



Mimer SQL

Getting Started on macOS

Version 11.0

Mimer SQL, Getting Started on macOS, Version 11.0, September 2025
© Copyright Mimer Information Technology AB

The contents of this manual may be printed in limited quantities for use at a Mimer SQL installation site. No parts of the manual may be reproduced for sale to a third party.
Information in this document is subject to change without notice. All registered names, product names and trademarks of other companies mentioned in this documentation are used for identification purposes only and are acknowledged as the property of the respective company. Companies, names and data used in examples herein are fictitious unless otherwise noted.

Produced and published by Mimer Information Technology AB, Uppsala, Sweden.

Mimer SQL Web Sites:
<https://developer.mimer.com>
<https://www.mimer.com>

Contents

.....	i
Getting Started.....	1
Licensing Mimer SQL	1
Documentation	1
Command line help and man pages.....	2
Useful links.....	2
Installing Mimer SQL	3
It is really simple to get going!	3
Using the Mimer SQL apps	3
From the Terminal application.....	4
Why do we need sudo access to install?	5
System resources	5
Physical memory.....	5
Virtual memory	5
Environment	6
Which components will be installed?.....	6
Running several Mimer SQL versions in parallel.....	7
Mimer SQL license key	7
Creating an initial database	8
Upgrading an existing database.....	9
Uninstalling the software	9
Database registration	9
The sqlhosts file	9
The Database Server	15
Database server management	15
mimadmin	15
mimdbserver.....	16
mimcontrol	16

Database home directory	16
Logging database events	16
Configuring a database server.....	17
The multidefs parameter file	17
Automatic database start and stop	23
Remote database access.....	24
Database TCP/IP connect dispatcher	24
The mimtcp command	24
Services setup	25
Networking Setup	25
Using odbc.ini data sources.....	26
Development and Example Environments	29
Database APIs	30
Embedded SQL	30
Module SQL	30
JDBC.....	30
ODBC.....	31
Mimer SQL C API.....	31
Python.....	32
PHP	32
Accessing the database	32
Setting up and running DbVisualizer	32
Running Mimer BSQL and other utilities.....	33
Environment Variables	33
macOS Commands	35
macOS Link Libraries	39
Index	41

Chapter 1

Getting Started

Welcome to Mimer SQL. This document describes how to install and set-up Mimer SQL on macOS. To get the most out of this document, you should be familiar with your macOS environment and know how to use the various macOS system tools, especially using the Terminal command line tool.

Mimer SQL provides small footprint, scalable and robust relational database solutions that conform to international ISO SQL standards. Due to its structural modularity, it is very well suited for high performance mission critical systems as well as for mobile and embedded appliances. In addition, Mimer SQL is equipped with an extensive multilingual support using collations.

Licensing Mimer SQL

When you install Mimer SQL, a default development edition license key is installed. This license key is limited in number of users and thus several features, including the DbVisualizer tool, may run into problems saying ‘Number of users exceeded’. A customized license key can be received for free, either it will come separately along with the installation, or it can be ordered from the Mimer SQL Support desk (info@mimer.com).

If you want to use Mimer SQL for any purpose other than development, you must purchase a commercial license. Contact your Mimer SQL distributor, <https://www.mimer.com/contactus/>, to purchase the license you require. Your new license key will be sent to you via e-mail. You apply the new license key by using the `mimlicense` tool in the installation.

Documentation

The *Mimer SQL Documentation Set*, *Mimer JDBC Driver Guide*, *Mimer SQL Release Notes*, and *Mimer SQL Getting Started on macOS* are available in the installation.

The Mimer SQL documentation set includes the following:

- *SQL Reference Manual*
- *Programmer's Manual*
- *System Management Handbook*
- *User's Manual*

The documentation mentioned, except for the Release Notes, are also available on the Mimer SQL Documentation site, <https://docs.mimer.com/>.

Command line help and man pages

For each command provided within the Mimer SQL installation, the options "`-?`" or `--help` can be used to retrieve a basic help text.

In addition, man pages are included in your Mimer SQL distribution. There are man pages for all commands available and for various configuration files, such as `sqlhosts` and `multidefs`. For general information about Mimer SQL, read the `mimersql` man page. Man pages are usually installed automatically at `/usr/local/share/man` when installing Mimer SQL.

Refer to the information provided by your operating system manufacturer concerning the macOS `man` and `manpath` commands.

Useful links

The Mimer SQL Developer Site contains lots of useful information, like FAQ's, Howto's and articles: <https://developer.mimer.com>.

All manuals for Mimer SQL are gathered at <https://docs.mimer.com>.

For general information on Mimer SQL, please see <https://www.mimer.com>.

Chapter 2

Installing Mimer SQL

The Mimer SQL software installation on macOS is expected to be completed in less than a minute, and creating the initial data dictionary and starting the database server will only take just a little longer.

Download a distribution package from <https://developer.mimer.com/downloads> and follow the instructions given below. The Mimer SQL distribution is provided as a PKG package file. It's a Universal Binary installation, i.e. it contains executables that run natively on both Apple silicon and Intel-based Mac computers.

It is really simple to get going!

To get up-and-running with Mimer SQL is usually made in a minute or two. Here is a quick step-by-step instruction using a sample macOS distribution package. The details will be given in the following sections.

You can either use the installed apps to get going, see *Using the Mimer SQL apps* on page 3, or via the macOS Terminal application, see *From the Terminal application* on page 4.

Using the Mimer SQL apps

1 Download package

Download Mimer SQL for macOS from <https://developer.mimer.com/products/downloads/>.

2 Install the package

Install Mimer SQL by double-clicking the downloaded .pkg file, using the Apple Installer.

3 Create a database

Open the `Mimer SQL Database Install` application, obtained when installing Mimer SQL.

4 Control the database

Use the application `Mimer SQL Database Admin` to control the created Mimer SQL database.

5 Access the database

Use the provided `DbVisualizer` app for database access. For example, if the example environment was chosen to be installed, connect to the database using the ident `MIMER_STORE` with password ‘GoodiesRUS’.

From the Terminal application

1 Download package

Download Mimer SQL for macOS from <https://developer.mimer.com/products/downloads/>.

2 Install the package

From the Terminal application, usually located in the Utilities folder, the distribution package can be installed as follows:

```
$ sudo installer -pkg mimersql1107-11.0.7F-41154_macosxi_64.pkg -target /
Password:
installer: Package name is Mimer SQL
installer: Installing at base path /
installer: The install was successful.
$
```

3 Create a database

Run `dbinstall` to create the initial Mimer SQL database named ‘`testdb`’. Specify `-e` to include an example environment:

```
# dbinstall -e testdb
```

4 Control the database

Verify the database server status by using the Mimer SQL administration tool `mimadmin`, which can be used to control the database:

```
# mimadmin testdb
```

5 Access the database

From the Terminal application, use `bsql` to access the database and the example environment as follows, using the ident `MIMER_STORE` with password ‘GoodiesRUS’:

```
# bsql testdb

Mimer SQL Command Line Utility Version 11.0.6C Beta Test
Copyright (C) Mimer Information Technology AB. All rights reserved.

Username: mimer_store
Password:
SQL>select * from categories;
category_id category
=====
1 Music
2 Books
3 Video

3 rows found

SQL>exit;
#
```

Why do we need sudo access to install?

To provide for a complete and proper easy-to-use installation, the procedure when installing Mimer SQL is doing all needed installation actions automatically. This includes updates to operating system locations, such as `/usr/local/bin`, `/usr/local/lib` and `/etc`. For example, the following tasks are handled:

- TCP/IP settings for Mimer SQL client/server access (using the `launchd` daemon with `/Library/LaunchDaemons/com.mimer.mimysqlPortListen.plist`)
- autostart settings for Mimer SQL databases (using the `launchd` daemon with `/Library/LaunchDaemons/com.mimer.mimysqlAutoStart11.plist`)
- desktop menu items
- system wide Mimer SQL database catalog (`/etc/sqlhosts`)
- system wide ODBC data source catalog (typically `/etc/odbc.ini` and `/etc/odbcinst.ini`)
- system wide Mimer SQL man-page setup (`/usr/man`, or `/usr/local/share/man`)
- easy access for Mimer SQL programs and libraries (`/usr/local/bin` and `/usr/local/lib`)

To achieve this, the installation requires sudo access (administration authority), or it has to be executed as root.

System resources

On an x86-64 based computer, at least macOS 10.15 (Catalina) has to be used.

On an ARM based computer, at least macOS 11.0 (Big Sur) has to be used.

Physical memory

The amount of physical memory used by the database server process is determined by parameters in the local database definition, whose initial default values are determined by looking at the amount of installed memory.

See *The multidefs parameter file* on page 17 for further details.

Also see the chapter *Managing a database server* in the *Mimer SQL System Management Handbook*, for the use of system resources.

Virtual memory

The amount of virtual memory that the database server process can use is limited by the operating system.

Environment

Which components will be installed?

The complete Mimer SQL distribution contains the following:

- Tools, libraries, examples, man-pages, etc.
- A complete documentation set in PDF format.
- An ODBC Driver, available in the `libmimodbc` shared library - see the chapter *Mimer SQL and the ODBC API*, in *Programmer's Manual*. This driver can be used for direct access to a Mimer SQL database, or it can be used with a third party ODBC Driver Manager, for example unixODBC or iODBC.
- A JDBC Driver, type-4, written in 100% Java - see the chapter *Mimer SQL and the JDBC API*, in *Programmer's Manual*.
- Various other database API's, like Embedded SQL, Module SQL and a native Mimer C API.
- GUI application for database installation, called Mimer SQL Database Install.
- GUI application for database administration, called Mimer SQL Database Admin.
- GUI application for database access and interaction, called DbVisualizer.
- GUI application for the PSM debugger.

The default installation location is `/opt`, where a sub directory named according to the package is created. For example, if Mimer SQL 11.0.0A is installed, an installation path like `/opt/mimersql1100-11.0.0A` is used. This Mimer SQL main installation directory then contains the following sub directories:

- `bin` - contains Mimer SQL tools, and other executable files.
- `doc` - contains Mimer SQL documentation.
- `examples` - contains example files.
- `include` - contains various header files that may be needed when developing with Mimer SQL.
- `lib` - contains library files.
- `man` - contains Mimer SQL man pages.
- `misc` - contains various additional files, like desktop menu system resources.

Additional Python interface

The Python interface towards Mimer SQL is downloaded and installed separately using the following command:

```
python -m pip install mimerpy
```

Additional PHP/PDO interface

Read about the details of the PHP/PDO interface to Mimer SQL in the article *PDO Driver for Mimer SQL*, located as <https://developer.mimer.com/article/mimer-sql-driver-for-pdo/>.

Running several Mimer SQL versions in parallel

If it is desirable to run two or more Mimer SQL versions in parallel on a host computer, this is fully feasible, but it requires a system knowledge and may involve manual measures. Be sure to do a proper setup of related environment variables to point to the version currently targeted, see *Environment Variables* on page 33.

Mimer SQL license key

To start the installed database server and to establish connections to the database, a license key is required. A key valid for development and evaluation only is included in the Mimer SQL distribution. This key is usually installed automatically during installation of the Mimer SQL package.

Whenever a user connects to a Mimer SQL database, the computer identification and the license key will be checked by the database server to determine access rights. If access is denied, the connect attempt will be aborted and an error message will be shown.

The Mimer SQL license key contains the following (encrypted) information:

- The maximum number of users that may use the database servers running on the same computer node at any one time.
- The maximum number of network users that may use the database servers running on the same computer node at any one time.
- The functionality modules which the key is valid for.
- The ID of the computer (in the case of a specific key) or a lifeboat key which is valid for any computer of the platform type for which it was issued (e.g. any macOS machine).
- Version number.
- Expiration date for the key.

The key data is case insensitive and space characters are ignored.

The `mimlicense` application is used to administrate the license key file. See *MIMLICENSE - Managing the license key* in *System Management Handbook* for information on how to use it. The following command will list the licenses installed:

```
# mimlicense -l
```

As mentioned above, for a production system a commercial license is required. A new key will also be needed if the key expires, if the number of Mimer SQL users is increased or if new Mimer SQL functionality is added.

The Mimer SQL license key is provided by your Mimer SQL distributor. In order to be able to generate the key, your Mimer SQL distributor must know the ID of the computer on which the database server will run.

The ID of a macOS machine is obtained by using the following command:

```
# mimlicense -i
```

Creating an initial database

Once the software is installed, the next step is to build a Mimer SQL database by using the `dbinstall` command. (On macOS there is also a GUI frontend application for the `dbinstall` command, named Mimer SQL Database Install. Here follows the description of the command line command.)

As mentioned before, the `dbinstall` command requires sudo access, or must be executed by root. If not started from a privileged shell sudo password will be asked for:

```
# dbinstall [<database name>]
```

If a database name is given, the `dbinstall` session is completed with default settings used as far as possible. Otherwise, during the `dbinstall` session, database name, database location, and password for the database administrator (i.e. SYSADM) will be asked for. There will also be options for installing example environments, etc. When the session is completed, a fully operational database is available - enabled for client/server access over TCP and automatic start at reboot.

Note: `dbinstall` creates all system databank files in the given database server home directory. In a production system it is recommended that the SYSDB, TRANSDB and LOGDB files are located on separate disks due to performance and reliability reasons. You can read more about this in the *Mimer SQL System Management Handbook* part of the *Mimer SQL Documentation Set* (found at the Documentation page).

Once the database is up and running it may be of interest to provide for remote access. To achieve this the database should be registered as a REMOTE on each node in the network from which it is to be accessed - see more on database registration below.

Now the database is ready for data storage, creating a storage structure built on idents and data objects using the data definition statements in Mimer SQL. See the article *The Example Database*, located as <https://developer.mimer.com/article/the-example-database/>, for an example on using various database elements.

To summarize, the `dbinstall` command performs all necessary installation steps to create an initial database and getting it up and running. The options available in `dbinstall` give opportunities to control and carry out the following:

- Deciding a database home directory
- Registering the database
- Deciding the SYSADM password
- Creating the system databanks, including the data dictionary
- Deciding owner of the database
- Setting up the networking environment
- Setting up autostart procedure
- Setting up a data source definition for ODBC use
- Creating an example database
- Creating a basic development setup with a user that has an OS_USER login (see the *Mimer SQL Reference Manual* for details)
- Creating the default database configuration file
- Starting the database created

Many of these tasks are described in a more general and detailed manner further on in this document.

Upgrading an existing database

If you are upgrading an existing database from an earlier version of Mimer SQL, please see the *Mimer SQL Release Notes* for detailed information. The Release Notes document is provided within each Mimer SQL distribution package. In short the steps are as follows:

- 1 Install the new Mimer SQL version in parallel with your existing Mimer SQL.
- 2 Stop the database.
- 3 Make sure the new Mimer SQL version is the one accessed, and run the `sdbgen -u` database command from the new Mimer SQL version.
- 4 Start the database with the database server program from the new Mimer SQL version.

Uninstalling the software

What happens to the databases?

The commands described below will remove the given software installation, but any databases using the installation will remain intact. Since databases may contain valuable data, the removal of databank files is not performed unless an explicit call to `dbuninstall`, specifying removal of data, is done.

If a database, and its databank files, is going to be removed, use the `dbuninstall` command. When executed, a question will be raised asking if specified database should be removed, i.e. permanently deleted.

```
# dbuninstall <database_name>
```

Database registration

The `sqlhosts` file

The database registration file is used to list all the databases that are accessible to a Mimer SQL application from the node on which it resides. All users must have read access to the `sqlhosts` file on the machine they are using in order to run applications and utilities accessing Mimer SQL databases. The standard location for this file is `/etc/sqlhosts`. By using the environment variable name `MIMER_SQLHOSTS`, another file can be used.

In a network environment, the name of a database must be registered on each node from which it is to be accessed. A database is created as a local database on the node where it resides, and it is defined as a remote database on each other node in the network from which access to it is required. For general information on how to make databases accessible, refer to *Registering the Database in System Management Handbook*.

The program `mimsqlhosts` can be used to manage the contents of the local `sqlhosts` file instead of editing it manually. To list the complete content of the `sqlhosts` file, simply use the following command:

```
# mimsqlhosts
```

When the `dbinstall` command is used to install a local database, an entry for it is automatically added to the LOCAL section of the `sqlhosts` file on that node, see *LOCAL section* on page 12.

If the file is not found, a default `sqlhosts` file is automatically generated. (See the `mimsqlhosts` and `sqlhosts` man-pages).

The `sqlhosts` file structure

The SQLHOSTS file contains three sections; DEFAULT, LOCAL and REMOTE.

The names of the local databases on the current node are listed in the LOCAL section, see *LOCAL section* on page 12, and the names of the remote databases accessible from the node are listed in the REMOTE section, see *REMOTE section* on page 12.

One of the local or remote databases can be set to be the default database for the node by specifying its name in the DEFAULT section, see *DEFAULT section* on page 12.

Database names may, in general, be up to 128 characters long and are case-insensitive.

A line of text beginning with the character sequence -- is interpreted as a comment in the `sqlhosts` file.

The default SQLHOSTS file

When the first Mimer SQL system is installed on a node, the following default sqlhosts file is automatically generated:

```
-- -----
-- S Q L H O S T S
-- =====
--
-- This file contains a list of all databases, local and remote, accessible
-- from the node where the file resides.
--
-- The DEFAULT label
-- -----
-- Name of default database. Can be either a REMOTE or LOCAL database name.
-- Can be overridden by setting MIMER_DATABASE to the name of a database.
--
-- The LOCAL label
-- -----
-- A list of all local databases on the current node, containing the
-- database name and a directory specification (Path).
-- UNIX Path -      database home, and directory path for databank lookup.
-- VMS  Path -      database home.
--
-- The REMOTE label
-- -----
-- A list of all remote databases containing the database name, the database
-- node, the protocol to be used, the protocol interface and the protocol
-- service to be used.
--
-- Protocol, Interface and Service may be defaulted by entering ''.
--
-- Node -      network node name for computer on which the database resides.
-- Protocol -   currently tcp is supported. (tcp or '' should be specified)
-- Interface -  currently not used ('' should be specified).
-- Service -   corresponds to the port number used in TCP/IP. The port number
--              Default is 1360, i.e. the port number reserved for MIMER.
--              On UNIX: The port number may either be a number or a name of a
--              service stored in the /etc/services file.
--
-- -----
-- =====
--DEFAULT:
-- -----
-- Database
-- -----
-- example_localdb
-- -----
-- =====
--LOCAL:
-- -----
-- Database      Path
-- -----
-- SINGLE
-- example_localdb      /directory
-- -----
-- =====
--REMOTE:
-- -----
-- Database      Node      Protocol Interface Service
-- -----
-- example_remotedb  server_nodename  ''      ''      1360
```

DEFAULT section

The DEFAULT section contains a single line that specifies the default database which will be used by a Mimer SQL application or command that does not explicitly specify a database to connect to, see *The Default Database* section in *System Management Handbook*.

The default database should be one of those listed in the LOCAL or REMOTE sections. If defining the `MIMER_DATABASE` environment variable, that setting usually overrides the DEFAULT setting in the sqlhosts file.

LOCAL section

The LOCAL section contains a list of all the local databases residing on the current machine, see *The Local Database* section in *System Management Handbook*.

Each line under the LOCAL keyword should contain two fields, separated by one or more blanks or tab characters. The first field specifies the database name, and the second specifies the location.

The location field is usually a single directory path, referred to as the database home directory. But, it may also be a colon (:) separated search path specification, where each directory included in the path list can hold databank files for the Mimer SQL database server. In that case the first directory in the search path is taken as the database home directory and the other directories in the search path will be used to locate databank files which have a file specification stored in the data dictionary without an explicit directory.

Using a path list is one way to arrange for having databank files on separate disks for optimal performance and reliability - see the *System Management Handbook*.

REMOTE section

The REMOTE section contains a list of all accessible databases that reside on other nodes in the network environment, see the section *Accessing a Database Remotely* in *System Management Handbook*.

Access to these databases is provided by using TCP/IP to establish a client/server connection to the remote machine.

Each entry in the REMOTE section contains up to five fields, separated by spaces and/or tab characters.

The DATABASE field specifies the name of the remote database.

The NODE field should specify the network node name of the remote machine. If the TCP/IP interface is used, the IP address may be specified here.

The PROTOCOL field should specify `tcp` or two single quotation marks ''.

The INTERFACE field is currently not used. Specify '' (two single quotation marks) here.

If using TCP/IP, the SERVICE field specifies the TCP/IP port number the database server uses. The default is 1360, which has been reserved by Mimer Information Technology AB for Mimer SQL client/server communication.

When TCP/IP is used, the value in the SERVICE field may be the actual port number, the name of a service stored in the `/etc/services` file or two single quotation marks '' for the default value 1360.

The remote section parameters are summarized below, depending on the protocol selected. The character sequence ' ' is two single quotation marks, and specifies the default value for a parameter:

Parameter	Explanation
DATABASE	Remote database name
NODE	TCP/IP node name or IP number
PROTOCOL	' ', TCP or tcp
INTERFACE	' ' (two single quotation marks)
SERVICE	TCP/IP_port_number, TCP/IP service name or ' '. (When ' ' is used to specify the default SERVICE, the TCP/IP port number 1360 will be used.)

Chapter 3

The Database Server

The Mimer SQL database server is a single, multi-threaded process with SMP scalability. Clients using TCP/IP can access the server. For clients running on the same platform, a shared-memory based communication method is used, usually called ‘local communication’.

The standard Mimer SQL database server program is named `mimexper` (there is also an in-memory database server available named `miminm`). When running the `dbinstall` command, the database server is automatically created and started - ready for duty.

On macOS there also is a specific GUI application provided to handle the database administration from the computer desktop. This is the Mimer SQL Database Admin app.

Database server management

The database server is usually controlled using the `mimadmin` command, the `mimdbserver` command or the `mimcontrol` command.

mimadmin

This is a menu based front-end tool involving different sub-commands to do various database server administration, like;

- Controlling a database server
- Monitoring a database server
- Managing database server registration
- List started database servers
- List installed license keys

The `mimadmin` command is used as follows:

```
# mimadmin [database]
```

If the database name is omitted, the setting of the `MIMER_DATABASE` environment variable is used. If `MIMER_DATABASE` isn’t defined, the database in the default section of the `/etc/sqlhosts` file is used.

See the `mimadmin` man-page for details.

mimdbserver

This is a command line based front-end tool involving different sub-commands. It handles the following operations:

- Controlling a database server
- Monitoring a database server

As an example, to manually start the database server, use the command as follows:

```
# mimdbserver -s [database]
```

If the database name is omitted, the setting of the `MIMER_DATABASE` environment variable is used. If `MIMER_DATABASE` is not defined, the database in the default section of the `/etc/sqlhosts` file is used.

See the `mimdbserver` man-page for details.

mimcontrol

The two commands described just above are front-end tools involving other commands. As can be seen in *macOS Commands* on page 35, there are many commands that can be used on their own to administer the database system. For example, the bottom-line tool for controlling a database server is the `mimcontrol` command.

As an example, to stop the database server program, use the following command:

```
# mimcontrol -t [database]
```

If the database name is omitted, the setting of the `MIMER_DATABASE` environment variable is used. If the database name is omitted and the `MIMER_DATABASE` is not defined, the command will not work.

See the `mimcontrol` man-page for details.

Database home directory

The database home directory is the catalog where the `SYSDB` system databank file resides. This path is registered in the `sqlhosts` file, usually located as `/etc/sqlhosts`. By using the environment variable `MIMER_SQLHOSTS`, another file can be pointed out as being the `sqlhosts` file.

The database home directory can be located using the following command:

```
# mimpath <database name>
```

The Mimer SQL system databank `SYSDB` file will be located in the database home directory and other databanks will typically be located relative to it, see *Locating Databank Files in System Management Handbook*.

Logging database events

Database events are written to the `mimer.log` file, located in the database home directory.

The following command can be used to list the log-file:

```
# mimdbfiles -L <database name>
```

Configuring a database server

The configuration file for an installed Mimer SQL database server is named `multidefs` and is located in the database home directory.

The content of the configuration file can be seen by using the command:

```
# mimdbfiles -C <database name>
```

The multidefs parameter file

The `multidefs` file holds the parameters adjustable for a database server. It is automatically created when creating the database using the `dbinstall` command. A default setup is made, but further configurations can be made manually if needed. Refer to the *Mimer SQL System Management Handbook* or open a discussion with Mimer SQL support representative.

If the `multidefs` file is not found when starting a database server, a new file will be created using the default values for all parameters. The actual default values used may vary and may depend on factors like machine type and the amount of physical memory available on the machine.

The `multidefs` settings can be modified after the database is created, and will be taken into account at the next server startup.

The following is an example of a default multidefs parameter file:

```
-- Mimer SQL version 11.0.7F Beta Test parameters generated 2023-03-17 00:01
Databases          100      # Max # of databanks (20-1000)
Tables             4000     # Max # of tables (500-1000000)
ActTrans           20000    # Max # of active trans (500-1000000)
SQLPool            1000     # Initial SQLPool (400-8388607 kB)
RequestThreads     8        # # of request threads (1-100)
BackgroundThreads  3        # # of background threads (1-100)
TcFlushThreads    1        # # of t-cache flush threads (0-20)
Users              100      # Max # of logged in users (1-5000)
DBCheck             1        # DB check [0=index, 1=all, 2=immediate,
                           3=im. index, 4=im. all] (0-4)
Pages4K             203347   # # of 4K bufferpool pages (11-2147480000)
Pages32K            18465    # # of 32K bufferpool pages (7-2147480000)
Pages128K           2151     # # of 128K bufferpool pages (0-2147480000)
DelayedCommit        0        # Delayed commit [0=Off, 1=On, 2=Disabled]
                           (0-2)
DelayedCommitTimeout 100     # Delayed commit timeout in milliseconds
                           (0-60000)
GroupCommitTimeout  2        # Group commit timeout in milliseconds (0-20)
Oper                .        # Receivers for messages
DumpPath             .        # Path for dump directory
TCPPort              inetd    # TCP/IP port
MaxSQLPool           216000   # SQLPool max size (2400-16777215 kB)
NetworkEncryption    1        # Client/server encryption [0=Off,
                           1=Optional, 2=Required] (0-2)
MemLock              0        # Lock bpool in memory [0=Off, 1=On] (0-1)
MiniDump             1        # Small bufferpool dump (no page content)
                           [0=Off, 1=On] (0-1)
BackgroundPriority   0        # Thread priority [0=Off, 1=On] (0-1)
AutoStart             1        # Autostart [0=Off, 1=On] (0-1)
DumpScript            ./dumper.sh %p # Dump Script
IOQueue              1024     # Max # of concurrent I/O requests (0-65535)
ServerType            3        # Server type [3=mimexper, 7=miminm] (3-7)
```

Comments in the file are introduced by the character sequence `--`, or by the character `!` or `#`.

A new multidefs file can also be generated manually. If no multidefs file is located in the database home directory, the following command will generate a new one, having the default values:

```
# mimdbserver -g <database name>
```

The parameters in multidefs

Parameter	Definition
Databanks	Specifies the maximum number of databank files that the database server can have open at any one time.
Tables	Specifies the maximum number of tables that can be accessed simultaneously by the database server.
ActTrans	Specifies the maximum number of transactions that can be active in the database server
SQLPool	Initial size of the SQLPool area in K bytes. This area contains information about each session, i.e. opened tables and databanks, compiled SQL programs, etc. The SQLPool area will expand automatically if it is too small, but it will not be larger than MaxSQLPool.
RequestThreads	The number of threads in the database server that can serve client requests.
BackgroundThreads	The number of background threads in the database server.
TcFlushThreads	Extra threads that run in the background to help clear the transaction cache. This is beneficial for systems with long-running transactions. The thread keeps the size of the transaction cache down by deleting records that are no longer used. When there are no long running transactions the cache can be cleared efficiently without scanning the cache so in this case the thread is not needed. Default is 1 thread. To get the same behavior as in Mimer SQL version 10.0, specify 0 threads for this parameter. For very large databases with long-running transactions more than 1 thread can be used.
Users	The maximum number of users that are allowed to connect to the database server. This parameter should not exceed the number of users specified in the Mimer SQL license key. This number is also used to calculate the size of the shared memory region used for local database server communication. About 70 Kbytes of shared memory will be allocated for each user.

DBCHECK	<p>A number which specifies what kind of check that should be performed when a databank is opened which previously was not closed properly.</p> <p>0 - check index pages</p> <p>Index pages only are checked in the foreground while applications that access the databank waits for the operation to complete.</p> <p>1 - check data pages</p> <p>A full databank check (involving index and data pages) provides for more secure operations, but may take much longer to execute than an index page check. When a full check is done, the index pages are checked in the foreground and the data pages are checked in the background so there is a smaller effect on performance.</p> <p>2 - Immediate restart, no check</p> <p>This option performs no checking when the file is opened. The system still verifies the integrity of each page through a checksum. A few pages may have been pre-allocated and these are not reclaimed when this option is used. If the option is subsequently changed these pages will be reclaimed the next time the databank is opened.</p> <p>3 - Immediate restart, check index pages</p> <p>This option performs a check of all index pages in the databank in the background. This is done concurrently with other operations on the system.</p> <p>4 - Immediate restart, check all pages</p> <p>This option performs a check of all pages in the databank in the background. This is done concurrently with other operations on the system.</p> <p>The Immediate restart options require a license key module called 'Imm Restart'. Databank checks can be avoided by always shutting down the database server properly with the <code>mimcontrol/mimdbserver</code> command, especially prior to shutting down the machine.</p>
PAGES4K	The number of 4 Kbytes pages in the bufferpool area containing pages from the databank files. The default value of this parameter is 12.5% of the total RAM memory in the machine.

Pages32K	The number of 32 Kbytes pages in the bufferpool area containing pages from the databank files. The default value of this parameter is 8.33% of the total RAM memory in the machine.
Pages128K	The number of 128 Kbytes pages in the bufferpool area containing pages from the databank files. The default value of this parameter is 5% of the total RAM memory in the machine.
DelayedCommit	<p>This option controls how quickly a transaction commit is secured on disk. It greatly affects the performance of the database server. For example, if a single user commits two transactions in quick sequence the database server may use a single I/O to secure both transactions when delayed commit is on. Transactions are never reordered by using the delayed commit option. I.e. it is not possible for a later transaction to be secured on disk before an earlier one. The database is thus always returned to a consistent state after a machine crash. However, if a transaction has been committed but not yet written to disk it will be lost if the database server or machine goes down in an uncontrolled fashion. Transactions that use the XA transaction protocol are automatically committed with delay commit disabled. The delayed commit option can be set to one of the following:</p> <p>0 - Default off</p> <p>In this mode delayed commit is not used unless a transaction is set to use delayed commit by the application. This is the default.</p> <p>1 - Default on</p> <p>In this mode all transactions where the delay mode has not been explicitly set are delayed. The transaction will be secured within the time-out period specified. If other transactions are committed before the time-out occurs the transactions may be combined into a single I/O to boost performance.</p> <p>2 - Disabled</p> <p>In this mode all transactions are secured to disk immediately and the application will not regain control after a commit until the transaction has been secured. This option overrides any application settings for delay commit.</p>

DelayedCommitTimeout	This specifies the number of milliseconds to wait before the transaction is written to disk. If a value of zero is specified transactions are not flushed until the server determines that the commit set page is full. In general, this is not recommended as transactions are likely to be lost if there is an uncontrolled machine stop. Default is 100 milliseconds.
GroupCommitTimeout	How many milliseconds to wait for other transactions to commit before proceeding with first transaction. If another transaction arrives within the timeout period it will be grouped with existing transactions before they are committed together with a single I/O rather. This improves overall performance but the delay prolongs commits time on a system with low load. Default is one millisecond.
Oper	This parameter gives a list of host system users, i.e. operators, or e-mail addresses that should receive e-mail notification of serious problems with the database server.
DumpPath	This parameter may specify an alternate path for the dump directories. The default is to create dump directories under the database home directory.
TCPPort	Specifies how the database server should handle incoming TCP/IP connection requests. If this parameter is set to - (a single dash), the TCP/IP capability will be disabled for the database server. The TCPPort parameter is, by default, set to <code>inetd</code> - which means that the TCP/IP port server program, <code>mimtcp</code> , will be used for establishing a connection to any Mimer SQL database server (of version 8 and later). In this case clients may connect to the port to which <code>mimtcp</code> listens, usually 1360, and the handshake will be passed over to the requested Mimer SQL database server. If a TCP/IP port number is specified, the database server will listen directly to that port.
MaxSQLPool	The maximum size (in kilobytes) of the SQLPool. The SQLPool memory area grows dynamically, but the size will never exceed this parameter. Use this parameter to control the maximum virtual size (maximum page file usage) for the database server process.

NetworkEncryption	<p>Controls the use of encryption of network communication over TCP/IP between server and clients.</p> <p>0 = Network encryption disabled</p> <p>Network encryption is not supported or not used.</p> <p>1 = Network encryption preferred</p> <p>Network encryption is enabled for version 11 clients. Older clients use unencrypted network communication. When this setting is used, older clients without support for network encryption are allowed to communicate with the database server over TCP/IP.</p> <p>Use this option when there is a mix of older and newer clients that communicate with the database server over TCP/IP.</p> <p>This is the default value.</p> <p>2 = Network encryption required</p> <p>The database server requires all clients to use encrypted communication when communicating over TCP/IP.</p> <p>Clients that do not support encryption are rejected at login with error code -18531.</p> <p>This option is recommended over option 1 when possible (i.e. when there are no older clients that need to be supported.)</p>
MemLock	A number which specifies whether the bufferpool and communication buffers should be locked in memory (1) or not locked in memory (0).
Minidump	Small bufferpool dump (no page content), if enabled.
BackgroundPriority	Specifies if the background threads should run at a higher priority than other server threads. During certain circumstances like in situations where the background threads cannot manage to shorten a transaction queue this can be an alternative.
AutoStart	<p>By default, this parameter is set to 1 which indicates that the database should be started automatically when the operating system goes into multi-user mode.</p> <p>If the parameter is set to 0 the database will not be started automatically at a computer restart.</p>

DumpScript	If the database server goes into an erroneous and unrecoverable state, it will produce dumps of the current internal database structures before it goes down. If this situation occurs, it is of great importance for the error detection process to get a macOS kernel stack trace from the location where the error was located. By defining this parameter to a command that can produce a kernel trace stack information will be automatically generated to mimer.log. The %p option used in the example setting in the beginning of this section, is used to get the current process ID as a parameter to the command given.
IOQueue	Specifies the maximum number of concurrent IO requests queued to the operating system. Default is 1024, but more advanced disk systems such as SAN's, battery backed caching IO controllers, PCI Express connected SSD's and NVMe SSD's can make use of larger queues. This can give a significantly higher database performance, but specifying a too large queue can overload the IO subsystems. Maximum queue length is 65535.
ServerType	This option decides which Mimer SQL database server program that should be started to operate the database files for the database: 3 - mimexper The Mimer SQL Experience database server. This is the standard database server. 7- miminm The Mimer SQL In-memory database server.

Automatic database start and stop

When Mimer SQL is installed, autostart is automatically enabled. The `mimservers` program will start or stop all local Mimer SQL v11.0 servers defined in the `sqlhosts` file. The setup is done by using the `mimautoset` command, invoked during installation. For details, see the man-page for `mimautostart`. To exclude a server from the automatic start/stop procedure, set the `AutoStart` parameter in the `multidefs` file for that server to 0. To see the autostart installation made, the following command can be used:

```
# mimautoset -lv
```

The `mimservers` command is used to manage all database servers installed. The following command will list the state for all database servers:

```
# mimservers -b
```

Remote database access

Database TCP/IP connect dispatcher

When a Mimer SQL database is created using the `dbinstall` command the definitions needed for remote access to the database is installed automatically.

When the setup is made, the `mimtcp` command is invoked by the operating system when an incoming Mimer SQL database connection request is identified on the target TCP/IP port. It finds out the database name in the handshake message and redirects the connection to the target database using the registered information in `/etc/sqlhosts`.

To see the setup made, the following command can be used:

```
# miminetd -l
```

Note: It is possible to let a Mimer SQL database server listen directly to a TCP/IP port, i.e. not using the `mimtcp` redirecting function. This is achieved by changing the `TCPPort` parameter in the `multidefs` file from the default `inetd` value to the actual port number used, usually 1360.

The `mimtcp` command

The `mimtcp` command is used to handle the handshake between a remote client and a database server. It should be used with, and be invoked by, an Internet Service Daemon - see *Networking Setup* on page 25.

Syntax

The overall syntax for MIMTCP is:

```
mimtcp [-l [-f filename]
mimtcp [--log [--file filename]
mimtcp [-v|--version] | [-?|--help]
```

Command-line Arguments

You can use the following arguments with MIMLOAD.

Argument	Function
<code>-l</code> <code>--log</code>	Enable logging.
<code>-f file</code> <code>--filename=file</code>	Define a log file.
<code>-v</code> <code>--version</code>	Display version information.
<code>-?</code> <code>--help</code>	Show help text.

If `mimtcp` is used without any option, no logging is performed by the program. If a string value is given in addition to the `-l` option, that value will be used as the log file. If the `-l` option is used without a value, the filename `mimtcp.log` will be used that will end up in the root home folder under a sub directory called `.mimer_log`.

The following example will start `mimtcp` with logging using the default log file, located in `~/.mimer_log/mimtcp.log`:

```
# mimtcp -l
```

Services setup

The `/etc/services` file holds the Internet network services list. The list is a mapping between names for internet services, and their underlying assigned port numbers and protocol types. The following excerpts from the file shows the header for the list and the `mimer` entries:

```
# Port Assignments:  
#  
# Keyword Decimal Description References  
# ----- ----- -----  
mimer      1360/tcp # MIMER  
mimer      1360/udp # MIMER
```

Having this definition done, the name `MIMER` can be used instead of 1360 when dealing with services.

Networking Setup

The `miminetc` command is used to handle the networking setup. This is done automatically during the installation.

Here are the `miminetc` options and a common setup display:

```
# miminetc "-?"  
Usage:  
  miminetc [-i|-l] [-q|-d]  
  miminetc [-v|-?]  
  
Options:  
  -i      Install.  
  -l      List status.  
  -q      Quiet operation.  
  -d      More detailed, verbose operation.  
  -v      Display version information.  
  -?      Display this usage text.  
  
Description:  
  Command used to integrate Mimer SQL in the operating system  
  Internet services like systemd/inetd/xinetd/launchd.  
#  
# miminetc -l  
miminetc: launchd: existing  
/Library/LaunchDaemons/com.mimer.mimersqlPortListen.plist is up-to-date  
miminetc: launchd: Mimer SQL is installed in the launchd environment  
(/Library/LaunchDaemons/com.mimer.mimersqlPortListen.plist)  
miminetc: launchd: daemon is executing  
miminetc: launchd: daemon found as /sbin/launchd  
#
```

Using odbc.ini data sources

The standard ODBC `odbc.ini` file and the Mimer SQL `sqlhosts` file are related to each other in both being repositories for databases, or data sources. When using ODBC to connect to a Mimer SQL database, data source names (DSN) defined in the `odbc.ini` file can be used. In this case the `odbc.ini` file is accessed first, and only if needed the ordinary database lookup is done in the `/etc/sqlhosts` file.

When a Mimer SQL database is created using the `dbinstall` command, it gets defined in the `sqlhosts` file in the `LOCAL` section. For example, if creating the database named `my_db` with the home directory `/Library/Application Support/MimerSQL/my_db`, it will end up in `/etc/sqlhosts` like this:

```
LOCAL:
my_db          /Library/Application Support/MimerSQL/my_db
```

If an ODBC Driver Manager is installed, there will also be an option to automatically define it in the global `odbc.ini` file, usually located as `/etc/odbc.ini`. Such a definition will look like the following:

```
[my_db]
Driver      = /usr/local/lib/libmimodbc.dylib
Database    = my_db
Host        = localhost
Port        = 1360
Trace       = No
TraceFile   = /tmp/mimersql.log
```

We can now look at a simple example where the Perl DBI/DBC-ODBC interface is used to connect to a Mimer SQL database:

```
#!/usr/bin/perl -w
use DBI;

$data_source="dbi:ODBC:my_db";
$username="sysadm";
$auth="sysadm_password";
$dbh = DBI->connect($data_source, $username, $auth) or die $DBI::errstr;
print "Connected! ($dbh)\n";
```

In this case the `my_db` definition in the `odbc.ini` file will be used, more precisely the attributes `Driver`, `Database`, `Host` and `Port` are used:

Driver	The ODBC driver to be used, specific to each database kind. For Mimer this is the <code>libmimodbc.dylib</code> shared library.
Database	The name of the database to be accessed, as defined in the <code>sqlhosts</code> file on the node where the database resides.
Host	The name of the computer node where the database resides. If this attribute is left out, the value of the <code>Database</code> attribute will be looked up in the <code>/etc/sqlhosts</code> file for further information about the connection setup.
Port	The port number to be used for the database communication. If this attribute is left out, the default '1360' will be assumed.

Assuming a Mimer SQL database on a remote computer is defined in the REMOTE section of the sqlhosts file as follows:

```
REMOTE:  
prod_db      typhon.mimer.se      tcp      ''      1360
```

Also, assuming we have the following DSN defined in the `odbc.ini` file:

```
[remote_prod]  
Driver = /usr/local/lib/libmimodbc.dylib  
Database = prod_db
```

To connect to the `prod_db` database on the `typhon.mimer.se` node using the program example above, we can simply change the data source definition in the program above to:

```
$data_source="dbi:ODBC:remote_prod";
```

The data source `remote_prod` will be looked up in `odbc.ini`. The database name `prod_db` will be encountered, but there is no host defined so an attempt will be made to find appropriate connection information for the given database in the `sqlhosts` file. When the node `typhon.mimer.se` and the port 1360 are identified for the database name, the connection will be completed.

The `ODBCINI` environment variable can be used to point out the `odbc.ini` file to be used.

Note: Tabs are not allowed in the `odbc.ini` file.

Chapter 4

Development and Example Environments

When installing Mimer SQL, there are options to install an initial development setup and an example database. The example database is described in detail in the article *The Example database*, found as <https://developer.mimer.com/article/the-example-database/>.

If these parts were not installed during the `dbinstall` session, they can be installed separately when needed. For the initial development setup, use the following command:

```
# mimdevenv <database name>
```

For the example database, use the following command:

```
# mimexampledbs <database name>
```

To access the example environments, you can use:

DbVisualizer	The GUI database front-end included in the installation package and located in the desktop setup for Mimer SQL.
bsql	The command line tool, usually available as <code>/usr/bin/bsql</code> .
Any ODBC or JDBC based SQL tool	Standard database access tools using these database API's can be adapted.

The username `MIMER_STORE` and password ‘GoodiesRUs’ (if you have used the default password) is used for the example database.

For further information on the API's described below, and on programming with Mimer SQL in general, please refer to the *Programmer's Manual* within the *Mimer SQL Documentation Set* - <https://docs.mimer.com/MimerSqlManual/latest/>.

Database APIs

Embedded SQL

An embedded SQL preprocessor is included. It enables SQL commands to be embedded in programs written in C, C++ and FORTRAN. The embedded syntax complies with the ISO standard for embedded SQL.

For a proper Unicode behavior, internationalized C/C++ programs must include the `locale.h` header file and call the `setlocale()` operating system function to initiate a specific language operation. This can be done by calling `setlocale()` as follows:

```
setlocale(LC_ALL, "");
```

In the examples directory under the installation path, e.g.

`/opt/mimersql1100-11.0.0A/examples`, there is a basic programming example provided along with a `readme` file named `readme_esql.txt`.

Module SQL

A Module SQL preprocessor is included. It enables separation of SQL code and a host application written in C, FORTRAN, COBOL or Pascal, into different source files, simplifying modularity and reuse of SQL code.

For a proper Unicode behavior, internationalized C programs must include the `locale.h` header file and call the `setlocale()` operating system function to initiate a specific language operation. This can be done by calling `setlocale()` as follows:

```
setlocale(LC_ALL, "");
```

In the examples directory under the installation path, e.g.

`/opt/mimersql1100-11.0.0A/examples`, there is a basic programming example provided along with a `readme` file named `readme_msq1.txt`.

JDBC

For database access from Java a JDBC driver is included in the distribution. The driver is a type 4 driver, which means that it is written entirely in Java. This provides the driver with full portability so that it can be copied or downloaded to any Java enabled platform. The driver uses TCP/IP to access a Mimer SQL server (version 8.2 or later) on any platform. For details on the JDBC drivers, please refer to the *Mimer JDBC Driver Guide*, <https://docs.mimer.com/MimerJdbcGuide/latest/>.

In the JDBC directory under the installation path, e.g.

`/opt/mimersql1100-11.0.0A/examples`, there is a basic programming example provided along with a `readme` file named `readme_java.txt`.

ODBC

The Mimer ODBC driver is a client library that enables applications to access Mimer SQL database servers running on any platform. The driver complies with the ODBC 3.52 specification.

There are various third party ODBC Driver Manager available on the market that enable applications to dynamically load drivers for different database products. But, you can also choose to link your applications directly to the Mimer ODBC driver, without using any Driver Manager. In the latter case we suggest usage of the provided ODBC header files, introduced by including the `mimcli.h` file.

For a proper Unicode behavior, internationalized programs must include the `locale.h` header file and call the `setlocale()` operating system function to initiate a specific language operation. This can be done by calling `setlocale()` as follows:

```
setlocale(LC_ALL, "");
```

In the ODBC environment it can be mentioned that `SQLWCHAR` refers to a four byte type (`wchar_t`).

In the examples directory under the installation path, e.g.

`/opt/mimersql1100-11.0.0A/examples`, there is a basic programming example provided along with a `readme` file named `readme_odbc.txt`.

Mimer SQL C API

Mimer SQL C API is a native C library suitable for tool integration and application development in environments where API standardization is not a requirement. The following characteristics describe the API:

- Simplicity
- Platform independence
- Small footprint
- Tight fit with the Mimer SQL application/database communication model.

This MimerAPI is provided in the `libmimerapi.dylib` shared library. The `mimerapi.h` header file provides prototypes and other handy defines. See Database API article for MimerAPI and the *Mimer SQL Programming Manual* (found in the *Mimer SQL Documentation Set* at the Documentation page.)

For a proper Unicode behavior, internationalized programs must include the `locale.h` header file and call the `setlocale()` operating system function to initiate a specific language operation. This can be done by calling `setlocale()` as follows:

```
setlocale(LC_ALL, "");
```

In the examples directory under the installation path, e.g.

`/opt/mimersql1100-11.0.0A/examples`, there is a basic programming example provided along with a `readme` file named `readme_mimerapi.txt`.

Python

A Mimer SQL database can be accessed from the Python programming language using the MimerPy adapter. This adapter allows the user to connect to Mimer SQL through Python, gaining access to the exceptional performance and powerful features provided by a Mimer SQL database.

For details and programming examples, please see the specific MimerPy guide found as <https://docs.mimer.com/MimerPython/latest/>, and the article at <https://developer.mimer.com/article/mimer-sql-python-adapter/>.

PHP

A Mimer SQL database can be accessed from HTML using the PHP/ODBC interface. PHP is a widely-used general-purpose scripting language that can be embedded into HTML, and is therefore especially suited for web development.

In the examples directory under the installation path, e.g.

`/opt/mimersql1100-11.0.0A/examples`, there is a basic programming example provided along with a readme file named `readme_php.txt`. This example uses the Apache HTTP Server which is an open source HTTP web server for a wide range of platforms.

Also see the article describing the PHP/PDO access method, <https://developer.mimer.com/article/mimer-sql-driver-for-pdo/>.

Accessing the database

Setting up and running DbVisualizer

DbVisualizer is a graphical front-end used to view and manage your database objects. It is started from the desktop using an icon in the Mimer SQL installation menu.

To use DbVisualizer you may need to install Java. Once Java is installed DbVisualizer is started. The very first time DbVisualizer is started two operations are automatically initiated:

- **First the New Connection Wizard is started**

This wizard will set up a connection to the Mimer SQL database:

- 1 It will initially prompt for a name of the connection. A common naming scheme is to use the database name followed by the username in parenthesis. For example 'testdb (MIMER_STORE)'.
- 2 In the second step the database driver should be selected. Select 'Mimer' from the drop down list.
- 3 In the third step you fill in the database name, username, and password used when accessing the Mimer SQL server. Enter the name of the target database. If the Mimer SQL Example database is installed, the username MIMER_STORE and password 'GoodiesRUs' (if you have used the default password) can be used. If your database is on another computer remember to change the Server field to the name of the computer. Before proceeding, make sure you test that your connection is working properly.

4 The wizard is now completed and various objects in the target database can now be explored by selecting them in the tree view to the left. Note that existing objects can be modified and new ones can be created by right-clicking on the objects or object types.

- **Secondly, when DbVisualizer is invoked for the first time**

The Driver Finder will locate the Mimer SQL JDBC Driver. Unless errors have occurred, this dialog can simply be closed.

Running Mimer BSQL and other utilities

In order to run most of the Mimer SQL utilities from a command prompt window, a target database to access must be specified. This can be furnished in different ways:

- Enter the database name on the command line, e.g.:

```
# bsql database_name
```
- As mentioned before, use the environment variable `MIMER_DATABASE`, e.g.:

```
# export MIMER_DATABASE=database_name
# bsql
```
- Use an ODBC data source. When installing a database using the `dbinstall` command, the default option is to define the database as an ODBC DSN (if such an ODBC environment is present).

The order of the three methods is significant as the first methods override the later ones. For example, specifying the database on the command line overrides the setting of the `MIMER_DATABASE` environment variable.

Environment Variables

The following table lists and explains the environment variables Mimer SQL uses.

Variable	Explanation
<code>HOME</code>	Used to locate the home directory from within various Mimer SQL programs.
<code>LD_LIBRARY_PATH</code>	Used on most platforms to locate shared libraries in runtime. On macOS the following resources are used until it finds a compatible Mach-O file: <code>\$LD_LIBRARY_PATH</code> , <code>\$DYLD_LIBRARY_PATH</code> , current working directory, <code>\$DYLD_FALLBACK_LIBRARY_PATH</code> .
<code>MIMER_DATABASE</code>	Used to point out which database to access. If not set, the default database, set in <code>/etc/sqlhosts</code> , is used.
<code>MIMER_EXTEND</code>	Used to change the number of pages to allocate when dynamically extending a databank file. If not set, the default is 128 pages (each of 4096 bytes). The variable must be set for the process starting the database server program.

Variable	Explanation
MIMER_HISTLINES	Used to change the number of command lines to be stored in the recorded history for a Mimer BSQL session. If not set, the default is 23.
MIMER_KEYFILE	If set, the given string is treated as the name of the license key file (overriding the <code>/etc/mimerkey</code> file).
MIMER_MODE	Used to indicate the mode for which the database should be accessed, that is, <code>SINGLE</code> or <code>MULTI</code> . Use single mode if accessing a database for which the database server program is not started. If not set, <code>MULTI</code> is assumed.
MIMER_NOEDIT	If set, the command line editing package for a Mimer BSQL session is disabled.
MIMER_ODBCINI	If set, the given string is treated as the name of the file for ODBC Data Source lookup. If not set, and if <code>ODBCINI</code> is not set, the home directory is searched for the <code>.odbc.ini</code> file (using the <code>HOME</code> environment variable).
MIMER_SQLHOSTS	The default <code>sqlhosts</code> file is <code>/etc/sqlhosts</code> . Another file can be used by defining the <code>MIMER_SQLHOSTS</code> environment variable to hold the path of the target <code>sqlhosts</code> file.
ODBCINI	Same as <code>MIMER_ODBCINI</code> . Overrides <code>MIMER_ODBCINI</code> if set.
PATH	Used to locate Mimer executables.
SHELL	Used shell when temporarily entering the operating system shell prompt from within Mimer SQL. If not set, <code>/bin/sh</code> is used.
TMPDIR	If set, it is used as the placeholder for temporary files created by Mimer SQL. If it is not set, the directory <code>\$HOME/.mimer_tmp</code> is used.

macOS Commands

Command	Function	Used by
bsql	SQL command interpreter. See <i>Mimer SQL User's Manual, Chapter 9, Mimer BSQL</i> for more information.	mimdevenv, mimexampledb
dbc	Databank check utility. See the chapter <i>Databank Check Functionality</i> in <i>System Management Handbook</i> for more information.	
dbfiles	Lists the databank files for a database server, as stored in the data dictionary.	mimdbfiles
dbinstall	Command used to install a new database, or update an existing one.	mimexampledb
dbopen	Opens all user defined databanks at once. See <i>Chapter 7, Databank Open Function</i> in <i>System Management Handbook</i> for more information.	
dbuninstall	Command used to remove a database, including its data files, registrations and related resources.	
esql	Embedded SQL preprocessor. See <i>Mimer SQL Programmer's Manual, Chapter 4, Embedded SQL</i> for more information.	
exload	Program used to create or delete the example environment. (I.e. MUSIC_STORE, see https://developer.mimer.com/article/the-example-database/)	dbinstall, mimexampledb
mimaddpath	Used to add a value to an environment variable (with duplicate check). The new definition is displayed – not installed. Especially used to update the shared library search path.	(internal use)
mimadmin	Menu-based database server administration utility.	
mimautoset	Switches on/off the automatic server start and stop functionality or gives the current state of this interaction with the operating system.	dbinstall

Command	Function	Used by
mimcontrol	Manages database servers. See <i>Chapter 4, Managing a Database Server in System Management Handbook</i> for more information.	dbinstall, mimadmin, mimlistdb
mimdbfiles	Lists the databank file names for a database server, as stored in the file system. Can also be used to change the ownership of the databank files (i.e. the new owner will be the one that is dedicated to manage the database server). In addition, the command can be used for displaying the database server log and configuration files.	dbinstall, dbuninstall, mimadmin
mimdbserver	A front-end to the mimcontrol and miminfo programs, used to control and monitor the database server.	dbinstall, dbuninstall, mimadmin, mimdbfiles
mimdevenv	Command used to create a beginner's development environment.	dbinstall
mimdumper	Creates or executes the <code>.dumper.sh</code> file for a database server. The functionality is used to get detailed operating system info about the process where the database server program is executed, especially in the case of a system failure.	dbinstall
mimexampledb	Installs the example database environment. (Invokes the <code>exload</code> program.)	
mimexec	<code>mimexec</code> command is used to execute a given program and stay attached. This command is used internally, especially when invoking terminal based programs using icons on the desktop	(mainly for internal use)
mimexper	The Mimer SQL Experience database server program. You start <code>mimexper</code> using the <code>mimcontrol</code> or <code>mimdbserver</code> commands.	<code>mimcontrol</code>
mimhome	Displays the home directory for the effective user. Especially used to find location for log and tmp files.	(mainly for internal use)

Command	Function	Used by
mimhosts	Program to manage and to do lookup in the <code>/etc/sqlhosts</code> file. See <i>The sqlhosts file</i> on page 9 for more information.	<code>dbinstall</code> , <code>dbuninstall</code> , <code>mimadmin</code> , <code>mimowner</code> , <code>mimdbfile</code> , <code>mimdbserver</code> , <code>mimdevenv</code> , <code>mimexampledb</code> , <code>mimexec</code>
miminstd	Command used to administer Mimer SQL in the operating system Internet services daemons.	<code>dbinstall</code>
miminfo	Program to monitor database servers. See <i>Chapter 4, Managing a Database Server</i> in <i>System Management Handbook</i> for more information.	<code>mimadmin</code>
miminm	Mimer SQL In-memory database server.	
miminstall	Command delivered with the distribution TAR file used to unpack and install Mimer SQL.	
mimjdbcver	Displays the version of the JDBC drivers delivered with Mimer SQL.	
mimlicense	Used to manage the license keys in <code>/etc/mimerkey</code> . See <i>Mimer SQL license key</i> on page 7 for more information.	<code>mimadmin</code> , <code>miminstall</code>
mimlink	Used to link Mimer SQL libraries, man pages and executables to <code>/usr/lib</code> , <code>/usr/man</code> and <code>/usr/bin</code> , respectively.	<code>miminstall</code>
mimlistdb	Lists started database servers.	<code>mimadmin</code> , <code>mimuninstall</code>
mimload	A command line front end to the Mimer SQL Load/Unload functionality.	
mimlocation	Displays the location of the Mimer SQL installation currently accessed.	(mainly for internal use)
mimmem	Lists current limits on memory usage.	
mimodbc	Program used to administer ODBC data sources and ODBC drivers (especially aimed at the managing iODBC data sources, see http://www.iodbc.org .)	<code>mimodbcadmin</code> , <code>mimodbcdm</code>
mimodbcadmin	Menu based ODBC data source and ODBC driver administration	<code>dbinstall</code> , <code>dbuninstall</code>

Command	Function	Used by
mimodbcadm	A front-end to the mimodbc program, used to administer ODBC data sources and drivers.	mimodbcadmin
mimowner	Displays the name of the operating system user that is dedicated to manage a specific database server.	mimadmin, mimdbfiles, mimdbserver
mimpath	Gets the path to databank locations.	dbinstall, dbuninstall, mimadmin, mimdbfiles, mimexampledb, mimowner
mimperf	Used to monitor a running database server.	
mimproc	Lists various system information for a running process.	.dumper.sh
mimrepadm	Program used to administrate the Mimer SQL replication dictionary.	
mimservers	Starts/stops all database servers (of current version) defined in /etc/sqlhosts.	mimautoset
mimsqlhosts	A front-end to the mimhosts program, used to control the database registration file /etc/sqlhosts.	dbinstall, mimadmin
mimstatln	Used to follow and display the source for a symbolic link.	(mainly for internal use)
mimsync	Program used to synchronize replicated Mimer SQL tables.	
mimsysconf	Displays the values of various host system configuration parameters, all related to the Mimer SQL system performance.	
mimtcp	Manages TCP port dispatching, i.e. distributing incoming connect-attempts to the requested database server.	
mimuninstall	Command to uninstall Mimer SQL, if installed via the tar package.	
mimunlink	Command used to remove symbolic links from /usr/bin, /usr/share/man and /usr/lib, previously created by the mimlink command.	mimuninstall
mimversion	Command used to get the installed Mimer SQL version.	mimadmin, mimodbcadmin

Command	Function	Used by
repserver	The Mimer SQL replication server program. See <i>REP SERVER - Replicating the Data in System Management Handbook</i> .	mimrepadm
sdbgen	Command used to create the system databanks for Mimer SQL. See <i>SDBGEN - Generating the System Databanks in System Management Handbook</i> .	dbinstall

macOS Link Libraries

Library	Description
libmimcomm.dylib	This is the shared library used when using Mimer JDBC with local communication, i.e. not via TCP/IP.
libmimdbs.dylib	This is the shared library used when accessing the database server in single user mode. It is automatically invoked when a single user access is identified.
libmimer.dylib	This shared library contains several of the client interfaces supported by Mimer SQL, i.e. Embedded SQL and ODBC.
libmimerapi.dylib	This is the shared library for the Mimer SQL C API.
libmimers.dylib	This is the Mimer setup library used by the unixODBC Driver Manager GUI interface.
libmimmicroapi.dylib	This is the shared library for the Mimer SQL C Micro API.
libmimodbc.dylib	This is the shared library for the Mimer ODBC database interface when the ODBC client is presuming the SQLWCHAR data type being 4 bytes.
libmimsql.dylib	This shared library contains the Mimer SQL database interface used for Embedded SQL client applications.
mimjdbc3.jar	This is the jar file to be used when accessing the Mimer JDBC database interface from a JAVA client using JRE 1.4 or later.
mimsqlxa.o	This object file should be linked in when using the XA functionality.
psmdebug.jar	This is an internal jar file for the PSM Debugger application.

Index

A

ARM 5
autostart 23

B

Big Sur 5
bsql 29

C

C language API 31
Catalina 5

D

database
 establishing 15
dbfiles 35
dbinstall 8, 35
dbopen 35
dbuninstall 9, 35
DbVisualizer 32

E

embedded SQL 30
environment variables 33
esql 35
example database 29
exload 35

I

Intel 3

J

JDBC 30

L

license key 7

M

M1 5
mimaddpath 35
mimadmin 35
mimautoset 23, 35
mimautostart 23
mimcontrol 36
mimdbfiles 36
mimdbserver 36
mimdevenv 36
mimdumper 36
Mimer SQL 1
MimerAPI 31
mimexampledb 36
mimexec 36
mimexper 36
mimhome 36
mimhosts 37
miminetd 25, 37
miminfo 37
miminm 37
miminstall 37
mimjdbcver 37
mimlicense 7, 37
mimlink 37
mimlistdb 37
MIMLOAD
 command-line arguments 24
 syntax 24
mimload 37
mimlocation 37
mimmem 37
mimodbc 37
mimodbcadmin 37
mimodbcdm 38
mimowner 38
mimpath 38
mimperf 38
mimproc 38
mimreadm 38
mimservers 23, 38
mimsqlhosts 9, 38
mimstatln 38

mimsync 38
mimsysconf 38
mimtcp 24, 38
mimuninstall 38
mimunlink 38
mimversion 38
Module SQL 30
multidefs 17

O

ODBC 31

P

PDO 6
PHP 32
PHP/PDO 6
Python 6, 32

R

repserver 39

S

sdbgen 39
services 25
silicon 3
sqlhosts 9
sudo 5

T

TCP/IP 24

U

Unicode 30, 31
Universal Binary 3

X

x86-64 5