Mimer SQL

Getting Started on Linux

Version 11.0

Mimer SQL, Getting Started on Linux, Version 11.0, October 2024

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

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

Contents

Chapter 1

Getting Started

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

Mimer SQL provides small footprint, scalable and robust relational database solutions that conform to international ISO SQL standards. Due to its structural modularity, Mimer SQL 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 covers basic usage for development purposes and enables 10 concurrent users.

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.

For further information see the section *Mimer SQL license key* on page *9*.

Documentation

The Mim*er SQL Documentation Set*, *Mimer JDBC Driver Guide*, *Mimer SQL Release Notes*, and *Mimer SQL Getting Started on Linux* 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/share/man` when installing Mimer SQL.

Refer to the information provided by your operating system manufacturer concerning the Linux `man` and `catman` commands, and the use of the `MANPATH` environment variable.

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 Linux 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 suitable distribution package from https://developer.mimer.com/downloads and follow the instructions given below. DIstribution packages are available as a complete distribution and as a headless server distribution package, without desktop integration etc. Distribution packages are provided in DEB, RPM and TAR formats, respectively.

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 Debian Linux 64-bit distribution package. The details will be given in the following sections.

1 Download package.

Download a Debian Mimer SQL for Linux (64-bit) from https://developer.mimer.com/products/downloads/.

2 Install the package.

Install Mimer SQL, in this case using the Debian package:

```
# sudo dpkg -i mimersql1100_11.0.0A-24298_amd64.deb
```

3 Create a database.

Create the initial Mimer SQL database named testdb, including an example environment:

```
# dbinstall -e testdb xpass
```

The system administrator (SYSADM user) password will be asked for.

4 Control the database.

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

```
# mimadmin testdb
```

5 Access the database.

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 11.0.8E
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/bin`, `/usr/lib` and `/etc`. For example, the following tasks are handled:

- TCP/IP settings for Mimer SQL client/server access (`/etc/inetd.d`, `/etc/xinetd.d` and/or `/etc/systemd/system`)
- autostart settings for Mimer SQL databases (`/etc/init.d`)
- desktop menu items (`/etc/xdg`, `/usr/share`)
- 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/share/man`)
- easy access for Mimer SQL programs and libraries (`/usr/bin` and `/usr/lib`)

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

Prerequisites

When using the RPM package, the RPM package manager environment must be installed. Installation using the RPM package means that a predefined installation is made to a default setup.

When using the DEB package, a package manager environment that can cope with DEB-files must be installed. Installation using the DEB package means that a predefined installation is made to a default setup.

The TAR installation procedure allows for a more flexible installation, but on the other hand, it requires some answers to questions and knowledge of operating system setup. The questions are given with explanations to each installation step and with default options.

The Linux operating system version and kernel version required, along with the target hardware for an installation package, is usually connected to a specific distribution package and information is provided on the Download page of the Mimer SQL Developer site, https://developer.mimer.com/products/downloads/. The most detailed description is obtained by drilling down to a specific package and clicking on its name.

System resources

Physical memory

The amount of physical memory used by the database server process is determined by parameters in the local database definition (see *The multidefs parameter file* on page *19*), whose initial default values are determined by looking at the amount of installed memory.

Virtual memory

The amount of virtual memory that the database server process can use is limited by the operating system. The virtual memory handling on Linux is platform specific - refer to the documentation for the specific Linux operating system you are using. (Often a paging file used).

Environment

Which components will be installed?

The Mimer SQL for Linux distribution is available in two different packages; as a complete installation, and also as a so called *headless package*, especially suitable for dedicated server machines and docker/embedded environments. The headless package is solely command line based and does not include any desktop related parts (such as the DbVisualizer database access tool), or any documentation but the Release Notes.

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.

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.
- `DbVisualizer` - contains all resources for the DbVisualizer tool that is bundled with Mimer SQL. Please note that an even more powerful version, DbVisualizer Pro, can be purchased from https://www.dbvis.com/. The DbVisualizer Pro features are enabled by installing a license key file.
- `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.
- `lib32` - contains 32-bit libraries for execution of 32-bit applications. (Only available in installation packages for 64-bit platforms).
- `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/.

Methods to install

For Linux platforms the software package is currently available in three different shapes:

- **RPM installation**

  This installation type needs the RPM package manager environment to be installed. An advantage is that possible dependencies to other software automatically are verified and arranged for.

- **DEB installation**

  This installation type needs a Debian package manager to be installed. In the example below the `dpkg` command will be used. Dependencies will be verified during this installation.

- **TAR installation**

  This installation type is platform independent and is the one that can be used for all UNIX/Linux platforms matching the requirements. If a customized installation is desired, for example if the software should be located in a non-default location, the TAR installation may be handy. On the other hand, a TAR installation is not integrated in commonly used package managers and must therefore be managed manually. Using the TAR installation method will NOT ensure that needed parts among other operating system packages are up to date.

Using the RPM distribution package

This is the procedure to follow when using an RPM distribution of Mimer SQL.

When using RPM, installed files are fully maintained by the RPM package manager. RPM will keep track of all files installed by RPM, and it will also check that all dependencies to system libraries are available and up-to-date.

To get a short description of an RPM file before installing it, you can use the following command:

```
# rpm -qpid mimersql1100-11.0.0A-24298.x86_64.rpm
```

An example installation using RPM could be as follows:

```
# sudo rpm -i mimersql1100-11.0.0A-24298.x86_64.rpm
```

From the Mimer SQL point of view, the RPM installation is a silent install. RPM can be instructed to be very verbose, by using the `-ivv` switch instead of `-i`, which will display the information known by, and performed by, RPM for the installed package.

You can run the installation procedure without actually installing anything by using the `--test` option as in the following example:

```
# rpm -i --test mimersql1100-11.0.0A-24298.x86_64.rpm
```

If an older version of an RPM package is already installed when a new RPM package is available, the upgrade switch can be used. See the following example:

```
# sudo rpm -U mimersql1100-11.0.0A-24298.x86_64.rpm
```

Note: An upgrade can only be done if the only difference in the package name is the package revision number, in this case 24298. Otherwise the new product is installed using the `-i` option, and then the old package is removed (see below.)

To get a listing of all installed rpm packages the `-qa` switch is used. Combined with `grep` the following command will display all Mimer SQL packages installed with the rpm command:

```
# rpm -qa | grep -i mimer
```

To get details of the package and to get instructions on how to continue, including how to create a database using the `dbinstall`, use the following command:

```
# rpm -qid mimersql1100-11.0.0A-24298
```

For further details about RPM, see the corresponding man-page, or visit the RPM documentation page.

Using the DEB distribution package

For platforms that use the Debian installation format, we suggest the `dpkg` command to be used. This is the procedure to follow when using an DEB distribution of Mimer SQL with `dpkg`.

To get a description of an DEB file before installing it, you can use the following command:

```
# dpkg-deb -I mimersql1100_11.0.0A-24298_amd64.deb
```

An example installation using a DEB file could be as follows:

```
# sudo dpkg -i mimersql1100_11.0.0A-24298_amd64.deb
```

From the Mimer SQL point of view, the DEB installation is a silent install, not displaying any details in operations performed.

Note: Installations that in fact is an upgrade of an existing package is handled automatically by the `dpkg` command.

To get a listing of all installed DEB packages the `-l` switch is used. Combined with `grep`, the following command will display all DEB packages installed for Mimer SQL with the command:

```
# dpkg -l | grep -i mimer
```

On your Debian linux system you can do `man dpkg` for more information on that command.

Using the TAR distribution package

This is the procedure to follow when using a TAR format distribution of Mimer SQL.

Unpack the distributed TAR archive by using a standard tar extract command, for example:

```
# tar xvf mimersql1100-11.0.0A-24298_linux26_64.tar
```

A subdirectory named according to the distribution has now been created in the current directory, holding the tar archive contents. There, the `miminstall` command is available, which should be executed to install the Mimer SQL software. Simply execute the command as follows:

```
# ./miminstall
```

During the miminstall session the license agreement should be accepted, a temporary location for unpacking should be chosen, and then the location for the Mimer SQL software in the file system should be specified.

The tar installation can also be executed in silent mode, mainly aimed for embedded installations.

Note: The `miminstall` command can be executed in a non-operational mode by using the `-n` option, meaning it only prints information about the installation steps without performing them.

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. The following must be regarded:

- If installing an RPM version while a tar installation is already made, it may be the case that host global configuration files such as `/etc/sqlhosts` for database registration and `/etc/mimerkey` for the license keys are reinstalled. In this case the original files are renamed to have the filename extension `.rpmsave`.
- If two or more packages of the same version is to be installed, only one can be installed using RPM or DEB installation packages. For additional installations the TAR package should be used where other installation directories than `/opt` must be used.
- If installing using RPM or DEB, the installation will always put itself as the preferred installation, located via `/usr/bin` and `/usr/lib`, for example. This is an installation option if installing with TAR. The commands `mimlink` and `mimunlink` can be used to adjust this after installations are done, but please note that it is essential that there is an installation linked to these locations since various default settings are pointing there.

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 node name 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 Linux 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. Also, expired keys may have to be renewed, or, when the number of Mimer SQL users is increased or new Mimer SQL functionality is added to the site, a new Mimer SQL license key will be needed. 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 Linux 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.

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
- 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 when uninstalling the software?

The commands described below, `mimuninstall`, `dpkg -r` and `rpm -e`, 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` is done, specifying removal of everything regarding the database, including data.

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>
```

Removing an RPM installation

To remove an installation installed using RPM, use the following RPM command:

```
# rpm -e mimersql1100-11.0.0A-24298
```

For more detailed information printout when uninstalling, the `-ev` or `-evv` options can be used.

The following command can be used to list installed Mimer SQL packages:

```
# rpm -qa | grep -i mimer
```

Removing a DEB installation

To remove a DEB package installation, use the following `dpkg` command:

```
# sudo dpkg -r mimersql1100
```

The following command can be used to list installed Mimer SQL packages:

```
# dpkg -l | grep -i mimer
```

Removing a TAR installation

To remove an installation that was installed using the `miminstall` command, use the `mimuninstall` command as follows:

```
# mimuninstall /opt/mimersql1100-11.0.0A
```

When running the `mimuninstall` command a question will be raised on if the `/etc/sqlhosts` and `/etc/mimerkey` files should be removed. These are global files to Mimer SQL, used by any installation, so the recommendation is to leave them since there may be other Mimer SQL installations using them.

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 *15*.

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 *15*, and the names of the remote databases accessible from the node are listed in the REMOTE section, see *REMOTE section* on page *15*.

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 *15*.

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`	`''` or `TCP`/`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.

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 *Linux Commands* on page *40*, 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>
```

In addition, main messages regarding the Mimer SQL database server are written to the Linux syslog facility. To extract those messages the following command can be used:

```
# cat /var/log/syslog | grep mimersql
```

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 11.0.8E parameters generated 2024-08-29 10:07
Databanks           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             206768      # # of 4K bufferpool pages (11-2147480000)
Pages32K            18775       # # of 32K bufferpool pages (7-2147480000)
Pages128K           2186        # # 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=None
                                  1=Optional, 2=Required (0-2)
MemLock             0           # Lock bpool in memory, 0=No 1=Yes (0-1)
MiniDump            1           # Small bufferpool dump (no page content),
                                  0=No 1=Yes (0-1)
BackgroundPriority 0            # Thread priority, 0=Default, 1=Highest,
                                  40=Lowest (0-40)
AutoStart           1           # Autostart, 0=No, 1=Yes (0-1)
DumpScript          ./.dumper.sh %p   # Dump Script
HugePages           0           # HugePages, 0=No, 1=2MB, 2=1GB (0-2)
IOQueue             1024        # Max # of concurrent I/O requests (0-65535)
ServerType          3           # Server type: 3=mimexper, 7=miminm (3-9)
```

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 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`	A number which specifies what kind of check that should be performed when a databank is opened which previously was not closed properly. **0 - check index pages** Index pages only are checked in the foreground while applications that access the databank waits for the operation to complete. **1 - check data pages** 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. **2 - Immediate restart, no check** 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. **3 - Immediate restart, check index pages** 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. **4 - Immediate restart, check all pages** This option performs a check of all pages in the databank in the background. This is done concurrently with other operations on the system. 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 `mimcontrol/mimdbserver` command, especially prior to shutting down the machine.
`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`	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: **0 - Default off** In this mode delayed commit is not used unless a transaction is set to use delayed commit by the application. This is the default. **1 - Default on** 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. **2 - Disabled** 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.

`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 if 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 `inetd` - which means that the TCP/IP port server program, `mimtcp`, 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 mimtcp 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`	Controls the use of encryption of network communication over TCP/IP between server and clients. **0 = Network encryption disabled** Network encryption is not supported or not used. **1 = Network encryption preferred** 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. Use this option when there is a mix of older and newer clients that communicate with the database server over TCP/IP. This is the default value. **2 = Network encryption required** The database server requires all clients to use encrypted communication when communicating over TCP/IP. Clients that do not support encryption are rejected at login with error code -18531. Named Pipes via OS-user login is not allowed. This option is recommended over option 1 when possible (i.e. when there are no older clients that need to be supported.)
`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). 0 = No 1 = Yes (default)
`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`	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. If the parameter is set to 0 the database will not be started automatically.

`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 Linux kernel stack trace from the location where the error was located. By defining this parameter to a command that can produce a kernel trace, such as pstack, 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.
`HugePages`	Enable use of larger memory pages: 0 = No 1 = 2MB 2 = 1GB See *HugePages* on page *27* for more information.
`IOQueue`	Specifies the maximum number of concurrent IO requests queued to the operating system. Default is 128, 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.
`IOThreads`	The number of threads in the database server that can serve I/O requests. This parameter is only present and used for some Linux implementations (if it is not present in a default generated multidefs file, the most common reason is that the native Asynchronous I/O is used.)
`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
```

HugePages

HugePages is a Linux kernel feature to enable use of larger memory pages. Normal pages are usually 4KB, and large pages can vary between 2MB to 1GB. By having the Mimer Bufferpool using HugePages, the operating systems use of page table entries and page state maintenance is reduced. It also increases the hit ratio in the CPU's Translation Lookaside Buffers (TLB).

Using HugePages

To enable HugePages in Linux, specify the kernel parameter `vm.nr_hugepages` in `/etc/sysctl.conf` file. (`/etc/sysctl.conf` is read during Linux boot.)

For test purposes, you can also use specify the number of hugepages after boot with `sysctl -w vm.nr_hugepages=value`. (This command is not writing to `/etc/sysctl.conf`, which means it will be reset at system reboot.)

Determine the default HugePage size by running the following command:

```
$ grep Hugepagesize /proc/meminfo
Hugepagesize:       2048 kB
```

Run `mimcontrol -c` to get the size of the bufferpool:

```
$ mimcontrol -c
...
Buffer pool size: 4096 MiB
```

Divide the size of the bufferpool with the HugePage size, to get the number of large pages. In this case, 4096 / 2 = 2048. Run `sysctl` with this value:

```
$ sudo sysctl -w vm.nr_hugepages=2048
vm.nr_hugepages = 2048
```

Run the following command to the see allocated HugePages:

```
$ grep Huge /proc/meminfo
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
HugePages_Total:    2048
HugePages_Free:     2048
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

Edit `/etc/sysctl.conf` and add the following line to ensure HugePages are allocated after system restarts:

```
vm.nr_hugepages=2048
```

Change the HugePages parameter in the multidefs file to 1 or 2, depending on the default HugePage size.

Start the Mimer database server:

```
$ mimcontrol -s
2022-04-05 16:48:45.08   <Information>
=====================================
Mimer SQL 11.0.6C Beta Test Apr  4 2022 Rev a37704m
Mimer SQL Experience server for database MIMERDB STARTED at /mimerdb
```

Verify the number of reserved pages with the following command:

```
$ grep Huge /proc/meminfo
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
HugePages_Total:    2048
HugePages_Free:     2004
HugePages_Rsvd:     2004
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

Background Thread Priority

In the multidefs configuration file there is a parameter called `BackgroundPriority` that can be used to raise the priority for the Mimer SQL database server background threads. If enabled, the following warning message can be obtained in the database server log file, `mimer.log`, as a notification on that the intention to increase the priority failed:

```
2022-06-22 09:18:07.48   <Warning>
Could not set priority for background thread
setpriority: [EACCES] Permission denied
```

To allow this setting to take place, the following command can be used:

```
sudo setcap CAP_SYS_NICE+iep mimexper
```

...or, if the Mimer SQL In-memory database server is used:

```
sudo setcap CAP_SYS_NICE+iep miminm
```

The database server then needs to be restarted.

OOM-killer setup

The `mimoomadjust` command can be used to manage the `oom_score_adj` setting, found for each process under the `/proc` environment. This setting is used by the operating system when selecting processes to be thrown out when the memory resource runs low in the system. Saying that, it can be understood that this command only works for systems having a `/proc` environment.

The command has three options - adjust (`-a`), reset (`-r`) and list (`-l`). It works for running database servers and takes a target database name as parameter. If the database name is omitted, the `MIMER_DATABASE` setting is used. To give a database a good chance of not being killed, the following command can be used:

```
mimoomadjust -a <database_name>
```

Please note that the `mimoomadjust` command needs sudo access to update the value, and that the `oom_score_adj` value adjustment only is valid as long as the process is alive.

To easily set this value when starting or restarting the database server, the option `-o` is available for the `mimdbserver` command. Doing the following command will restart the database defined in the `MIMER_DATABASE` environment variable and set the `oom_score_adj` value:

```
mimdbserver -Xo
```

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. Depending on what support the host machine can offer, one or several configurations may be installed on the host system. Locations are as follows per feature provided and available:

inetd	The internet services daemon. Here the `/etc/inetd.conf` file is used for the Mimer SQL configuration.
xinetd	The extended internet services daemon. Here the `/etc/xinetd.conf` file or the `/etc/xinetd.d` directory is used for the Mimer SQL configuration.
systemd	The system and service manager. Here the `/etc/systemd/system` directory is used for the Mimer SQL configuration.

In all these cases 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 *31*.

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
`-l` `--log`	Enable logging.
`-f file` `--filename=file`	Define a log file.
`-v` `--version`	Display version information.
`-?` `--help`	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

There are different system features available to administer this depending on platform and operating system versions. Currently inetd, xinetd and systemd are supported, where the first two are described more in detail below.

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

inetd setup

The Linux command `inetd` is the Internet services daemon, the server process for the Internet standard services. It is usually started up at system boot time. The configuration file `/etc/inetd.conf` lists the services that inetd should handle. An excerpt from the file shows the syntax used in the file:

```
#
# Syntax for socket-based Internet services:
#  <service_name> <socket_type> <proto> <flags> <user> <server_pathname>
<args>
#
```

When `dbinstall` is executed, and the `inetd.conf` file is found, the following line is added to the configuration file:

```
mimer stream tcp nowait root /usr/bin/mimtcp mimtcp -l
```

This indicates that mimtcp should be started for the mimer service. The `-l` option is used standalone which implies that the default log file should be used.

When the inetd configuration is changed, for example if mimer is added like described above, the inetd daemon must reread it. This is triggered by sending the HUP signal to the inetd process (located using the `ps -ef` command):

```
# ps -ef | grep inetd
root      8796     1  0  2006 ?        00:00:12 inetd
# kill -HUP 8796
#
```

xinetd setup

The Linux command `xinetd` stands for "the extended Internet services daemon". It is the successor to inetd and works in a slightly different way. Instead of having tasks started at system initialization time, and be dormant until a connection request arrives, xinetd is the only daemon process started and it listens on all service ports for the services listed in its configuration file. When a request comes in, xinetd starts the appropriate server.

The default xinetd definitions for Mimer SQL can be found in the file `mimersql.xinetd` in the installation directory called `misc`:

```
$ cat /opt/MimerSQL-11.0.5A/misc/mimersql.xinetd
# default: on
# description: The MIMER service allows remote users to access the
#              Mimer SQL database servers on this node.
service mimer
{
        port            = 1360
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/bin/mimtcp
        server_args     = -l
        log_on_failure  += USERID
        disable         = no
        protocol        = tcp
}
$
```

If the `/etc/xinetd.d` directory is found when `dbinstall` is executed, the `mimersql.xinetd` file is copied there and is given the name `mimer`.

If the `/etc/xinetd.d` is not found, but `/etc/xinetd.conf` is found, the `mimersql.xinetd` contents is added at the end of the `/etc/xinetd.conf` file.

When the xinetd configuration is changed, for example if mimer is added like described above, the xinetd daemon must reread it. This is triggered by sending the HUP signal to the xinetd process (located using the `ps -ef` command):

```
# ps -ef | grep xinetd
root      8796     1  0  2006 ?        00:00:12 xinetd
# kill -HUP 8796
```

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 `/usr/local/MimerSQL/my_db`, it will end up in `/etc/sqlhosts` like this:

```
LOCAL:
   my_db          /usr/local/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/lib/libmimodbc.so
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 `libmimodbc.so` shared library.
Database	The name of the database to be accessed, as defined in the sqlhosts 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 Database attribute will be looked up in the `/etc/sqlhosts` file for further information about the connection setup.
Port	The port number to 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/lib/libmimodbc.so
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:

```
# mimexampledb <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 `/usr/bin/bsql`.
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_msql.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 Linux 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.so` 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/.

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.

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: dbsql (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 modify 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
  ```

- 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 in Linux.

Variable	Explanation
`HOME`	Used to locate the home directory from within various Mimer SQL programs.
`LD_LIBRARY_PATH`	Used on most platforms to locate shared libraries in runtime.
`MIMER_DATABASE`	Used to point out which database to access. If not set, the default database, set in `/etc/sqlhosts`, is used.
`MIMER_EXTEND`	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.
`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 `/etc/mimerkey` file).

Variable	Explanation
MIMER_MODE	Used to indicate the mode for which the database should be accessed, that is, SINGLE or MULTI. Use single mode if accessing a database for which the database server program is not started. If not set, MULTI 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 ODBCINI is not set, the home directory is searched for the .odbc.ini file (using the HOME environment variable).
MIMER_SQLHOSTS	The default sqlhosts file is /etc/sqlhosts. Another file can be used by defining the MIMER_SQLHOSTS environment variable to hold the path of the target sqlhosts file.
ODBCINI	Same as MIMER_ODBCINI. Overrides MIMER_ODBCINI 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, /bin/sh 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 $HOME/.mimer_tmp is used.

Linux Commands

Command	Function	Used by
bsql	SQL command interpreter. See *Mimer SQL User's Manual, Chapter 9, Mimer BSQL* for more information.	mimdevenv, mimexampledb
dbc	Databank check utility. See the chapter *Databank Check Functionality* in *System Management Handbook* 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

Command	Function	Used by
`dbopen`	Opens all user defined databanks at once. See *Chapter 7, Databank Open Function* in *System Management Handbook* for more information.	
`dbuninstall`	Command used to remove a database, including its data files, registrations and related resources.	
`esql`	Embedded SQL preprocessor. See *Mimer SQL Programmer's Manual, Chapter 4, Embedded SQL* 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`
`mimcontrol`	Manages database servers. See *Chapter 4, Managing a Database Server* in *System Management Handbook* for more information.	`dbinstall`, `mimadmin`, `mimlistdb`
`mimdbfiles`	Lists the databank filenames 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`

Command	Function	Used by
`mimdbvis`	Installs the DbVisualizer tool bundled with Mimer SQL. For details on DbVisualizer, see https://developer.mimer.com/documentation/dbvisualizer/.	`miminstall`, `mimuninstall`
`mimdesktop`	Installs Mimer SQL items into the desktop menu system. (Linux only)	`miminstall`, `mimuninstall`
`mimdevenv`	Command used to create a beginner's development environment.	`dbinstall`
`mimdumper`	Creates or executes the `.dumper.sh` 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 `exload` program.)	
`mimexec`	`mimexec` 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 `mimexper` using the `mimcontrol` or `mimdbserver` commands.	`mimcontrol`
`mimhome`	Displays the home directory for the effective user. Especially used to find location for log and tmp files.	`(mainly for internal use)`
`mimhosts`	Program to manage and to do lookup in the `/etc/sqlhosts` file. See *The sqlhosts file* on page *12* for more information.	`dbinstall`, `dbuninstall`, `mimadmin`, `mimowner`, `mimdbfile`, `mimdbserver`, `mimdevenv`, `mimexampledb`, `mimexec`
`miminetd`	Command used to administer Mimer SQL in the operating system Internet services daemons.	`dbinstall`
`miminfo`	Program to monitor database servers. See *Chapter 4, Managing a Database Server* in *System Management Handbook* for more information.	`mimadmin`

Command	Function	Used by
`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 `/etc/mimerkey`. See *Mimer SQL license key* on page *9* for more information.	`mimadmin`, `miminstall`
`mimlink`	Used to link Mimer SQL libraries, man pages and executables to `/usr/lib`, `/usr/man` and `/usr/bin`, respectively.	`miminstall`
`mimlistdb`	Lists started database servers.	`mimadmin`, `mimuninstall`
`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.)	`mimodbcadmin`, `mimodbcdm`
`mimodbcadmin`	Menu based ODBC data source and ODBC driver administration	`dbinstall`, `dbuninstall`
`mimodbcdm`	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`

Command	Function	Used by
`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`
`psmdebug`	PSM debugger, see *Mimer SQL Programmer's Manual, Chapter 10, The Mimer SQL PSM Debugger*.	
`repserver`	The Mimer SQL replication server program. See *REPSERVER - Replicating the Data* in *System Management Handbook*.	`mimrepadm`
`sdbgen`	Command used to create the system databanks for Mimer SQL. See *SDBGEN - Generating the System Databanks* in *System Management Handbook*.	`dbinstall`

Linux Link Libraries

Library	Description
`libmimcomm.so`	This is the shared library used when using Mimer JDBC with local communication, i.e. not via TCP/IP.
`libmimdbs.so`	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.so`	This shared library contains several of the client interfaces supported by Mimer SQL, i.e. DBI and ODBC.
`libmimerapi.so`	This is the shared library for the Mimer SQL C API.
`libmimerS.so`	This is the Mimer setup library used by the unixODBC Driver Manager GUI interface.
`libmimmicroapi.so`	This is the shared library for the Mimer SQL C Micro API.
`libmimodbc.so`	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.so`	This shared library contains the Mimer DBI 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

O

P

R

S

T

U