



MIMER

Installation and Usage Guide for VMS

Version 8.1

Copyright © 1998 Sysdeco Mimer AB

MIMER version 8.1 Installation and Usage Guide for VMS

May, 1999

Copyright © 1999 Sysdeco Mimer AB.

Published by Sysdeco Mimer AB,

P.O.Box 1713,

S-751 47 Uppsala, Sweden.

Tel +46(0)18-18 50 00.

Fax +46(0)18-18 51 00.

Internet: <http://www.mimer.com>

Produced by Sysdeco Mimer AB, Uppsala, Sweden.

All rights reserved under international copyright conventions.

The contents of this manual may be printed in limited quantities for use at a MIMER installation site. No parts of the manual may be reproduced for sale to a third party.

CONTENTS

1	INTRODUCTION	
1.1	Document objectives	1-1
1.2	Definition of terms	1-1
2	MIMER INSTALLATION	
2.1	VMS system requirements.....	2-1
2.2	Unpacking the MIMER distribution file.....	2-2
2.2.1	Unpacking a VMS saveset from a tape	2-3
2.2.2	Unpacking a VMS saveset from a file	2-3
2.2.3	Unpacking a self-extracting VMS ZIP file	2-3
2.3	The MIMSETUP8 command	2-4
2.3.1	MIMSETUP8 syntax	2-4
2.3.2	Logical names defined by MIMSETUP8.....	2-5
2.4	Installing the MRS license key	2-6
2.5	Establishing a MIMER database	2-6
2.6	Removing a MIMER installation	2-7
3	MIMER INSTALLATION COMPONENTS	
3.1	Distributed files	3-1
3.1.1	Root directory files	3-1
3.1.2	Documentation files (MIMDOC8)	3-2
3.1.3	Example files (MIMEXAMPLES8)	3-2
3.1.4	Executable programs (MIMEXE8).....	3-3
3.1.5	Library files (MIMLIB8).....	3-4
3.2	Shared images	3-5
4	RUNNING MIMER APPLICATIONS	
4.1	Executing MIMER applications.....	4-1
4.1.1	Using the DCL command RUN	4-1
4.1.2	Using the DCL\$PATH logical name	4-2
4.1.3	Using VMS command definitions.....	4-2
4.1	Defining how MIMER applications are run	4-2
4.2	Selecting a MIMER installation	4-3
5	MANAGING A DATABASE SERVER	
5.1	The MIMCONTROL command.....	5-1
5.1.1	Privileges needed for MIMCONTROL execution.....	5-1
5.1.2	Prerequisites for database server startup.....	5-2
5.1.3	MIMCONTROL/STATUS/DCL.....	5-3
5.2	Automatic database start and stop	5-4
5.3	The MIMTCP server	5-4
5.3.1	Manually starting a MIMTCP server.....	5-5

6	CREATING MIMER APPLICATIONS	
6.1	Developing embedded SQL applications	6-1
6.2	The ESQL command	6-2
6.2.1	ESQL syntax	6-2
6.2.2	File handling conventions	6-4
6.2.3	Points to note when compiling applications	6-4
6.3	Linking applications	6-4
6.4	Embedded SQL example programs	6-5
6.4.1	Building the simple EXAMPLE program	6-5
6.4.2	Building the DSQLSAMP example	6-5
6.4.3	Building the BLOBSAMP example	6-5
6.4.4	Building the examples performing PSM calls	6-6
A	DATA TYPES USED IN MIMER	
A.1	Internal MIMER representation	A-1
A.2	External data types supported by MIMER	A-1
B	USING MIMER7 APPLICATIONS WITH MIMER8	
B.1	Re-link applications	B-1
B.2	Remap shareable libraries	B-2
B.3	Client-server access	B-2
B.4	Local client/server access	B-3

1 INTRODUCTION

1.1 Document objectives

This publication describes the installation and usage of version 8.1 of the MIMER relational database server under VMS. It gives VMS-specific instructions about installing the software, creating databases and managing database servers.

A working knowledge of system management within the VMS environment is required for the installation of MIMER. Relevant information may be found in the *VMS System Management Guide* (published by Compaq).

Note: Most VMS manuals can be found online at the following Internet address: <http://www.openvms.digital.com:8000/>.

General familiarity with the concepts and facilities provided by the MIMER system is an advantage.

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 System Management Handbook*, the *MIMER/SQL Programmer's Manual* and the *MIMER Release Notes*.

1.2 Definition of terms

The following terms, trademarks and acronyms are used in this document:

API	Application Programming Interface.
BSQL	Batch SQL, a program used for execution of SQL statements which are read from a command file or entered interactively.
CLD	A CLD file contains definitions for new DCL commands.
Data source	ODBC term for a database.
Databank	“Databank” is the MIMER term for the physical file in which one or more MIMER 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.
DCL\$PATH	A logical name used by DCL to find executable programs. By including the MIMEXE8 directory in the definition of this logical name, all MIMER 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.
MIMER8	The general term used for version 8 of MIMER.
MIMERxxxx	Symbolic name for the distribution directory tree, unique for each MIMER release, where "xxxx" stands for the current version number, e.g. "812B".
MRS	MIMER Release and Security - manages the MIMER software licenses.
ODBC	Microsoft's 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 MIMER databank may have one or more shadows. If the databank is shadowed, there will be one file for each shadow. A shadow is a copy of the original (master) databank and is continuously updated by MIMER/DB. 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. (If database shadowing is being used, the <i>MIMER Shadowing Handbook</i> should be consulted).

SQL	Structured Query Language, standardized language for database manipulation.
SQLHOSTS	A file containing lookup information for all accessible MIMER databases, relative to the current node.
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.

(All other trademarks are the property of their respective holders.)

2 MIMER INSTALLATION

This chapter describes the installation of the MIMER software and the preparations needed to get a database system up and running.

Tasks to be performed during installation of MIMER

- Unpack the distribution file to a directory tree.
- Software setup.
- Enter the MRS license into the MIMKEY8 file.
- Establish the MIMER database(s) that reside on or are to be accessed from the VMS machine.
- Add a MIMSETUP8 command in the SYSSTARTUP_VMS.COM file so that the MIMER installation is set up each time the system boots.

2.1 VMS system requirements

To be able to use MIMER8 on the OpenVMS Alpha, version 7.1 of the operating system, or later, is needed.

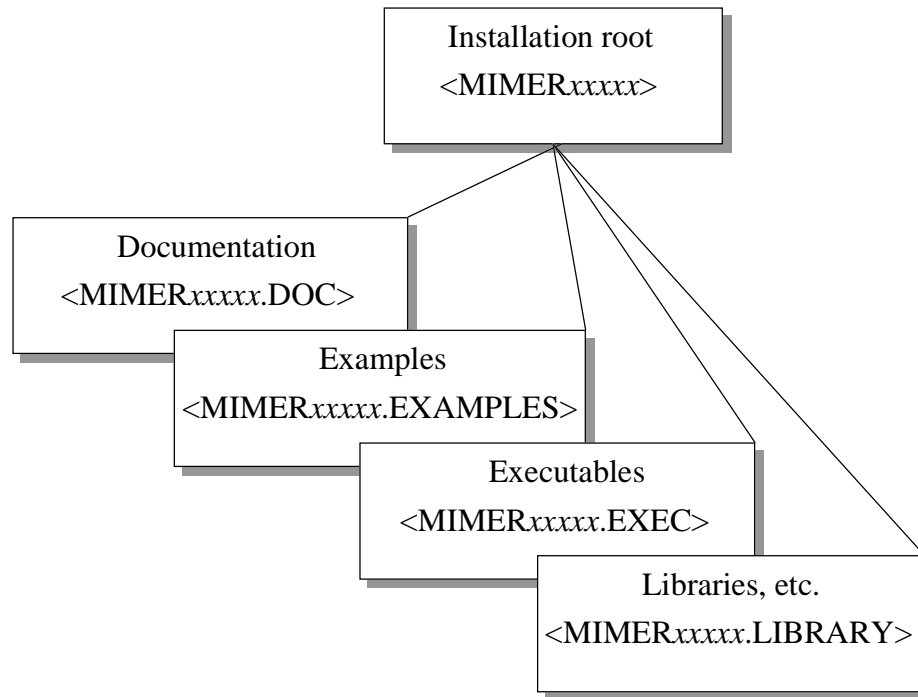
If you are using VMS 7.1, the following VMS patches must be installed:

- ALPSYSA01_071
- ALPSYSB02_071
- ALPTHREADS_03071

The installed MIMER product requires about 50000 disk blocks (25 Mb) including on-line documentation.

2.2 Unpacking the MIMER distribution file

The MIMER8 software resides in a directory tree, illustrated in the figure below. The name of the installation root directory in the tree contains the word “MIMER” and the version number of the product, e.g.: “MIMER812B” (generally denoted as MIMER_{xxxxx}).



The directory structure when unpacked and directory contents.

The name of the root directory is unique for each MIMER release, which makes it easier to install new versions without disturbing any old version of the product.

Note: The three consecutive dots in the “[...]” construction used in the VMS 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 installation will fail.

2.2.1 Unpacking a VMS saveset from a tape

When distributing the MIMER software for VMS on tape, VMS savesets are primarily used. Unpack it as follows:

```
$ MOUNT/FOREIGN tape:
$ SET DEFAULT disk: [000000]
$ BACKUP tape: /SAVE_SET [ ... ]
$ DISMOUNT tape:
```

2.2.2 Unpacking a VMS saveset from a file

The VMS saveset format can be used to create a distribution file. Such a file usually has the file type .BCK. Unpack it as follows:

```
$ SET DEFAULT disk: [000000]
$ BACKUP file.BCK /SAVE_SET [ ... ]
```

2.2.3 Unpacking a self-extracting VMS ZIP file

Self-extracting ZIP files (created using Info-ZIP) are used when distributing MIMER on the Internet.

Such files look like ordinary executable (.EXE) files. For example, the file containing MIMER version 8.1.2B would be named "MIMER812B.EXE".

To unpack the contents, simply execute the file. The MIMER distribution directory will be created under the current directory.

```
$ SET DEFAULT disk: [000000]          ! Create tree at root level
$ RUN disk: [directory]MIMERxxxxxx
```

The auto-extract facility may not always apply the correct file protections to the files it extracts, therefore it is recommended that the following commands be run 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) [MIMERxxxxxx...]*.*
$ SET FILE/PROT=(S:RWE,O:RWE,G:RE,W:RE) MIMERxxxxxx.DIR
```

2.3 The MIMSETUP8 command

Before the MIMER8 software can be used, certain setup operations must be performed. The MIMER environment, i.e. locations for programs, libraries, data files, documentation, etc., must be defined for users and applications.

The MIMSETUP8 command procedure is found in the MIMER root directory. It defines the logical names needed to run MIMER applications.

Note: The VMS user running MIMSETUP8 may require some of the following privileges: SYSPRV, CMKRNL, SYSNAM (see below for details).

2.3.1 MIMSETUP8 syntax

The syntax for the MIMSETUP8 command is as follows:

```
$ @disk:[MIMERxxxxx]MIMSETUP8 [lnm-table]
```

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

Valid values are: SYSTEM, GROUP, JOB, PROCESS.

If the user specifies SYSTEM or GROUP, the following shared images will be installed (see Section 3.2), if they are not installed already:

```
MIMLIB8:MIMDBP8.EXE
MIMLIB8:MIMDB8.EXE
MIMLIB8:MIMDBS.EXE
```

Therefore, a SYSTEM or GROUP level setup must be performed at least once for a MIMER installation in order to get these essential shared images installed.

It is generally recommended that MIMSETUP8 is executed to update the SYSTEM logical name table, so that the definitions are available to all users.

In order to define logical names SYSTEM wide, the user must have SYSPRV, CMKRNL and SYSNAM privileges. When logical names are defined in the SYSTEM table, they are defined in executive mode.

To define logical names GROUP wide, the user must have SYSPRV and CMKRNL privileges.

Since SYSTEM or GROUP wide setups have to be re-executed each time the VMS system is booted, it is recommended that the command(s) be entered into the system startup command file (SYS\$MANAGER:SYSTARTUP_VMS.COM).

MIMSETUP8 can be performed at the PROCESS or JOB level by an individual user 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 MIMSETUP8 is used without specifying the *lnm-table* parameter, a **PROCESS** level setup is performed by default.

If the parameter *lnm-table* is preceded by a hyphen (“-”), the MIMSETUP8 command procedure will remove the effects of any MIMER setup previously performed for the specified table, including de-installation of shareable images.

MIMSETUP8 examples

- 1) Definition of logical names SYSTEM wide, i.e. all VMS users may access the MIMER installation. Shareable images are installed.

```
$ @SDEPT2:[MIMER812B]MIMSETUP8 SYSTEM
```

- 2) Any user can override the default definition of the logical names. This is useful when a user wishes to test an alternate MIMER installation. No shareable images are installed.

```
$ @SDEPT2:[MIMER812B]MIMSETUP8
```

- 3) A user can remove a MIMER setup which was previously made GROUP wide by running MIMSETUP (MIMROOT8 was defined by MIMSETUP). Shareable images are de-installed.

```
$ @MIMROOT8:[000000]MIMSETUP8 -GROUP
```

2.3.2 Logical names defined by MIMSETUP8

The MIMSETUP8 command procedure defines the logical names listed below:

MIMROOT8	A concealed logical name pointing to the root of the MIMER distribution.
MIMER_SQLHOSTS	Points to the SQLHOSTS file which contains one entry for every accessible MIMER database. Normally this logical name is set to SYSS\$SPECIFIC:[SYSMGR]SQLHOSTS.DAT
MIMDB8	Logical name pointing to the MIMER/DB shareable library. Used when starting MIMER applications.
MIMDBP8	Logical name pointing to the MIMDBP8 image. Used when starting MIMER applications.
MIMDOC8	Points to the directory containing on-line documentation.
MIMEXAMPLES8	Points to the examples directory in the MIMER distribution.
MIMEXE8	Points to the directory containing executable programs in the MIMER distribution.
MIMKEY8	Points to the MRS key file. Normally set to SYSS\$SPECIFIC:[SYSMGR]MIMKEY8.DAT.
MIMLIB8	Points to the directory containing application libraries, CLD files, etc.

2.4 Installing the MRS license key

Before the steps involved in establishing MIMER database(s) can be performed, the MRS license must be installed (see Section 3.8 of the *MIMER System Management Handbook* for details about the MRS key).

To get the MRS license key data the node name must be provided to the MIMER distributor.

The key data provided by the MIMER distributor should be entered into the file MIMKEY8.DAT in SYSS\$SPECIFIC:[SYSMGR], pointed to by the logical name MIMKEY8.

Note: The VMS user entering the MRS key must have appropriate access to the MIMKEY8 file and should use an editor that handles logical names correctly.

Make sure that all VMS users authorized to run MIMER have read access to the MIMKEY8 file (file protection GROUP:R or WORLD:R, depending on the site organization).

Note: The MIMER software has a built-in default MRS license which permits limited usage of various MIMER facilities - see the *MIMER Release Notes*, under the chapter for VMS, for details about access and expiry dates.

2.5 Establishing a MIMER database

Having installed the MIMER software and the MRS key, the database(s) which are to reside on the VMS machine (local databases) can be established and any databases residing on other network nodes which are to be accessed from the VMS machine (remote databases) must be made accessible from it.

The remaining activities involved in setting up the MIMER installation are described in the *MIMER System Management Handbook* (see below for specific chapter references) and the remainder of this guide covers various additional points that relate to the VMS platform.

Refer to Chapter 2 of the *MIMER System Management Handbook* for background information which is useful for understanding the issues and the different components involved in establishing a MIMER database.

Refer to Chapter 3 of the *MIMER System Management Handbook* for details of the actual steps to be followed when establishing local and remote MIMER databases.

Before a database can be accessed, the database server for it must be started on the machine it resides on. Refer to Chapter 4 of the *MIMER System Management Handbook* for details on controlling and managing database servers.

Note: The VMS user establishing MIMER databases on a VMS node must have write access to the file pointed to by the logical name MIMER_SQLHOSTS and should use a text editor that handles logical names correctly.

2.6 Removing a MIMER installation

To remove a MIMER installation, perform the steps listed below. If you plan to remove any MIMER databases (see Section 3.11 of the *MIMER System Management Handbook*), please do this before removing the MIMER installation.

- Check that no MIMER applications or database servers are using the installation.
- Run the MIMSETUP8 command procedure to de-install shared images and deassign logical names (see Section 2.3)

```
$ disk:[MIMERxxxxx]MIMSETUP8 -SYSTEM
```

- Delete the MIMER directory tree. Note that you may have to issue the DELETE command more than once:

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

- If there are no other MIMER installation trees in the system, you may want to delete the MRS license key file (MIMKEY8) and the SQLHOSTS file:

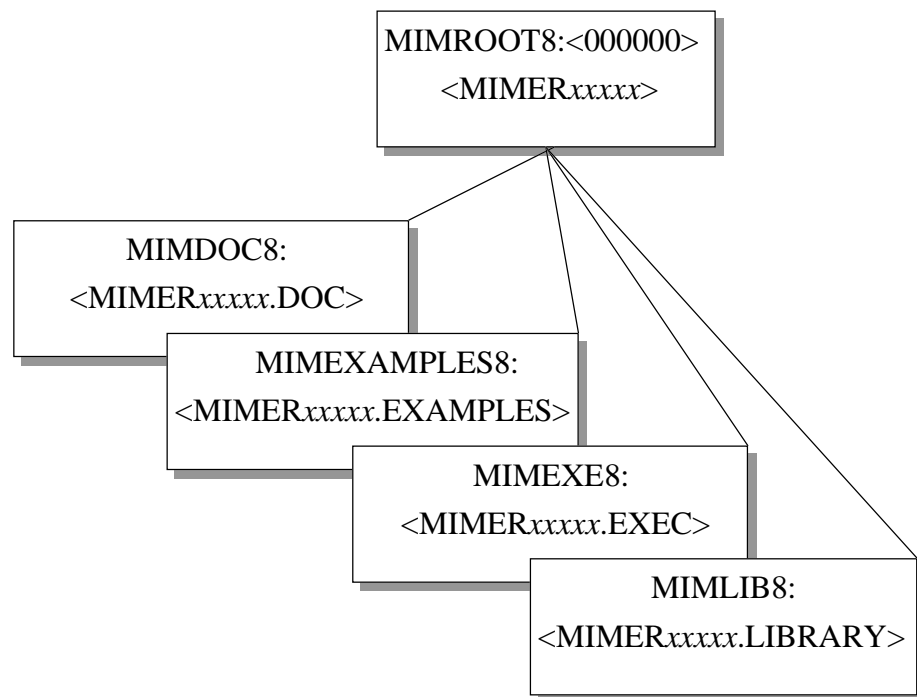
```
$ DELETE SYS$SPECIFIC:[SYSMGR]MIMKEY8.DAT.*
$ DELETE SYS$SPECIFIC:[SYSMGR]SQLHOSTS.DAT.*
```

- If you have added any MIMER-related commands to the VMS startup file (SYS\$MANAGER:SYSTARTUP_VMS.COM) or the VMS shutdown file (SYS\$MANAGER:SYSHUTDOWN.COM), you should remove those commands.

3 MIMER INSTALLATION COMPONENTS

3.1 Distributed files

The distributed files are all located in a directory structure, illustrated in the figure below. The figure also shows the logical names that are set to point to the different placeholders.



The directory structure and the related logical names.

3.1.1 Root directory files

MIMSETUP8.COM	Command procedure that defines logical names and installs images (see Section 2.3).
MIMTCP.LOG	Log file for all MIMTCP processes (see Section 5.3).
VERSION.DAT	Contains the version number of the MIMER distribution.

3.1.2 Documentation files (MIMDOC8)

RN81MIM.PDF	MIMER Release Notes.
SHADOW.PDF	MIMER Shadowing Handbook.
SQLPROG.PDF	MIMER/SQL Programmer's Manual.
SQLREF.PDF	MIMER/SQL Reference Manual.
SQLUSERS.PDF	MIMER/SQL User's Manual.
SYSMAN.PDF	MIMER System Management Handbook.
VMSGUIDE.PDF	MIMER Installation and Usage Guide for VMS. (This guide.)

3.1.3 Example files (MIMEXAMPLES8)

BLOBSAMP.EC	Example of a program written in embedded C that stores and retrieves binary data (see Section 6.4.3).
CREHOTDB.DAT	Contains SQL commands that create an example database (see Section 3.10).
DSQL.EC	Embedded C examples that demonstrates the use of dynamic SQL (see Section 6.4.2).
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 (see Section 6.4.1).
EXAMPLES.DAT	Text file containing examples of SQL commands. This assumes that the example database is loaded (see Section 3.10 of the <i>MIMER System Management Handbook</i>).
FREQCALL.EC	Example of a program written in embedded C that calls a stored procedure (see Section 6.4.4).
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 Section A.4 of the <i>MIMER System Management Handbook</i>).
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 (see Section 6.4.4).
WAKECALL.ECO	Same as WAKECALL.EC, but written in COBOL.
WAKECALL.EFO	Same as WAKECALL.EC, but written in FORTRAN.

3.1.4 Executable programs (MIMEXE8)

BSQL.EXE	Program that executes SQL statements which are entered interactively or read from a command file (described in the <i>MIMER/SQL User's Manual</i>).
DBC.EXE	Program that can check if a databank file is internally consistent (described in the <i>MIMER System Management Handbook</i>).
DBOPEN.EXE	Program that opens and restarts all databanks in a database (described in the <i>MIMER System Management Handbook</i>).
DBSERVER.EXE	The database server. Do not start this program directly, use MIMCONTROL to do this.
ESQL.EXE	Pre-processor for embedded SQL (see Section 6.2).
MIMCONTROL.EXE	The MIMCONTROL command, which is used to control database servers (see Section 5.1).
MIMINFO.EXE	Program that can display status information for a database server.
MIMTCP.EXE	The program executed by the MIMTCP server (see Section 5.3). Do not start this program directly.
SDBGEN.EXE	Program used to create the initial MIMER system databank files in a database (described in the <i>MIMER System Management Handbook</i>).
UP7TO8.EXE	Program that converts a MIMER version 7 database to MIMER8.

UTIL.EXE Utility functions for a database (see the *MIMER System Management Handbook*).

3.1.5 **Library files (MIMLIB8)**

LR.OLB Library with entries for backward compatibility.

LRU.OLB Library with entries for backward compatibility.

MDR.OLB Library with entries for backward compatibility.

MIMDB8.EXE Shareable library image containing the code for the MIMER database client API (see Section 3.2).

MIMDBP8.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 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 software.

MIMER.OPT Options file used for linking MIMER applications (see Section 6.3).

MIMODBC8.EXE ODBC driver library.

3.2 Shared images

The MIMER distribution contains a number of shareable images which are located in the MIMLIB8 directory (see Section 3.1.5).

The shared images are installed by the MIMSETUP8 command procedure when defining logical names in the SYSTEM or GROUP name table.

MIMER applications are linked with the shareable library MIMLIB8:MIMDB8.EXE. When the application image is activated, the VMS system locates the correct shareable image by using the logical name MIMDB8 (which MIMSETUP8 has defined as MIMLIB8:MIMDB8). This allows users to run applications with other versions of MIMER by using the MIMSETUP8 procedure, without having to re-link the applications.

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

Note: All logical names that MIMSETUP8 defines in the SYSTEM logical name table are defined in executive mode. This means that privileged or protected images will run the MIMER version defined in the SYSTEM table even if there is another MIMER version defined in one of the other tables!

4 RUNNING MIMER APPLICATIONS

This chapter describes how to run applications in the MIMER8 environment.

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

This chapter describes

- Defining whether VMS-style or Unix-style command-line flags are accepted by the applications which are supplied as part of the MIMER installation.
- Selecting a MIMER installation - if several MIMER installations reside on the same computer it is essential that users access the correct one.

4.1 Executing MIMER applications

This section describes the various ways an application can be executed under VMS and also describes how to set up the MIMER-supplied programs to use Unix-style or VMS-style command-line flags.

4.1.1 Using the DCL command RUN

The DCL command RUN can be used as follows:

```
$ RUN disk:<directory.app>my_appl.exe
```

It is not possible to supply any flags or other input parameters on the command-line when the RUN command is used.

Some of the MIMER-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 program in the *MIMER System Management Handbook* for specific details).

4.1.2 Using the DCL\$PATH logical name

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

In order for the programs supplied by MIMER to be run by specifying the program name followed by the **Unix-style** command line flags and parameters, the MIMEXE8 directory must be included 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 MIMEXE8 , disk:<directory.app>
```

will allow the programs supplied by MIMER to be run by specifying the program name followed by the Unix-style command line flags and parameters. It will also allow all programs found in the *app* subdirectory of *directory* on *disk:* to be run by specifying the program name.

Example using Unix-style command-line flags:

```
$ bsq1 -s mydb
```

4.1.3 Using VMS command definitions

It is possible to set up the programs supplied by MIMER so that they may be run by specifying the program name followed by the **VMS-style** command line flags and parameters.

This is done by defining the MIMER programs as DCL command verbs.

Issuing the following VMS command will define all the MIMER-supplied programs as DCL command verbs:

```
$ SET COMMAND MIMLIB8:MIMER
```

Example using VMS-style command-line flags:

```
$ BSQL/SINGLE MYDB
```

Note: If a MIMER-supplied program is set up as a DCL command verb, the Unix-style command-line flags and parameters **cannot** be used, even if MIMEXE8 is included in DCL\$PATH.

It is possible to undefine a DCL command by issuing the following command:

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

Care should be taken when using this DCL command, because any DCL command verb can be undefined.

4.2 Selecting a MIMER installation

A host computer may have several versions of the MIMER database system installed simultaneously. Access through a specific version is done through logical names (MIMEXE8, etc.). Since the major version number (currently 8) is included in the logical names, MIMER version 7 and version 8 can run concurrently without any interference.

If several versions of MIMER8 are installed, the MIMSETUP8 command procedure can be used to specify exactly which version a program should work with. Normally, a system wide definition of the logical names is made. However, a user may specify another MIMER version by running the MIMSETUP8 command procedure and specifying a GROUP, JOB or PROCESS logical name definition (see Section 2.3 for details on MIMSETUP8).

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 version specified in the GROUP or SYSTEM logical name tables.

5 MANAGING A DATABASE SERVER

This chapter contains information relevant to the administration of a database server under VMS.

For general information on managing database servers, refer to Chapter 4 of the *MIMER System Management Handbook*.

This chapter describes

- Using the MIMCONTROL command under VMS.
- The addition of commands to the SYSTARTUP_VMS and SYSHUTDWN files to automatically start or stop database servers when the system comes up or goes down.

5.1 The MIMCONTROL command

Database servers on VMS are controlled by using the MIMCONTROL command (refer to Section 4.2.1 of the *MIMER System Management Handbook* for details).

Note: The MIMCONTROL program cannot be started by using the DCL command RUN.

5.1.1 Privileges needed for MIMCONTROL execution

A user running the MIMCONTROL command must have either:

SETPRV privilege

or

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

5.1.2 Prerequisites for database server startup

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

- The system databank (SYSDB) must have been created.
- There must be an entry for the database in the local section of the SQLHOSTS file.
- 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 MIMSETUP8 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 MIMLIB8:MIMDBP8.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.
- An MRS key must have been obtained to authorize startup of the database server.

5.1.3 MIMCONTROL/STATUS/DCL

The MIMCONTROL/STATUS/DCL command is a special form of the MIMCONTROL/STATUS command described in Section 4.2.1 of the *MIMER System Management Handbook*.

When the /DCL option is used, the MIMCONTROL command is silent and the status information for the database server is returned in DCL symbols. This is useful for command procedures.

The MIMCONTROL command sets the DCL symbol MIMER_STATUS to a comma-separated string. The lexical function F\$ELEMENT() can be used to get individual values from the MIMER_STATUS string. The returned string has the following components:

server-state,logins,db-directory,connections,server-pid,start-date

The *server-state* value shows the state of the database server. The value is one of the following strings:

STOPPED	The database server is stopped
STARTING	The database server is becoming operational
RUNNING	The database server is operational
STOPPING	The a stop command as been issued to the database server
CRASHING	The database server has crashed

The *logins* value is either ENABLED if new logins are permitted or DISABLED if the server has been ordered to reject new logins.

The *db-directory* field shows the home directory for the database.

The *connections* field shows the number of clients connected to the server.

The *server-pid* field gives the VMS PID of the database server process.

The *start-date* field gives the date and time when the database server was started.

If the database server is not operational, the MIMCONTROL command may return only the first three fields in the MIMER_STATUS symbol.

5.2 Automatic database start and stop

If you want the database server to start automatically whenever the system is booted, you should edit the SYSS\$MANAGER:SYSTARTUP_VMS.COM file and add some commands at the end. The following example sets up a MIMER installation (MIMSETUP8) and then starts two database servers:

```
$ @SDEPT2:[MIMER812B]MIMSETUP8 SYSTEM
$ SET COMMAND MIMLIB8:MIMER
$ MIMCONTROL/START PRODUCTION
$ MIMCONTROL/START INVENTORY
```

If you want to perform a controlled shutdown of the database server whenever the VMS system is shut down, you should edit the SYSS\$MANAGER:SYSHUTDOWN.COM file and add some commands at the end. The following example stops two database servers:

```
$ SET COMMAND MIMLIB8:MIMER
$ MIMCONTROL/STOP PRODUCTION
$ MIMCONTROL/STOP INVENTORY
```

5.3 The MIMTCP server

In MIMER8, a MIMTCP server listening to a specific port (usually port 1360) will be started whenever a database server is started. The TCP/IP port number this 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.

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.

There is no explicit command for stopping MIMTCP servers as there is generally no need to do this. The MIMTCP process does not have to be stopped or restarted when database servers are stopped or (re)started.

To manually stop the MIMTCP server process listening to port 1360 (for example), execute the following commands:

```
$ SHOW LOG MIMTCP_1360    ! Find PID of process
$ STOP/ID=xxxxxxxxx      ! Delete the process found
```

5.3.1 Manually starting a MIMTCP server

It is normally not necessary to start a MIMTCP server manually because the server process is started automatically when a database server is started.

It is possible to start several MIMTCP servers listening on several ports. This will make version 8 servers accessible from all ports.

A user starting a MIMTCP server must have either:

SETPRV privilege

or

SYSPRV, TMPMBX, OPER, NETMBX and SYSNAM privileges.

To manually start a MIMTCP server process listening to port 1377 (for example), execute the following command:

```
$ RUN/PRIV=(SYSPRV,TMPMBX,OPER,NETMBX,SYSNAM)/DETACH/INPUT=1377 -  
$_ /OUTPUT=MIMROOT8:[000000]MIMTCP.LOG MIMEXE8:MIMTCP
```

The /INPUT parameter is used to specify the port number on which the server process listens.

The /OUTPUT parameter is used to specify the server process log file. Note that several server processes can share the same log file.

6 CREATING MIMER APPLICATIONS

The two usual ways of creating applications that access a MIMER database server are to use SQL embedded into a host programming language or to use the ODBC database API with the C host language.

The ODBC product is not available on VMS, therefore applications using this can only access a MIMER database server via the client/server interface from another type of platform.

To create embedded SQL applications the ESQL preprocessor is used, as described in this chapter.

6.1 Developing embedded SQL applications

The host languages supported by MIMER/ESQL are COBOL, FORTRAN and C.

The output generated by the ESQL preprocessor is intended to be compiled by one of the compilers listed below. Other compilers, from other software distributors, may or may not be able to compile the ESQL output. MIMER cannot guarantee the result for any compilers other than those listed below.

The following compilers (all sold by Compaq) are supported on the VMS platform:

- DEC C
- DEC FORTRAN
- DEC COBOL

Note: When using COBOL, the source program must be formatted according to the ANSI rules. Use the /ANSI option when compiling the resulting COBOL program.

6.2 The ESQL command

On VMS, the ESQL command can be invoked in two different ways:

- 1) One way is to define the ESQL command by issuing the following VMS command, using the MIMER.CLD file:

```
$ SET COMMAND MIMLIB8:MIMER
```

This will establish ESQL as a DCL command verb. All command options must be given in the usual DCL syntax, where each command option is preceded by a slash character (“/”).

- 2) The other way to invoke the ESQL command is by having the logical name DCL\$PATH include the MIMEXE8 directory:

```
$ DEFINE DCL$PATH MIMEXE8
```

When the ESQL command is invoked this way, the UNIX style options are assumed (see below).

Note: The ESQL program cannot be started by using the DCL command RUN.

6.2.1 ESQL syntax

The syntax for the ESQL command is as follows:

```
$ esql language [options] input-file [output-file]
```

Parameter	Parameter description
<i>language</i>	Use one of the language options, i.e. C, COBOL or FORTRAN, listed in the table following this one.
<i>options</i>	Optional. LINE and/or SILENT.
<i>input-file</i>	File containing input to be processed by ESQL.
<i>output-file</i>	Optional. Output file name.

Parameter options	VMS-style	UNIX-style
<p><i>language: C</i></p> <p>Use this option if the input file is written using the C host language.</p>	/C	-c
<p><i>language: COBOL</i></p> <p>Use this option if the input file is written using the COBOL host language.</p>	/COBOL	-cob(ol)
<p><i>language: FORTRAN</i></p> <p>Use this option if the input file is written using the FORTRAN host language.</p>	/FORTRAN	-for(tran)
<p><i>option: SILENT</i></p> <p>With this option ESQL acts silently, i.e. it suppresses the reporting of syntax errors on SYS\$OUTPUT.</p>	/SILENT	-s(ilent)
<p><i>option: LINE</i></p> <p>Generate #line directives (only relevant for the C language). The #line preprocessing directive forces the C compiler to produce diagnostic messages with respect to the source code that is <u>input</u> to the preprocessor, rather than the C source that is generated (and subsequently compiled by the C compiler).</p>	/LINE	-l(ine)
<p><i>input-file</i></p> <p>The file to preprocess. If given without file extension, the extension described in Section 6.2.2 is assumed.</p>	<i>file</i>	<i>file</i>
<p><i>output-file</i></p> <p>The result file. If not specified, the output file will be generated with the same name as the input file, but with the default file extension (see Section 6.2.2).</p>	/OUTPUT= <i>file</i>	<i>file</i>

6.2.2 File handling conventions

The default file extensions for input and output files depend on the host language used:

Language	Input	Output
C	.EC	.C
COBOL	.ECO	.COB
FORTRAN	.EFO	.FOR

6.2.3 Points to note when compiling applications

The floating point data types supported on Alpha/VMS are F_FLOAT (4 bytes) and G_FLOAT (8 bytes). Please use the /G_FLOAT option when compiling programs that are to be linked with MIMER libraries.

MIMER does not currently support D_FLOAT or the use of the IEEE floats, S_FLOAT and T_FLOAT. (H_FLOAT is also not supported by MIMER on Alpha/VMS since that data type is not supported natively by the Alpha architecture).

64-bit integers are not supported by MIMER on Alpha/VMS.

6.3 Linking applications

All MIMER applications should be linked with the options file MIMER.OPT, as shown in the following example:

```
$ LINK main,MIMLIB8:MIMER/OPT
```

The MIMER.OPT file includes the following:

- MIMLIB8:LR.OLB (object library)
- MIMLIB8:MDR.OLB (object library)
- MIMLIB8:MIMDB8.EXE (shareable library)

When an image linked in this fashion is activated, it will translate the logical name MIMDB8 to get the name of the actual shareable library to be used. The logical name is defined by the MIMSETUP8 command procedure. This means that when a new version of MIMER is installed, the application will automatically use it without having to be re-linked.

6.4 Embedded SQL example programs

6.4.1 Building the simple EXAMPLE program

The source code of a simple example program is included in the distribution. There is one version of the program for each supported host language. The programs are written according to international SQL standards and it should be possible to use them on any standard-compliant database management system without modification.

The program logs in to the default database with the user ident SYSADM and password SYSADM (change the program as appropriate). The program will then select and print all the rows of the X/Open standard system catalog view INFORMATION_SCHEMA.TABLES, i.e. lists all tables in the system.

The following example illustrates how EXAMPLE.EFO is compiled, linked and run. The programs written in the other languages can be linked in a similar manner.

```
$ ESQL/FORTRAN MIMEXAMPLES8:EXAMPLE      ! Preprocess source code
$ FORTRAN EXAMPLE                        ! Compile preprocessed code
$ LINK EXAMPLE,MIMLIB8:MIMER/OPT         ! Link an executable program
$ RUN EXAMPLE                            ! Run it
```

6.4.2 Building the DSQSAMP example

The DSQL.EC file demonstrates how a C program can be constructed, using dynamic SQL syntax according to international SQL standards.

DSQL.EC contains a collection of routines that define a simple but convenient API for dynamic SQL, which can be called from other C programs.

The DSQSAMP.C file contains a program that calls the routines in DSQL.EC. The DSQSAMP program allows the user to enter a SQL statement that will be executed directly, similar to the BSQL program (see the *MIMER/SQL User's Manual*).

To build the DSQSAMP example program, the following commands can be used:

```
$ ESQL/C MIMEXAMPLES8:DSQL                ! Preprocess source code
$ COPY MIMEXAMPLES8:DSQL.H SYS$DISK:[ ]   ! Get header file
$ COPY MIMEXAMPLES8:DSQSAMP.C SYS$DISK:[ ] ! Get main program
$ CC DSQL                                  ! Compile preprocessed code
$ CC DSQSAMP                               ! Compile sample program
$ LINK DSQSAMP,DSQL,MIMLIB8:MIMER/OPT     ! Link executable program
$ RUN DSQSAMP                             ! Run it
```

6.4.3 Building the BLOBSAMP example

The BLOBSAMP.EC file contains a program that demonstrates the use of the VARCHAR data type to store BLOB's. The program copies the contents of any file into the database as a BLOB, it then performs a SELECT to retrieve it again and compares the result with the original file. The file name should be specified to the program as an input parameter and the program assumes that the user ident SYSADM (password SYSADM) has access to a databank where a table can be created.

To build the BLOBSAMP example program, the following commands can be used:

```
$ ESQL/C MIMEXAMPLES8:BLOBSAMP          ! Preprocess source code
$ CC BLOBSAMP                            ! Compile sample program
$ LINK BLOBSAMP,MIMLIB8:MIMER/OPT        ! Link executable program
$ BLOBSAMP:==$disk:[current.dir]BLOBSAMP ! Define BLOBSAMP
$ BLOBSAMP user password file            ! Run it
```

6.4.4 Building the examples performing PSM calls

The WAKECALL.EC and FREQCALL.EC files contain program code that demonstrates how to embed calls to stored procedures. The procedures used are included in the example database (described in Section 3.10), which must be installed in order to execute the programs successfully.

To build the WAKECALL and FREQCALL example programs, the following commands can be used:

```
$ ESQL/C MIMEXAMPLES8:FREQCALL          ! Preprocess source code
$ CC FREQCALL                           ! Compile sample program
$ LINK FREQCALL,MIMLIB8:MIMER/OPT        ! Link executable program
$ RUN FREQCALL                           ! Run it

$ ESQL/C MIMEXAMPLES8:WAKECALL          ! Preprocess source code
$ CC WAKECALL                            ! Compile sample program
$ LINK WAKECALL,MIMLIB8:MIMER/OPT        ! Link executable program
$ RUN WAKECALL                           ! Run it
```

Versions of the WAKECALL and FREQCALL examples are also supplied in .EFO files for FORTRAN and in .ECO file for COBOL.

A DATA TYPES USED IN MIMER

A.1 Internal MIMER representation

MIMER uses only two data representations internally: numeric and character. These two types are used to store all data in the database.

The numeric data type stores numbers in packed BCD format with a maximum precision of 45 digits. Every number stored has an exponent with a range of -999 to +999.

The character data type uses the ISO 8859-1 standard character set, also known as Latin 1. This standard specifies graphic characters and the representation of each character. It is a proper extension of the DEC multinational character set, with a few exceptions.

On most platforms, including VMS, the ISO 8859-1 character set is used by MIMER to transfer data to and from the application, i.e. the same character set as is used internally.

Apart from VT100 and VT200 that only supports the DEC multinational character set, the VT300 series terminals support both DEC multinational and ISO 8859-1. Since the two character sets are quite similar it is recommended that any VMS site using VT300 terminals uses the ISO character set.

The ISO 8859-1 character set is presented in Appendix B of the *MIMER/SQL Reference Manual*.

A.2 External data types supported by MIMER

Any value stored in the database may be read into host language variables as described in Appendix A of the *MIMER/SQL Programmer's Manual*. MIMER will perform all the necessary conversions and will signal an error if the value to be converted is not compatible with the destination type.

B USING MIMER7 APPLICATIONS WITH MIMER8

General database upgrade information is provided in the *MIMER Release Notes*. This appendix describes the specific task of using a MIMER7 application with a MIMER8 database server.

If a MIMER7 application is to be used with a MIMER8 database server, there are four main approaches to getting communication established:

- Re-link the application with the MIMDB8 library.
- Use the sharable images distributed with MIMER8.
- Connect to the database using the client/server interface, i.e. from a MIMER7 client node to a MIMER8 server node.
- Connect to the database using the client/server interface locally, i.e. the MIMER7 client and MIMER8 server are on the same node.

Please note that the MIMER7 application needs a MIMER7 license (located in MIMKEY7). The MIMER8 database server needs a MIMER8 license located in MIMKEY8. A single machine will need both licenses if it runs MIMER7 applications and a MIMER8 database server.

B.1 Re-link applications

MIMER7 applications can be re-linked for use in a MIMER8 environment, accessing a MIMER8 database server. This method is recommended for applications running on the same machine as the database server.

In this case the MIMER.OPT file distributed with MIMER8 should be used, instead of the one distributed with MIMER7. Other MIMER7 libraries that may be used by the application, such as SH (Screen Handler), FM (Forms Manager), etc., should be linked in as before.

B.2 Remap shareable libraries

A MIMER7 application can access a MIMER8 database server by mapping in the MIMER8 sharable image instead of the MIMER7 one. This method is recommended for Mimer version 7 tools, such as PG or QL.

This can be achieved by defining the following logical names, after executing MIMSETUP7:

```
$ DEFINE MIMDB7M MIMDB8
$ DEFINE MIMDB7S MIMDB8
```

B.3 Client-server access

MIMER7 applications and tools can access a MIMER8 database server through the client/server interface without any modification to the application. This method is recommended when the version 8 database server is on a remote node in a network.

The entry created for a MIMER8 database server in the SQLHOSTS file is described in Appendix B of the *MIMER System Management Handbook*. In MIMER8, the SERVICE field of the SQLHOSTS file should specify the specific port (TCP/IP) or network object (DECNET) to which the database server listens. If the NODE field is set to the name of the current node and the SERVICE field specifies a local database server, the connection will be established to the local server, using the client/server interface.

The database server needs an MRS license (called "NETS") in order to be able to accept client/server connections.

B.4 Local client/server access

It is possible to use the client/server interface locally, i.e. using a remote database definition which actually points to a local database server.

To achieve this, the MIMER_SQLHOSTS file must be updated in a specific way using a 6th parameter in the remote definition, as shown in the following example (current node is "STARTREK", i.e. where the MIMER8 database "M8SERVER" runs):

```
--
-- =====
DEFAULT:
--
-- Database
-----
SINGLE
-----
LOCAL:
--
-- Database          Path
-----
SINGLE              SYS$DISK:[ ]
M8SERVER         DISK:[DIRECTORY]
-----
REMOTE:
--
-- Database          Node          Protocol Interface Service
-----
example_database    server_nodename  ''         ''         1360
M8ACCESS         STARTREK      TCP      ''         1360      M8SERVER
```

To access the "M8SERVER" database, the MIMER7 application must connect to the database "M8ACCESS".