC, iODBC and DataDirect.

- **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.

- **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 doing a reconnect without replay).

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

- **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.

- **Native Support for Python data types**: the Zope DA can optionally use Python datetime object, decimal objects in result sets to simplify integration. It also supports Zope's DateTime objects, mxDateTime objects, strings and tuples.

- **Lazy Connect**: the mxODBC Zope DA only connects to the database backends when a connection is actually requested. This results in a better use of resources compared to other Zope DAs.

- Fully compatible to the popular **Znolk SQL Wizard** Product and other similar products relying on the common database schema access methods `.tables()` and `.columns()`.

1.2 Requirements

mxODBC Zope DA needs these environment on Windows, Unix or Mac OS X for successful installation:

Windows

- All Windows platforms starting with Windows 2000 are supported.

- Zope 2.12 or later needs to be installed and working. If you use Plone 4.x or Plone 5.0, you will already have this installed.

- Python 2.6 or 2.7 needs to be installed and working. You normally have Python already installed if you are using Zope 2.12/Plone 4.x or later,

- 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 2.12 or later needs to be installed and working. If you use Plone 4.x or Plone 5.0, you will already have this installed.

- Python 2.6 or 2.7 needs to be installed and working. You normally have Python already installed if you are using Zope 2.12/Plone 4.x or later.

- 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.

Mac OS X

- On Mac OS X 10.4/10.5 Intel and PPC 32-bit are supported. On Mac OS X 10.6 and later, only Intel 64-bit is supported as platform.

- Zope 2.12 or later needs to be installed and working. If you use Plone 4.x or later, you will already have this installed.

- Python 2.6 or 2.7 needs to be installed and working. You normally have Python already installed if you are using Zope 2.12/Plone 4.x or later,

- Mac OS X uses a variant of iODBC as system ODBC manager. On Mac OS X 10.4 and 10.5 this comes pre-installed with the system. On Mac OS X 10.6 and later, the ODBC manager is available from Apple as separate download. Alternatively, you can use the ODBC Manager which is maintained by Actual Technologies. Please use the ODBC manager GUI interfaces to configure the data sources.

- 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 Distribution (`egenix-mx-base`), the eGenix.com mxODBC Distribution (`egenix-mxodbc`) and the mxODBC Zope DA Product in form of binary archive files and Python egg files.

The following sub-sections will guide you through the download and installation process.

See the next section on downloading the software for details.

2.1 Download the Software

2.1.1 Automatic download

The mxODBC Zope DA is normally distributed and installed in form of Python egg archives which are built for automatic download and made available through a special package index on the eGenix.com website.

A separate **manual download is normally not needed**, since the zc.buildout installation and build tools used by Zope 2.12 and later or Plone 4.x/5.0 and later will automatically find and download the software from the eGenix.com website as needed.

Manual installation is no longer supported with mxODBC Zope DA.

2.1.2 Manual download

If you do need to download the egg archives eGenix makes available, e.g. because your server doesn't have Internet access, or is behind a firewall, please read on.

You can download the binary egg archives for your combination of platform, Python version and Unicode variant from the eGenix.com website 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, Mac OS X)

All recent versions of these operating systems are supported.

Python Build Version (2.6, 2.7)

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

```
bin/zopepy -V
```

This will print out the Python version number.

Python Build Architecture (32 bit or 64 bit)

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

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

```
bin/zopepy -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 (if you are running Plone with a different Python interpreter, please replace bin/zopepy with the one you are using):

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

This will either print out "UCS2" or "UCS4".

[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.

Plone's Universal Installer defaults to building a UCS2 Python, so that's what you'll like have installed. Things may be different if you're using a Plone version that came with your Linux distribution.

2.2 Installation in Plone 4.x or Plone 5.0

This section explains the installation of the mxODBC Zope DA in Plone 4.x or Plone 5.0.

Plone 4/5 have standardized on a *zc.buildout*-based approach to software management, which is what we'll use in the following sections.

2.2.1 Before You Start

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

> Please make sure that you **do not have egenix-mx-base or egenix-mxodbc installed separately**, since the installation will not succeed in such a setup.
>
> If you have not installed them manually in your Python installation, also **make sure that you don't have any of these buildout recipes installed:** `collective.recipe.mxodbc`, `collective.recipe.mxbase` or `collective.recipe.mxzopeda`.

Upgrading

zc.buildout will automatically upgrade your mxODBC Zope DA to the latest release.

If you don't want this to happen, add an entry with the exact version number to the `[versions]` section of the buildout.cfg or versions.cfg file, e.g.

```
[versions]
egenix-mxodbc-zopeda = 2.2.0
```

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/mxodbc_zopeda_license.py
2100-8789-0322-0926-2568-6429/mxodbc_zopeda_license.txt
2100-8089-0312-0926-2668-6529/mxodbc_zopeda_license.py
2100-8089-0312-0926-2668-6529/mxodbc_zopeda_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 your Plone instance directory, i.e. the directory with the buildout.cfg file, usually named zinstance/ for stand-alone installations or zeocluster/ for ZEO installations.

> The files will be copied to the right Plone instance directory location via a zc.buildout recipe where Zope can find them, so **you should not remove these license directories**.

2.2.2 Step-by-step Installation Guide

We assume that you have already installed Plone and unzipped the license files to the instance or ZEO cluster directory as explained in the previous section.

Step 1

Determine whether you are using a UCS2 or UCS4 build of Python.

> **Windows users** always need the UCS2 version. Users of the **Plone Universal Installer** will most likely also need the UCS2 version, since Python's default configuration is to build a UCS2 interpreter.

To find out which variant your Python version was compiled with, run the following command (if you are running Plone/Zope with a different Python interpreter, please replace bin/zopepy with the one you are using):

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

This will either print out "UCS2" or "UCS4".

Plone's Universal Installer defaults to building a UCS2 Python, so that's what you'll like have installed. Things may be different if you're using a Plone version that came with your Linux distribution.

Step 2

Open your buildout.cfg file in a text editor and make the following changes, depending on your requirements.

Step 3a: Stand-alone installation

This step explains the configuration for a stand-alone installation. If you are installing mxODBC Zope DA to a ZEO cluster client, please see *step 3b*.

In the buildout.cfg file, please add/adapt the following content:

```
[buildout]
…

### Add eGenix Index to the buildout setup
#
# IMPORTANT: Use the URL
#     https://downloads.egenix.com/python/index/ucs2/
# if your Python version is a UCS build. If you have a UCS4
# build of Python, use the URL
#     https://downloads.egenix.com/python/index/ucs4/
#
find-links =
    …
    https://downloads.egenix.com/python/index/ucs2/

# If your buildout.cfg has an entry allow-hosts, you need
# to add our server to that list as well:
allow-hosts =
    …
    downloads.egenix.com
```

```
### Add eGenix mxODBC Zope DA eggs
#
# ThreadLock and Products.ZSQLMethods are dependencies of the
# egenix-mxodbc-zopeda, but included here for completeness, since
# you have to add versions for these further below.
#
# The new egenix-mxodbc-zopeda-license part takes care of
# automatically installing your license files in the instance.
#
eggs =
    …
    egenix-mxodbc-zopeda
    ThreadLock
    Products.ZSQLMethods

parts =
    …
    egenix-mxodbc-zopeda-license

…

### Install the Zope instance licenses for egenix-mxodbc-zopeda
#
# This part copies the license files you extracted to the
# instance directory to the right directory on the instance's
# Python path.
#
# IMPORTANT: You need to replace ***license-serial*** with the
# directory containing the mxodbc_zopeda_license.py file for
# your instance.
#
[egenix-mxodbc-zopeda-license]
recipe = collective.recipe.template
input = ***license-serial***/mxodbc_zopeda_license.py
output = ${instance:location}/lib/python/mxodbc_zopeda_license.py

### Define versions of packages to be used
#
# buildout will automatically use the latest version it finds
# for building instances. This may not always be what you
# want, so it's usually better to pin down the version you're
# interested in.
#
[versions]
…
egenix-mxodbc-zopeda = 2.2.0
ThreadLock = 2.13.0.1
Products.ZSQLMethods = 2.13.4
collective.recipe.template = 1.9
```

In the above file, you have to make two adjustments:

- Adjust the **URL used in the find-links directive** to use either the ucs2/ or the ucs4/ version of the eGenix PyPI-style distribution index.

- Adjust the **egenix-mxodbc-zopeda version** to the one that you would like to use.

- Replace the *****license-serial***/ path component** with the license directory containing the license for the instance you are configuring. The directories extracted from the license.zip file are usually named after the license serial, e.g. 2100-8789-0322-0926-2568-6429/.

- The **ThreadLock** egg which includes a Python C extension will be downloaded from our indexes as well. We provide binaries for all supported platforms to avoid **the need to have a compiler installed** for the buildout run. Please see our *ThreadLock distribution release announcement* for details.

Step 3b: ZEO client installation

This step explains the configuration for a ZEO cluster client installation. If you are installing mxODBC Zope DA to a stand-alone server, please see *step 3a*.

The following assumes that you have a ZEO cluster with two clients, the default installation of Plone when requesting a ZEO installation. Please adapt as necessary.

In the buildout.cfg file, please add/adapt the following content:

```
[buildout]
…

### Add eGenix Index to the buildout setup
#
# IMPORTANT: Use the URL
#     https://downloads.egenix.com/python/index/ucs2/
# if your Python version is a UCS build. If you have a UCS4
# build of Python, use the URL
#     https://downloads.egenix.com/python/index/ucs4/
#
find-links =
    …
    https://downloads.egenix.com/python/index/ucs2/

# If your buildout.cfg has an entry allow-hosts, you need
# to add our server to that list as well:
allow-hosts =
    …
    downloads.egenix.com

### Add eGenix mxODBC Zope DA eggs to instance and zopepy
#
# ThreadLock and Products.ZSQLMethods are dependencies of the
# egenix-mxodbc-zopeda, but included here for completeness, since
# you have to add versions for these further below.
#
# The new egenix-mxodbc-zopeda-license part takes care of
# automatically installing your license files in the instance.
#
eggs =
    …
    egenix-mxodbc-zopeda
    ThreadLock
    Products.ZSQLMethods
```

```
parts =
    …
    egenix-mxodbc-zopeda-license-client1
    egenix-mxodbc-zopeda-license-client2

…

### Install the Zope instance licenses for egenix-mxodbc-zopeda
#
# This part copies the license files you extracted to the
# instance directory to the right directory on the client
# instance's Python path.

# ZEO Client 1
#
# IMPORTANT: You need to replace ***license-serial-1*** with the
# directory containing the mxodbc_zopeda_license.py file for
# your client1 instance.
#
[egenix-mxodbc-zopeda-license-client1]
recipe = collective.recipe.template
input = ***license-serial-1***/mxodbc_zopeda_license.py
output = ${client1:location}/lib/python/mxodbc_zopeda_license.py

# ZEO Client 2
#
# IMPORTANT: You need to replace ***license-serial-2*** with the
# directory containing the mxodbc_zopeda_license.py file for
# your client1 instance.
#
[egenix-mxodbc-zopeda-license-client2]
recipe = collective.recipe.template
input = ***license-serial-2***/mxodbc_zopeda_license.py
output = ${client2:location}/lib/python/mxodbc_zopeda_license.py

### Define versions of packages to be used
#
# buildout will automatically use the latest version it finds
# for building instances. This may not always be what you
# want, so it's usually better to pin down the version you're
# interested in.
#
[versions]
…
egenix-mxodbc-zopeda = 2.2.0
ThreadLock = 2.13.0.1
Products.ZSQLMethods = 2.13.4
collective.recipe.template = 1.9
```

In the above file, you have to make two adjustments:

- Adjust the **URL used in the find-links directive** to use either the ucs2/ or the ucs4/ version of the eGenix PyPI-style distribution index.

- Adjust the **egenix-mxodbc-zopeda version** to the one that you would like to use.

- Replace the *****license-serial-1***/ and ***license-serial-2***/ path components** with the respective license directory containing the license for the client instance you are configuring. The directories extracted from the license.zip file are usually named after the license serial, e.g. 2100-8789-0322-0926-2568-6429/.

- The **ThreadLock** egg which includes a Python C extension will be downloaded from our indexes as well. We provide binaries for all supported platforms to avoid **the need to have a compiler installed** for the buildout run. Please see our *ThreadLock distribution release announcement* for details.

Step 4

Run buildout in the instance directory:

```
./bin/buildout
```

This will rebuild your instance using the newly added eGenix mxODBC Zope DA product.

Step 5

To complete the installation, restart Plone/Zope. This will then automatically register the eGenix mxODBC Zope DA.

You can then add it as new connection objects to folders in your Plone installation using the Zope Management Interface (ZMI) and use the connections in external methods.

2.3 Installation in Zope 2.12 or 2.13

Since Zope 2.12 and 2.13 have standardized on a *zc.buildout*-based approach to software management just like Plone, you can follow the same instructions as for Plone in order to install the mxODBC Zope DA. Please see section 2.2 Installation in Plone 4.x or Plone 5.0.

3. Configuration

The configuration of access to a database involves two steps:

1. Configuration of the database as ODBC data source
2. 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 to all platforms.

Connection Pooling by the ODBC Manager

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 side-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.

Platform Default ODBC Manager

The *Platform Default* ODBC manager on Windows is always the Windows ODBC manager.

On 64-bit Windows platforms, Windows comes with two versions of the Windows ODBC manager: a 32-bit version and a 64-bit version. The 32-bit version of mxODBC Zope DA will choose the 32-bit one, the 64-bit version of mxODBC Zope DA the 64-bit 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 unixODBC/iODBC/DataDirect 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 websites of the ODBC managers:

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

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

DataDirect - *http://www.datadirect.com/*

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 ODBC managers installed on the installation machine for the mxODBC Zope DA to work.

The Zope DA will log messages at Zope startup time to the <Zope Instance Directory>/log/events.log which informs you about the successfully added interfaces for unixODBC, iODBC and/or DataDirect. 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.

Platform Default ODBC Manager

The *Platform Default* ODBC manager depends on which ODBC manager mxODBC Zope DA finds during startup. It select the first one found from the above given list, i.e. unixODBC, iODBC, DataDirect.

Note that in mxODBC Zope DA versions 2.0 and earlier, the order was iODBC, unixODBC. This was changed in 2.1, since the unixODBC ODBC manager is more popular than the iODBC one and provides better Unicode support.

3.1.4 Mac OS X Platform

On Mac OS X, please configure the ODBC manager through the standard system *ODBC Administrator*. Open the finder and navigate to Applications / Utilities / ODBC Administrator.

Internally, the ODBC Administrator builds upon the open-source ODBC manager **iODBC**, so the comments related to iODBC also apply to the Mac OS X ODBC manager.

If you are running Mac OS X 10.6 or later and don't have the ODBC Administrator installed, you can *download and install it from Apple*.

Please consult the Mac OS X help and your database/ODBC driver documentation for details on how to setup data sources in the Mac OS X ODBC Administrator.

> If you are running on Mac OS X 10.6 and have **problems finding the data sources** configured with the ODBC Administrator in the mxODBC Zope DA data source list or connecting to them, please see *this Mac Dev Center article* for a fix.

Platform Default ODBC Manager

The *Platform Default* ODBC manager depends on which ODBC manager mxODBC Zope DA finds during startup. It select the first one found from the above given list, i.e. unixODBC, iODBC.

Unless you have installed the unixODBC ODBC manager by hand or via Mac Ports and configured an appropriate linker setup, this will select the iODBC ODBC manager as default.

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 ODBC Managers iODBC, unixODBC and DataDirect

On Unix the mxODBC Zope DA uses an already installed iODBC, unixODBC or DataDirect 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, unixODBC or DataDirect ODBC 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.

Note: **unixODBC changed their ABI from 2.2.12 to 2.2.13**, but kept the library version at version 1 (libodbc.so.1). In 2.3.0 they then changed the library version to version 2 (libodbc.so.2). Some Linux distributions kept the old version name (e.g. Ubuntu). mxODBC Zope DA 2.2 and later are compiled against version 2 of the library, so if you are getting **linker errors mentioning not finding libodbc.so.2**, you should consider adding a symlink from libodbc.so.2 to libodbc.so.1, provided that your installed unixODBC is version 2.12.13 or later.

3.2.3 MS SQL Server Native Client for Windows

On Windows, Microsoft provides a free, robust and full featured ODBC driver for MS SQL Server.

We recommend using this driver when accessing SQL Server from Windows systems.

3.2.4 MS SQL Server Native Client for Linux

This is a free ODBC driver from Microsoft which was ported from the existing mature SQL Server Native Client driver version 11 on Windows. It is currently only available for 64-bit Linux variants.

The driver is more robust than the FreeTDS ODBC driver and provides better Unicode support, but also has the same issues as the SQL Server Native Client driver on Windows.

Please see the *mxODBC documentation* for details on setting up the driver.

3.2.5 FreeTDS ODBC Driver (access MS SQL Server from Unix)

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.

Please note that the driver's current versions (0.9x) still lack a few ODBC features which you may need for production work. Unicode support was added just recently in version 0.91.

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 standard 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 Microsoft, 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.

3.2.6 Oracle Instant Client ODBC Driver

Oracle provides a client driver set called "*Oracle Instant Client*", which also includes a good quality ODBC driver for accessing Oracle databases.

We recommend always using the latest version of this driver, since Oracle frequently updates it. The latest versions usually also provide full support for most older Oracle database versions, so you can e.g. use a version 12 driver with an Oracle 10g database.

Some additional notes for this driver:

- On AIX 32-bit, we have found problems with the ODBC driver when trying to open multiple connections at once. Try to use the "*Use Connection Serialization*" option to work around this issue. The AIX 64-bit version of the driver does not seem to have this problem.

3.2.7 IBM DB2 ODBC Driver

The DB2 ODBC Driver for Windows 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.

3.2.8 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.

3.2.9 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.

3.2.10 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.

3.2.11 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*.

3.2.12 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.

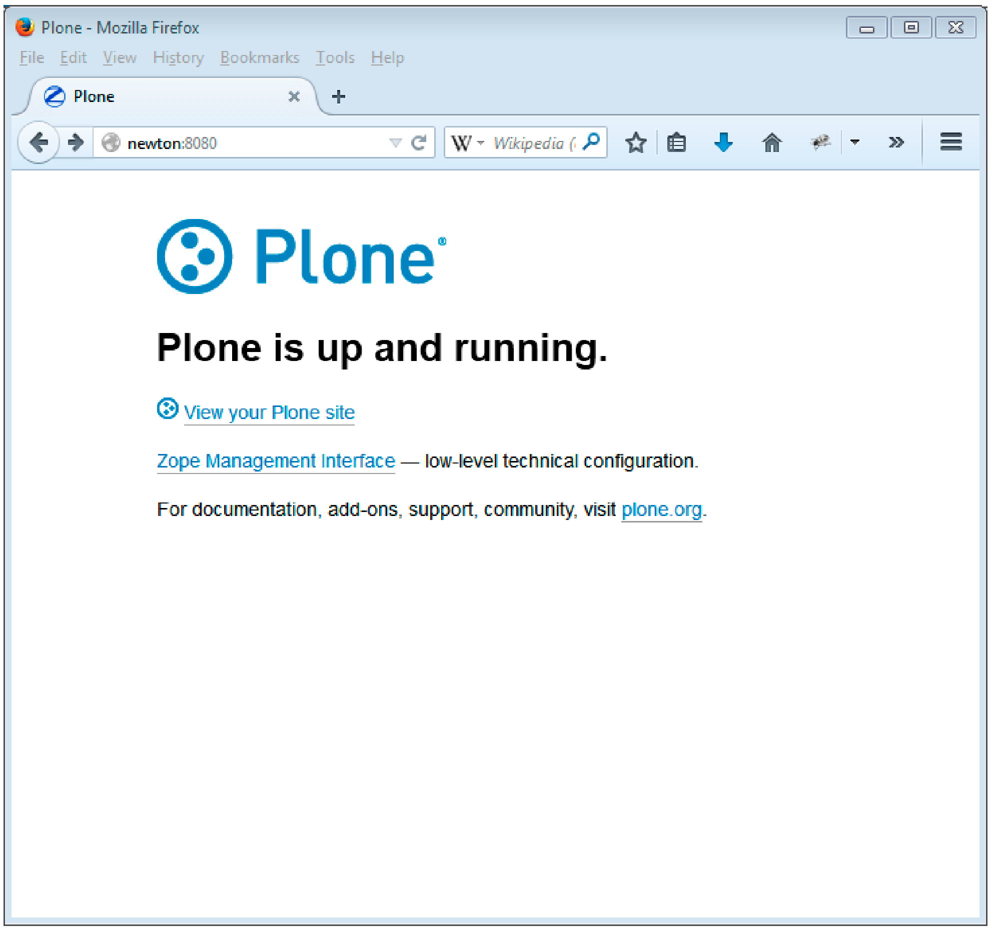
Per default, the mxODBC Zope DA fetches and returns the first available result set from the ODBC driver, so this may not be the one you actually want from the stored procedure.

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.



Plone is up and running.

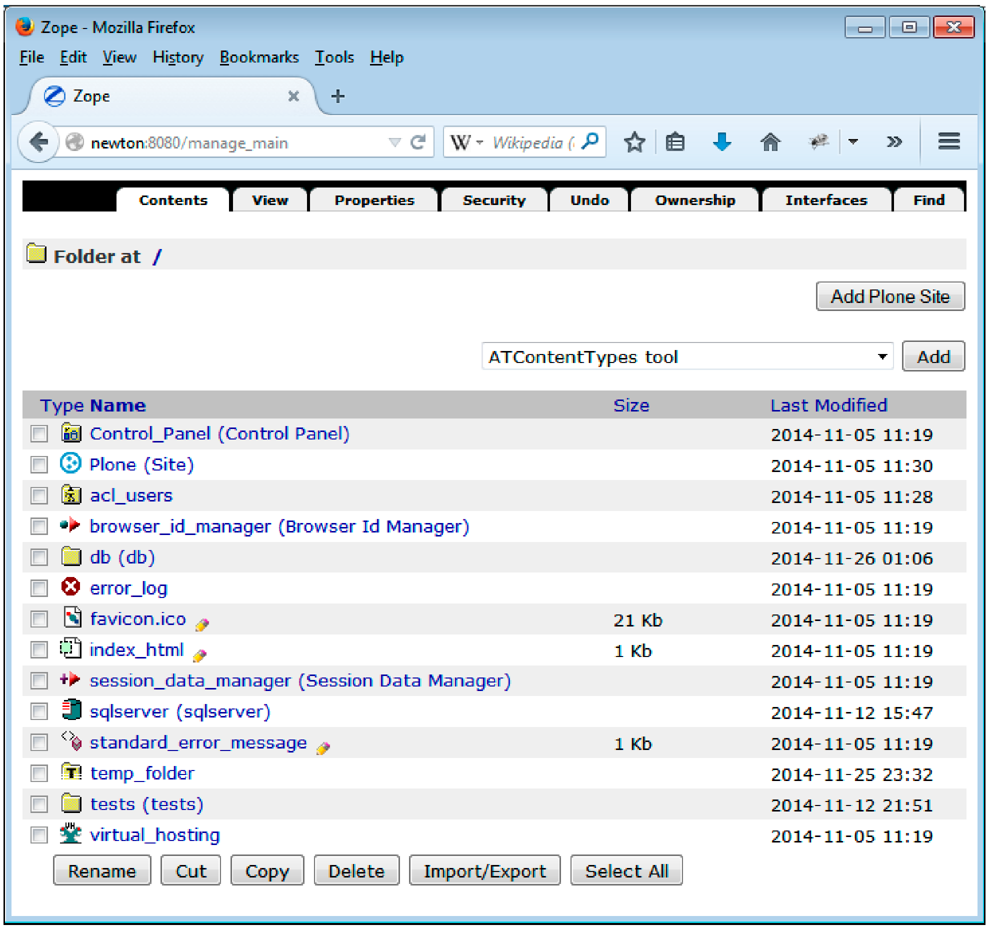
View your Plone site

Zope Management Interface — low-level technical configuration.

For documentation, add-ons, support, community, visit plone.org.

In order to add connection objects in Plone you have to enter the Zope Management Interface (ZMI).

Clicking on the link will open up a window with the ZMI.


Zope - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Zope
newton:8080/manage_main
Wikipedia
Contents View Properties Security Undo Ownership Interfaces Find
Folder at /
Add Plone Site
ATContentTypes tool
Add
Rename Cut Copy Delete Import/Export Select All

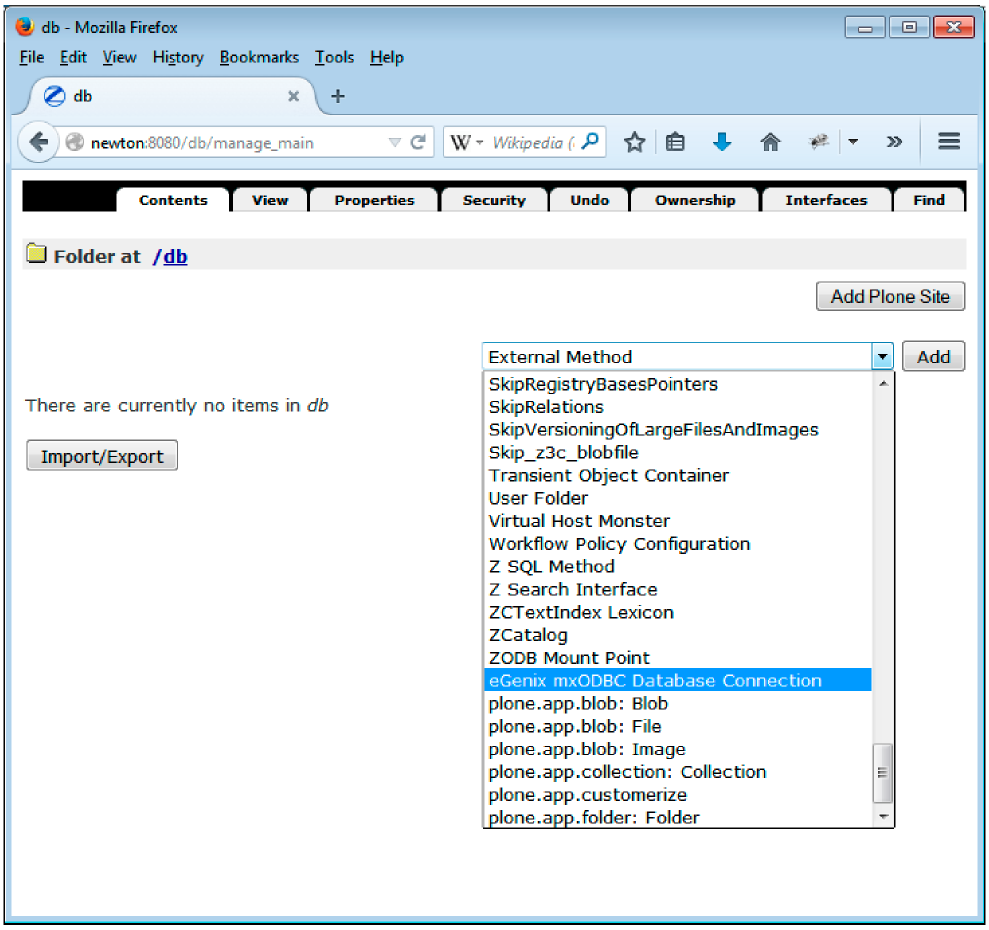
Type Name	Size	Last Modified
Control_Panel (Control Panel)		2014-11-05 11:19
Plone (Site)		2014-11-05 11:30
acl_users		2014-11-05 11:28
browser_id_manager (Browser Id Manager)		2014-11-05 11:19
db (db)		2014-11-26 01:06
error_log		2014-11-05 11:19
favicon.ico	21 Kb	2014-11-05 11:19
index_html	1 Kb	2014-11-05 11:19
session_data_manager (Session Data Manager)		2014-11-05 11:19
sqlserver (sqlserver)		2014-11-12 15:47
standard_error_message	1 Kb	2014-11-05 11:19
temp_folder		2014-11-25 23:32
tests (tests)		2014-11-12 21:51
virtual_hosting		2014-11-05 11:19

It is usually best to create a folder for the connection objects, but you can also create the connection objects at the root level.

Please note that most of the settings you make during the creation process can be adjusted after object creation in the *Properties* dialog, so you don't have to know all the details in advance.

3.3.1 Adding mxODBC Zope DA Connection Objects

After installation and restart, the “Add Object” list box in the Zope Management Interface (ZMI) should show an entry “eGenix mxODBC Zope Database Connection”.

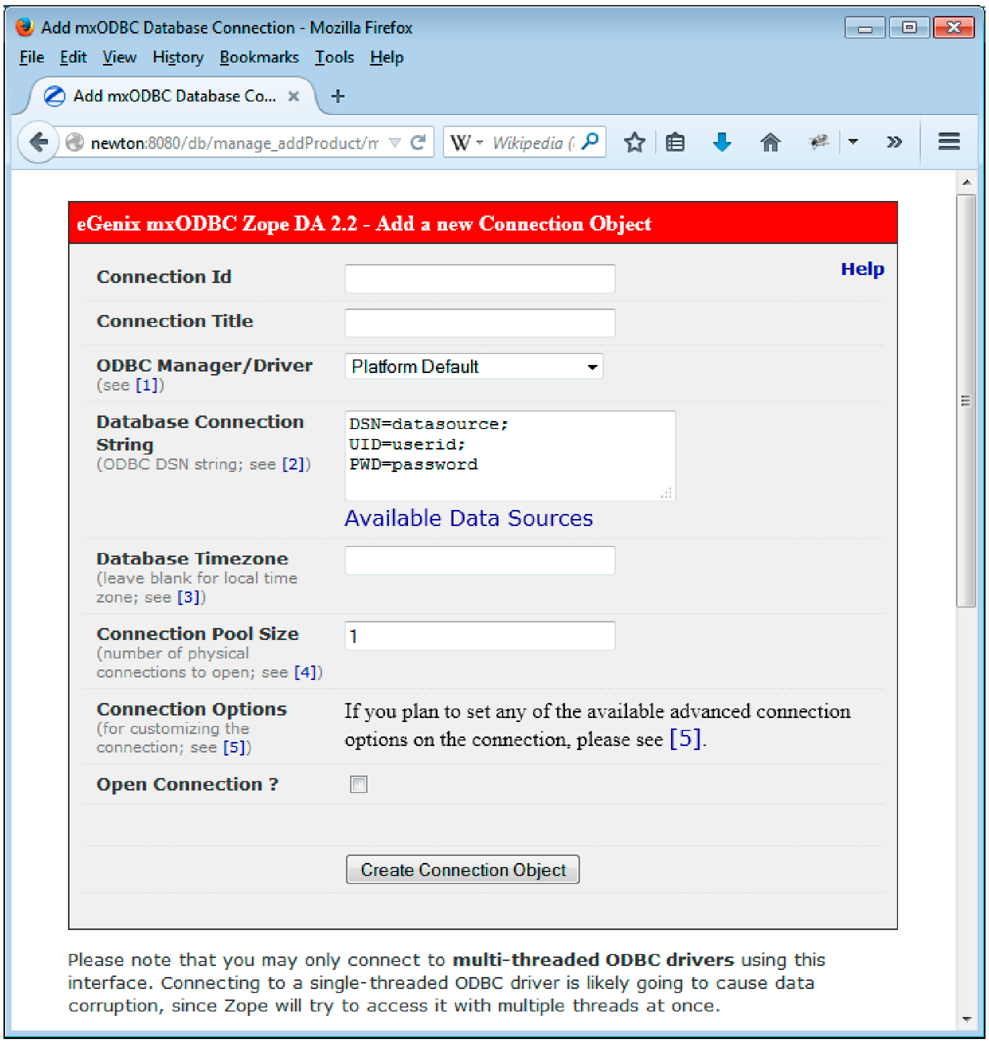


There are currently no items in *db*

```
SkipRegistryBasesPointers
SkipRelations
SkipVersioningOfLargeFilesAndImages
Skip_z3c_blobfile
Transient Object Container
User Folder
Virtual Host Monster
Workflow Policy Configuration
Z SQL Method
Z Search Interface
ZCTextIndex Lexicon
ZCatalog
ZODB Mount Point
eGenix mxODBC Database Connection
plone.app.blob: Blob
plone.app.blob: File
plone.app.blob: Image
plone.app.collection: Collection
plone.app.customerize
plone.app.folder: Folder
```

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.


Add mxODBC Database Connection - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Add mxODBC Database Co...
newton:8080/db/manage_addProduct/m
Wikipedia

eGenix mxODBC Zope DA 2.2 - Add a new Connection Object

Connection Id		Help
Connection Title		
ODBC Manager/Driver (see [1])	Platform Default	

Database Connection String (ODBC DSN string; see [2])	DSN=datasource;
	UID=userid;
	PWD=password
	Available Data Sources	

Database Timezone (leave blank for local time zone; see [3])		
Connection Pool Size (number of physical connections to open; see [4])	1	
Connection Options (for customizing the connection; see [5])	If you plan to set any of the available advanced connection options on the connection, please see [5].	
Open Connection ?	☐	
	Create Connection Object	

Please note that you may only connect to **multi-threaded ODBC drivers** using this interface. Connecting to a single-threaded ODBC driver is likely going to cause data corruption, since Zope will try to access it with multiple threads at once.

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. Please see section 3.1 ODBC Data Source Configuration for more information on how the Platform Default manager is chosen.

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

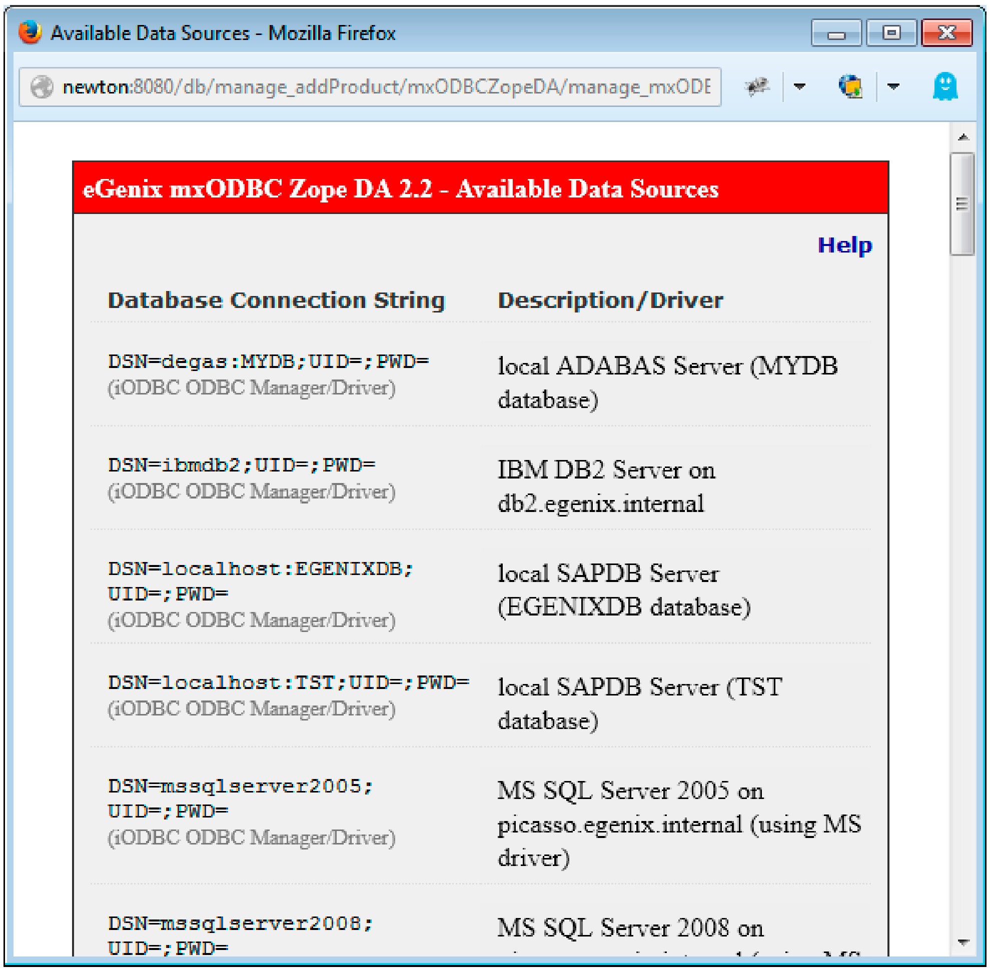
The connection string has to be formatted according to the ODBC DSN standard:

```
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 Data Sources" 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.


Available Data Sources - Mozilla Firefox
newton:8080/db/manage_addProduct/mxODBCZopeDA/manage_mxODBC
eGenix mxODBC Zope DA 2.2 - Available Data Sources
Help

Database Connection String	Description/Driver
`DSN=degas:MYDB;UID=;PWD=` (iODBC ODBC Manager/Driver)	local ADABAS Server (MYDB database)
`DSN=ibmdb2;UID=;PWD=` (iODBC ODBC Manager/Driver)	IBM DB2 Server on db2.egenix.internal
`DSN=localhost:EGENIXDB;UID=;PWD=` (iODBC ODBC Manager/Driver)	local SAPDB Server (EGENIXDB database)
`DSN=localhost:TST;UID=;PWD=` (iODBC ODBC Manager/Driver)	local SAPDB Server (TST database)
`DSN=mssqlserver2005;UID=;PWD=` (iODBC ODBC Manager/Driver)	MS SQL Server 2005 on picasso.egenix.internal (using MS driver)
`DSN=mssqlserver2008;UID=;PWD=`	MS SQL Server 2008 on

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.

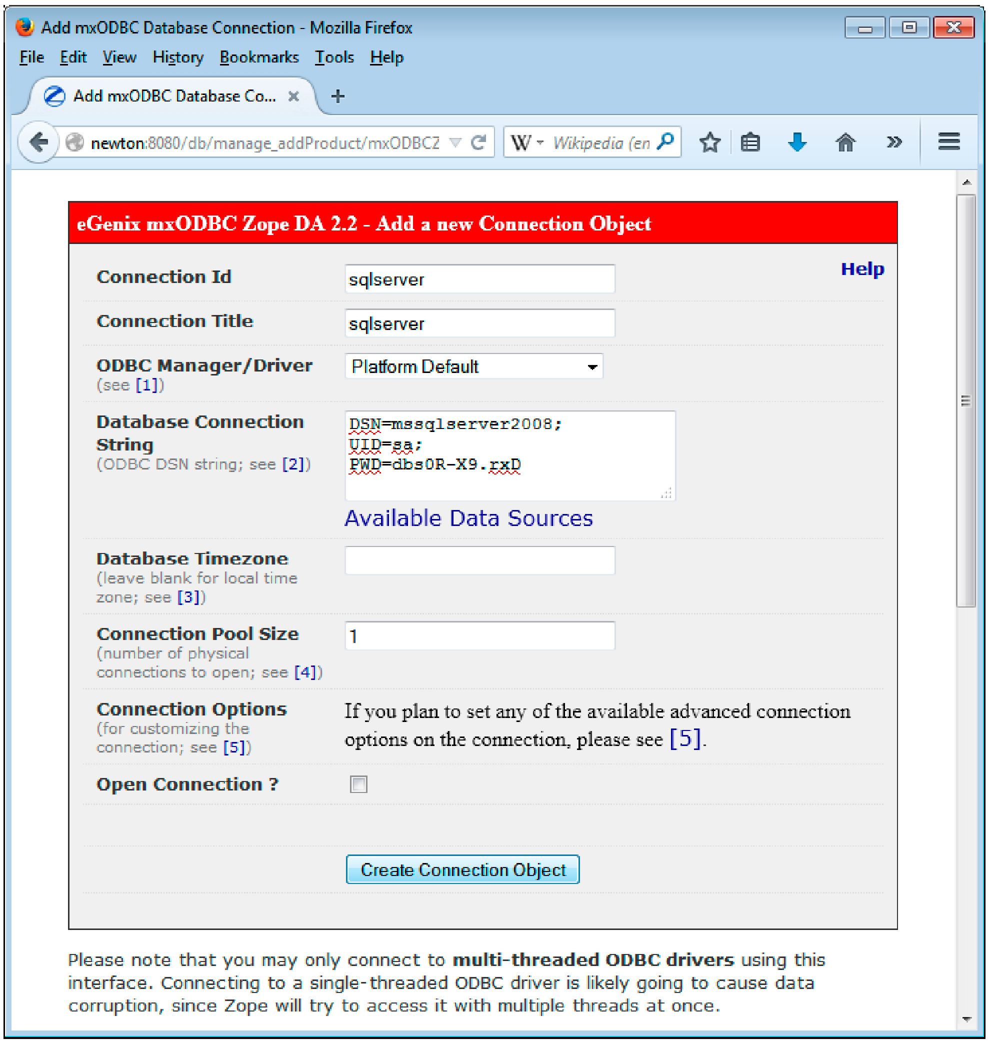
3.3.7 Open Connection

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 open/closed 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

After you have entered the object id, title and details about the connection, you are ready to create the connection.


Add mxODBC Database Connection - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Add mxODBC Database Co...
newton:8080/db/manage_addProduct/mxODBCZ
Wikipedia (en

eGenix mxODBC Zope DA 2.2 - Add a new Connection Object

Help

Connection Id	sqlserver
Connection Title	sqlserver
ODBC Manager/Driver (see [1])	Platform Default

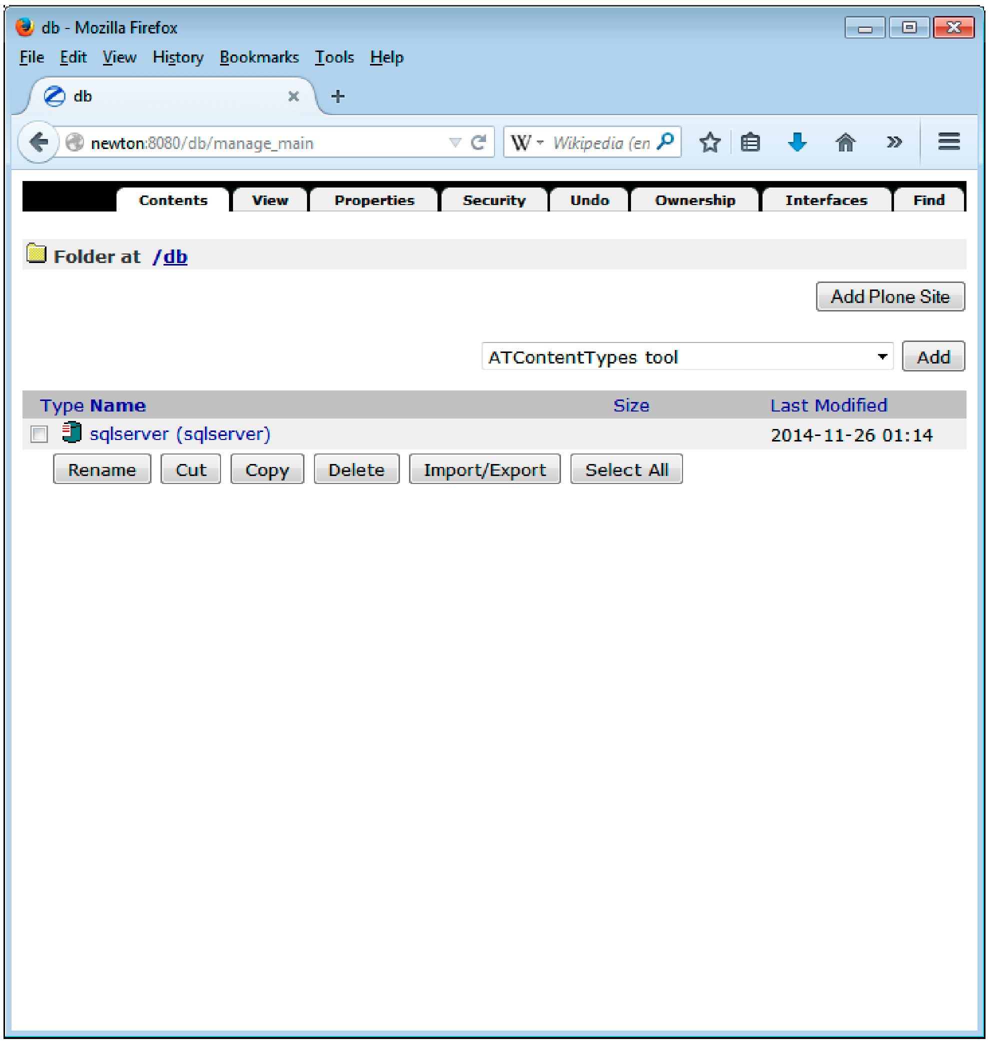
Database Connection String (ODBC DSN string; see [2])	DSN=mssqlserver2008;
	UID=sa;
	PWD=dbs0R-X9.rxD
	Available Data Sources

Database Timezone (leave blank for local time zone; see [3])	
Connection Pool Size (number of physical connections to open; see [4])	1
Connection Options (for customizing the connection; see [5])	If you plan to set any of the available advanced connection options on the connection, please see [5].
Open Connection ?	☐
	Create Connection Object

Please note that you may only connect to **multi-threaded ODBC drivers** using this interface. Connecting to a single-threaded ODBC driver is likely going to cause data corruption, since Zope will try to access it with multiple threads at once.

Hitting the Create Connection Object 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 *sqlserver* in the folder.

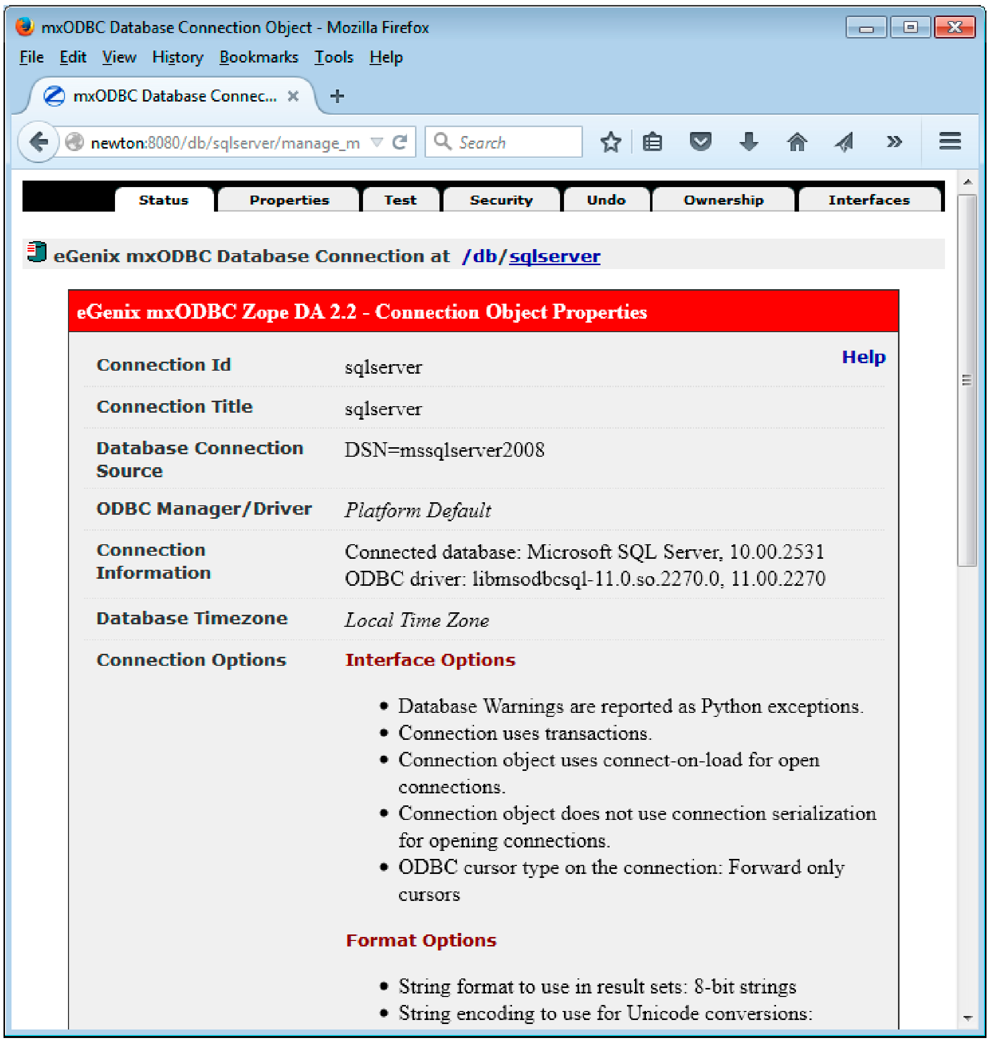

db - Mozilla Firefox
File Edit View History Bookmarks Tools Help
db
newton:8080/db/manage_main
Wikipedia (en
Contents View Properties Security Undo Ownership Interfaces Find
Add Plone Site
ATContentTypes tool
Add
Rename Cut Copy Delete Import/Export Select All

Folder at /db

Type Name	Size	Last Modified
sqlserver (sqlserver)		2014-11-26 01:14

3.3.9 Connection Status

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


mxODBC Database Connection Object - Mozilla Firefox
File Edit View History Bookmarks Tools Help
mxODBC Database Connec...
newton:8080/db/sqlserver/manage_m
Search
Status
Properties
Test
Security
Undo
Ownership
Interfaces
Help

eGenix mxODBC Database Connection at /db/sqlserver

eGenix mxODBC Zope DA 2.2 - Connection Object Properties

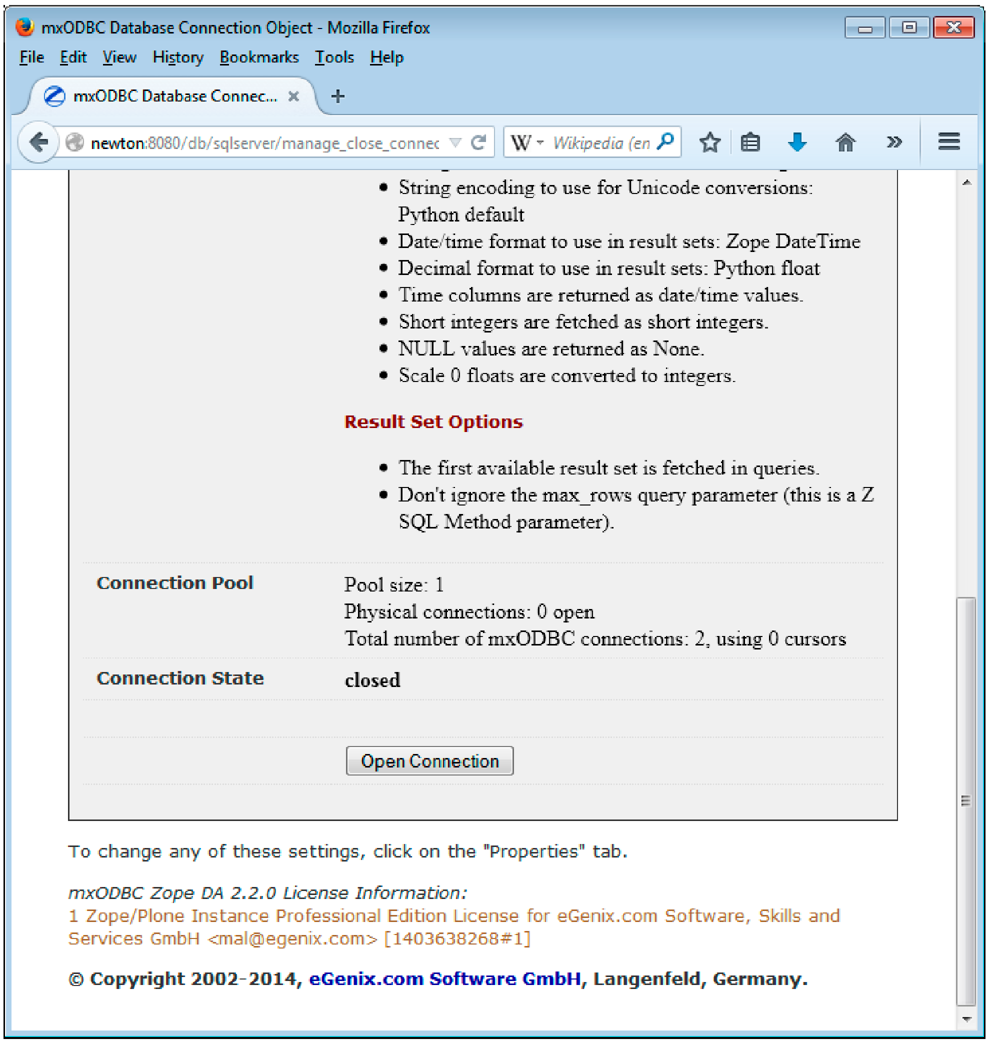
Connection Id	sqlserver
Connection Title	sqlserver
Database Connection Source	DSN=mssqlserver2008
ODBC Manager/Driver	*Platform Default*
Connection Information	Connected database: Microsoft SQL Server, 10.00.2531 ODBC driver: libmsodbcsql-11.0.so.2270.0, 11.00.2270
Database Timezone	*Local Time Zone*
Connection Options	**Interface Options**

- Database Warnings are reported as Python exceptions.
- Connection uses transactions.
- Connection object uses connect-on-load for open connections.
- Connection object does not use connection serialization for opening connections.
- ODBC cursor type on the connection: Forward only cursors

Format Options

- String format to use in result sets: 8-bit strings
- String encoding to use for Unicode conversions:

This shows the connection object details as well as the currently active connection options and the state of the connection pool associated with this connection object.


mxODBC Database Connection Object - Mozilla Firefox
File Edit View History Bookmarks Tools Help
mxODBC Database Connec...
newton:8080/db/sqlserver/manage_close_connec
Wikipedia (en

- String encoding to use for Unicode conversions: Python default
- Date/time format to use in result sets: Zope DateTime
- Decimal format to use in result sets: Python float
- Time columns are returned as date/time values.
- Short integers are fetched as short integers.
- NULL values are returned as None.
- Scale 0 floats are converted to integers.

Result Set Options

- The first available result set is fetched in queries.
- Don't ignore the max_rows query parameter (this is a Z SQL Method parameter).

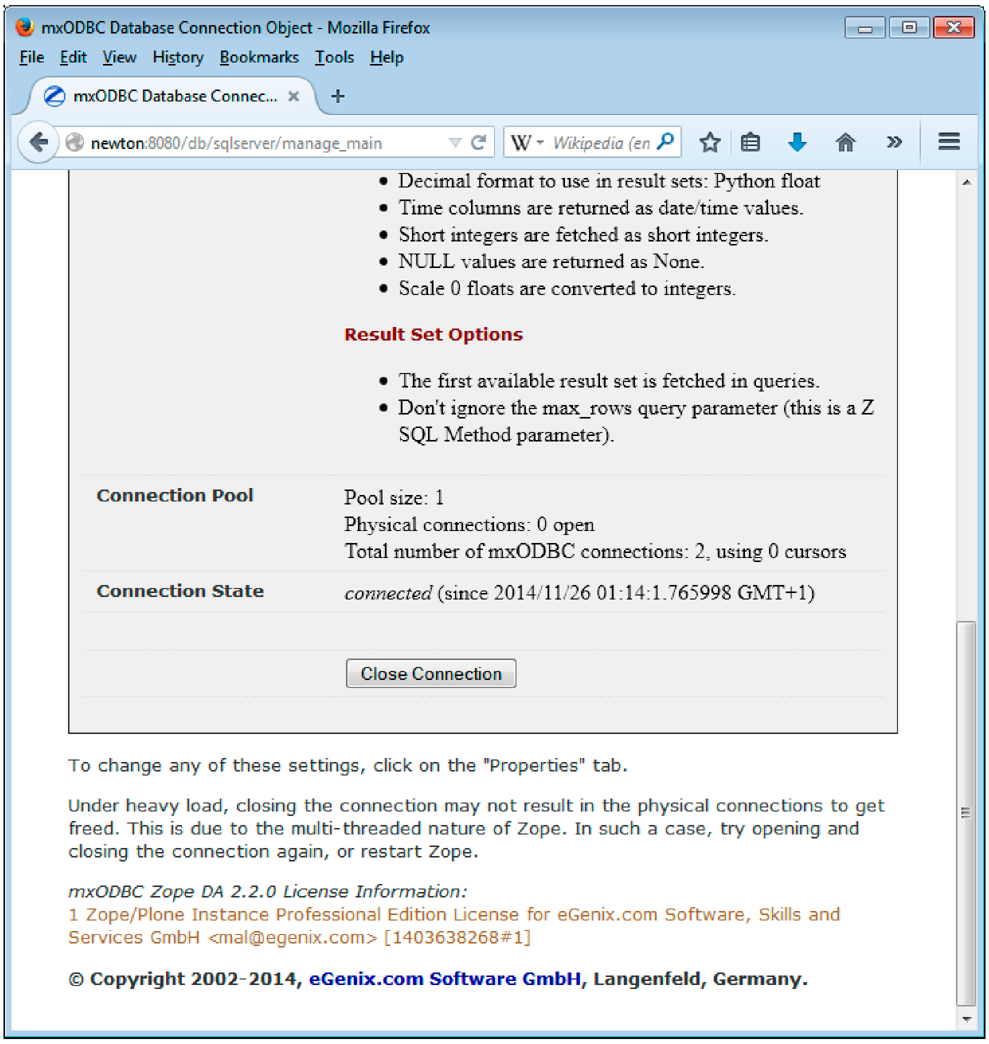
Connection Pool	Pool size: 1 Physical connections: 0 open Total number of mxODBC connections: 2, using 0 cursors
Connection State	**closed**
	Open Connection

To change any of these settings, click on the "Properties" tab.

mxODBC Zope DA 2.2.0 License Information:
1 Zope/Plone Instance Professional Edition License for eGenix.com Software, Skills and Services GmbH <mal@egenix.com> [1403638268#1]

© Copyright 2002-2014, eGenix.com Software GmbH, Langenfeld, Germany.

If the connection object is currently in the closed state, you can open the connection by clicking on the Open Connection button.


mxODBC Database Connection Object - Mozilla Firefox
File Edit View History Bookmarks Tools Help
mxODBC Database Connec...
newton:8080/db/sqlserver/manage_main
Wikipedia (en

- Decimal format to use in result sets: Python float
- Time columns are returned as date/time values.
- Short integers are fetched as short integers.
- NULL values are returned as None.
- Scale 0 floats are converted to integers.

Result Set Options

- The first available result set is fetched in queries.
- Don't ignore the max_rows query parameter (this is a Z SQL Method parameter).

Connection Pool	Pool size: 1 Physical connections: 0 open Total number of mxODBC connections: 2, using 0 cursors
Connection State	*connected* (since 2014/11/26 01:14:1.765998 GMT+1)
	Close Connection

To change any of these settings, click on the "Properties" tab.

Under heavy load, closing the connection may not result in the physical connections to get freed. This is due to the multi-threaded nature of Zope. In such a case, try opening and closing the connection again, or restart Zope.

mxODBC Zope DA 2.2.0 License Information:
1 Zope/Plone Instance Professional Edition License for eGenix.com Software, Skills and Services GmbH <mal@egenix.com> [1403638268#1]

© Copyright 2002-2014, eGenix.com Software GmbH, Langenfeld, Germany.

Likewise, if the connection is open, clicking on Close Connection will close the connection.

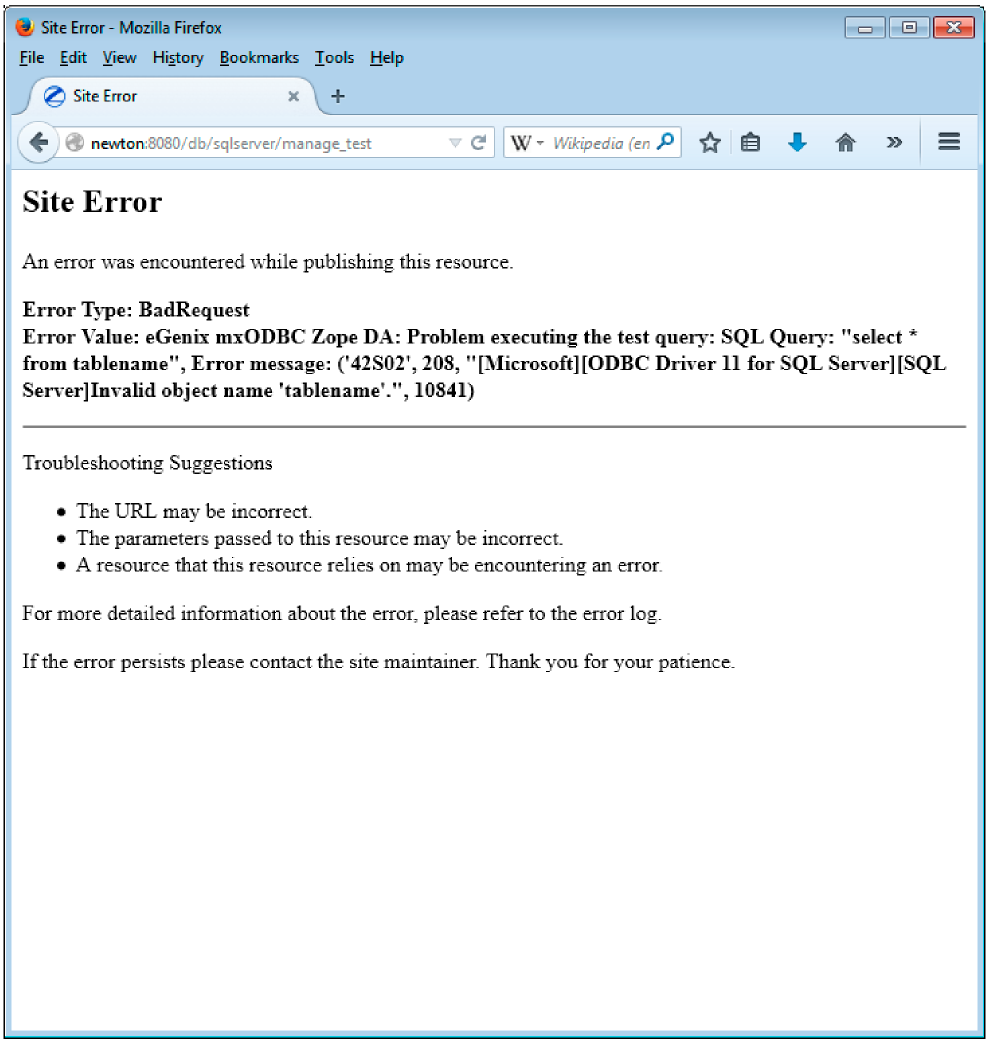
3.3.10 Open/closed state of connection objects

The closed/open state of the connection object is stored persistently in the connection object, meaning that Zope will open the database connection when loading an open connection object or load the closed connection object without opening a database connection.

In case of a failure to connect an open database connection upon loading the connection object, the object will enter a special "**open, but not connected**" state. In this mode, the mxODBC Zope DA will automatically try to reconnect to the database every time the object is loaded, allowing the application to continue working without restart after the database connection problem has been resolved. Any queries issued on a connection object in the "open, but not connected" will fail with an error.

3.3.11 Error messages

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


Site Error - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Site Error
newton:8080/db/sqlserver/manage_test
Wikipedia (en

Site Error

An error was encountered while publishing this resource.

Error Type: BadRequest
Error Value: eGenix mxODBC Zope DA: Problem executing the test query: SQL Query: "select * from tablename", Error message: ('42S02', 208, "[Microsoft][ODBC Driver 11 for SQL Server][SQL Server]Invalid object name 'tablename'.", 10841)

Troubleshooting Suggestions

- The URL may be incorrect.
- The parameters passed to this resource may be incorrect.
- A resource that this resource relies on may be encountering an error.

For more detailed information about the error, please refer to the error log.

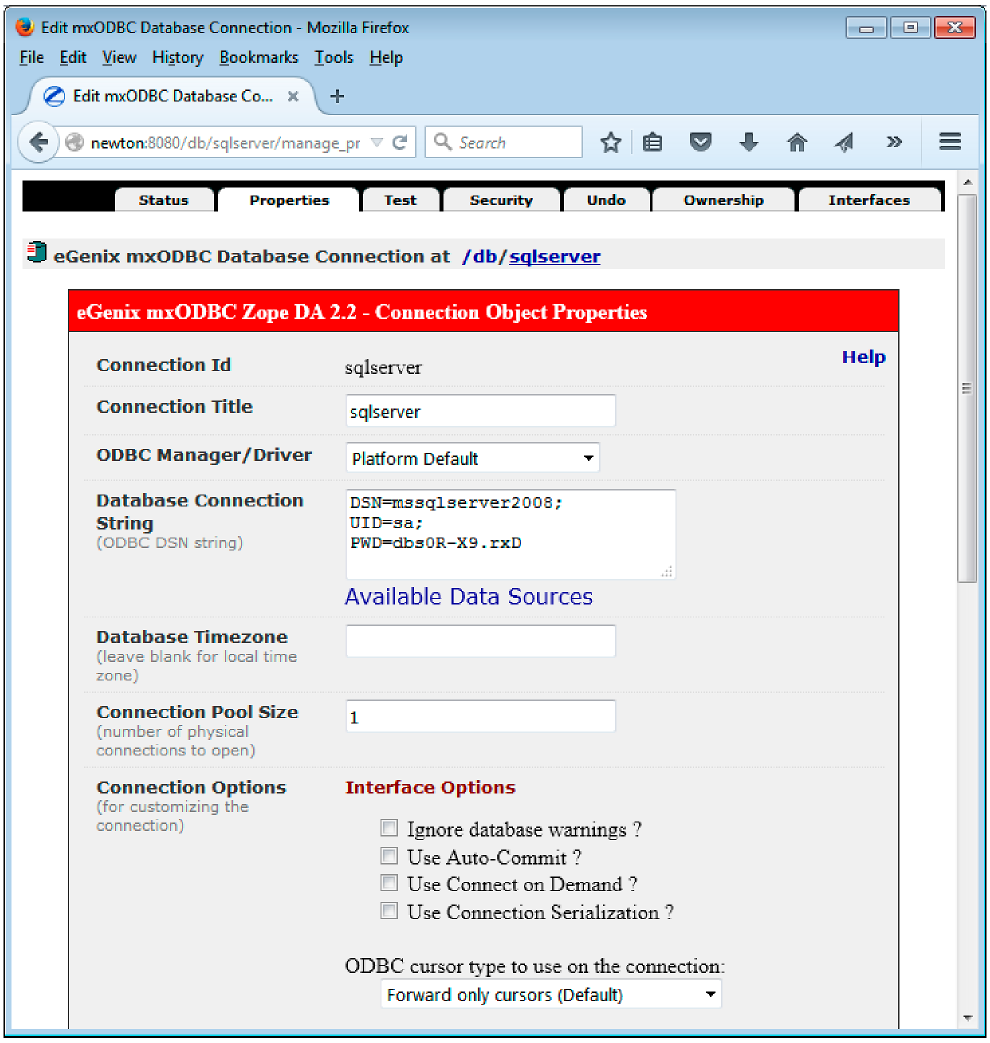
If the error persists please contact the site maintainer. Thank you for your patience.

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.


Edit mxODBC Database Connection - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Edit mxODBC Database Co...
newton:8080/db/sqlserver/manage_pr
Search
Status Properties Test Security Undo Ownership Interfaces

eGenix mxODBC Database Connection at /db/sqlserver

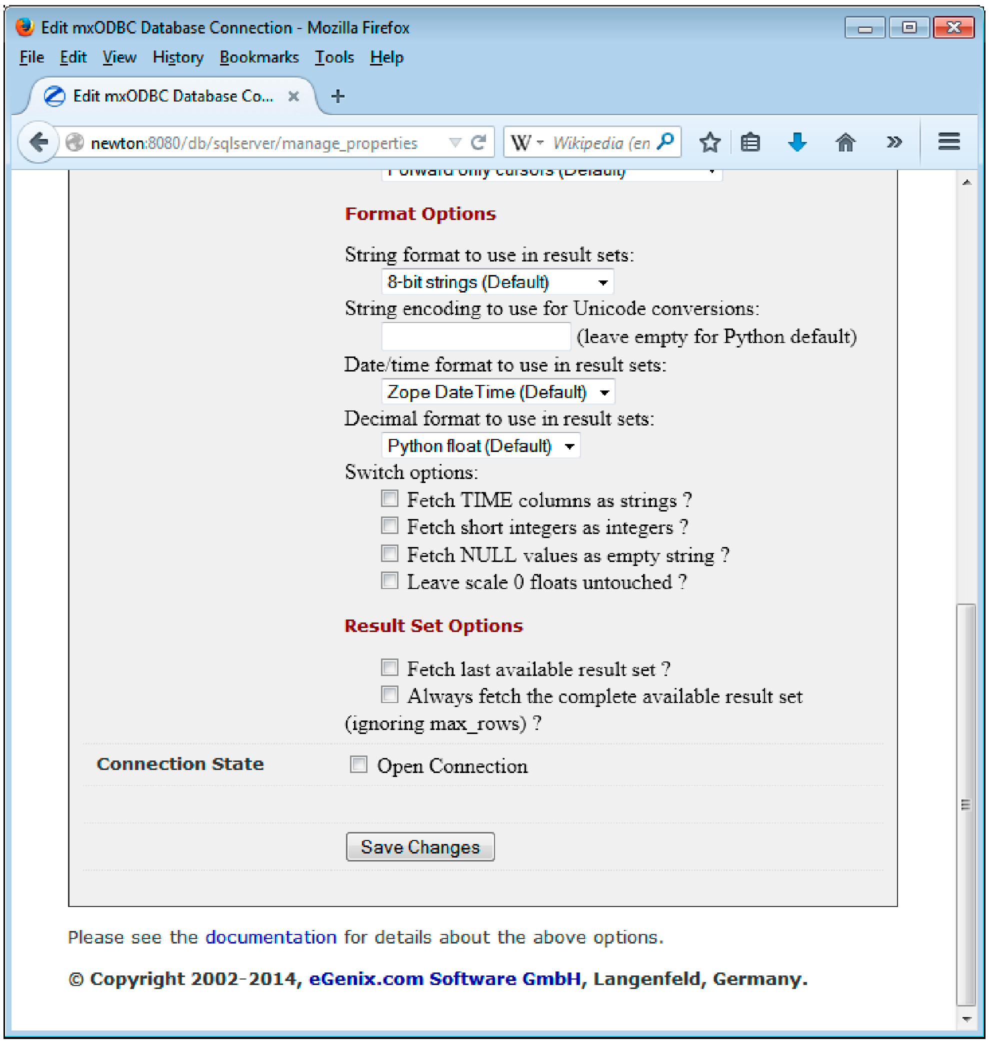
eGenix mxODBC Zope DA 2.2 - Connection Object Properties

Help

Connection Id	sqlserver
Connection Title	sqlserver
ODBC Manager/Driver	Platform Default

Database Connection String (ODBC DSN string)	DSN=mssqlserver2008;
	UID=sa;
	PWD=dbs0R-X9.rxD
	Available Data Sources

Database Timezone (leave blank for local time zone)	
Connection Pool Size (number of physical connections to open)	1
Connection Options (for customizing the connection)	Interface Options ☐ Ignore database warnings ? ☐ Use Auto-Commit ? ☐ Use Connect on Demand ? ☐ Use Connection Serialization ? ODBC cursor type to use on the connection: Forward only cursors (Default)


Edit mxODBC Database Connection - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Edit mxODBC Database Co...
newton:8080/db/sqlserver/manage_properties
Wikipedia (en

Forward only cursors (Default)

Format Options

String format to use in result sets:
8-bit strings (Default)
String encoding to use for Unicode conversions:
(leave empty for Python default)
Date/time format to use in result sets:
Zope DateTime (Default)
Decimal format to use in result sets:
Python float (Default)
Switch options:
☐ Fetch TIME columns as strings ?
☐ Fetch short integers as integers ?
☐ Fetch NULL values as empty string ?
☐ Leave scale 0 floats untouched ?

Result Set Options

☐ Fetch last available result set ?
☐ Always fetch the complete available result set (ignoring max_rows) ?

Connection State ☐ Open Connection

Save Changes

Please see the documentation for details about the above options.

© Copyright 2002-2014, eGenix.com Software GmbH, Langenfeld, Germany.

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

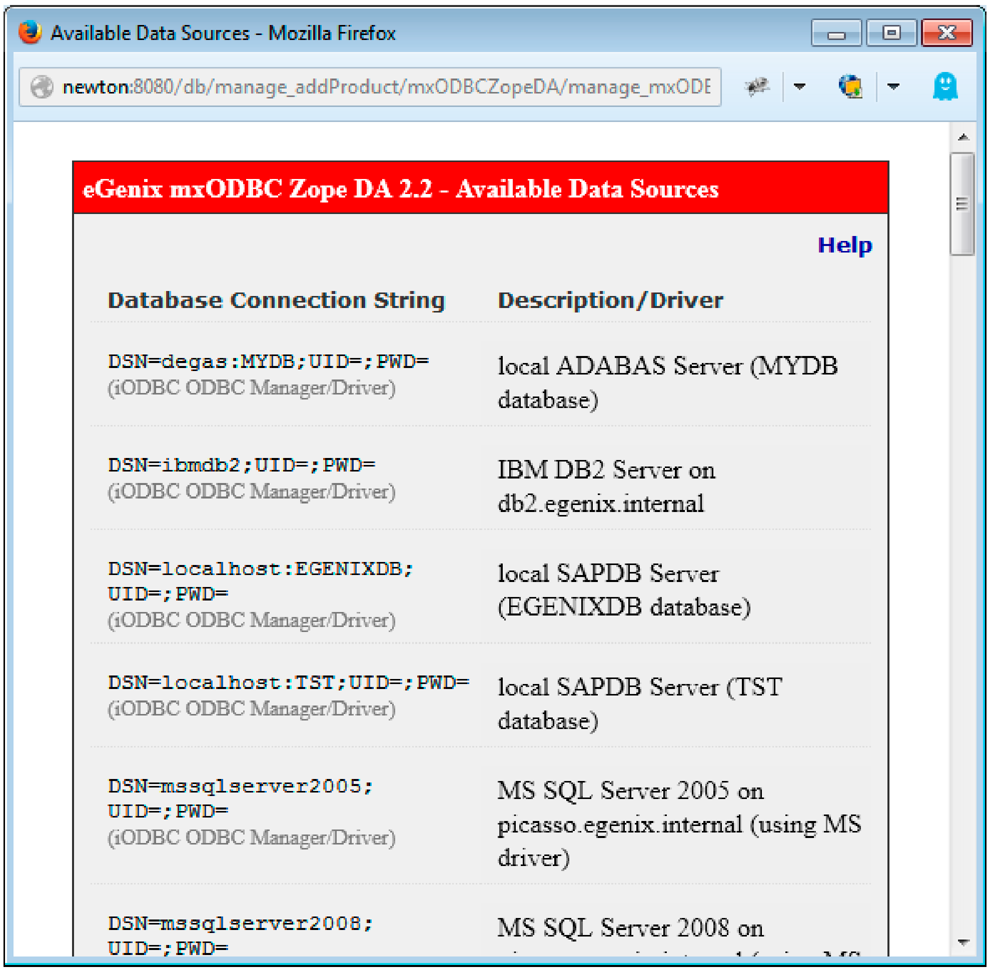
The connection string has to be formatted according to the ODBC DSN standard:

```
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 Data Sources" 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.


Available Data Sources - Mozilla Firefox
newton:8080/db/manage_addProduct/mxODBCZopeDA/manage_mxODBC
eGenix mxODBC Zope DA 2.2 - Available Data Sources
Help

Database Connection String	Description/Driver
DSN=degas:MYDB;UID=;PWD= (iODBC ODBC Manager/Driver)	local ADABAS Server (MYDB database)
DSN=ibmdb2;UID=;PWD= (iODBC ODBC Manager/Driver)	IBM DB2 Server on db2.egenix.internal
DSN=localhost:EGENIXDB; UID=;PWD= (iODBC ODBC Manager/Driver)	local SAPDB Server (EGENIXDB database)
DSN=localhost:TST;UID=;PWD= (iODBC ODBC Manager/Driver)	local SAPDB Server (TST database)
DSN=mssqlserver2005; UID=;PWD= (iODBC ODBC Manager/Driver)	MS SQL Server 2005 on picasso.egenix.internal (using MS driver)
DSN=mssqlserver2008; UID=;PWD=	MS SQL Server 2008 on

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.

Interface Options

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.

Use Connection Serialization

The mxODBC Zope DA will normally rely on the ODBC driver being thread safe and permitting to open connections on multiple threads simultaneously.

However, some ODBC drivers have issues with this. They work fine in multi-threaded mode once a connection has been made, but crash when trying to open multiple connections at the same time from different threads.

As work-around, the mxODBC Zope DA can make sure that the connections on a particular Zope connection object are opened in serialized mode, i.e. one after another, even if multiple threads request opening a new connection.

If you enable "*Use Connection Serialization*", this work-around will be enabled. Default is not to enable the work-around and assume the ODBC driver is thread-safe as mandated by the ODBC standard.

> Note that you have to do this on a per-connection object basis.

ODBC Cursor Type

mxODBC support selecting multiple different ODBC cursor types to be used on a connection. This option allows configuring the used ODBC cursor type.

Each cursor type has its own characteristics and affects the way a cursor sees the result set while fetching data from it. mxODBC Zope DA allows selecting a cursor type per connection.

The following settings are available:

`mxODBC default`

This is the default setting, which corresponds to "Forward only cursors" starting with mxODBC Zope DA 2.2.[2]

`Forward only cursors`

The cursor only scrolls forward. This is the default setting in case mxODBC finds that the database does not support scrollable cursors.

`Keyset driven cursors`

Keysets are sets of columns in the result set that provide unique keys to the rows in the result set. Keyset driven cursors fix the memberships and order of the rows in the result set using these keysets. Unlike static cursors, they don't create a copy of the result set.

[2] Please note that in mxODBC Zope DA 2.1 this setting was dependent on the used database backend.

`Dynamic cursors`

Dynamic cursors are the opposite of static cursors. All changes to the result set after opening it are visible on the next fetch operation.

`Static cursors`

The result set is made static by creating a static copy of the result set after opening the cursor. As a result, any changes to the result set after opening the cursor will not be visible to the client. mxODBC uses these cursors as default for backends which support scrollable cursors.

Please note that using cursor types other than the default "Forward only cursors" may have a significant effect on the performance of fetch operations.

Especially for **MS SQL Server** and **IBM DB2** we recommend using "Forward only cursors", if your application doesn't need the more advanced cursor types, since this results in a significant performance boost compared to the "Static cursors" setting.

Format Options

String format to use in result sets

While the mxODBC Zope DA fully supports Unicode, many Zope applications still rely on 8-bit strings for text data. For this reason, the mxODBC Zope DA defaults to returning text data as plain "*8-bit strings*".

If you want to use Unicode for representing text data, i.e. to avoid problems when dealing with different character sets, you can enable the setting "*Unicode*" to have the Zope DA return text data as Python Unicode objects.

If the ODBC or database don't support Unicode, the Zope DA will automatically convert the string data it receives from the database to Unicode based on the given string encoding (see next section).

In case you want to avoid any such automatic conversion, you can also set the option to "8-bit strings and Unicode". The Zope DA will then return text data from the database in the database provided format, ie. 8-bit strings or Unicode.

String encoding to for Unicode conversions

When using the "Unicode" or "8-bit strings" setting, the mxODBC Zope DA will sometimes have to convert input or output data between Unicode and 8-bit strings, e.g. if you set the string format to Unicode and the database sends 8-bit string, the Zope DA will automatically convert the 8-bit string data to Unicode.

In order for this to work, the mxODBC Zope DA will have to know the encoding used for the 8-bit string text data.

Leave the field empty if you want the Zope DA to use the Python default encoding, which usually is US-ASCII.

If you want to adjust the setting to a different encoding, you can use any of the available and supported Python encodings listed in the Python documentation. Common encodings are *utf-8*, *latin-1* and *cp1252*.

Date/time format to use in result sets

The Zope DA supports many different date/time formats. This option allows you to select which type of date/time object you want to receive in the result sets.

The default is to use Zope's own DateTime object. It is also possible to adjust the setting to receive mxDateTime objects, Python datetime module objects, broken down tuples as used in the Python time module or strings.

When using the "Date/Time Strings" format, the database and/or ODBC driver decides which date/time layout to use. Please consult your database and ODBC driver documentation for details.

Decimal format to use in result sets

Earlier Python version did not have support for working with decimal numbers, i.e. numbers with fixed precision as used for e.g. monetary values.

Since Python 2.5 and later come with a built-in decimal type, the Zope DA supports using this type for database interfacing as well.

In order to remain backwards compatible, the default is still set to use floats.

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 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.

Result Set Options

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[3]. This is done by enabling the connection option "*Use Auto-Commit*" in the connection object management interface.

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 select the "*mxDateTime*" date/time format in the connection object management interface.

[3] 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 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

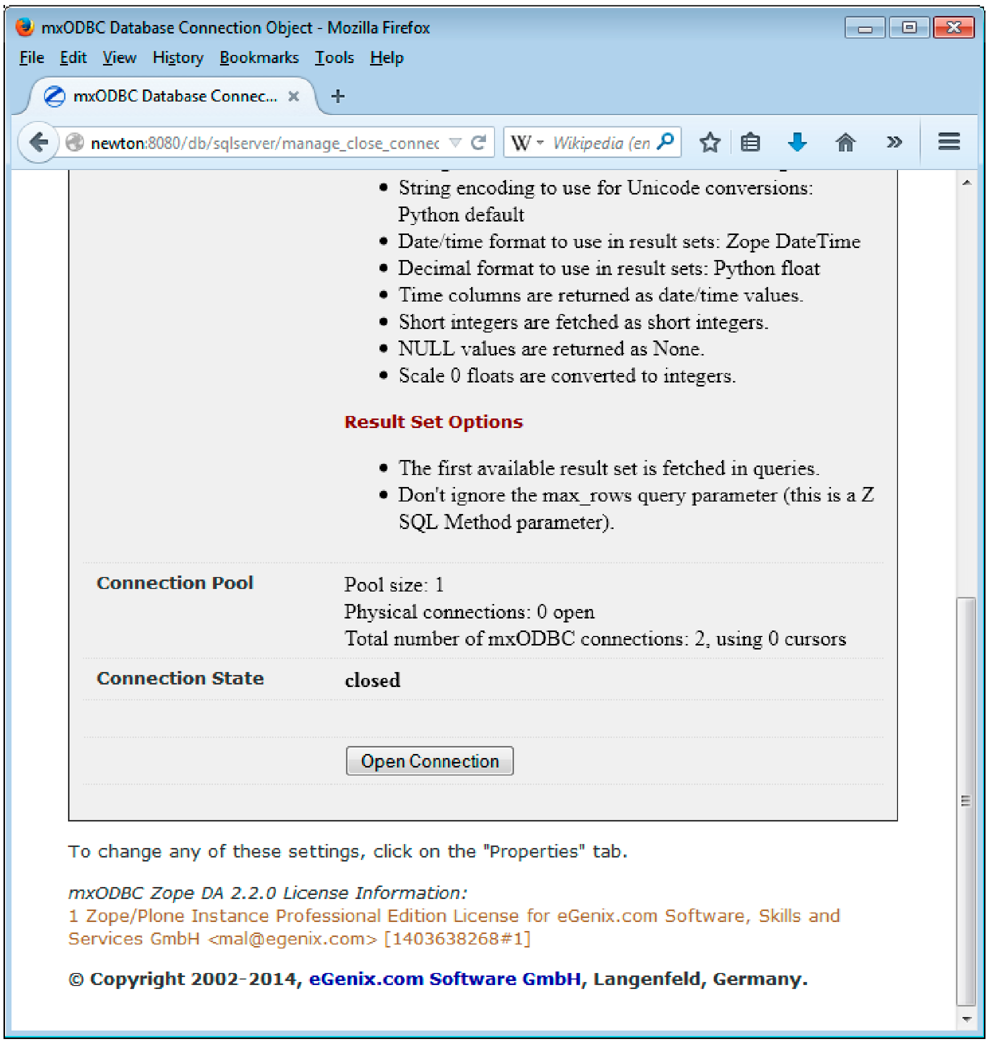
The mxODBC Zope DA Connection object keep persistent state about the connectivity information, i.e. the information whether a connection was put in open or closed state is maintained across Zope restarts (unlike with other Zope DAs).

The mxODBC Zope DA maintains the state at four 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.
- **Open**: the connection has the open state, but no physical connection to the data source could be established. This is a special state meant to ease problem recovery in production systems. The connection will only use this state for open connections that failed to connect on load. It is not a persistent state.
- **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


mxODBC Database Connection Object - Mozilla Firefox
File Edit View History Bookmarks Tools Help
mxODBC Database Connec...
newton:8080/db/sqlserver/manage_close_connec
Wikipedia (en

- String encoding to use for Unicode conversions: Python default
- Date/time format to use in result sets: Zope DateTime
- Decimal format to use in result sets: Python float
- Time columns are returned as date/time values.
- Short integers are fetched as short integers.
- NULL values are returned as None.
- Scale 0 floats are converted to integers.

Result Set Options

- The first available result set is fetched in queries.
- Don't ignore the max_rows query parameter (this is a Z SQL Method parameter).

Connection Pool	Pool size: 1 Physical connections: 0 open Total number of mxODBC connections: 2, using 0 cursors
Connection State	**closed**
	Open Connection

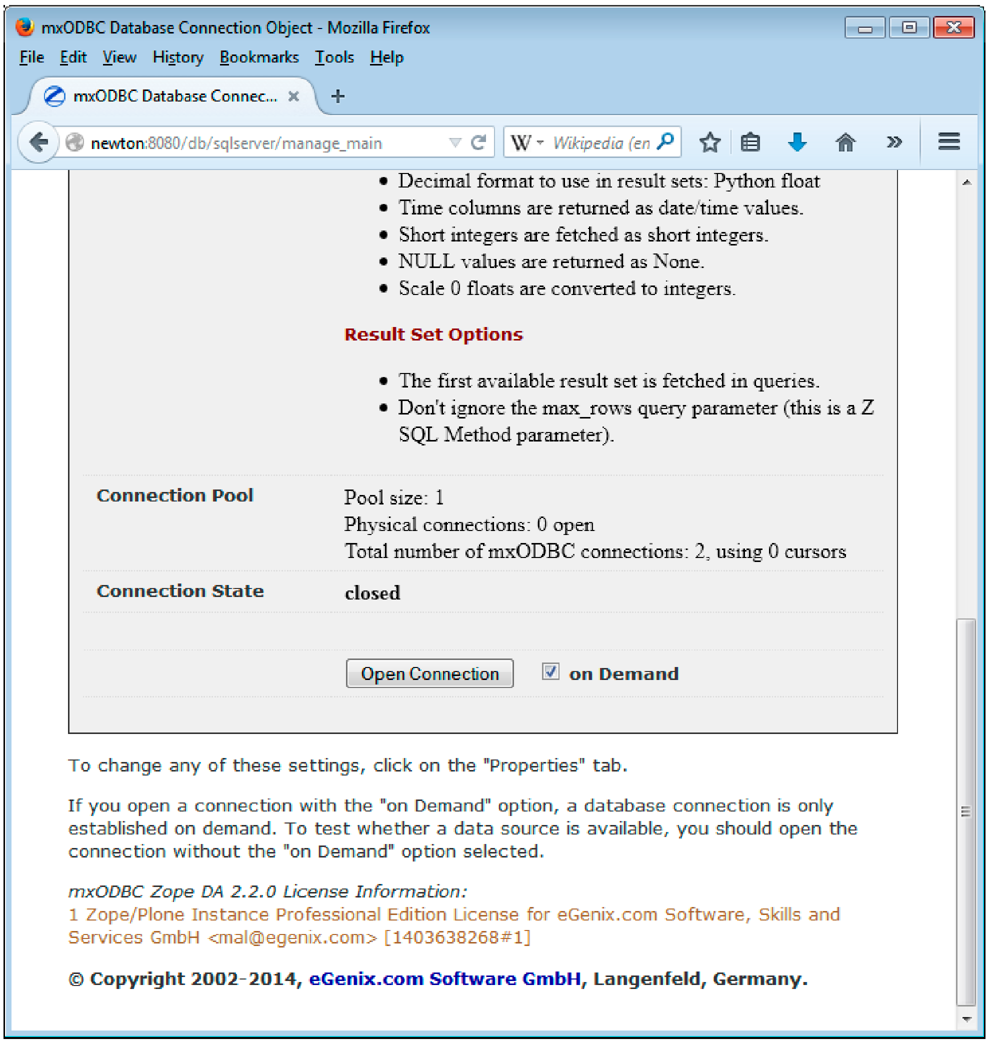
To change any of these settings, click on the "Properties" tab.

mxODBC Zope DA 2.2.0 License Information:
1 Zope/Plone Instance Professional Edition License for eGenix.com Software, Skills and Services GmbH <mal@egenix.com> [1403638268#1]

© Copyright 2002-2014, eGenix.com Software GmbH, Langenfeld, Germany.

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*".


mxODBC Database Connection Object - Mozilla Firefox
File Edit View History Bookmarks Tools Help
mxODBC Database Connec...
newton:8080/db/sqlserver/manage_main
Wikipedia (en

- Decimal format to use in result sets: Python float
- Time columns are returned as date/time values.
- Short integers are fetched as short integers.
- NULL values are returned as None.
- Scale 0 floats are converted to integers.

Result Set Options

- The first available result set is fetched in queries.
- Don't ignore the max_rows query parameter (this is a Z SQL Method parameter).

Connection Pool	Pool size: 1 Physical connections: 0 open Total number of mxODBC connections: 2, using 0 cursors
Connection State	**closed**
	Open Connection ☑ **on Demand**

To change any of these settings, click on the "Properties" tab.

If you open a connection with the "on Demand" option, a database connection is only established on demand. To test whether a data source is available, you should open the connection without the "on Demand" option selected.

mxODBC Zope DA 2.2.0 License Information:
1 Zope/Plone Instance Professional Edition License for eGenix.com Software, Skills and Services GmbH <mal@egenix.com> [1403638268#1]

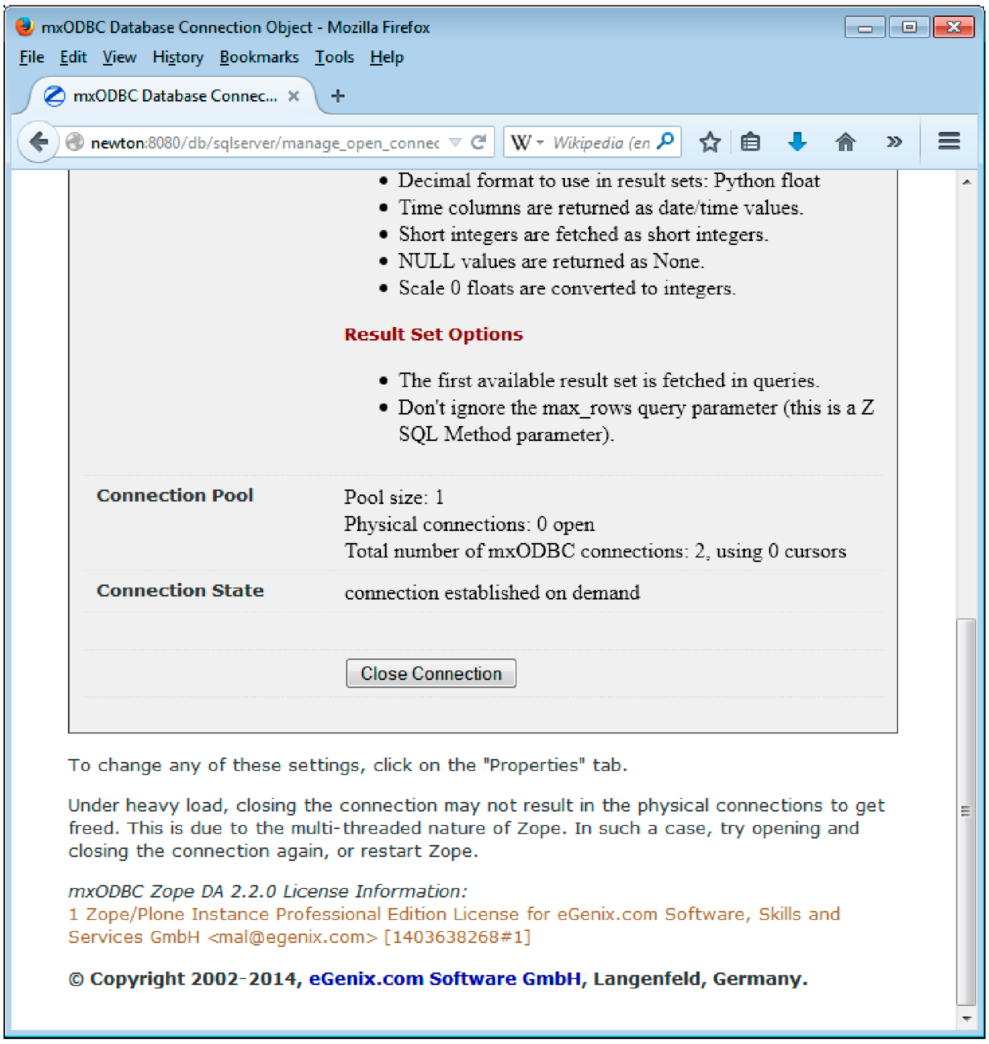
© Copyright 2002-2014, eGenix.com Software GmbH, Langenfeld, Germany.

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:


mxODBC Database Connection Object - Mozilla Firefox
File Edit View History Bookmarks Tools Help
mxODBC Database Connec...
newton:8080/db/sqlserver/manage_open_connec
Wikipedia (en

- Decimal format to use in result sets: Python float
- Time columns are returned as date/time values.
- Short integers are fetched as short integers.
- NULL values are returned as None.
- Scale 0 floats are converted to integers.

Result Set Options

- The first available result set is fetched in queries.
- Don't ignore the max_rows query parameter (this is a Z SQL Method parameter).

Connection Pool	Pool size: 1 Physical connections: 0 open Total number of mxODBC connections: 2, using 0 cursors
Connection State	connection established on demand

To change any of these settings, click on the "Properties" tab.

Under heavy load, closing the connection may not result in the physical connections to get freed. This is due to the multi-threaded nature of Zope. In such a case, try opening and closing the connection again, or restart Zope.

mxODBC Zope DA 2.2.0 License Information:
1 Zope/Plone Instance Professional Edition License for eGenix.com Software, Skills and Services GmbH <mal@egenix.com> [1403638268#1]

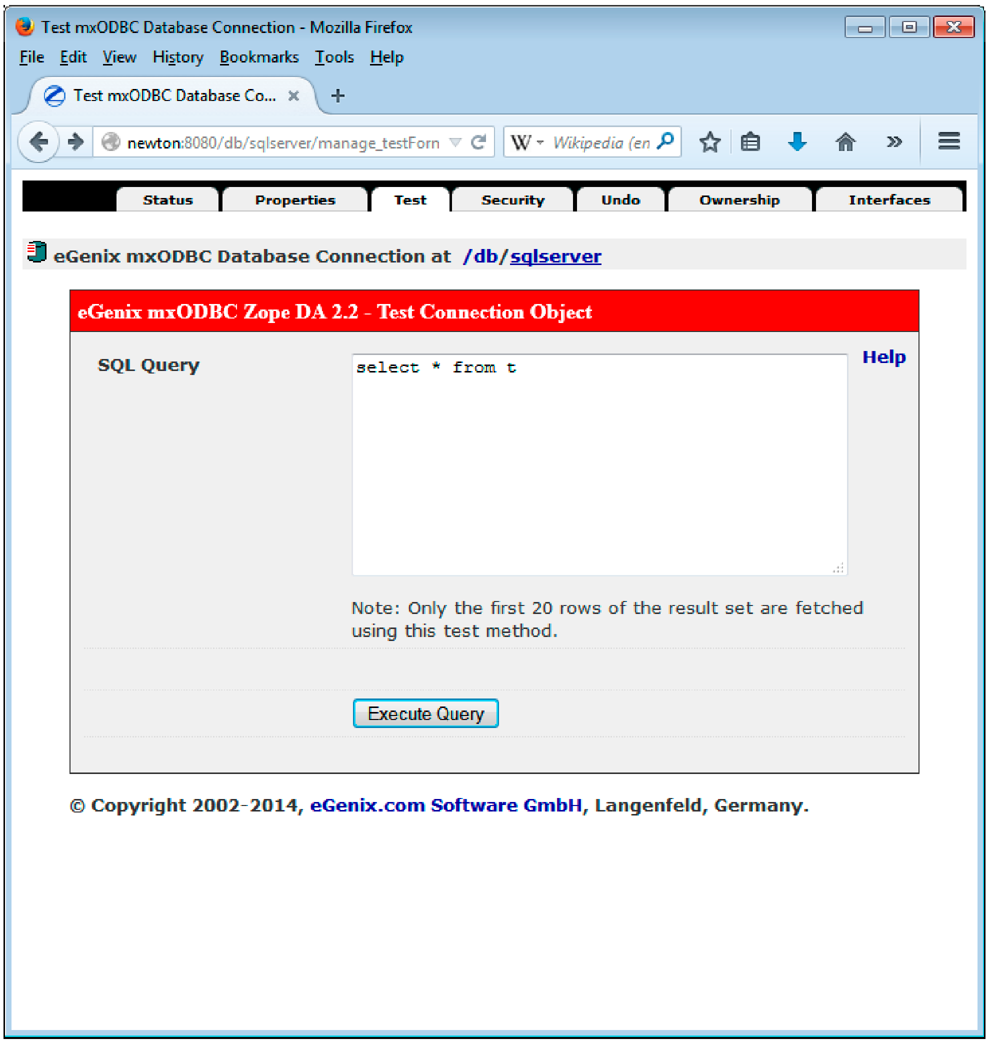
© Copyright 2002-2014, eGenix.com Software GmbH, Langenfeld, Germany.

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.

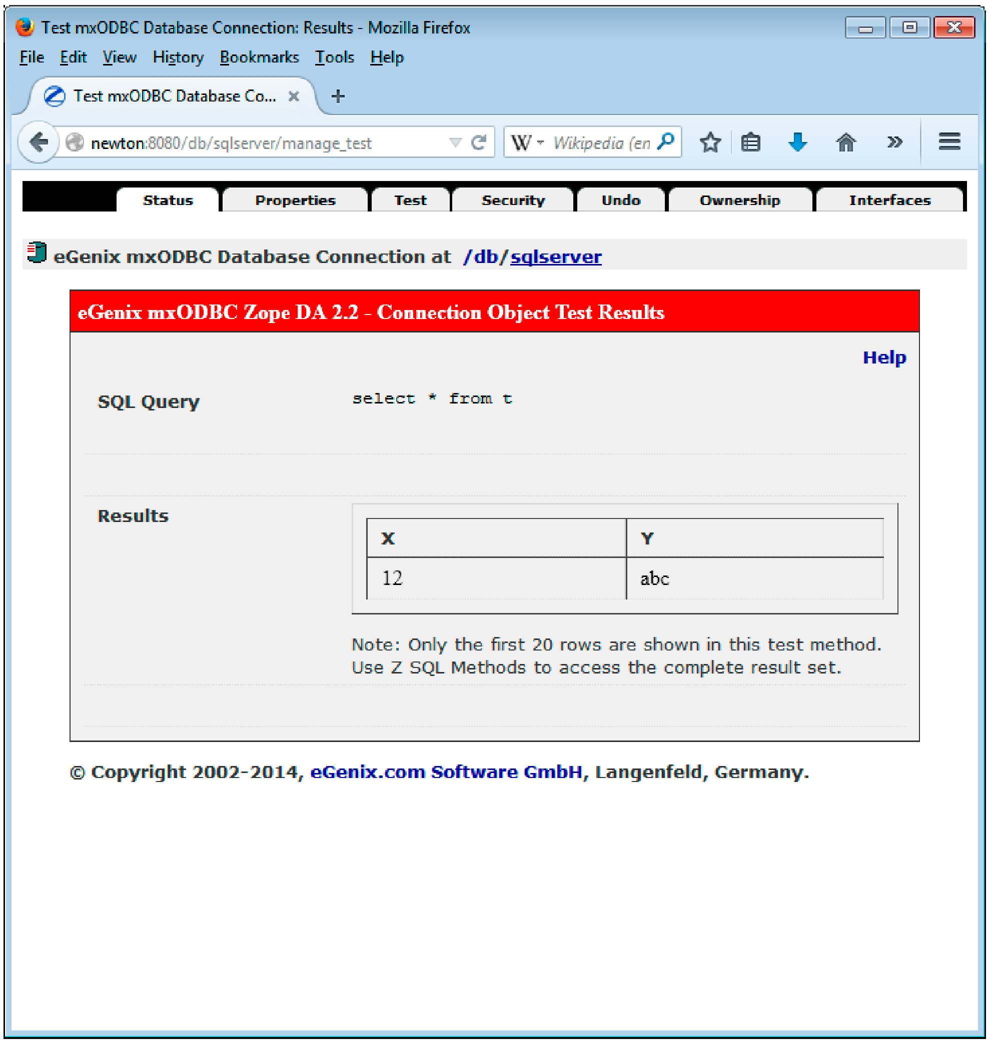

Test mxODBC Database Connection - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Test mxODBC Database Co...
newton:8080/db/sqlserver/manage_testForn
Wikipedia (en
Status
Properties
Test
Security
Undo
Ownership
Interfaces
eGenix mxODBC Database Connection at /db/sqlserver
eGenix mxODBC Zope DA 2.2 - Test Connection Object
SQL Query
Help
Execute Query

```
select * from t
```

Note: Only the first 20 rows of the result set are fetched using this test method.

© Copyright 2002-2014, eGenix.com Software GmbH, Langenfeld, Germany.

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


Test mxODBC Database Connection: Results - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Test mxODBC Database Co...
newton:8080/db/sqlserver/manage_test
Wikipedia (en
Status
Properties
Test
Security
Undo
Ownership
Interfaces
eGenix mxODBC Database Connection at /db/sqlserver
Help

eGenix mxODBC Zope DA 2.2 - Connection Object Test Results

SQL Query

```
select * from t
```

Results

X	Y
12	abc

Note: Only the first 20 rows are shown in this test method.
Use Z SQL Methods to access the complete result set.

© Copyright 2002-2014, eGenix.com Software GmbH, Langenfeld, Germany.

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.

Please see the *Zope documentation* for details on how to use Z SQL Methods. The *ZSQL Method User's Guide* also provides a good overview.

4.5 Using External Methods with the mxODBC Zope DA

If you want to use more of the available functionality that the mxODBC Zope DA exposes to Plone/Zope via the Python interface (see 5 Python Interface), a good way to do this is by deploying Zope *External Methods* which allow calling into Python modules and functions from without Plone/Zope.

The following sections will explain how to create such *External Methods*.

4.5.1 Create a Python module for use by External Methods

First, you have to create a Python module to hold the functions you want to call as external methods.

In a typical Plone/Zope installation this is done by creating a directory Extensions/ in the zinstance/ directory of your instance:

```
cd zinstance
mkdir Extensions
```

You can then create a Python module in this directory, let's say mxodbc_tests.py.

In this module, create a function called `test_callproc()`, which looks like this:

```
from mx import Log, DateTime

def test_callproc(self, REQUEST):
```

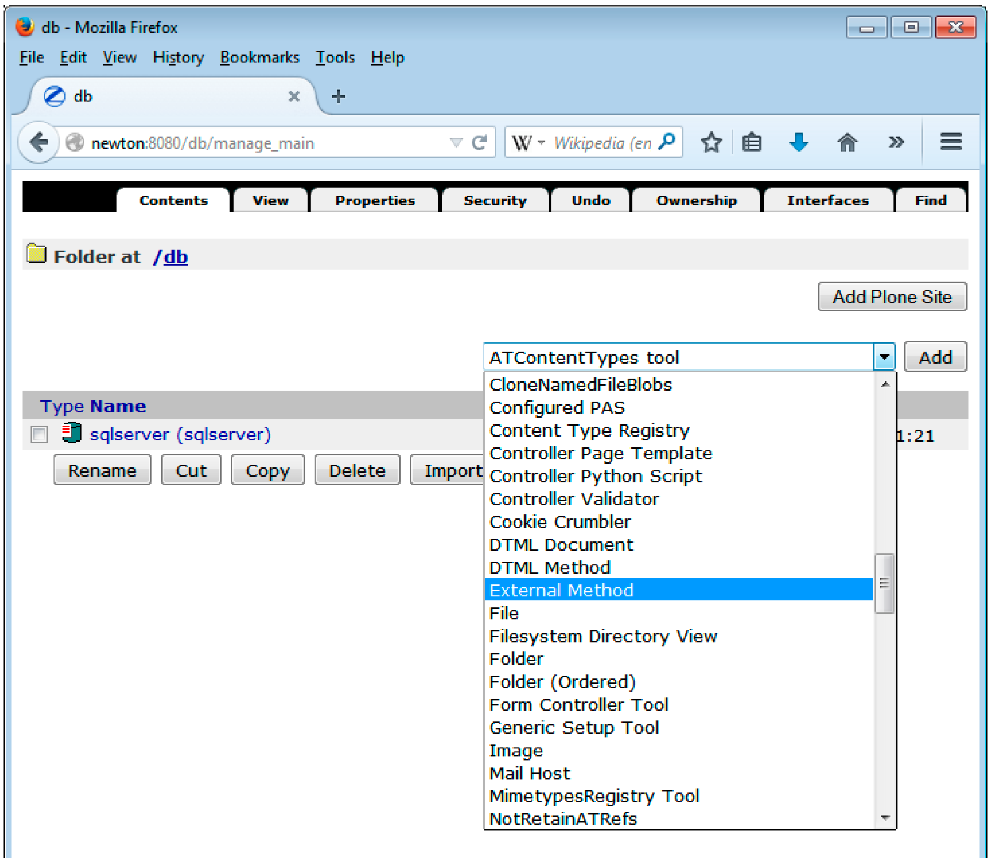
```
""" Test the .callproc() method.

    These tests are specific to MS SQL Server and need a
    mxODBC Zope DA connection object "sqlserver" in the local
    context.

    The tests generate AssertionErrors in case of problems.

"""
request_time = DateTime.now()
request = REQUEST
response = request.RESPONSE

# Get database connection
sqlserver = self.sqlserver()

# Create SP
try:
    sqlserver.execute('DROP PROCEDURE test_sp_result_set')
except sqlserver.DatabasePackage.ProgrammingError:
    pass
sqlserver.execute(
    """
    CREATE PROCEDURE test_sp_result_set
    @a INTEGER
    AS
      SELECT @a * 3;
   """)

# Call SP and check results
args, resultsets = sqlserver.callproc('test_sp_result_set', 4)
assert args == [4], 'test_sp_result_set args'
info, data = resultsets[0]
assert data == [(12,)], 'test_sp_result_set data'

# Create SP
try:
    sqlserver.execute('DROP PROCEDURE test_sp_float')
except sqlserver.DatabasePackage.ProgrammingError:
    pass
sqlserver.execute(
    """
    CREATE PROCEDURE test_sp_float
    @a FLOAT,
    @b FLOAT OUTPUT
    AS
      SET @b = @a * 3.14;
   """)

# Call SP and check results
args, resultsets = sqlserver.callproc(
   'test_sp_float',
   2.10, 0.0,
   parametertypes=(sqlserver.SQL.PARAM_INPUT,
                   sqlserver.SQL.PARAM_OUTPUT,
                  ))
assert args == [2.1, 6.594], 'test_sp_float args'
info, data = resultsets[0]
assert data == (), 'test_sp_float data'

# Create SP
try:
    sqlserver.execute('DROP PROCEDURE test_sp_float_inout')
    except sqlserver.DatabasePackage.ProgrammingError:
        pass
    sqlserver.execute(
        """
        CREATE PROCEDURE test_sp_float_inout
        @a FLOAT,
        @b FLOAT OUTPUT
        AS
          SET @b = @b * 3.14 + @a;
       """)

    # Call SP and check results
    args, resultsets = sqlserver.callproc(
        'test_sp_float_inout', 0.5, 2.10,
        parametertypes=(sqlserver.SQL.PARAM_INPUT,
                        sqlserver.SQL.PARAM_INPUT_OUTPUT,
                       ))
    assert args == [0.5, 7.094], 'test_sp_float_inout args'
    info, data = resultsets[0]
    assert data == (), 'test_sp_float_inout data'

    return ('Tests passed.')
```

The above test functions exercises a few called procedure features using MS SQL Server as backend.

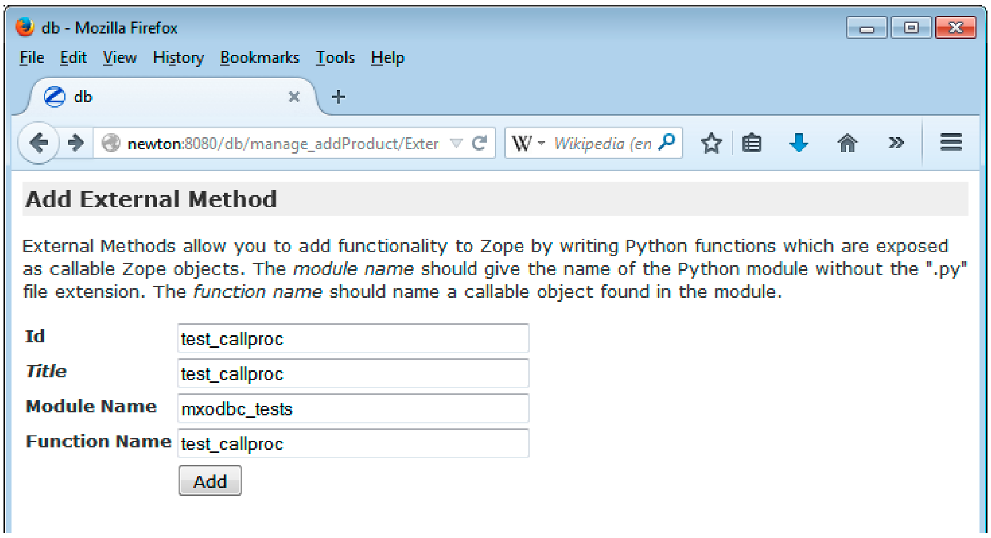
4.5.2 Create a Zope External Method

in order to call the `test_callproc()` function from within Zope, you need to register it as External Method object.

Using the Zope Management Interface (ZMI), create an External Method object in a folder:



This will open the dialog for entering the ID and name of the External Method:


db - Mozilla Firefox
File Edit View History Bookmarks Tools Help
db
newton:8080/db/manage_addProduct/Exter

Add External Method

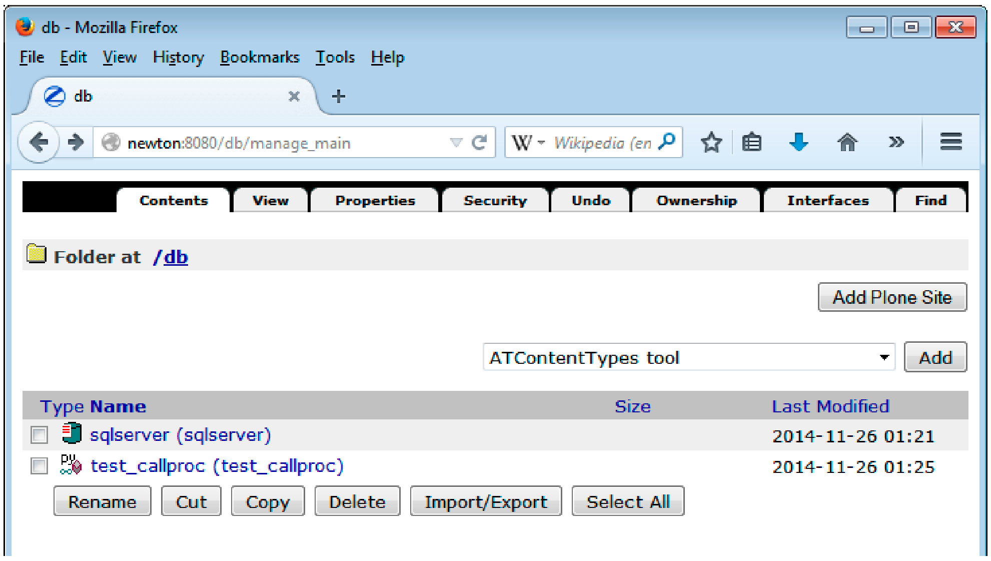
External Methods allow you to add functionality to Zope by writing Python functions which are exposed as callable Zope objects. The *module name* should give the name of the Python module without the ".py" file extension. The *function name* should name a callable object found in the module.

Id	test_callproc
Title	test_callproc
Module Name	mxodbc_tests
Function Name	test_callproc

Enter the module name you created and the function name to call when calling the External Method from within Zope.

In the example, we've used the names from the previous section.

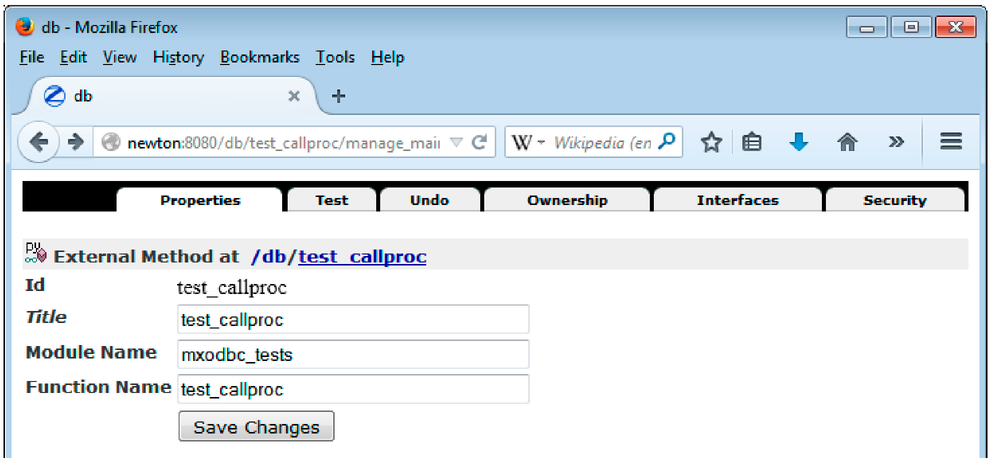
Finally, click Add.



Folder at /db

Type Name	Size	Last Modified
sqlserver (sqlserver)		2014-11-26 01:21
test_callproc (test_callproc)		2014-11-26 01:25

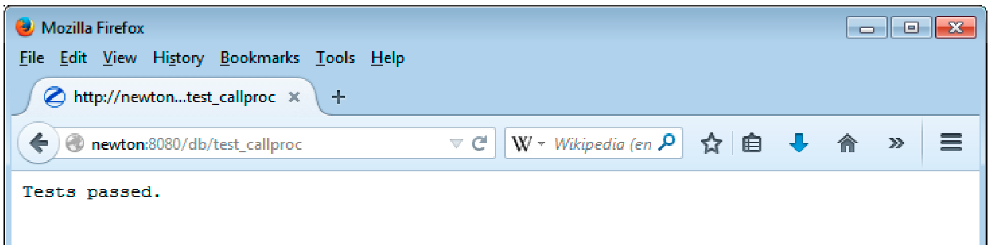
Clicking on the new External Method object will allow you to test the method:



External Method at /db/test_callproc

Id	test_callproc
Title	test_callproc
Module Name	mxodbc_tests
Function Name	test_callproc

Clicking on the Test tab will run the function call and since we've used a test case will result in the following text to be returned:


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://newton...test_callproc
newton:8080/db/test_callproc
Wikipedia (en

```
Tests passed.
```

Of course, you can use such functions to return any kind of data to your Plone/Zope application.

4.5.3 Accessing the connection object

As you can see in the example function, accessing the connection object from the context folder is easy:

```
def test_callproc(self, REQUEST):
    …
    # Get database connection
    sqlserver = self.sqlserver()
```

The sqlserver object will now refer to a connected database *Products.mxODBCZopeDA.ZopeDA.DatabaseConnection* object (see 5.1.1 Class "Products.mxODBCZopeDA.ZopeDA.DatabaseConnection") or *DatabaseConnection* for short.

You can directly run queries or statements on this object. Zope will take care of managing the transaction for you.

Since the Zope Z SQL Method access to database connections are fairly limited, we have exposed a lot more methods on this object in order to give you access to the broad set of features in mxODBC.

4.5.4 Catching connection exceptions

In some cases, you will have to catch exceptions from the connection object methods.

Since mxODBC support multiple different ODBC managers, it is not always clear how to access the correct exception classes. To make this easier, we have added a reference to the right ODBC manager interface subpackage to the connection object itself as attributes **`connection.DatabasePackage`**. This subpackage contains the exception object by their usual Python DB-API compatible name. Please see the mxODBC documentation or the Python DB-API 2.0 standard for details on the various exception objects.

Catching exceptions can thus be done by referencing the mxODBC exception object as attribute on the `connection.DatabasePackage` object, e.g.

```
try:
    sqlserver.execute('DROP PROCEDURE test_sp_result_set')
except sqlserver.DatabasePackage.ProgrammingError:
    pass
```

4.5.5 Accessing connection specific SQL constants

You will sometimes need to access SQL constants which are specific to the underlying ODBC manager or connection. A typical case is when defining the input/output parameter types of stored procedures.

mxODBC Zope DA makes just as easy as accessing mxODBC exceptions by placing the mxODBC SQL object as attribute on the *DatabaseConnection* object, e.g.

```
args, resultsets = sqlserver.callproc(
    'test_sp_float_inout', 0.5, 2.10,
    parametertypes=(sqlserver.SQL.PARAM_INPUT,
                    sqlserver.SQL.PARAM_INPUT_OUTPUT,
                   ))
```

As you can see, `sqlserver.SQL` provides access to the needed SQL constants, regardless of which ODBC manager was used for the connection.

4.5.6 Low-level mxODBC access

Warning: If you want to go all the way down to the mxODBC connection object level, this is possible as well, but please be aware that you can easily cause the connection pooling available in the mxODBC Zope DA to break by e.g. reconfiguring the mxODBC connection objects in ways which are not detected by the mxODBC Zope DA.

The underlying **Python DB-API 2.0 compatible** mxODBC connection object is exposed by the mxODBC Zope DA on the *DatabaseConnection* objects as .connection attribute, e.g.

```
args = sqlserver.connection.callproc(
    'test_sp_float_inout', [0.5, 2.10],
    parametertypes=(sqlserver.SQL.PARAM_INPUT,
                    sqlserver.SQL.PARAM_INPUT_OUTPUT,
                    ))
assert args == [0.5, 7.094], 'test_sp_float_inout args'
```

Please see the *mxODBC documentation* for details on how to use the mxODBC connection objects.

Since these objects are compatible to Python DB-API 2.0 connection objects, you can also use available Python DB-API resources such as the *PEP 249 standard document* or one of the many Python books as reference.

Transactions

As long as you only use these connection objects through the higher level mxODBC Zope DA *DatabaseConnection* objects, your implementation will benefit from the mxODBC Zope DA transaction mechanisms which hook directly into the Zope transaction mechanism.

If you create your own mxODBC connection objects, these won't automatically be hooked into the Zope transaction mechanism. You will have to manage this yourself.

4.6 Calling Stored Procedures

Stored procedures are often used to implement business logic directly in the database backend. The mxODBC Zope DA can access such stored procedures using two different mechanisms explained in the following sections:

- Z SQL Methods
- External Methods

4.6.1 Using Z SQL Methods

Stored procedures can be called using Z SQL Methods if they only use input parameters and pass back their results (if any) using result sets. This restriction is due to the way Z SQL Methods work. They cannot pass back modified parameters or return values from stored procedures.

There are no special requirements with respect to parameter types. The syntax to use for the procedure calls depend on the database backend. Please refer to the database documentation for details.

4.6.2 Using External Methods

If you want to use output or in/out parameters, you have to use External Methods to access the `DatabaseConnection .callproc()` method.

Apart from providing access to the modified parameters, the `.callproc()` method also removes the need to use database specific SQL in the call. The ODBC driver used for accessing the database backend will take care of this for you.

This is a copy of the `connection.callproc()` documentation found in section 5.1.1:

`.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.

If parametertypes is given as keyword parameter, it defines the parameter types of the parameters used in the procedure. Default is to assume input parameter types for all parameters. Please see the *mxODBC documentation* for details on how to use parametertypes.

When using the method, you pass in the procedure name, followed by the parameters.

If you are using output or in/out parameters, you have to declare these using the keyword parameter `parametertypes`. This provides a position mapping of the parameters to parameter types. Possible values are:

`connection.SQL.PARAM_INPUT`

for input parameters

`connection.SQL.PARAM_INPUT_OUTPUT`

for input/output parameters

`connection.SQL.PARAM_OUTPUT`

for output parameters

The method returns a tuple which includes a copy of the - possibly modified - parameters passed to the method.

Example:

```
sqlserver.execute(
    """
    CREATE PROCEDURE test_sp_float
    @a FLOAT,
    @b FLOAT OUTPUT
    AS
      SET @b = @a * 3.14;
   """)

# Call SP and check results
args, resultsets = sqlserver.callproc(
   'test_sp_float',
   2.10, 0.0,
   parametertypes=(sqlserver.SQL.PARAM_INPUT,
                   sqlserver.SQL.PARAM_OUTPUT,
                  ))
assert args == [2.1, 6.594], 'test_sp_float args'
info, data = resultsets[0]
assert data == (), 'test_sp_float data'
```

When running this code as External Method, `args` will be a list with the modified output data and `resultsets` will have one empty result set. Please see section 4.5 Using External Methods with the mxODBC Zope DA for details on how to embed such code in an External Method.

Note that for output and input/output parameters, the type of the parameter used as input value (possibly a placeholder value such as 0.0 in the above example) has to have the same type as the output parameter expected from the procedure.

4.7 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).

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_format = 0`

Date/time format to use. Possible values: 0 - Zope DateTime, 1 - mxDateTime, 2 - Python datetime, 3 - Python tuples, 4 - Python strings.

`.decimal_format = 0`

Decimal format to use. Possible values: 0 - Python floats, 1- Python decimals.

`.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.

`.string_encoding = ''`

String encoding to use for Unicode / 8-bit string conversion. This must be set to an encoding known to Python. Empty for the Python default encoding.

`.string_format = 0`

String format to use. Possible values: 0 - 8-bit strings, 1 - Unicode, 2 - mixed 8-bit and Unicode.

`.cursor_type = 0`

Defines the ODBC cursor type to be used. Possible values: 0 - mxODBC default, 1 - forward only cursors, 2 - keyset driven cursors, 3- dynamic cursors, 4- static cursors.

`.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_format = format_code`

Fetch date/time values in different ways.

`decimal_format = format_code`

Fetch decimal in different ways.

`string_format = format_code`

Fetch string values in different ways.

`string_encoding = encoding_string`

Encoding to use for strings on the connection.

`cursor_type = format_code`

Defines the ODBC cursor type to be used.

`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.

`serialize_connects = boolean_flag`

Use connection serialization for this connection object ? Enabling this option causes the Zope DA to force serialization of connection attempts to work around ODBC driver issues. Default is not to use serialized connections.

`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.

If parametertypes is given as keyword parameter, it defines the parameter types of the parameters used in the procedure. Default is to assume input parameter types for all parameters. Please see the *mxODBC documentation* for details on how to use parametertypes.

`.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, direct=-1)`

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.

If direct is given, mxODBC will use direct, unprepared execution for the SQL query in case it is set to 1 (True) and prepared execution in case it is set to 0 (False).

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 = 80`

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. 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.

Note: The included mx.ODBC package is **only available and usable from within Zope**. For stand-alone installations, please see the *mxODBC product page*.

6.1 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.

6.2 Unix Platform

mxODBC Zope DA binary packages for Unix only ship with the `mx.ODBC.iODBC`, `mx.ODBC.unixODBC` and `mx.ODBC.DataDirect` subpackages of mxODBC.

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

6.3 Mac OS X Platform

mxODBC Zope DA binary packages for Mac OS X only ship with the `mx.ODBC.iODBC` and `mx.ODBC.unixODBC` subpackages of mxODBC. *iODBC* uses the system-provided ODBC Administrator, while *unixODBC* is an optional separate installation on Mac OS X.

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

7. 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-mxodbc* (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

7.1 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 1.3.0

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, among other parameters, in terms of the following definitions:

1) "CPU" refers to a central processing unit which is able to store and/or execute the Software (a server, personal computer, virtual machine, 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.

Additional terms may be defined as part of the Proof of Authorization.

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. The number of cores per processor does not count towards this license limitation. Virtual machines always count as one (1) CPU. 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:	one (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>

EGENIX.COM PROOF OF AUTHORIZATION

1 Developer CPU License (Example)

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

5. 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.

6. 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

7. Authorizations

7.1 Application Development

eGenix.com hereby authorizes Licensee to copy, install, compile, modify and use the Software on the following Developer Installation Targets for the purpose of developing products using the Software as integral part.

Developer Installation Targets: one (1) Developer CPU

7.2 Redistribution

eGenix.com hereby authorizes Licensee to redistribute the Software bundled with a product developed by Licensee on the Developer Installation Targets ("the Product") subject to the terms and conditions of this License Agreement for installation and use in combination with the Product on the following Redistribution Installation Targets, provided that:

1. Licensee shall not and shall not permit or assist any third party to sell or distribute the Software as a separate product;

2. Licensee shall not and shall not permit any third party to

    i. market, sell or distribute the Software to any end user except subject to the terms and conditions of this License Agreement,

    ii. rent, sell, lease or otherwise transfer the Software or any part thereof or use it for the benefit of any third party,

    iii. use the Software outside the Product or for any other purpose not expressly licensed hereunder;

3. the Product does not provide functions or capabilities similar to those of the Software itself, i.e. the Product does not introduce commercial competition for the Software as sold by eGenix.com;

4. Licensee has obtained Developer CPU Licenses for all developers and CPUs used in developing the Product.

Redistribution Installation Targets:

any number of CPUs capable of running the Product and the Software

8. Proof

This Proof of Authorization was issued by

<name>, <title>

Langenfeld, <date>

Proof of Authorization Key:

<license key>