Mimer SQL

OpenVMS Guide

Version 11.0

Mimer SQL, OpenVMS Guide, Version 11.0, January 2018

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

Introduction

Since the first release of Mimer on VMS in 1982 (Mimer version 3.1 on VAX VMS 3), a large number of Mimer SQL users have deployed their solutions on the OpenVMS platform.

About this Guide

This guide describes how to install and use version 11.0 of the Mimer SQL relational database server under OpenVMS. It contains OpenVMS specific instructions about installing the software, creating databases and managing database servers and is for general users, system administrators and programmers who use Mimer SQL on the OpenVMS operating system.

About Mimer SQL for OpenVMS

You can download a full version of Mimer SQL for OpenVMS for free from https://developer.mimer.com/downloads. This distribution is for development and evaluation. It contains a complete copy of Mimer SQL Version 11.0 and a built-in software key, with support for up to 10 concurrent database connections.

Mimer SQL Run-time License

To use Mimer SQL for OpenVMS in production, you need a run-time license key.

Please contact your local Mimer representative, see https://www.mimer.com/contact/, or send an e-mail to info@mimer.com.

The Mimer SQL Database Server

The Mimer SQL database server is a single, multi-threaded process. By using DECThreads, good SMP scalability is achieved.

Clients using TCP/IP or Decnet can access the server. For clients running on the same host as the server, a special shared-memory based communication method is used.

Embedded SQL

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

Module SQL

MSQL (Module SQL) enables you to call SQL statements in a host program written in C/C++, COBOL, Fortran or Pascal, without embedding the actual SQL statements in the host program. The SQL statements are explicitly put into a separate SQL module, that is written in the Module language and maintained separately from the host program.

JDBC Driver

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 on any platform.

By using a special connection URL, the JDBC driver can also load a native library (`MIMER$LIB:MIMCOMM.EXE`). This allows the JDBC driver to use native code for all communication methods which can improve the performance.

The driver, `MIMJDBC3.JAR`, supports the JDBC 3 standard, and is for Java 1.4 or later.

ODBC Driver

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

By using the ODBC driver on OpenVMS, you can develop ODBC applications and execute them on the OpenVMS platform.

Unlike other platforms, OpenVMS does not include a Driver Manager that enables OpenVMS applications to dynamically load drivers for different database products. Until such a Driver Manager becomes available on OpenVMS, you can link your applications directly to the Mimer ODBC driver.

Note that in order to run an ODBC application on a Windows platform, the Windows ODBC driver has to be installed on the client side. This driver can then access Mimer database servers on any platform (including OpenVMS). There is no need to install any special software on the server side in order to use ODBC.

Mimer SQL C API

The native Mimer SQL C API is included in the distribution. This API is intended for programs written in C and is described further in the *Mimer SQL Programmer's Manual*.

Utilities

Mimer SQL includes the following utilities:

Utility	Description
BSQL	BSQL executes SQL statements which are entered interactively or read from a command file. It is described in the *Mimer SQL User's Manual*.

Utility	Description
DBC	DBC checks if a databank file is internally consistent. It is described in the *Mimer SQL System Management Handbook*.
DBOPEN	DBOPEN opens and restarts all databanks in a database. It is described in the *Mimer SQL System Management Handbook*.
ESQL	ESQL is a pre-processor for embedded SQL.
EXLOAD	Utility to load the example database. See *Mimer SQL System Management Handbook*.
MIMCONTROL	MIMCONTROL provides facilities for managing the operation of a database server, for example, starting, controlled shutdown, etc.
MIMINFO	MIMINFO displays status information for a database server.
MIMLICENSE	Utility for managing Mimer SQL licenses.
MIMLOAD	Utility to load or unload data from the database. See *Mimer SQL System Management Handbook*.
MIMPERF	Utility to display performance counters in the server. Can be used interactively or together with the T4 tool.
MIMREPADM	Administration utility for Mimer Replication.
MIMSYNC	Utility to bring two replicated Mimer systems back into sync.
REPSERVER	Replication server.
SDBGEN	SDBGEN generates the Mimer SQL system databanks SYSDB, TRANSDB, LOGDB and SQLDB.
SQLMONITOR	Utility to display how the current SQL statements are using the server resources.
TCPCONTROL	Procedure to manage MIMTCP processes.

The Mimer SQL distribution also contains examples of database programs and SQL statements. See *Appendix A Distributed Files* for more information.

OpenVMS System Requirements

Mimer SQL version 11.0 requires OpenVMS V8.4-1H1 (from VSI) or later.

Mimer uses OpenSSL for cryptography, and requires SSL1 version 1.0 or later to be installed.

User Requirements

To install Mimer SQL, you should have a working knowledge of system management within the OpenVMS environment.

Documentation Resources

You can find relevant information in the OpenVMS System Management Guide (published by VSI). Most OpenVMS manuals can be found at https://vmssoftware.com/resources/documentation/

You should be familiar with the concepts and facilities provided by the Mimer SQL system.

Other documents that are referred to in this document or that may be of interest when dealing with the tasks described here are: the *Mimer SQL System Management Handbook*, the *Mimer SQL Programmer's Manual*, and the *Mimer SQL Release Notes*.

You can find them at https://developer.mimer.com/documentation/

The documentation is also included in the Mimer SQL distribution as PDF (Adobe Portable Document Format) files. After a SETUP has been made with an installed Mimer version, the documentation can be found in the directory pointed to by the logical name MIMER$DOC.

Documentation Conventions

Convention	Example	Explanation
All uppercase	COMMIT	Indicates command names, SQL reserved words, and keywords.
`Monospace`	`$ MIMCONTROL/START DB5`	Indicates directory names, file names, code examples and interactive screen displays.
`[ ]`	`[timeout]`	Encloses optional items.
`[ | ]`		Vertical bars separating optional items indicate that you can choose none, one or more than one item.
Italics	*Mimer SQL User's Guide*	Indicates a cross reference or the title of a guide.

Terms and Definitions

The following terms and acronyms are used in this document:

Term	Explanation
API	Application Programming Interface.
BSQL	BSQL, a program used to execute SQL statements which are read from a command file or entered interactively.
Data source	ODBC term for a database.

Term	Explanation
Databank	Databank is the Mimer SQL term for the physical file in which one or more Mimer SQL tables are stored. A databank corresponds to one file in the operating system. A database may contain several databanks.
Database	A database is a collection of databanks, tables, shadows, etc., all defined as objects in the data dictionary. A computer may have several databases operating simultaneously, but no information is shared between them. Each database has a unique name, registered in the SQLHOSTS file.
Database home directory	The directory where the SYSDB databank file is located, also recorded in the SQLHOSTS file.
DCL	Digital Command Language.
Dynamic SQL	SQL statements constructed at runtime and passed to the database management system for execution.
Embedded SQL	The term used for SQL statements when they are embedded in a traditional host language.
ESQL	The preprocessor for embedded SQL.
Java	A platform independent language invented by Sun and now owned by Oracle. See https://www.oracle.com/technetwork/java/index.html.
JDBC	Java DataBase Connectivity. A set of Java interfaces for accessing relational databases. See https://www.oracle.com/technetwork/java/javase/jdbc/index.html.
MIMER*xxxxx*	Symbolic name for the installed directory tree, unique for each Mimer SQL release, where "*xxxxx*" stands for the current version number, e.g. "1101A".
ODBC	Open Database Connectivity, a specification for a database API in the C language, independent of any specific DBMS or operating system.
PSM	Persistent Stored Modules, the term used by ISO/ANSI for stored procedures.
Shadow	A shadow is a copy of the original (master) databank and is continuously updated by Mimer SQL. A Mimer SQL databank may have one or more shadows. If the databank is shadowed, there will be one file for each shadow. If the master databank is lost, it is possible to continue operations from the shadow databank without stopping the database server. A databank must have the TRANS or LOG option to be shadowed. For more information, see the *Mimer SQL System Management Handbook*.
SQL	Structured Query Language, standardized language for database manipulation.

Term	Explanation
SQLHOSTS	A file containing lookup information for all accessible Mimer SQL databases, relative to the current node. It is located at `SYS$MANAGER:SQLHOSTS.DAT`.
Table	Tables (or relations) hold all the information in a relational database. A table is stored in a databank. It may not be split across databanks, but a databank may contain several tables.

Chapter 2

Installing Mimer SQL

This chapter describes how to install the Mimer SQL software on OpenVMS. It also documents how to remove a Mimer SQL installation.

Overview

- Execute the distribution file to obtain the installation files
- Use `PRODUCT INSTALL` to install Mimer SQL
- Add commands in `SYS$MANAGER:SYSTARTUP_VMS.COM`, `SYS$MANAGER:SYSHUTDWN.COM` and `SYS$MANAGER:SYLOGIN.COM`
- Execute `SYS$COMMON:[MIMERxxxxx]SETUP.COM` to define Mimer commands for this process
- Install your Mimer SQL license key

Execute the distribution file to obtain the installation files

The software is distributed as a self-extracting ZIP file. To unpack the ZIP file, just execute it. The installation files will be extracted and placed in the current directory.

```
$ RUN MIMER-I64VMS-MIMER1106A.ZIPEXE
UnZipSFX 6.00 of 20 April 2009, by Info-ZIP (http://www.info-zip.org).
  inflating: MIMER-I64VMS-MIMER1106A-U1100-6A-1.PCSI$COMPRESSED
```

Use PRODUCT INSTALL to install Mimer SQL

Use the command `PRODUCT INSTALL` to install Mimer SQL, just like any other layered product. Please note that the product name includes the version number which is unusual. New versions of Mimer will have new version numbers in the product name which will make all versions of Mimer to appear as different products. This allows multiple versions to be installed at the same time on a machine. The user can select any version by running its `SETUP` command procedure.

```
$ PRODUCT INSTALL MIMER1106A

Performing product kit validation of signed kits ...
%PCSI-I-CANNOTVAL, cannot validate SIV$DKA300:[PER.SLASK]MIMER-I64VMS-
MIMER1106A-U1100-6A-1.PCSI$COMPRESSED;1
-PCSI-I-NOTSIGNED, product kit is not signed and therefore has no manifest
file

The following product has been selected:
    MIMER I64VMS MIMER1106A U11.0-6A       Layered Product

Do you want to continue? [YES]

Configuration phase starting ...

You will be asked to choose options, if any, for each selected product and for
any products that may be installed to satisfy software dependency
requirements.

Configuring MIMER I64VMS MIMER1106A U11.0-6A

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
    MIMER I64VMS MIMER1106A U11.0-6A       DISK$I64SYS:[VMS$COMMON.]

Portion done: 0%...10%...20%...30%...40%...50%...60%...70%...90%...100%

The following product has been installed:
    MIMER I64VMS MIMER1106A U11.0-6A       Layered Product

MIMER I64VMS MIMER1106A U11.0-6A

    Insert the following lines in SYS$MANAGER:SYSTARTUP_VMS.COM:
        @SYS$STARTUP:MIMER$STARTUP_1106A
    Insert the following lines in SYS$MANAGER:SYSHUTDWN.COM:
        @SYS$STARTUP:MIMER$SHUTDOWN_1106A

    Users need to add the following lines to their login command procedure:
        $ @SYS$SYSROOT:[MIMER1106A]SETUP VMS
```

Add commands in system startup files

As instructed by the `PRODUCT INSTALL` command, you should add commands to execute Mimer startup and shutdown procedures.

The procedure `SYS$STARTUP:MIMER$STARTUP_1106A` installs required shareable images. Add a command to do this in `SYS$MANAGER:SYSTARTUP_VMS.COM`.

The procedure `SYS$STARTUP:MIMER$SHUTDOWN_1106A` deinstalls shareable images. Add a command to do this in `SYS$MANAGER:SYSHUTDWN.COM`.

Any process that is running Mimer SQL must execute the SETUP command procedure for the version of Mimer SQL it wants to use. By adding the SETUP command in SYS$MANAGER:SYLOGIN.COM, a default SETUP is provided for all processes. This makes it easier to manage a version upgrade of Mimer SQL since the command is in a central place.

Execute SYS$COMMON:[MIMERxxxxx]SETUP.COM for this process

To continue using the newly installed Mimer SQL product in the current process you should execute the SETUP procedure.

```
$ @SYS$SYSROOT:[MIMER1106A]SETUP VMS
MIMER version 11.0.6A Beta Test Jun 29 2021 Rev 35581
Node name: SIV VMS IA64
```

An alternative is to log out and log in again. The new process should have executed the SYS$MANAGER:SYLOGIN.COM procedure where you have added the SETUP command.

The SETUP procedure is described further in section *The Mimer SQL SETUP command procedure* on page *9*.

Install your Mimer SQL license key

Mimer requires a license key to be installed. Use the MIMLICENSE command to administrate Mimer licenses. A license for testing purposes is included in the Mimer distribution and can be reached by the logical name MIMER$EXAMPLES.

The MIMLICENSE command is described further in section *Installing the Mimer SQL License Key* on page *11*.

```
$ MIMLICENSE /FILE=MIMER$EXAMPLES:DEFAULTKEY.MCFG

Mimer SQL License Manager  Version 11.0.6A Beta Test
Copyright (C) Mimer Information Technology AB. All rights reserved.

License key file: SYS$SPECIFIC:[SYSMGR]MIMERKEY_IA64.DAT

Number Expires    Version Module       Users Netusers
------ ---------- ------- ------------ ----- --------
    -6 2022-01-31 11.0    Beta test                  *** test and development
                           Mimer SQL       10       10
                           Imm restart
                           XA

Key file updated

Note: The database server must be restarted to enable the new key settings
```

The Mimer SQL SETUP command procedure

Use the SETUP command procedure in the top directory of a particular Mimer installation to define logical names and commands for the current process. For example:

```
$ @SYS$SYSROOT:[MIMER1106A]SETUP VMS
MIMER version 11.0.6A Beta Test Jun 29 2021 Rev 35581
Node name: SIV VMS IA64
```

The parameter to SETUP specifies what style of commands you wish to use. You can also clear any definitions made by a previous SETUP command.

@SYS$SYSROOT:[MIMER1106A]SETUP VMS

This command specifies that Mimer version 11.0.6A should be used. Logical names are defined in the process logical name table. Mimer commands are defined in VMS style (like: MIMCONTROL/START).

Note that if you create a subprocess (by using the SPWAN command), the command definitions will not be active in the there. Use the SETUP command again in the subprocess.

If no parameter is given to the SETUP procedure, VMS will be assumed.

@SYS$SYSROOT:[MIMER1106A]SETUP UNIX

This command specifies that Mimer version 11.0.6A should be used. Logical names are defined in the process logical name table. Mimer commands are defined in UNIX style (like: mimcontrol --start).

UNIX command definitions will be transferred to any subprocess you start with the SPAWN command.

@SYS$SYSROOT:[MIMER1106A]SETUP CLEAR

Clears all logical names and command definitions made by an earlier invocation of the SETUP procedure.

Logical Names Defined by SETUP

The SETUP command defines the logical names listed below

Logical name	Description
MIMER$ROOT	Points to the Mimer installation directory, such as DKA100:[SYS0.SYSCOMMON.MIMER1106A.]
MIMER_SQLHOSTS	Points to the file defining database names
MIMER$LIB	Location of header files, etc
MIMER$DOC	Location of Mimer documentation
MIMER$EXAMPLES	Directory containing example files
MIMER$EXE	Location of Mimer executable files
MIMER$SQL	Specifies selected version of MIMER$SQL image
MIMER$ODBC	Selected version of MIMER$ODBC image
MIMER$MIMERAPI	Selected version of MIMER$MIMERAPI image
MIMER$DBP	Points to shareable image for shared memory communication
MIMCOMM	Selected version of MIMCOMM image
MIMCOMM64	Selected version of MIMCOMM64 image

Installing the Mimer SQL License Key

Mimer SQL for OpenVMS is free for development and evaluation. A development and evaluation license key is included in the Mimer SQL distribution and is automatically installed.

This means that, as long as you use Mimer SQL for development and/or evaluation, you can set up a complete Mimer SQL environment and work with Mimer SQL without adding any additional license keys.

Mimer SQL for OpenVMS in Production

If you want to use Mimer SQL in production, you must purchase a valid run-time license key and then install it. Please contact your local Mimer SQL representative, see https://www.mimer.com/contact/, or send an e-mail to info@mimer.com.

Node Name

Your representative will need to know the node name of the computer on which the Mimer SQL database server will run.

To obtain the node name, use `SETUP`:

```
$ @SYS$SYSROOT:[MIMERxxxx]SETUP
```

Receiving Your Run-time License Key

Your run-time license key and instructions for installing it will be e-mailed to you.

When you receive the file, save it in an accessible directory.

The MIMLICENSE Utility

You use the `MIMLICENSE` utility to administrate the license key file. You can add, remove and update keys using `MIMLICENSE`.

You can also use `MIMLICENSE` to list and describe the contents of the key file.

Note: When entering the Mimer SQL license key, you must have appropriate access to the key file.

Adding a License Key using MIMLICENSE

To add a license key

1 Assuming a `SETUP` with VMS-style commands, enter the following:

```
$ mimlicense /FILE=file_name.mcfg
```

For example, if the license key file you received was named `mimerkey1234.mcfg`, you would enter the following:

```
$ mimlicense /FILE=mimerkey1234.mcfg
```

MIMLICENSE Syntax

The `MIMLICENSE` program is controlled by options specified on the command-line.

Argument	Function
`/ADD=`*`hexcode`*	Adds a license key.
`/FILE=`*`file-name`*	Adds a license key from a `.mcfg` file.
`/DELETE=`*`key-id`*	Deletes the specified key.
`/REMOVE`	Removes all keys. (Each key must be verified.)
`/LIST`	Lists the contents of the key file.
`/COMBINED`	Describes what the combined keys permits.
`/SILENT`	Silent mode, i.e. execution with no output.

The Mimer SQL license keys are stored in the file:

```
SYS$SPECIFIC:[SYSMGR]MIMERKEY.DAT
```

Removing a Mimer SQL Installation

Caution: If you plan to remove any Mimer SQL databases, see *Removing a Mimer SQL Database* on page *20* **before** removing the Mimer SQL installation.

To remove a Mimer SQL installation:

1. Check that no Mimer SQL applications or database servers are using the installation.
2. Remove the selected Mimer SQL version by using the `PRODUCT REMOVE` command:

```
$ PRODUCT REMOVE MIMER1106A

The following product has been selected:
    MIMER I64VMS MIMER1106A U11.0-6A       Layered Product

Do you want to continue? [YES]

The following product will be removed from destination:
    MIMER I64VMS MIMER1106A U11.0-6A       DISK$I64SYS:[VMS$COMMON.]

Portion done: 0%...20%...30%...40%...50%...60%...70%...80%...90%...100%

The following product has been removed:
    MIMER I64VMS MIMER1106A U11.0-6A       Layered Product
```

3. If there are no other Mimer SQL installation trees in the system, you may want to delete the SQLHOSTS file and the Mimer SQL license key file:

```
$ DELETE SYS$SPECIFIC:[SYSMGR]SQLHOSTS.DAT.*
$ DELETE SYS$SPECIFIC:[SYSMGR]MIMERKEY_IA64.DAT.*
```

4. If you have added any Mimer SQL-related commands to the OpenVMS startup file `SYS$MANAGER:SYSTARTUP_VMS.COM`, or the OpenVMS shutdown file `SYS$MANAGER:SYSHUTDWN.COM`, or the OpenVMS system login file `SYS$MANAGER:SYLOGIN.COM`, remove these commands.
5. If you have added any database-specific commands to the system startup file `SYS$MANAGER:SYSTARTUP_VMS.COM`, or the shutdown file `SYS$MANAGER:SYSHUTDWN.COM`, you should remove them.

Chapter 3

Establishing a Database

This chapter describes how to establish a local database and how to access a remote database already established in the network. It also describes how to upgrade and remove a database.

Refer to the *Mimer SQL System Management Handbook* for background information which is useful for understanding the issues and the different components involved in establishing a Mimer SQL database.

Overview

Having installed Mimer SQL, you can now establish a local database on the node on which you unpacked the Mimer SQL distribution.

To establish a database on an OpenVMS node, you must carry out the following steps:

Step 1 Create a home directory for your Mimer SQL database

See *Creating a Home Directory* on page *15*.

Step 2 Prepare access to the database by editing the SQLHOSTS file

See *Editing the SQLHOSTS File* on page *15*.

Step 3 Run SDBGEN to generate the system databanks and the SYSADM ident

See *Generating System Databanks and SYSADM* on page *17*.

Creating a Home Directory

First of all you must create a home directory for your database, for example:

```
$ CREATE/DIR somedisk:[TESTDB]
```

Editing the SQLHOSTS File

The SQLHOSTS file is used to list all the databases that are accessible to a Mimer SQL application from the node on which it is installed.

On a VMS node, the SQLHOSTS file is located by translating the logical name `MIMER_SQLHOSTS`.

The `SETUP` command will define it to be:

```
SYS$SPECIFIC:[SYSMGR]SQLHOSTS.DAT
```

A default SQLHOSTS file is installed the first time you run `SETUP`.

The SQLHOSTS file contains three sections:

- **LOCAL**

  The LOCAL section contains the names of the local databases on the current node, see *Adding a Local Database* on page *16*.

- **REMOTE**

  The REMOTE section contains the names of remote databases accessible from the node, see *Accessing a Remote Database* on page *19*.

- **DEFAULT**

  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 *Specifying the Default Database* on page *16*.

Note: In the SQLHOSTS file, a line of text beginning with the character sequence `--` is interpreted as a comment.
The maximum length for the name of a database on an OpenVMS node is 30 characters.

Adding a Local Database

To add a local database:

1. Open the SQLHOSTS file in a text editor and locate the LOCAL section.
2. Under Database, enter the name of the database.
3. Under Path, enter the name of the directory which is to be the database's home directory.

Example of a LOCAL Entry

```
LOCAL:
--
-- Database           Path
-- ------------------------------------------------------------------
   TESTDB             DISK:[TESTDB]
--==================================================================
```

Specifying the Default Database

The DEFAULT section in the SQLHOSTS file contains a single line that specifies the default database. This is the database which will be used if a database is not explicitly specified when logging on.

The default database must be listed in either the LOCAL or the REMOTE section.

Example of a Default Entry

```
DEFAULT:
--
-- Database
-------------------------------------------------------------------------
   TESTDB
-------------------------------------------------------------------------
```

Generating System Databanks and SYSADM

You generate the Mimer SQL system databanks SYSDB, TRANSDB, LOGDB and SQLDB by running the SDBGEN program.

When you run SDBGEN, it also generates the system administration ident SYSADM.

SDBGEN loads the system tables and defines the data dictionary views detailed in the *Mimer SQL Reference Manual*.

Note: A databank created for one SYSDB cannot be accessed by using a different SYSDB even if identical data dictionary definitions are created in it.

SDBGEN

The SDBGEN command has two purposes. Either to create a new set of system databank files, or to upgrade database files created in an earlier version of Mimer SQL to version 11.0. Upgrade can be done for databank files created by Mimer SQL version 8.1 and later.

For more information on upgrading, see *Mimer SQL Release Notes*.

SDBGEN Syntax

Assuming a `SETUP` is done with VMS-style commands, you run SDBGEN from the command line, using arguments.

The syntax for creating databank files is as follows:

```
SDBGEN [/PASSWORD=passw] [dbase] [syssz] [tfn] [tsz] [lfn] [lsz] [sfn] [ssz]
```

sdbgen Command-line Arguments

Argument	Function
`/PASSWORD=`*`password`*	Password for SYSADM
`dbase`	Database name
`syssz`	Size of SYSDB
`tfn`	Filename for TRANSDB
`tsz`	Size of TRANSDB
`lfn`	Filename for LOGDB
`lsz`	Size of LOGDB
`sfn`	Filename for SQLDB
`ssz`	Size of SQLDB

Generating the System Databanks

For example, the following SDBGEN call:

```
$ SDBGEN /PASSWORD=ooops TESTDB
```

will generate a database named `my_database` and the database administration ident SYSADM will be assigned the password `ooops`.

If you do not enter the `password` parameter, SDBGEN will prompt you for all parameters that are missing, including the password for SYSADM.

If you enter the `password` parameter, SDBGEN will not prompt for any missing parameters, it will use default values.

If you do not enter the `dbase` parameter, the environment variable MIMER_DATABASE is used to determine which database the databank files should be created for.

Setting the Initial Size

You can specify the initial size for each of the Mimer SQL system databanks.

The size for the databanks is specified in Mimer SQL pages. The size of a Mimer SQL page is 4 kilobytes.

SYSADM Password

When you run SDBGEN, the database administration ident SYSADM is created and you must specify a password (passwords are case-sensitive) for this ident.

SYSADM Password Case

DCL converts all VMS-style commands to uppercase and all Unix-style commands to lowercase. To control the case used in your password, you may have use quotes.

The following table shows how to use quotes to set the password case.

Entering:	Sets the password to:
`SDBGEN/PASS=oops`	`OOPS`
`SDBGEN/PASS="oops"`	`oops`
`SDBGEN -p OOPS`	`oops`
`SDBGEN -p "OOPS"`	`OOPS`

SYSADM Password Security

For security reasons, the password specified for SYSADM is not echoed on the screen when you enter it.

You should change the password at appropriate intervals using Mimer SQL with the ALTER IDENT statement.

Caution:	Take care to safeguard the SYSADM password, because if it is lost, it cannot be retrieved from the system and it is not possible to set a new one.

Accessing a Remote Database

You can access databases that reside on other nodes on the network by editing the REMOTE section in the SQLHOSTS file and adding information about the remote database.

For more information on the SQLHOSTS file, see *Editing the SQLHOSTS File* on page *15*.

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

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

The fields in the REMOTE section specify the following:

Field	Explanation
DATABASE	The DATABASE field specifies the name of the remote database.
NODE	The NODE field specifies the network node name of the remote machine. If you are using the TCP/IP interface, you can specify the IP address here.
PROTOCOL	You can specify DECNET or TCP depending on the type of network protocol to be used to create the client/server connection. The default, specified by `''` (two single quote characters), is TCP.
INTERFACE	For TCP/IP, this field specifies whether IPv4 or IPv6 should be used. Specify 4 for IPv4 only Specify 6 for IPv6 only Enter an empty string with `''` (two quotes) to use either IPv4 or IPv6 as indicated by the Node name lookup.
SERVICE	**TCP/IP** If using the TCP/IP protocol, enter the TCP/IP port number the database server uses. The default is 1360. **DECNET** If using DECNET, enter the database name. The server listens to the network object using the same name as the database. (An old Mimer SQL version 7 database server using DECNET listens to the network object named “MIMER”).

Adding a Remote Database

To add a remote database:

1. Open the SQLHOSTS file in a text editor and locate the REMOTE section.
2. Fill in the fields, as specified above, according to your network configuration.

Example of a REMOTE Entry

```
REMOTE:
--
-- Database           Node               Protocol Interface Service
-- ------------------ ------------------ -------- --------- -------
   REMTEST            STARTREK           TCP      ''        1360
```

Mimer SQL System Settings

You can edit your startup and shutdown files to automatically start and stop database servers at system startup and shutdown.

Automatic Database Server Start

If you want the Mimer SQL database server to start automatically whenever the system is booted, you must edit the `SYS$MANAGER:SYSTARTUP_VMS.COM` file.

The following example starts two Mimer SQL database servers:

```
$ MIMCONTROL/START TESTDB
$ MIMCONTROL/START INVENTORY
```

Automatic Database Server Shutdown

If you want to perform a controlled shutdown of the database server whenever the OpenVMS system is shut down, you must edit the `SYS$MANAGER:SYSHUTDWN.COM` file and add the relevant commands at the end.

The following example stops two database servers:

```
$ MIMCONTROL/STOP TESTDB
$ MIMCONTROL/STOP INVENTORY
```

Removing a Mimer SQL Database

To remove a database, perform the following steps:

1. Check that no one is using the database.
2. Check that no database server is started against the database you are going to remove.
3. Create a list of all databank files by doing the following:

```
$ BSQL/SINGLE database
Username: SYSADM
Password: xxxxxx
SQL> SELECT DATABANK_FILENAME FROM SYSTEM.DATABANKS;
SQL> EXIT;
```

4. Using the list generated in the previous step, locate and delete all the physical databank files. If the file name does not contain a directory specification, the directory will be the home directory of the database.
5. Delete any directories that have been specifically created to hold databank files for the database.
6. Delete the database entry in:

   ```
   SYS$SPECIFIC:[SYSMGR]SQLHOSTS.DAT
   ```

Chapter 4

Managing a Database Server

This chapter contains a short guide to administrating database servers under OpenVMS. It also describes how to use the MIMCONTROL command under OpenVMS.

For general information on managing database servers, refer to the *Mimer SQL System Management Handbook*.

The MIMCONTROL Command

Before you can access a database, the Mimer SQL database server must be started on the node it resides on.

You start, stop and control database servers on OpenVMS using the MIMCONTROL command.

Note: You cannot start the MIMCONTROL program by using the DCL command RUN.

Required Privileges

To use MIMCONTROL you must have either:

- SETPRV privilege

or

- CMKRNL, CMEXEC, SHMEM, SYSPRV, WORLD, TMPMBX, OPER, NETMBX, PSWAPM, DETACH, ALTPRI, PRMGBL, SYSGBL, SYSLCK and SYSNAM privileges.

If the resident memory feature is used (the `BPResident` parameter in MULTIDEFS.DAT is set), you must have the `VMS$MEM_RESIDENT_USER` process right.

Database Server Parameters – the MULTIDEFS File

When you start a database server on an OpenVMS machine for the first time, MIMCONTROL will create a MULTIDEFS file containing default parameter values for the database server. These parameters are based on the amount of memory installed on the machine.

You may also generate the MULTIDEFS file manually by using the MIMCONTROL/GENERATE command. For example:

```
$ MIMCONTROL/GENERATE TESTDB
```

It is not possible to change the parameters for a running database server.

You can fine-tune database server performance by adjusting the parameters as required.

Refer to the *Mimer SQL System Management Handbook* for details.

MIMCONTROL Syntax

You run the MIMCONTROL program is using arguments specified on the command-line.

For example:

```
$ MIMCONTROL/START TESTDB
```

Starts the database server and provides access to TESTDB.

If you run the MIMCONTROL command without any options it displays help on command-line arguments.

MIMCONTROL Command-line Arguments

For more information on MIMCONTROL arguments and their combinations, see the *Mimer SQL System Management Handbook*.

Argument	Function
/STATUS/DCL	Output status information about the specified database server into the symbol MIMER_STATUS for use in a command procedure. For details about the output string resulting from this option, see *MIMCONTROL (/STATUS/DCL)* on page *25*.
/STATUS	Output status information about the specified database server.
/DISABLE	Disable new user connections to the database server. Users already connected are not affected.
/ENABLE	Enable new user connections to the database server.
/KILL	Kill the database server immediately. This should only be used in emergency situations when a normal stop does not work. The next time the database is started, all databanks that were open at the time the server was killed will be automatically checked. Connected users will receive an error the next time they attempt to access the database.
/LOGOUT=chan	Force logout of the specified channel number. Use channel numbers displayed by the USERS option of the MIMINFO command, see the *Mimer SQL System Management Guide*.

Argument	Function
`/START [=timeout]`	Start the database server. If the server does not become operational within the specified number of seconds, the server will be killed. (The default timeout is 600 seconds.)
`/HOLD`	This switch can only be used together with the `/START` switch. After the server has been started, the MIMCONTROL command will not return to the DCL prompt, but will wait for the server to stop. This can simplify writing scripts that automatically restart database servers. The termination status from the MIMCONTROL command will be the final status code from the database server process.
`/STOP [=timeout]`	Stop a database server. Any remaining users will be logged out. If the database server does not stop within the specified number of seconds, the server will be killed. (The default timeout is 120 seconds.)
`/WAIT [=timeout]`	Wait for all connected users to log out. If there are still users connected after the timeout period expires, the command fails. The timeout period should be given in seconds. If no timeout period is specified wait will be performed without any timeout.
`/DUMP`	Create a dump directory and produce dumps of all internal database server areas to files in that directory. The files produced can be examined by using MIMINFO, see the *Mimer SQL System Management Guide*.
`/GENERATE`	Create a new MULTIDEFS.DAT file with default for parameters, if the file is missing.
database	Specifies the name of the database to access. If a database name is not specified, the default database will be controlled. The default database is determined by setting the MIMER_DATABASE logical name. The DEFAULT setting in SQLHOSTS is not used for MIMCONTROL.

MIMCONTROL (/STATUS/DCL)

The `MIMCONTROL/STATUS/DCL` command is a special form of the `MIMCONTROL/STATUS` command which returns the database server status information in the form of a single string containing a comma-separated list which is useful when writing command procedures.

On OpenVMS, the `MIMCONTROL` command is silent and sets the DCL symbol `MIMER_STATUS` to the value of the status string.

You can use the lexical function F$ELEMENT() to extract the list elements. For example:

```
$ MIMCONTROL/STATUS/DCL
$ SHOW SYMBOL MIMER_STATUS
MIMER_STATUS == "Running,Enabled,LOKE_0:[PER.LOKE],0,A2,2017-02-16
16:10,121323127"
$ DIR=F$ELEMENT(2,",",MIMER_STATUS)
$ USERS=F$ELEMENT(3,",",MIMER_STATUS)
$ PID=F$ELEMENT(4,",",MIMER_STATUS)
$ SHOW SYMBOL DIR
DIR = "LOKE_0:[PER.LOKE]"
$ SHOW SYMBOL USERS
USERS = "0"
$ SHOW SYMBOL PID
PID = "A2"
```

The MIMPERF Command

The database server maintains a number of performance related counters counting events such as transactions, disk I/O, buffer pool accesses, client/server accesses and much more. The MIMPERF command can be used to monitor these counters.

MIMPERF can access remote databases running under any type of host. It can display statistics from Mimer database servers from version 9 and up.

VSI has a tool named T4 that continually collects performance counters from the VMS host system. The collected data is stored in simple CSV (Comma Separated Values) files. These files can be analyzed by tools such as TLViz or Excel. The MIMPERF tool has an option to generate CSV files so that the Mimer server data can be used together with other data collected by T4.

For more information about T4, please see the VSI web site:

https://vmssoftware.com/products/t4/

For more information on how to use MIMPERF with T4, please read MIMER$DOC:MIMPERF_T4.TXT.

MIMPERF Syntax

```
$ MIMPERF [options] database
```

The options can be given either is VMS style (preceded by a /) or in UNIX style, depending on how the Mimer SETUP command was executed. Please note that UNIX style options in uppercase should be enclosed in double quotes on VMS.

Most counters are displayed as the number of events that happened per second. If this number is greater than zero but less than one, it may be displayed as "0." (a zero followed by a decimal point) to distinguish it from zero (meaning that no such events happened).

MIMPERF Command-line Arguments

VMS style	Unix style	Description
/SYSTEM	--system, -S	Display general database server counters interactively
/TRANSACTION	--transaction, -T	Display counters detailing transaction rates

VMS style	Unix style	Description
/COMMUNICATION	--communication, -C	Display counters from the communication between the database server and connected clients
/POOL	--pool, -P	Show the buffer pool usage and hit rates
/COMPILER	--compiler, -Q	Show how the compilers for SQL statements and PSM procedures are used
/SPACE	--space, -A	Show the space usage in the SQL pool
/EXPLAIN	--explain, -x	Use this switch together with any of the switches above to display a short explanation of the counters
/T4=file	--t4=file, -4 file	Collect all Mimer counters to a T4 compatible CSV file
/INTERVAL=n	--interval=n, -i n	Set the sample interval to n seconds. The default is 10 seconds for the interactive displays and 60 seconds for T4.
/ENDING="[YYYY-MM-DD]HH:MM[:SS]"	--ending=t, -e t	Stop collection at the specified time. Note that there should be a space between the (optional) date and the time. If a space is included, the value must be enclosed in double quotes.

The MIMTCP Server

If you are using the TCP/IP protocol, a MIMTCP server listening to a specific port (usually port 1360) will be started the first time the database server is started.

TCP/IP Port Number

The TCP/IP port number that the MIMTCP server will listen to is specified in the `TCPPort` parameter in the MULTIDEFS.DAT file. If several database servers specify the same port number, they will share the same MIMTCP server.

When a client connects to the TCP/IP port, the MIMTCP server will accept the connection. The client specifies the database to which a connection is to be established and the MIMTCP server will hand over the connection to the appropriate database server. All further communication between the client and the database server is then done directly without involving the MIMTCP server.

System Logical Names

Whenever a MIMTCP server starts, it will define the system logical name "`MIMTCP_xxxx`" (where `xxxx` is the port number) to be the PID of the MIMTCP server process. This makes it easy to find the MIMTCP server process that is listening to a particular TCP/IP port.

Controlling MIMTCP servers

The command procedure MIMER$EXE:TCPCONTROL.COM can be used to manage MIMTCP processes. The first parameter given to the procedure controls what the procedure should do. If no parameter is given, a short help message is displayed. The following example shows how the MIMTCP server for port 1360 is stopped and a new server for port 1337 is started:

```
$ @MIMER$EXE:tcpcontrol
Usage: TCPCONTROL STATUS        ! Display status for all MIMTCP processes
       TCPCONTROL START  [port] ! Start MIMTCP process for a port
       TCPCONTROL STOP   [port] ! Stop the MIMTCP process for a port
       TCPCONTROL STOP   ALL    ! Stop all running MIMTCP processes
$ @MIMER$EXE:tcpcontrol status
     Pid    Port Version     Username   Started
00000227    1360 1101A       SYSTEM     25-AUG-2017 22:14:29.22
$ @MIMER$EXE:tcpcontrol stop 1360
MIMTCP process for port 1360 STOPPED
$ @MIMER$EXE:tcpcontrol start 1337
Starting MIMTCP process version 1101A
%RUN-S-PROC_ID, identification of created process is 00004D04
$ @MIMER$EXE:tcpcontrol status
     Pid    Port Version     Username   Started
00004D04    1337 1101A       SYSTEM     18-AUG-2017 12:09:11.96
```

Starting and stopping MIMTCP servers explicitly using the TCPCONTROL procedure is rarely needed. When the MIMCONTROL/START command is used, a MIMTCP process will be started automatically. This process will normally be active until the machine is shut down. Since the MIMTCP process does not hold any resources, it is not necessary to shut it down explicitly in the machine shutdown procedure.

Using a Memory Resident Buffer Pool

The buffer pool can be created in two different ways. If the MULTIDEFS.DAT parameter `BPResident` is left blank (the default) the buffer pool is allocated in normal process memory and is backed by the paging file. This means that the buffer pool is subject to normal virtual memory paging; if the system memory requirements increases the operating system may page out parts of the buffer pool.

Since the paging file is used as backing store the paging file quota of the database server process is increased to a suitable value. The working set quotas are also increased. These quotas are ultimately limited by the system parameter WSMAX, so when a large buffer pool is created with this method the WSMAX parameter must be increased accordingly.

At database server startup, the WSMAX parameter is checked. If it is insufficient an error message is displayed with the required value.

If the `BPResident` parameter specifies a name, the buffer pool is allocated as a memory resident area, i.e. physical memory in the machine is reserved. This has several advantages:

- Since physical memory is used, the buffer pool contents are always available and is never swapped out.
- The paging files are not used for the buffer pool; they do not need to be extended.
- The buffer pool does not use working set quota for the server process.
- The buffer pool uses a larger virtual page size which makes memory accesses more efficient.
- It is possible to reserve a large memory area for the buffer pool at VMS boot time.

The name that `BPResident` specifies will be use to name the memory resident global section. Please note that it is case sensitive. The name must be unique (the same section can not be used by two different database servers).

The user that starts a server using a resident memory area must have the process right VMS$MEM_RESIDENT_USER. This can be granted to a user by using AUTHORIZE:

```
$ SET DEF SYS$SYSTEM
$ MCR AUTHORIZE
UAF> GRANT/IDENTIFIER VMS$MEM_RESIDENT_USER SMITH
%UAF-I-GRANTMSG, identifier VMS$MEM_RESIDENT_USER granted to SMITH
UAF> EXIT
```

It is also possible to create a resident memory reservation so that the VMS system puts aside resident memory for the buffer pool at system boot time. By doing this and then running AUTOGEN, the VMS system can be appropriately tuned. This is described in the VMS document *System Manager Manual Vol 2, section 3.11 Reserved Memory Registry*. The name used in the registration must match the name used in the BPResident parameter.

Determining Buffer Pool Size

The buffer pool size is calculated by the MIMCONTROL/STATUS command. This command reads the parameter configuration in the MULTIDEFS.DAT file and calculates the required buffer pool size. The size is displayed in KBytes or MBytes and is rounded upwards so the value can be used for a resident memory reservation.

To see the exact buffer pool size, use the MIMCONTROL/STATUS/DCL command. The last value returned in the MIMER_STATUS symbol is the buffer pool size in bytes.

Process Quotas for the Database Server

When MIMCONTROL/START is used to start a database server process quotas are calculated according to the MULTIDEFS.DAT parameter file.

The size of the database server is calculated. The size includes communication areas (70K per `User` as specified in MULTIDEFS.DAT), thread stacks, local data, initial SQL Pool (`SQLPool` parameter) and code. If `BPResident` is not specified, the buffer pool size is also included.

The WSDEFAULT and WSQUOTA (working set quotas) of the server process is set to the calculated server size.

The PGFLQUOTA (page file quota) is set to the calculated server size plus the size the SQL pool can grow to according to the MULTIDEFS.DAT parameter `MaxSQLPool`. This means that the `MaxSQLPool` parameter can be used to control the paging file quota for the process.

The WSEXTENT quota is set to the WSMAX system parameter.

Troubleshooting Tips

In order to successfully start a database server, the following conditions must be fulfilled:

- The system databank file, `SYSDB110.dbf`, must have been created. See *Generating System Databanks and SYSADM* on page *17*
- There must be an entry for the database in the local section of the SQLHOSTS file. See *Editing the SQLHOSTS File* on page *15*
- The ProcName of the MULTIDEFS file must not specify a process name prefix that is identical to that of another running multi-user system.
- There must not be any other node in a cluster which has started the same database server.
- The database must not be in use in single-user access mode at the time the database server is started.
- The file `SYS$MANAGER:MIMERKEY.DAT` must contain a valid Mimer SQL license key.

Chapter 5

Running Mimer BSQL Scripts

This chapter describes how to run BSQL scripts in the Mimer SQL environment.

Running a BSQL Script

BSQL can be used to run SQL commands from within a script. There are several ways to use BSQL and some examples are given here.

You can use the READ command from within BSQL to read a file containing SQL statements:

```
$ CREATE Q.SQL
SELECT 1+1 FROM SYSTEM.ONEROW;
$ BSQL/USER=SYSADM/PASSW=SYSADM
READ 'Q.SQL';
```

To use BSQL from within a command procedure (.COM file), you do like this:

```
$ BSQL
SYSADM
SYSADM
SELECT 1+1 FROM SYSTEM.ONEROW;
$ ! next DCL command beginning with dollar ends program input.
```

To execute a single SQL command, the `/QUERY` switch can be used with BSQL:

```
$ BSQL/USER=SYSADM/PASSW=SYSADM/QUERY="SELECT 1+1 FROM SYSTEM.ONEROW"
```

The VMS command PIPE can be used to create Unix-like pipes with Unix-like redirection. This command reads SQL statements from the file Q.SQL:

```
$ PIPE BSQL/USER=SYSADM/PASSW=SYSADM < Q.SQL > RESULT.TXT
```

Old-time VMS users would achieve the same thing by redefining `SYS$INPUT` and `SYS$OUTPUT`. By defining them in USER mode, the definitions are dropped automatically at image (program) exit:

```
$ DEFINE/USER SYS$INPUT Q.SQL
$ DEFINE/USER SYS$OUTPUT RESULT.TXT
$ BSQL/USER=SYSADM/PASSW=SYSADM
```

A problem with all the examples above is that the password for the Mimer ident used is stored in a file. This should generally be avoided unless the security of the file can be guaranteed.

The problem can be solved by adding an OS_USER login to the user. A user with an OS_USER login can log in without providing a password. (This does not work over TCP/IP.)

For example:

```
$ BSQL /USER=SYSADM /PASSW="SYSADM"
SQL> CREATE IDENT PER AS USER;
SQL> ALTER IDENT PER ADD OS_USER 'PER';
SQL> EXIT;
```

Now VMS user PER can do:

```
$ BSQL /USER="" /QUERY="SELECT 1+1 FROM SYSTEM.ONEROW"
```

Chapter 6

Using the JDBC Driver

The Mimer SQL distribution includes a JDBC driver. This driver enables Java programs running on OpenVMS to access any Mimer SQL database server running at least version 8.2.

The JDBC driver is a 'type 4' driver which means that it is written entirely in Java, and can be moved to any platform supporting Java.

The driver, MIMJDBC3.JAR, supports the JDBC 3 standard, and is for Java 1.4 or later.

For more information about Java and JDBC, please see:

- `SYS$COMMON:[JAVA*.DOCS]INDEX.HTML` – Java information for more recent versions
- `MIMER$DOC:MIMJDBEN.PDF` – Information on the Mimer JDBC driver.
- https://www.oracle.com/technetwork/java/javase/jdbc/index.html – JDBC technology information.

Using the JDBC Driver

To use the JDBC driver, you must first set up the OpenVMS Java environment.

Defining Java Commands

Use the following command to define the Java commands (Java version 1.4.2 is used in the example):

```
$ @SYS$MANAGER:JAVA$142_SETUP
```

Setting CLASSPATH

To use a Mimer JDBC driver, the Java environment must be able to find it.

The logical name CLASSPATH is used for this purpose. This logical name contains a list of directories and Java archives (`.ZIP` and `.JAR` files).

On OpenVMS you can use either the JAVA$CLASSPATH or CLASSPATH logical name to specify a Java classpath. The JAVA$CLASSPATH logical name is easier to use since it uses standard OpenVMS file specifications.

Example

```
$ DEFINE JAVA$CLASSPATH MIMER$LIB:MIMJDBC3.JAR, SYS$DISK:[]
```

It is also possible to use the CLASSPATH logical name. This logical name uses a Unix syntax to specify a search path. Please read the OpenVMS Java documentation for details.

The following example sets the CLASSPATH logical name to include the Mimer JDBC 1 driver.

Since the equivalence string becomes rather long, and must be enclosed in quotes, a DCL string is constructed.

Example

```
$ SHOW LOG CLASSPATH
"CLASSPATH" = "/sys$common/java/lib/JDK118_CLASSES.ZIP:."
(LNM$PROCESS_TABLE)
$ CLASSPATH=F$TRNLNM("CLASSPATH")+":/MIMER$LIB/MIMJDBC3.JAR"
$ DEFINE CLASSPATH "''CLASSPATH'"
```

The Mimer JDBC driver should now be accessible.

Note: Using logical names that enclose directory specifications with < and > is problematic in Java. Please make sure you use `[` and `]` instead.

Verifying the Environment

Since the driver contains a `main()` function, it is possible to execute it as a program for testing purposes.

Use the `-version` switch to verify that the Java environment can locate and use the Mimer JDBC driver. Note that quotes must be used since Java package names are case sensitive.

```
$ JAVA "com.mimer.jdbc.Driver" -version
Mimer JDBC driver version 3.31 (12385)
```

Testing the Connection

Use the `-ping` switch to test that the driver can make a connection with a Mimer SQL v10.1 database server.

Please read the JDBC driver guide for an explanation of the syntax of the connection URL.

```
$ java "com.mimer.jdbc.Driver" -ping
"jdbc:mimer:local://SYSADM:SYSADM@/targetdb
Database connection established.
getDatabaseProductName():    Mimer SQL Experience
getDatabaseProductVersion(): 11.00.0000 Mimer SQL Experience 11.0.0a

Ping tests:
  0     0 ms
  1     0 ms
  2     1 ms
  3     0 ms
  4     0 ms
  5     1 ms
  6     0 ms
  7     0 ms
  8     0 ms
  9     0 ms
avg     0 ms      min     0 ms      max     1 ms
```

Finally, compile and execute the JDBC example program. You should copy the example program to a private directory and edit it in order to set the connection URL string, database user name and passwords.

```
$ SET DEF [SOMEWHERE.PRIVATE]
$ COPY MIMER$EXAMPLES:EXAMPLE.JAVA []
$ ! Edit the example. Alter the URL and username/password
$ EDIT EXAMPLE.JAVA
$ JAVAC EXAMPLE.JAVA
$ JAVA "Example"
```

Appendix A

Distributed Files

When installing Mimer SQL with the `PRODUCT INSTALL` command, a new directory named `[MIMER`*xxxxx*`]` is created under `SYS$COMMON:`, where *xxxxx* is the version number of the product. A few files are placed outside the Mimer directory in system directories.

Files in System Directories

The names of files placed in system directories are suffixed with *_xxxxx*, where *xxxxx* is the version number of the Mimer product.

File Name	Description
`SYS$SHARE:MIMCOMM_xxxxx.EXE`	Image for fast Mimer access in 32-bit Java
`SYS$SHARE:MIMCOMM64_xxxxx.EXE`	Image for fast Mimer access in 64-bit Java
`SYS$SHARE:MIMER$DBP_xxxxx.EXE`	Protected image for shared memory communication
`SYS$SHARE:MIMER$DBS_xxxxx.EXE`	Executable code for single-user access
`SYS$SHARE:MIMER$MIMERAPI_xxxxx.EXE`	Shareable image with the Mimer SQL C API
`SYS$SHARE:MIMER$ODBC_xxxxx.EXE`	Image containing the ODBC interface
`SYS$SHARE:MIMER$SQL_xxxxx.EXE`	Image containing the API for Embedded SQL
`SYS$STARTUP:MIMER$SHUTDOWN_xxxxx.COM`	Command procedure that deinstalls shareable images. Should be called from SYS$MANAGER:SYSHUTDWN.COM
`SYS$STARTUP:MIMER$STARTUP_xxxxx.COM`	Command procedure that installs shareable images. Should be called from SYS$MANAGER:SYSTARTUP_VMS.COM
`SYS$SYSROOT:[000000]MIMERxxxxx.DIR`	Directory tree with files for this specific version of Mimer SQL

Files in the Mimer Root Directory

File Name	Description
SETUP.COM	Command procedure defining logical names and commands for a process.
VERSION.DAT	Contains the version number of the Mimer SQL distribution.

Documentation Files (MIMER$DOC)

File Name	Description
DTOA_LICENSE.TXT	License text for the DTOA library used by Mimer SQL,
MIMER_LICENSE.TXT	License text for Mimer SQL
MIMJDBEN.PDF	JDBC usage guide.
MIMPERF_T4.TXT	Describes how to add Mimer SQL performance metrics to the T4 tool.
MIMSQLEN.PDF	The Mimer SQL documentation.
MIMVMS.PDF	The VMS guide.
README.TXT	A short description of Mimer SQL and how to get started.
RELNOTEN.PDF	Mimer SQL Release Notes.

Example Files (MIMER$EXAMPLES)

File Name	Description
BLOBSAMP.EC	Example of a program written in embedded C that stores and retrieves binary data.
COMINGSOON.EC	Example program in embedded C that calls a PSM procedure.
DEFAULTKEY.MCFG	Mimer license for up to ten simultaneous users. For testing purposes.
DSQL.EC	Embedded C examples that demonstrates the use of dynamic SQL.
DSQL.H	Header file for the dynamic SQL example.
DSQLSAMP.C	Main program for the dynamic SQL example.
DYNCALL.EC	Example program in embedded C that demonstrates a dynamic query.
EXAMPLE.EC	Very simple embedded C example.
EXAMPLE.ECO	Very simple embedded COBOL example.
EXAMPLE.EFO	Very simple embedded FORTRAN example.
Example.java	Java example program using JDBC.
EXDAT.SQL	A BSQL script that populates an example database.
EXDEF.SQL	A BSQL script that creates example users, tables and procedures.
FREQCALL.EC	Example of a program written in embedded C that calls a stored procedure.
FREQCALL.ECO	Same as FREQCALL.EC, but written in COBOL.
FREQCALL.EFO	Same as FREQCALL.EC, but written in FORTRAN.
MIMERAPIEX1.C	Simple Mimer SQL C API test program.
MIMERAPIEX2.C	Simple Mimer SQL C API test program.
MIMERAPIEX3.C	Simple Mimer SQL C API test program.
OSQL.C	An ODBC example
SINGLEDEFS.DAT	Contains a template for database parameters in single-user mode, see the *Mimer SQL System Management Handbook*.
SQLHOSTS.DAT	Contains a template for the SQLHOSTS.DAT file. The actual SQLHOSTS file is pointed to by the MIMER_SQLHOSTS logical name (normally SYS$MANAGER:SQLHOSTS.DAT). Do not edit this template file.

File Name	Description
T4$MIM_MON.COM	Command procedure that helps with integrating Mimer SQL with the T4 tool. See MIMER$DOC:MIMPERF_T4.TXT for more information
WAKECALL.EC	Example of a program written in embedded C that calls a stored procedure.
WAKECALL.ECO	Same as WAKECALL.EC, but written in COBOL.
WAKECALL.EFO	Same as WAKECALL.EC, but written in FORTRAN.

Executable Programs (MIMER$EXE)

File Name	Description
BSQL.EXE	Program that executes SQL statements which are entered interactively or read from a command file. It is described in the *Mimer SQL User's Manual*.
DBC.EXE	Program that can check if a databank file is internally consistent. It is described in the *Mimer SQL System Management Handbook*.
DBOPEN.EXE	Program that opens and restarts all databanks in a database. It is described in the *Mimer SQL System Management Handbook*.
ESQL.EXE	Pre-processor for embedded SQL.
EXLOAD.EXE	Program to load the example database.
MIMCONTROL.EXE	The MIMCONTROL command, which is used to control database servers,*The MIMCONTROL Command* on page *23*.
MIMEXPER.EXE	The Mimer SQL Experience database server. Do not start this program directly. Use MIMCONTROL to start a database server and specify which type of server to start by setting the parameter ServerType in the MULTIDEFS.DAT file.
MIMINFO.EXE	Program that can display status information for a database server.
MIMINM.EXE	The Mimer SQL InMemory database server. Do not start this program directly. Use MIMCONTROL to start a database server and specify which type of server to start by setting the parameter ServerType in the MULTIDEFS.DAT file.
MIMLICENSE.EXE	Application used to administrate the license key(s).
MIMLOAD.EXE	Utility to load and unload data.
MIMPERF.EXE	A program to display server performance counters.
MIMREPADM.EXE	Administration utility for Mimer Replication.
MIMSYNC.EXE	Utility to bring two replicated Mimer systems back into sync.

File Name	Description
MIMTCP.EXE	The program executed by the MIMTCP server, see *The MIMTCP Server* on page *28*. Do not start this program directly.
REPSERVER.EXE	Mimer Replication server.
SDBGEN.EXE	Program used to create the initial Mimer SQL system databank files in a database, described in the *Mimer SQL System Management Handbook*.
SQLMONITOR.EXE	SQL Monitor tool. Described in the *Mimer SQL System Management Handbook*.
TCPCONTROL.COM	Procedure to manage MIMTCP processes.

Library Files (MIMER$LIB)

File Name	Description
MIMER.CLD	Command definitions for all the executable programs supplied with the Mimer SQL software.
MIMER.HLB	HELP library for Mimer.
MIMER$MIMERAPI.OPT	Options file used for linking Mimer SQL C API applications.
MIMER$ODBC.OPT	Options file used for linking ODBC applications.
MIMER$SQL.OPT	Options file used for linking Mimer SQL embedded applications.
MIMERAPI.H	Header file for Mimer SQL C API programs.
MIMERRORS.H	Header file for Mimer SQL C API programs.
MIMJDBC3.JAR	JDBC 3 driver.
MIMODBC.H	C-definitions for Mimer ODBC specific descriptor attributes, working with 64-bit integers.

Appendix B

Data Types Used in Mimer SQL

The following sections explain how to compile applications using floating point data types, and what data types Mimer SQL uses internally and externally.

Compiling Applications Using Floating Point Data Types

OpenVMS supports various floating point types. Mimer SQL uses the types that the C compiler uses as default. Unfortunately, this differs between the Alpha and Integrity platforms.

F_FLOAT	4 bytes	Used by Mimer on Alpha
S_FLOAT	4 bytes	Used by Mimer on Integrity
G_FLOAT	8 bytes	Used by Mimer on Alpha
T_FLOAT	8 bytes	Used by Mimer on Integrity
D_FLOAT	8 bytes	Not supported by Mimer
H_FLOAT	16 bytes	Not supported by Mimer

External Data Types Supported by Mimer SQL

Any value stored in the database may be read into host language variables as described in the *Mimer SQL Programmer's Manual*.

Mimer SQL will perform all the necessary conversions and will signal an error if the value to be converted is not compatible with the destination type.

Index