Mimer SQL

OpenVMS Guide

Version 10.0.8

Mimer SQL, OpenVMS Guide, Version 10.0.8, October 2018

Produced and published by Mimer Information Technology AB, Uppsala, Sweden.
P.O. Box 1713,
SE-751 47 Uppsala, Sweden.
Tel +46(0)18-780 92 00.
Fax +46(0)18-780 92 40.

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 10.0 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 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 e-mail 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.

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.

Three drivers are supplied. Each driver support a different JDBC standard. Pick the one that best matches your needs.

JDBC Driver	JDBC Standard	Java environment
MIMJDBC1.JAR	JDBC 1	Java 1.1.8 or later
MIMJDBC2.JAR	JDBC 2	Java 1.2 to Java 1.3
MIMJDBC3.JAR	JDBC 3	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.

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*.
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*.

Utility	Description
EXPTOLOAD	A program that converts export files generated by the old UTIL program to a format suitable for MIMLOAD. 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*.
MIMREPADM	Administration utility for Mimer Replication.
MIMSYNC	Utility to bring two replicated Mimer systems back into sync.
PSMDEBUG	Java-based utility for debugging PSM procedures.
REPSERVER	Replication server.
SDBGEN	SDBGEN generates the Mimer SQL system databanks SYSDB, TRANSDB, LOGDB and SQLDB.
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

For latest news regarding the OpenVMS version to use with Mimer SQL, please see our developer site: https://developer.mimer.com/vms.

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 HP).

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. You can find the files in the `[MIMER1008B.DOC]` directory. Once `MIMESETUP` is executed, you can use the `MIMDOC` logical name for convenient access to the documentation directory.

Documentation Conventions

Convention	Example	Explanation
All uppercase	COMMIT	Indicates command names, SQL reserved words, and keywords
`Monospace`	`$ SET COMMAND MIMLIB:MIMER`	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	Batch SQL, a program used to execute SQL statements which are read from a command file or entered interactively.
CLD	A Command Language Definition (CLD) file that contains definitions for new DCL commands.
Data source	ODBC term for a database.
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.

Term	Explanation
Database home directory	The directory where the SYSDB databank file is located, also recorded in the SQLHOSTS file.
DCL	Digital Command Language.
DCL$PATH	A logical name used by DCL to find executable programs. By including the MIMEXE directory in the definition of this logical name, all Mimer SQL programs can be started by typing their names directly. Some of the programs also accept UNIX-style command options when started in this fashion.
Dynamic SQL	SQL statements constructed at runtime and passed to the database management system for execution.
Embedded SQL	The term used for SQL statements when they are embedded in a traditional host language.
ESQL	The preprocessor for embedded SQL.
Java	A platform independent language invented by Sun and now owned by Oracle. See http://www.oracle.com/technetwork/java/index.html
JDBC	Java DataBase Connectivity. A set of Java interfaces for accessing relational databases. See http://www.oracle.com/technetwork/java/javase/jdbc/index.html
MIMERxxxxx	Symbolic name for the distribution directory tree, unique for each Mimer SQL release, where "*xxxxx*" stands for the current version number, e.g. "1008B".
ODBC	Open Database Connectivity, a specification for a database API in the C language, independent of any specific DBMS or operating system.
PSM	Persistent Stored Modules, the term used by ISO/ANSI for Stored Procedures.
Shadow	A shadow is a copy of the original (master) databank and is continuously updated by Mimer SQL. A Mimer SQL databank may have one or more shadows. If the databank is shadowed, there will be one file for each shadow. If the master databank is lost, it is possible to continue operations from the shadow databank without stopping the database server. A databank must have the TRANS or LOG option to be shadowed. For more information, see the *Mimer SQL System Management Handbook*.
SQL	Structured Query Language, standardized language for database manipulation.
SQLHOSTS	A file containing lookup information for all accessible Mimer SQL databases, relative to the current node. It is located at `SYS$MANAGER:SQLHOSTS.DAT`.

Term	Explanation
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

The following steps provide an overview of how to install Mimer SQL for OpenVMS.

Step 1 Unpack the distribution file to a directory tree

To unpack the file, simply execute it. Note that the `MIMER1008B.EXE` file must reside in your current directory, so you may have to copy the file.

A new directory will be created in your current default directory. For example:

```
$ COPY MIMER1008B.EXE somedisk:[000000]
$ SET DEF somedisk:[000000]
$ RUN MIMER1008B
```

For more information, see *Unpacking the Mimer SQL Distribution File* on page *8*.

Step 2 Set-up the Mimer SQL Environment

Using MIMSETUP, you set up locations for programs, libraries, data files, documentation, etc., for users and applications. For example:

```
$ @somedisk:[MIMER1008B]MIMSETUP SYS
```

For more information, see *Setting-up the Mimer SQL Environment* on page *9*.

Step 3 (Optional) Install your Mimer SQL License Key

A default key, for test and development only, is automatically installed, that is, you can finish the installation without adding a key.

But, if you are going to put Mimer SQL into production you must install a run-time license.

See *Installing the Mimer SQL License Key* on page *11*.

Step 4 (Optional) Add a MIMSETUP command in the SYSTARTUP_VMS.COM file

In order to set-up Mimer SQL each time the system boots, include the MIMSETUP command in the system startup file: `SYS$MANAGER:SYSSTARTUP_VMS.COM`, for example:

```
$ @disk:[MIMER1008B]MIMSETUP SYSTEM
```

When you have carried out the steps above, you are ready to create your first database, see *Chapter 3, Establishing a Database*.

Unpacking the Mimer SQL Distribution File

The most common way of acquiring the Mimer SQL distribution for OpenVMS is to download it, in ZIP file format, from https://developer.mimer.com/downloads. The ZIP file is created using Info-ZIP.

Once downloaded, the file looks like an ordinary executable (`*.exe`) file, for example, the file containing Mimer SQL version 10.0.8B is named `MIMER1008B.EXE`.

Unpacking the ZIP File

To unpack the contents of the ZIP file, execute the file. Note that the `MIMER1008B.EXE` file must reside in your current directory, so you may have to copy the file.

The Mimer SQL distribution directory will be created under the current directory. For example:

```
$ COPY MIMER1008B.EXE somedisk:[000000]
$ SET DEF somedisk:[000000]
$ RUN MIMER1008B
```

Note that you must download an executable file that matches your server platform, Alpha or Integrity. If you try to execute the wrong distribution file, you will get an error message.

Ensuring Correct File Protections

Note: The three consecutive dots in the "[…]" construction used in the OpenVMS command examples that follow, together with savesets, are an essential part of the command syntax. If they are omitted, all files on the distribution media will be assigned to a single directory and the Mimer SQL installation will fail.

The auto-extract facility may not always apply the correct file protections to the files it extracts.

Therefore, we recommend that you run the following commands to ensure that the correct file protections are applied to the files in the tree:

```
$ SET FILE/PROT=(S:RWED,O:RWED,G:RE,W:RE) [MIMERxxxxx...]*.*
$ SET FILE/PROT=(S:RWE,O:RWE,G:RE,W:RE) MIMERxxxxx.DIR
```

The Mimer SQL Directory Tree

Once you have successfully unpacked Mimer SQL, you can review the directory structure in which the Mimer SQL Version 10.0.8 software resides.

The name of the installation root directory in the tree contains the word 'MIMER' and the version number of the product, for example: 'MIMER1008B' (generally denoted as MIMER*xxxxx*).

The name of the root directory is unique for each Mimer SQL release. This makes it easier to install new versions without affecting any previous versions of the product.

For more information about the files installed, see *Appendix A Distributed Files*.

Setting-up the Mimer SQL Environment

Before you can run Mimer SQL, you must carry out certain setup operations. These include defining locations for programs, libraries, data files and documentation for users and applications.

You set up Mimer SQL using the MIMSETUP command procedure. MIMSETUP defines the logical names needed to run Mimer SQL applications.

You can find the MIMSETUP command procedure in the Mimer SQL root directory.

Note: When running MIMSETUP, you may require some of the following privileges: SYSPRV, CMKRNL, SYSNAM, see *Privileges* on page *10* for details.

MIMSETUP Syntax

The syntax for the MIMSETUP command is as follows:

```
$ @disk:[MIMERxxxxx]MIMSETUP [-][lnm-table]
```

The parameter `lnm-table` specifies which logical name table to use when defining the logical names required to access a Mimer SQL installation.

If the parameter `lnm-table` is preceded by a hyphen (`-`), the MIMSETUP command procedure will remove the effects of any Mimer SQL setup previously performed for the specified table, including uninstalling shareable images.

Valid Values

Valid values for `lnm-table` are:

- SYSTEM
- GROUP
- JOB
- PROCESS

In general, we recommend that you execute MIMSETUP to update the SYSTEM logical name table so that the definitions are available to all users.

SYSTEM or GROUP

If you specify SYSTEM or GROUP, the following shared images will be installed if they are not installed already:

- MIMLIB:MIMDBP.EXE
- MIMLIB:MIMDB.EXE
- MIMLIB:MIMDBS.EXE

Therefore, for a Mimer SQL installation, you must perform a SYSTEM or GROUP level setup at least once in order to get these essential shared images installed. For more information, see *Shared Images* on page *43*.

PROCESS or JOB

You can perform MIMSETUP at the PROCESS or JOB level to set up logical names that may be different to those available from the SYSTEM or GROUP level (no shared image installation is involved in a PROCESS or JOB level setup).

If you run MIMSETUP without specifying the `lnm-table` parameter, a PROCESS level setup is performed by default.

Privileges

In order to run a SYSTEM-wide MIMSETUP, you must have SYSPRV, CMKRNL and SYSNAM privileges.

When logical names are defined in the SYSTEM table, they are defined in executive mode.

To run a GROUP-wide MIMSETUP, you must have SYSPRV and CMKRNL privileges.

System Startup Command File

Since all setups have to be re-executed each time the OpenVMS system is booted, we recommend that you enter the command(s) in the system startup command file: `SYS$MANAGER:SYSTARTUP_VMS.COM`.

For example:

```
$ @disk:[MIMER1008B]MIMSETUP SYSTEM
```

MIMSETUP Examples

Defining Logical names SYSTEM Wide

The following example defines logical names SYSTEM wide, that is, all OpenVMS users may access the Mimer SQL installation. Shareable images are installed.

```
$ @SDEPT2:[MIMER1008B]MIMSETUP SYSTEM
```

Overriding the Default Definition of the Logical Names

The following example shows how any user can override the default definition of the logical names. This is useful when a user wants to test an alternative Mimer SQL installation. No shareable images are installed.

```
$ @SDEPT2:[MIMER1008B]MIMSETUP
```

Removing a Mimer SQL Setup

The following example demonstrates how to remove a Mimer SQL setup which was previously made GROUP wide by running MIMSETUP (MIMROOT was defined by MIMSETUP). Shareable images are uninstalled.

```
$ @MIMROOT:[000000]MIMSETUP -GROUP
```

Logical Names Defined by MIMSETUP

The MIMSETUP command defines the logical names listed below:

Logical Name	Description
MIMCOMM	Points to a native communication library for JDBC.
MIMDB	Logical name pointing to the Mimer SQL shareable library. Used when starting Mimer SQL applications.
MIMDBP	Logical name pointing to the `MIMDBP` image. Used when starting Mimer SQL applications.
MIMDOC	Points to the directory containing on-line documentation.
MIMER_SQLHOSTS	Points to the SQLHOSTS file which contains one entry for every accessible Mimer SQL database. Normally this logical name is set to `SYS$SPECIFIC:[SYSMGR]SQLHOSTS.DAT`.
MIMEXAMPLES	Points to the examples directory in the Mimer SQL distribution.
MIMEXE	Points to the directory containing executable programs in the Mimer SQL distribution.
MIMLIB	Points to the directory containing application libraries, CLD files, etc.
MIMODBC	Points to the Mimer ODBC shared library.
MIMROOT	A concealed logical name pointing to the root of the Mimer SQL distribution.

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 e-mail info@mimer.com.

Node Name and OpenVMS version

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

There are three ways of obtaining the node name:

1 Use MIMSETUP, for example:

```
$ @disk:[MIMERxxxxx]MIMSETUP
```

2 If the OpenVMS node is part of a cluster, the `scsnode` parameter describes its name:

```
$ WRITE SYS$OUTPUT F$GETSYI("SCSNODE")
```

3 If the OpenVMS system is not clustered, the node name parameter may be blank. In this case, you should check the `SYS$NODE` logical name instead:

```
$ SHOW LOG SYS$NODE
```

To get the OpenVMS version:

Use the command:

```
$show sys /noproc
```

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 that SET COMMAND MIMLIB:MIMER is done, see *Setting the Command Style* on page *20*, enter the following:

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

For example, if the license key file you received was named `1234.mcfg`, you would enter the following:

```
$ mimlicense /FILE=1234.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.

On OpenVMS 8.4 or earlier, the Mimer SQL license keys are stored in the file:

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

On OpenVMS 8.4-1 or later, the Mimer SQL license keys are stored in the file:

```
SYS$SPECIFIC:[SYSMGR]MIMERKEY_IA64.DAT
```

Removing a Mimer SQL Installation

> **Caution:** If you plan to remove any Mimer SQL databases, see *Removing a Mimer SQL Database* on page *21* **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. Run the MIMSETUP command procedure to uninstall shared images and deassign logical names:

   ```
   $ disk:[MIMERxxxxx]MIMSETUP -SYSTEM
   ```

3. Delete the Mimer SQL directory tree, for example:

   ```
   $ SET DEF disk:[000000]
   $ SET PROC/PRIV=BYPASS
   $ DELETE [.MIMERxxxxx...]*.*.*
   $ DELETE [.MIMERxxxxx...]*.*.*
   $ DELETE [.MIMERxxxxx...]*.*.*
   $ SET PROC/PRIV=NOBYPASS
   ```

   Note: You may have to issue the DELETE command more than once, because the DELETE command will not remove directory files unless they are empty.

4. 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*.DAT.*
   ```

5. 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`, remove the commands.
6. 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 MIMSETUP command will define it to be:

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

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

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 10.0. Upgrade can be done for databank files created by Mimer SQL version 7.1 and later.

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

SDBGEN Syntax

Assuming that SET COMMAND MIMLIB:MIMER is done, 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 2 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. (A Mimer SQL 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 set-up logical names and install images, command style and automatic database server startup and shutdown.

Setting-up Logical Names and Install Images

In order to set-up Mimer SQL logical names and install images each time your OpenVMS system starts up, you must edit your startup file.

Edit `SYS$MANAGER:SYSSTARTUP_VMS.COM` to include the following line:

```
$ @disk:[MIMER1008B]MIMSETUP SYSTEM
```

Setting the Command Style

You can use either OpenVMS- or UNIX-style commands.

To set-up your system to automatically accept the style you prefer, you can edit the `LOGIN.COM` file.

OpenVMS-style Commands

In `LOGIN.COM`, add the following line:

```
$ SET COMMAND MIMLIB:MIMER
```

UNIX-style Commands

In `LOGIN.COM`, add the following line:

```
$ DEFINE DCL$PATH MIMEXE
```

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,A2,2012-02-16
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 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 MIMEXE: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:

```
$ @mimexe: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
$ @mimexe:tcpcontrol status
     Pid    Port Version     Username   Started
00000227    1360    1007C       SYSTEM   25-MAY-2013 22:14:29.22
$ @mimexe:tcpcontrol stop 1360
MIMTCP process for port 1360 STOPPED
$ @mimexe:tcpcontrol start 1337
Starting MIMTCP process version 1007C
%RUN-S-PROC_ID, identification of created process is 00004D04
$ @mimexe:tcpcontrol status
     Pid    Port Version     Username   Started
00004D04    1337    1007C       SYSTEM   18-NOV-2012 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.

The maximum supported buffer pool size is 16 GBytes.

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, `SYSDB100.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.
- The MIMSETUP command procedure must have defined the logical names to be SYSTEM-wide or GROUP-wide.
- There must not be any logical names in the JOB or PROCESS tables that override the SYSTEM or GROUP definitions.
- The shareable image in the file named MIMLIB:MIMDBP.EXE must be properly installed.
- 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 SQL Applications

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

It covers information that applies to the utilities included in the Mimer SQL installation as well as to applications that may have been created to access a Mimer SQL database.

This chapter also describes:

- Defining whether OpenVMS-style or UNIX-style command-line flags are accepted by the utilities which are supplied as part of the Mimer SQL installation.
- Selecting a Mimer SQL installation – if several Mimer SQL installations reside on the same computer, it is essential that users access the correct one.

Executing Mimer SQL Utilities

This section describes the various ways an application can be executed under OpenVMS and also describes how to set up the Mimer SQL-supplied utilities to use UNIX-style or OpenVMS-style command-line flags.

Using the DCL command RUN

The DCL command `RUN` can be used as follows:

```
$ RUN MIMEXE:BSQL
```

You cannot specify any flags or other input parameters on the command-line when you use the RUN command.

Some of the Mimer SQL-supplied programs allow parameters and options to be supplied via logical names, e.g. `MIMER_DATABASE` to supply a database name and `MIMER_MODE` to define the database access mode.

See documentation for the programs in the *Mimer SQL System Management Handbook* for specific details.

Using OpenVMS Command Definitions

You can set up the programs supplied by Mimer SQL so that they can be run by specifying the program name followed by the VMS-style command line flags and parameters.

You do this by defining the Mimer SQL programs as DCL command verbs.

Use the following OpenVMS command to define all the Mimer SQL-supplied programs as DCL command verbs:

```
$ SET COMMAND MIMLIB:MIMER
```

Example using OpenVMS-style command-line arguments:

```
$ BSQL/SINGLE TESTDB
```

You can un-define a DCL command by issuing the following command:

```
$ SET COMMAND/DELETE=command-name
```

Caution: Take care when using this DCL command, because any DCL command verb can be un-defined.

Using the DCL$PATH Logical Name

The `DCL$PATH` logical name defines a list of directories in which the OpenVMS operating system will look when trying to locate the executable for a specified program name.

Utilities started this way accept UNIX-style command options.

Note: If you set-up a Mimer SQL-supplied utility as a DCL command verb, the UNIX-style command-line flags and parameters are not used, even if MIMEXE is included in DCL$PATH.

In order for the utilities supplied by Mimer SQL to be run by specifying the utility name followed by the UNIX-style command line flags and parameters, you must include the `MIMEXE` directory in the directory list defined in `DCL$PATH`.

If there are other directories containing executables for programs that are to be run this way, those directories must also be included in the directory list defined in `DCL$PATH`.

For example, the following `DCL$PATH` definition:

```
$ DEFINE DCL$PATH MIMEXE,disk:<directory.app>
```

will allow the utilities supplied by Mimer SQL to be run by specifying the utility name followed by the UNIX-style command line flags and parameters.

It will also allow all programs found in the specified application directory to be run by specifying the program name.

Example using UNIX-style command-line flags:

```
$ bsql -s testdb
```

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 MIMER.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 MIMER.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 MIMER.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 creating OS_USER idents. An OS_USER is a user inside Mimer with the same name as your VMS user. An OS_USER 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 OS_USER;
SQL> EXIT;
```

Now VMS user PER can do:

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

Running the PSM Debugger

You can use the Mimer SQL PSM Debugger for debugging PSM procedures that are stored in the Mimer SQL database server. The PSM debugger is written in Java and requires a Java 2 environment.

Since the debugger accesses the Mimer SQL server by using the TCP protocol, you can execute the Debugger on any machine that has adequate Java support, such as a Windows or Linux machine.

If you want to execute the PSM debugger on the OpenVMS platform, you must first make sure that the Java 2 environment is active.

You can display the current Java version using the following command:

```
$ JAVA -VERSION
```

If you need to install a newer Java version on OpenVMS, you can download the installation kit from the following site: http://h18012.www1.hp.com/java/download/

Starting the PSM Debugger

To start the PSM debugger, use the following command:

```
$ JAVA -JAR MIMLIB:PSMDEBUG.JAR
```

More information about the PSM debugger can be found in the included on-line help file.

Selecting a Mimer SQL Installation

A host computer can have several versions of the Mimer SQL database system installed simultaneously.

Access to a specific version is done through logical names (MIMEXE, etc.). If several versions of Mimer SQL version are installed, you can use the MIMSETUP command procedure to specify exactly which version a program should work with.

Normally, you do a system wide definition of the logical names. However, you can specify another Mimer SQL version by running the MIMSETUP command procedure and specifying a GROUP, JOB or PROCESS logical name definition (see *MIMSETUP Syntax* on page *9* for details on MIMSETUP).

Note: When starting a database server, any JOB or PROCESS logical names will not be inherited by the database server process. The database server will use the Mimer SQL version specified in the GROUP or SYSTEM logical name tables.

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.

Three drivers are supplied. Each driver support a different JDBC standard. Pick the one that best matches your needs.

JDBC Driver	JDBC Standard	Java environment
MIMJDBC1.JAR	JDBC 1	Java 1.1.8 or later
MIMJDBC2.JAR	JDBC 2	Java 1.2 to Java 1.3
MIMJDBC3.JAR	JDBC 3	Java 1.4 or later

For more information about Java and JDBC, please see:

- `SYS$COMMON:[SYSHLP.JAVA.RELEASE_NOTES]JDK118_VMS_RELEASE_NOTES.HTML` – Java 1.1.8 information (old)
- `SYS$COMMON:[JAVA*.DOCS]INDEX.HTML` – Java information for more recent versions
- `MIMDOC:MIMJDBEN.PDF` – Information on the Mimer JDBC driver.
- http://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

When using Java 1.1.8, use the following commands to define the Java commands:

```
$ DEASSIGN JAVA$USE_DCL
$ @SYS$MANAGER:JAVA$SETUP
```

When using a more recent Java version, use the following 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$CLASSPATH MIMLIB:MIMJDBC2.JAR, SYS$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")+":/MIMLIB/MIMJDBC1.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 2.14
```

Testing the Connection

Use the `-ping` switch to test that the driver can make a connection with a Mimer SQL v10.0 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://SYSADM:PASSWORD@mynode/testdb"
Database connection established.
getDatabaseProductName():    MIMER/DB
getDatabaseProductVersion(): 10.00.0001 MIMER/DB 10.0.01

Ping tests:
  0     2 ms
  1     2 ms
  2     1 ms
  3     1 ms
  4     1 ms
  5     1 ms
  6     0 ms
  7     2 ms
  8     1 ms
  9     1 ms
avg     1 ms      min     0 ms      max     2 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 MIMEXAMPLES:EXAMPLE.JAVA []
$ ! Edit the example. Alter the URL and username/password
$ EDIT EXAMPLE.JAVA
$ JAVAC EXAMPLE.JAVA
$ JAVA "Example"
```

Appendix A

Distributed Files

The Mimer SQL distributed files are all located in a directory structure.

Root Directory Files

File Name	Description
DEFAULTKEY.MCFG	Default Mimer SQL license key for OpenVMS 8.4-1 or later.
DEFAULTKEY_OLD.MCFG	Default Mimer SQL license key for OpenVMS 8.4 or older.
MIMSETUP.COM	Command procedure that defines logical names and installs images, see *Setting-up the Mimer SQL Environment* on page *9*.
VERSION.DAT	Contains the version number of the Mimer SQL distribution.

Documentation Files (MIMDOC)

File Name	Description
MIMJDBEN.PDF	JDBC usage guide.
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 (MIMEXAMPLES)

File Name	Description
BLOBSAMP.EC	Example of a program written in embedded C that stores and retrieves binary data.
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.
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.
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.
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.
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 (MIMEXE)

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*.
DBSERVER.EXE	The database server. Do not start this program directly, use MIMCONTROL to do this.
ESQL.EXE	Pre-processor for embedded SQL.
EXLOAD.EXE	Program to load the example database.
EXPTOLOAD.EXE	Program converting old export files to the new mimload format.
MIMCONTROL.EXE	The MIMCONTROL command, which is used to control database servers,*The MIMCONTROL Command* on page *23*.
MIMINFO.EXE	Program that can display status information for a database server.
MIMLICENSE.EXE	Application used to administrate the license key(s).
MIMLOAD.EXE	Utility to load and unload data.
MIMREPADM.EXE	Administration utility for Mimer Replication.
MIMSYNC.EXE	Utility to bring two replicated Mimer systems back into sync.
MIMTCP.EXE	The program executed by the MIMTCP server, see *The MIMTCP Server* on page *26*. Do not start this program directly.
PSMDEBUG.JAR	The PSM Debugger. For more information, see *Running the PSM Debugger* on page *34*.
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*.
TCPCONTROL.COM	Procedure to manage MIMTCP processes.

Library Files (MIMLIB)

File Name	Description
LR.OLB	Library with entries for backward compatibility.
LRU.OLB	Library with entries for backward compatibility.
MDR.OLB	Library with entries for backward compatibility.
MIMCOMM.EXE	Library for shared memory communication in JDBC.
MIMDB.EXE	Shareable library image containing the code for the Mimer SQL database client API, see *Shared Images* on page *43*.
MIMDBP.EXE	Protected shareable library image containing code that performs secure and fast shared memory based communication with local database servers.
MIMDBS.EXE	Shareable library image containing code for running a Mimer SQL database in single-user mode. This library is mapped in dynamically when required.
MIMER.CLD	Command definitions for all the executable programs supplied with the Mimer SQL software.
MIMER.OPT	Options file used for linking Mimer SQL applications.
MIMJDBC1.JAR	JDBC 1 driver.
MIMJDBC2.JAR	JDBC 2 driver.
MIMJDBC3.JAR	JDBC 3 driver.
MIMODBC.EXE	ODBC driver library.
MIMODBC.H	C-definitions for Mimer ODBC specific descriptor attributes, working with 64-bit integers.
MIMSQLXA.OBJ	Object file containing an `xa_switch_t` entry defining mimsqlxa, which is used when accessing the XA routines in Mimer SQL.

Shared Images

The Mimer SQL distribution contains a number of shareable images which are located in the MIMLIB directory.

The shared images are installed by the MIMSETUP command procedure when defining logical names in the SYSTEM or GROUP name table.

Mimer SQL applications are linked with the shareable library `MIMLIB:MIMDB.EXE`.

When the application image is activated, the OpenVMS system locates the correct shareable image by using the logical name `MIMDB` (which MIMSETUP has defined as `MIMLIB:MIMDB`). This allows users to run applications with other versions of Mimer SQL by using the MIMSETUP procedure, without having to re-link the applications.

If you start an image that is installed with privileges or if you do not have read (R) access to the image file, OpenVMS will only use logical names defined in executive mode when activating shareable images. This is a security precaution that OpenVMS takes to avoid activating non-trusted shareable images together with trusted images.

Note: All logical names that MIMSETUP defines in the SYSTEM logical name table are defined in executive mode. This means that privileged or protected images will run the Mimer SQL version defined in the SYSTEM table even if there is another Mimer SQL version defined in one of the other tables!

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.

Appendix C

Using Version 9 Applications With Mimer SQL Version 10

A Mimer SQL version 9 application is linked with the `MIMDB9` shared library which provides client-side functionality for the application. The `MIMDB9` shared library communicates with the database server.

The API exported by the `MIMDB9` library is compatible with the new `MIMDB` library used in version 10. Likewise the client/server protocol used between the `MIMDB9`/`MIMDB` libraries and the database server is also compatible between versions. This means that there are several ways to run a Mimer SQL version 9 application with a Mimer SQL version 10 server.

Relink the Application

The preferred method is to relink the version 9 application with the new `MIMDB` shared library. This will make the application a true version 10 application and eliminate the setup needed by the other approaches.

This is the preferred method if the sources or object files of the application is still available.

Remap the Shareable Libraries

By defining the logical name `MIMDB9` to point to `MIMDB`, all version 9 applications will use the new `MIMDB` shared library

```
$ DEFINE MIMDB9 MIMDB
```

(Define the logical name in the `SYSTEM` or `GROUP` logical name tables as appropriate.)

This method is very similar to relinking the application, but requires extra setup. This setup will also affect all version 9 applications simultaneously, which may lead to undesired effects unless care is exercised.

Continue With the Old Shared Library

Since the client/server protocol used between the `MIMDB9` shared library and the database server is compatible between version 9 and version 10, the old version 9 application can access the version 10 server directly. There is no need to do anything special to the version 9 application.

The drawback with this method is that new features, optimizations and bug corrections in the new `MIMDB` shared library will not be available to the application; it will function as it did before and rely on the compatibility of the client/server protocol.

This method works for all kinds of communication protocols available; shared-memory, TCP/IP and DECNET. There is no need to do anything special in the `SQLHOSTS` file for this to work.

However, be aware that if you should try to access a database in single-user mode, the `MIMDB9` shared library will map the version 9 single-user library. This library will fail if it tries to open a version 10 database in single-user mode.

Index

J

L

M

O

P

R

S

T

U

Z