Mimer SQL

Packaging Guide for Windows

Version 11.0

Mimer SQL, Packaging Guide for Windows, Version 11.0, April 2023

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

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

Contents

Introduction 1
- **Document Overview** 1
- **Definitions and Abbreviations** 2
- **References** 2

Creating the Installation Package 3
- **Mimer SQL Installation overview** 3
- **The Mimer SQL distribution package** 3
- **Mimer SQL Installation command line** 4
- **Content of the Mimer SQL installation** 4
- **Upgrading existing installations** 6
- **Controlling installation content** 6
- **Creating a Mimer SQL Installation with WiX** 7
- **Creating a Mimer SQL Installation with InstallShield** 7
- **Mimer SQL Data Provider installation** 8
- **Adding a Mimer SQL License Key** 8
- **Defining a Database** 9
- **Defining a Remote Database** 9
  - Example 1 9
  - Example 2 9
- **Defining and Configuring a Local Database** 10
  - Error Handling 11
  - Generating the System Databanks 11
  - SDBGEN Parameters 11
  - Starting the Database Server 12
- **More Useful Examples** 12
  - Example 1 12
  - Example 2 12
  - Example 3 13
  - Example 4 13
  - Example 5 13

Example Configuration Program 15

Main Program 15

I_SetupLicense 16
I_DefineDatabase 16
I_CreateSystemDatabanks 16
I_StartupDatabaseServer 17
I_CreateApplicationObjects 17
I_InstError 18

About Previous Mimer SQL Versions 19

Upgrading a Mimer SQL Installation 19

Upgrading a Mimer SQL Database 20

Mimer Configuration Parameters 21

Mimer SQL Specific Parameters and Commands 21

Data Source Parameters 22
Local Database Parameters 22
Local Database Server Parameters 26
Remote Database Parameters 26
License Key Parameters 27

Chapter 1

Introduction

The purpose of this document is to describe how the relational database system Mimer SQL can be packaged together with an application.

By including Mimer SQL, end-users can install both the application and Mimer SQL in a single installation.

The document describes in detail how this is performed in a Windows environment with Mimer SQL version 11.0. The following Windows operating systems are supported: Windows 8.1, Windows 10 and Windows 11.

The document is intended for Value Added Resellers (VAR's) that package and sell Mimer SQL together with their own application.

Document Overview

This document describes how to create an installation package that can install Mimer SQL together with an application.

The installation package will perform the following:

- Run a silent setup of Mimer SQL. See *Mimer SQL Installation command line* on page *4* for more information.
- Add a Mimer SQL license key, see *Adding a Mimer SQL License Key* on page *8*.
- Define a database, see *Defining a Database* on page *9*.
- Generate the system databanks, see *Generating the System Databanks* on page *11*.
- Start the database server, see *Starting the Database Server* on page *12*.
- Create the database objects needed by the application, see *I_CreateApplicationObjects* on page *17*.

The first three steps are sufficient for a Mimer SQL client-only installation.

See *About Previous Mimer SQL Versions* on page *19*, to read how the Mimer SQL installation program handles other versions of Mimer SQL on the system where the installation is taking place and how to upgrade already existing Mimer SQL systems from older versions to version 11.0.

Definitions and Abbreviations

Definition	Description
ODBC Data Source	The ODBC data source name corresponding to the Mimer SQL database that the user wants to access.
ODBC Driver	A routine library where the database specific ODBC functions are implemented. Drivers are specific to a single DBMS.
ODBC Driver Manager	A routine library that manages access to drivers for the application. The Driver Manager loads and unloads drivers and passes calls to ODBC functions to the correct driver.

Abbreviation	Short for	Description
API	Application Programming Interface	A documented set of general routines designed for some limited area of use.
DBMS	DataBase Management System	Software which is used to manage the database.
ODBC	Open Database Connectivity	Microsoft's specification for an API that defines a standard set of routines with which an application can access data in a data source.
SQL	Structured Query Language	Query and management language for relational databases.

References

- Mimer SQL Reference Manual
- Mimer SQL System Management Handbook
- ODBC Programmer's Reference and SDK Guide volumes 1 and 2, Microsoft
- WiX installer: https://wixtoolset.org/

Chapter 2

Creating the Installation Package

This chapter describes the steps necessary to bundle the Mimer SQL relational database management system with an application. This enables you, as an application developer, to install both your application and Mimer SQL with one integrated installation script.

Mimer SQL Installation overview

On Windows there are two types of installation packages - the Mimer SQL installation packages and the Mimer SQL .NET Data Provider package.

The Mimer SQL distributions, from version 11, are created with the WiX installation package (http://wixtoolset.org/). In previous versions InstallShield was used to build the installation.

From version 11 the installer has been simplified to, by default, install all the components of the Mimer SQL software.

The Mimer SQL installation contains a number of components that has been packaged into a single executable with a WiX installation package bundle. The bundle contains the following components:

- Mimer SQL Windows installer package (.msi-file)
- Java Runtime Environment (JRE)
- Microsoft Visual C/C++ runtime libraries

The installation package is for 64-bit Windows.

The Mimer SQL distribution package

The Mimer SQL version 11.0 installation is available in different packages:

1. The first one is a small executable that is downloaded. When run, it inspects the local machine to determine which components are needed and then downloads and installs only these components. For example, if a suitable version of Java or C runtime library exists, they are never downloaded.
2. The second type of package has all the Mimer SQL components and runtime libraries packaged into a single (much larger) executable. It can be used in an environment where there is no access to the Internet.

3 In the third type all the components and runtime libraries are extracted and exists as individual .msi (Windows Installer) files and executables.

The third type can be created from the first or second with the command switch /layout. This is described further in the next section.

Only 64-bit versions of Windows are supported. The 64-bit installation also installs support for 32-bit applications that are run on 64-bit Windows.

The version 11.0 installation has been developed using the WiX Toolset version 3.14. The installation is a so-called WiX Bundle.

Mimer SQL Installation command line

The following command line switches determine what type of operation is performed with the installation package:

Switch	Description
`/install`	Install Mimer SQL on the local machine.
`/repair`	Repair the current Mimer SQL installation.
`/uninstall`	Uninstall the current Mimer SQL installation.
`/layout`	Unpacks the installation into the components as described in 3) in the previous section. The components are placed next to the executable used for the /layout operation.

In addition, a number of switches complements the actual behavior of the installation program:

Switch	Description
`/passive`	Do not prompt the user for anything.
`/quiet`	Do not display any progress information while installing.
`/norestart`	Do not perform any restart of the system, even if this is needed to complete the installation.

A so-called silent installation is thus a combination of: `/install /passive /quiet`

These command line arguments can be used with any of the packages available.

Content of the Mimer SQL installation

Here is a description of the installation files of the Mimer SQL distribution:

File	Description
MimerOnlineInstaller.exe	This is the unpacked small Mimer SQL installation executable.

MimerSQLInstall_x64.msi	This is the Windows Installer package for Mimer SQL. Please note that you cannot use this package directly. You must always use MimerOnlineInstaller.exe, or the file MimerSQLInstaller_x64.exe if present, for correct operation.
mimuninst_x64.exe	64-bit helper executable to uninstall old Mimer SQL version that are based on InstallShield. I.e. version 9.2 to 10.1.
mimuninst_x86.exe	32-bit helper executable to uninstall old Mimer SQL version that are based on InstallShield. I.e. version 9.2 to 10.1.
OpenJDK11U-jre_x64_windows_hotspot_11.0.13_8.msi	64-bit Java runtime installation. This is needed for DbVisualizer, PSM debugger etc.
OpenJDK11U-jre_x86-32_windows_hotspot_11.0.13_8.msi	32-bit Java runtime installation.
vcredist_x64.exe	64-bit Visual C runtime libraries.
vcredist_x86.exe	32-bit Visual C runtime libraries.

Please note that the actual version number used above may change between different subreleases of Mimer SQL. The filenames above are from the version 11.0.7B distribution.

The MimerOnline.exe installer controls the installation and invokes the appropriate packages as needed depending on the target machine configuration.

Upgrading existing installations

Mimer SQL only allows a single version of Mimer SQL to be installed on a computer. While installing, an existing version is automatically removed and replaced with the new version. However, this is only done if the installation version is a later version than the one installed.

There are two modes of upgrades. If the existing installation is an older major version of Mimer SQL, the older version is uninstalled before applying the new version. For example, if we are installing version 11.0 and the existing installation is using Mimer SQL 10.1, then version 10.1 is uninstalled first and then version 11.0 is installed.

If the version being installed is an upgrade of the same major version a Windows installer upgrade is done where only changed files, registry entries, etc. are updated. For example, this happens if version 11.0.7B is installed on a machine with version 11.0.6A installed.

From a packaging perspective there is no difference between these two modes of operation.

If a later version of Mimer SQL is installed, installation of an earlier version is not allowed. To achieve this, the existing, later, version must be uninstalled first before applying the older version.

The package automatically handles uninstallation of previous version of Mimer SQL starting with version 9.2 on Windows when necessary. If version 9.1, or earlier, is found the installation requires a manual uninstall first as there is not enough information available to do this automatically.

Controlling installation content

The Mimer SQL installation has been simplified in version 11.0. The entire package is, by default, always installed. This makes the installation extremely easy for any user. Mimer SQL is a very compact product so saving a few bytes of disc by only installing some components has not been a priority.

Having said this, it is actually still possible to control what parts of Mimer SQL is installed on a computer. By default, the entire package is installed. But, through the command line it is possible to tell the installation program to exclude certain functionality.

Variable	Description
`ExcludeServer`	Do not include database server executables. I.e. only perform a Client install.
`ExcludeDbVisualizer`	Do not include DbVisualizer. I.e. no GUI for interactive access to view databases will be present.
`ExcludeReplication`	Exclude the Mimer SQL replication functionality.
`ExcludeDevelopment`	Exclude development and sample files.
`ExcludeDocumentation`	Exclude documentation help files.
`ExcludeJava`	Do not install Java on the machine.

A number of these may be combined on the command line to exclude several parts of the installation.

If all parts above are excluded, only the Mimer SQL client libraries and Mimer Administrator are installed.

Example command line:

```
MimerOnlineInstaller.exe /passive ExcludeServer=1 ExcludeDbVisualizer=1
```

Please note that the exact spelling of the keywords is needed for the above functionality to work. There are no checks for actual variable names passed to the installer. They are simply ignored.

If you set the variable to zero the component will be installed. For example, `ExcludeDocumentation=0` will install the documentation set. It is recommended that everything is always installed.

Creating a Mimer SQL Installation with WiX

To include Mimer SQL with a WiX based installation you need to create a so-called WiX bundle. The bundle is described by an XML file that describes the various installation components.

The installation is performed by including an ExePackage element with the Mimer SQL installation file. This can, for example, look like this:

```
<Chain>
    <ExePackage
        SourceFile="mimerinstall\MimerOnlineInstallerV1107b.exe"
        InstallCommand="/install /quiet /norestart"
        Compressed="yes"
        Vital="yes"
        Permanent="no">
      <ExitCode Value="1641" Behavior="forceReboot"/>
      <ExitCode Value="3010" Behavior="forceReboot"/>
    </ExePackage>
```

The chain element contains each executable or .msi to install. The above depicts a Mimer SQL example. The parameters let the Mimer SQL installation run silently. If a reboot is needed, it is postponed until the ongoing installation is complete.

Creating a Mimer SQL Installation with InstallShield

There are two ways to run executables from InstallShield:

- From an InstallShield script use `LaunchApplication` or `LaunchAppAndWait` InstallScript functions.
- Or place the executable in a custom action in the `InstallExecute` sequence.

Other installation products typically feature similar type of support for running executables during the installation.

Mimer SQL Data Provider installation

The Mimer SQL Data Provider is a .NET installation. It follows the typical pattern for .NET programs by allowing several versions of the software to be installed in parallel.

The installation performs the following:

1. Install the Mimer SQL Data Provider dynamic link libraries
2. If this is the Mimer SQL Data Provider with the highest version number, it also installs the dynamic link libraries in the global assembly cache and registers them in machine.config.
3. Performs integration with supported versions of Microsoft Visual Studio. This also includes integration of documentation in the Visual Documentation set. XML files are included that enable statement completion with associated help text.
4. Adds a help file and shortcut to start menu

For typical installations only 1) above is needed. It is therefore possible to install only the Mimer SQL Data Provider dynamic link libraries. This can be done together with the application executable using the Mimer SQL Data Provider.

The Mimer SQL Data Provider is also available as a NuGet package. When using the NuGet packages the dynamic link libraries needed are stored in the Visual Studio build tree. These dynamic link libraries can simply be included with the application. The provider client can thus be used with no installation other than copying the .dll files along with the customer application.

In this type of installation there is not integration with Visual Studio, and machine.config is not updated. This means that MimerProviderFactory is not available through `DbProviderFactories.GetFactory` unless explicitly set up with a call to `DbProviderFactories.RegisterFactory`.

Adding a Mimer SQL License Key

Depending on your VAR license agreement with Mimer Information Technology, you may need to add a Mimer SQL license key to your installation script.

You use the ODBC routine `SQLConfigDataSource` to add the license key.

Note: Using the ODBC routine `SQLConfigDataSource` to add a license key can be compared to interactively adding a license key using the Mimer Administrator.

The following C example adds a Mimer SQL license key:

```
rc = SQLConfigDataSource(NULL,ODBC_ADD_SYS_DSN,"MIMER",
     "InstallationNo=12345\0"
     "LicenseKey=23CB23B7C8D33E2F206A23CDEF67\0"
     "Description=This is my license key\0\0");
```

See *License Key Parameters* on page *27* for more information about the `SQLConfigDataSource` license key parameters.

If you prefer command line installation the program `confmim.exe` that is installed with the Mimer SQL installation can be used to set up and manage License keys, databases, and ODBC data sources. Use `confmim -?` on the command line to find out more. The above install command would be:

```
confmim --add --installationno=12345
--licensekey=23CB23B7C8D33E2F206A23CDEF67 --description="My license key"
```

Defining a Database

Once you have installed Mimer SQL and added a license key, the next step is to define and configure a database. The database can either be a local or a remote database. You can also associate the database with an ODBC data source name.

You use the ODBC routine `SQLConfigDataSource` to define and configure Mimer SQL databases and data sources from a program. The Mimer specific parameters are documented in *Chapter 5, Mimer Configuration Parameters*.

You use `SQLConfigDataSource` to perform the same tasks as you would normally perform using the Mimer Administrator. It is possible to override any defaults for parameters and, if you like, to invoke the same dialog boxes as the Mimer Administrator uses to enable the user of your application to perform customization for their specific environment.

Defining a Remote Database

The following sections document examples of how to define a remote database. For more examples, see *Chapter 3, Example Configuration Program*. For more information on the parameters used, see *Chapter 5, Mimer Configuration Parameters*.

If any parameters that are specific to a local or remote database definition are given, the system will create the appropriate database definition. In the examples below, the parameters `NODE` and `PROTOCOL` indicates that a remote database definition should be created.

Example 1

The following C example will create the definition for a remote database on the host `db.mimer.com`. The client will communicate using the TCP/IP protocol.

```
rc = SQLConfigDataSource(NULL,ODBC_ADD_SYS_DSN,"MIMER",
     "DSN=MyRemoteODBC\0DATABASE=MyRemoteMIMER\0"
     "NODE=db.mimer.com\0"
     "PROTOCOL=tcp\0"
     "DESCRIPTION=This is my remote database\0\0");
```

The example above creates a new, system-wide, data source `MyRemoteODBC` and also a remote Mimer SQL definition for the database `MyRemoteMIMER`.

As the first (`hwndParent`) parameter is set to NULL, the operation is performed without displaying any dialog boxes.

And the corresponding command line database setup:

```
confmim --add --database=MyMimerDb --node=db.mimer.com --protocol=tcp
--description="This is my remote database"
confmim --DSN=MyRemoteODBC --system --database=MyMimerDb
```

Example 2

Alternately, you can achieve the same result as in example 1 with the following two calls:

```
rc = SQLConfigDataSource(NULL,ODBC_ADD_SYS_DSN,"MIMER",
     "DATABASE=MyRemoteMIMER\0"
     "NODE=db.mimer.com\0"
     "PROTOCOL=tcp\0"
     "DESCRIPTION=This is my remote database\0\0");
```

and

```
rc = SQLConfigDataSource(NULL,ODBC_ADD_SYS_DSN,"MIMER",
     "DSN=MyRemoteODBC\0DATABASE=MyRemoteMIMER\0"
     "DESCRIPTION=This is my remote database\0\0");
```

First, the remote database definition is created and then the data source. This shows some important points regarding how `SQLConfigDataSource` is implemented.

When the input parameters are analyzed, the system decides which objects to work with. If a DSN parameter is specified, a data source is created.

Since the parameter is `ODBC_ADD_SYS_DSN`, this means that any existing definitions of `MyRemoteODBC` and `MyRemoteMimer` are overwritten.

If you want to change the `NODE` argument for an existing database definition the following call would do that:

```
rc = SQLConfigDataSource(NULL,ODBC_CONFIG_SYS_DSN,"MIMER",
     "DATABASE=MyRemoteMIMER\0"
     "NODE=newnodename\0\0");
```

And the corresponding command line:

```
confmim --config --database=MyRemoteMimer --node=newnode
```

Defining and Configuring a Local Database

To create a local database, you specify one or more local parameters in a call to `SQLConfigDataSource`.

The following C program will create a local database definition with 1000 4K pages in the bufferpool.

```
rc = SQLConfigDataSource(NULL,ODBC_ADD_SYS_DSN,"MIMER",
       "DATABASE=MyLocalMimer\0"
       "DIRECTORY=C:\DB6\0"
       "Pages4K=1000\0\0");
```

As the first argument (`hwndParent`) is set to NULL, the call will not display any dialog boxes.

`DATABASE` and `DIRECTORY` are required parameters that have to be specified when the first argument (`hwndParent`) is set to NULL.

The number of 4K pages will be set to 1000 and all other parameters will be set to their default values.

And the corresponding command line:

```
confmim --add --database=MyLocalMimer --directory=C:\DB6 --pages4k=1000
```

You can view the changes made by the installation program in the Registry Editor (`regedit.exe`) under the following keys:

```
HKEY_LOCAL_MACHINE:
  Software\Mimer\Mimer SQL\SQLHosts\xxx
  Software\ODBC\ODBC.INI\yyy
HKEY_CURRENT_USER:
  Software\ODBC\ODBC.INI\zzz
```

Where `xxx` is the database name, `yyy` is system data source name and `zzz` is user data source name.

Error Handling

Note that if the configuration is handled as a number of calls, then the error handling, from the installation program's point of view, can much more easily determine what went wrong.

In Example 1, it is possible that a data source was created even though the database definition was not. See *I_InstError* on page *18* for more information on error handling.

Generating the System Databanks

When you have defined the database, you must generate the system databanks.

You can generate system databanks in your own program by calling `SQLConfigDataSource` with the `SDBGEN` command. In this mode no dialog boxes, only progress bars, are displayed and any missing directories are created automatically.

Consider the following example:

```
rc = SQLConfigDataSource(hWnd,ODBC_CONFIG_SYS_DSN,"MIMER",
     "DATABASE=MyLocalMimer\0"
     "SDBGEN=-pSYSPSW MyLocalMimer 1000 \"\" 2000\0\0");
```

The above call will start a system databank generation for the database `MyLocalMimer`. The initial size for SYSDB will be 1000 and for TRANSDB it will be 2000 4K blocks.

Note that the default file name for TRANSDB is set using `""`, backslashes (`\`) are used as escape characters.

Whenever you use the `-p` option, the program is run in silent mode. In this mode no dialog boxes are displayed and any missing directories are created automatically.

If an error occurs, such as disk space exhausted, a dialog box is displayed where file names and/or sizes may be changed. However, in this case the fields for SYSADM password are disabled.

If you do not use option `-p`, the values specified will be used as the default values in the system databank generation dialog box.

When using the SDBGEN command with `SQLConfigDataSource` you should, for consistency, always specify `-p` when the `hWndParent` parameter is NULL. Otherwise, a dialog box is displayed even though you have requested `SQLConfigDataSource` not to do so.

If you want to use a command's default value, specify the default value using two double quotes (`""`). All commands except `database-name` are optional.

SDBGEN Parameters

The following example demonstrates all of `SDBGEN`'s parameters:

```
rc = SQLConfigDataSource(hWnd,ODBC_CONFIG_SYS_DSN,"MIMER",
     "DATABASE=MyLocalMimer\0"
     "SDBGEN=-pSYSPSW " /* SYSADM password      */
     "MyLocalMimer "    /* Database name        */
     "1000 "            /* Initial SYSDB size   */
     "\"\" "            /* TRANSDB file name    */
     "1000 "            /* Initial TRANSDB size */
     "\"\" "            /* LOGDB file name      */
     "1000 "            /* Initial LOGDB size   */
     "\"\" "            /* SQLDB file name      */
     "1000 ");          /* Initial SQLDB size   */
```

By using two double quotes ("") in the example above, SDBGEN will assign the default file names to the TRANSDB, LOGDB and SQLDB system databanks.

Note: We highly recommend that you let the `SDBGEN` program determine the appropriate location for the system databanks.

`SDBGEN` does this by examining the available hard drives on the system and spreading the files over the disks and taking into account recovery and performance issues.

It is, of course, possible to invoke the `sdbgen.exe` (console based program) or `sdbgenw.exe` (windows wizard for running system databank generation) directly on the command line. Command line arguments are the same as described above.

Starting the Database Server

By now you have installed Mimer SQL and created a local database with associated system databanks. The next step is to start the local database server.

To start the server for the database `MyLocalMimer`, use the following call:

```
rc = SQLConfigDataSource(NULL,ODBC_CONFIG_SYS_DSN,"MIMER",
       "DATABASE=MyLocalMimer\0"
       "DbServer=START\0\0");
```

And the corresponding `confmim` command line:

```
confmim --config --database=MyLocalMimer --dbserver=start
```

More Useful Examples

This section contains a few more useful example calls to `SQLConfigDataSource`. For more examples, see *Chapter 3, Example Configuration Program*.

Example 1

This is the simplest possible call to create: a system data source, a Mimer SQL database and the system databank, and then start the database server.

```
rc = SQLConfigDataSource(NULL,ODBC_ADD_SYS_DSN,"MIMER",
       "DSN=DB6\0"
       "DATABASE=DB6\0"
       "DIRECTORY=C:\DB6\0"
       "SDBGEN=-pSYSPSW DB6\0"
       "DBSERVER=START\0\0");
```

For the command line a number of commands are needed:

```
confmim --add --database=DB6 --directory=C:\DB6
confmim --add --DSN=DB6 --system --database=DB6
sdbgenw --password=SYSPSW --silent DB6
confmim --config --dbserver=start --database=DB6
```

Example 2

The following call passes control to the user to perform all customization of a database.

The user may not change the data source name (as governed by the specification of `SQLConfigDataSource`), so the application knows how to connect to the database through the application defined data source name.

```
rc = SQLConfigDataSource(hWnd,ODBC_ADD_SYS_DSN,"MIMER",
     "DSN=FIXEDDSN\0"
```

```
     "DATABASE=\0"
     "DIRECTORY=\0\0");
```

Please note that specifying an empty `DIRECTORY` allows `SQLConfigDataSource` to identify that a local database should be created.

Example 3

In this example, the database server is stopped and then the definition is deleted along with the data source definition.

```
rc = SQLConfigDataSource(NULL,ODBC_REMOVE_SYS_DSN,"MIMER",
     "DSN=DB6\0"
     "DATABASE=DB6\0"
     "DBSERVER=STOP\0\0");
```

For the command line two commands are needed:

```
confmim --config --dbserver=stop --database=DB6
confmim --delete --database=DB6
```

When deleting database definition all corresponding ODBC data sources are deleted.

Example 4

This example demonstrates how to remove a Mimer SQL license key.

```
rc = SQLConfigDataSource(NULL, ODBC_REMOVE_SYS_DSN,
     "MIMER",
     "InstallationNo=1234\0\0");
```

And the corresponding `confmim` command line:

```
confmim --delete --installationno=1234
```

Example 5

This example demonstrates how to remove an ODBC data source.

```
rc = SQLConfigDataSource(NULL, ODBC_REMOVE_SYS_DSN,
     "MIMER",
     "DSN=DB8\0\0");
```

And the corresponding `confmim` command line:

```
confmim --delete --DSN=DB8
```

Chapter 3

Example Configuration Program

The following example program is a console mode program that does everything a typical installation program needs to do. The program is organized into one subroutine for each task needed.

This program is run after the silent installation of Mimer SQL has completed, that is, Mimer SQL needs to be installed at this point.

In the example program the routines are prefixed by `I_`. All other calls are either to the C runtime library or ODBC.

Note: If you receive "error LNK2001: unresolved external symbol _vsnwprintf_s", add `legacy_stdio_definitions`.lib to additional dependencies in linker input. See https://msdn.microsoft.com/en-us/library/bb531344.aspx for more information.

Main Program

The main program looks as follows:

```
#include <windows.h>
#include <odbcinst.h>
#include <stdio.h>
#include <stdlib.h>
#include <sqlext.h>
void main()
{
    I_SetupLicense();
    I_DefineDatabase();
    I_CreateSystemDatabanks();
    I_StartupDatabaseServer();
    I_CreateApplicationObjects();
    exit(0);
}
```

I_SetupLicense

The first example subroutine adds a Mimer SQL license key. Note that different machines usually have different installation numbers and license keys. This routine is only needed when an application is distributed with a license key.

```
void I_SetupLicense()
{
    BOOL rc;
    rc = SQLConfigDataSource(NULL, ODBC_ADD_SYS_DSN,
         "MIMER",
         "InstallationNo=12345\0"
         "LicenseKey=23CB23B7C8D33E2F206A23CDEF67\0"
         "DESCRIPTION=This is my license key\0\0");
    if (!rc) {
        I_InstError("Error adding Mimer SQL license:");
    }
}
```

I_DefineDatabase

This routine defines the ODBC data source and the Mimer SQL database parameters:

```
void I_DefineDatabase()
{
    BOOL rc;
    rc = SQLConfigDataSource(NULL, ODBC_ADD_SYS_DSN,
         "MIMER",
         "DSN=MyODBC\0DATABASE=MyMIMER\0"
         "DIRECTORY=c:\\MyDirectory\0"
         "USERS=200\0"
         "DESCRIPTION=This is my sample database\0\0");
    if (!rc) {
        I_InstError("Error creating database definition:");
    }
}
```

I_CreateSystemDatabanks

This routine starts a process that runs the Mimer SQL system databank generation program in silent mode.

```
void I_CreateSystemDatabanks()
{
    BOOL rc;
    rc = SQLConfigDataSource(NULL, ODBC_CONFIG_SYS_DSN,
         "MIMER",
         "DATABASE=MyMimer\0"
         "SDBGEN=-pSYSPSW MyMimer\0\0");
    if (!rc) {
        I_InstError("Error creating System Databanks:");
    }
}
```

I_StartupDatabaseServer

The following code starts the database server:

```
void I_StartupDatabaseServer()
{
    BOOL rc;
    rc = SQLConfigDataSource(NULL, ODBC_CONFIG_SYS_DSN,
         "MIMER",
         "DATABASE=MyMimer\0"
         "DbServer=START\0\0");
    if (!rc) {
        I_InstError("Error starting database server:");
    }
}
```

I_CreateApplicationObjects

This routine sets up the Mimer SQL database environment needed by the application. For clarity, error handling has been left out of this routine.

```
void I_CreateApplicationObjects()
{
    RETCODE rc;
    HENV hEnv;
    HDBC hCon;
    HSTMT hStmt;
    rc = SQLAllocEnv(&hEnv);
    rc = SQLAllocConnect(hEnv, &hCon);
    rc = SQLConnect(hCon,
         "MyODBC", SQL_NTS,
         "SYSADM", SQL_NTS,
         "SYSPSW", SQL_NTS);
    rc = SQLAllocStmt(hCon, &hStmt);
/*
 *  Add application specific object creation here
 */
    rc = SQLExecDirect(hStmt,
         "CREATE IDENT …", SQL_NTS);
    .
    .
    .
/*
 *  Done, logout from database system
 */
    rc = SQLDisconnect(hCon);
    rc = SQLFreeEnv(hEnv);
}
```

To make the sample complete, you should add code for creating the Mimer SQL objects such as databanks, users, tables, and views needed to run the application.

I_InstError

An example of the error handling used by the other examples follows:

```
void I_InstError(char *pszOperation)
{
    RETCODE rc;
    WORD iError, cbErrorMsg;
    DWORD fErrorCode;
    char szErrorMsg[1000];
    printf("%s\n", pszOperation);
    for (iError = 1; iError <= 8; iError++) {
        rc = SQLInstallerError(iError,
             &fErrorCode,
             szErrorMsg,
             sizeof(szErrorMsg),
             &cbErrorMsg);
        if (rc == SQL_NO_DATA_FOUND ||
            rc == SQL_ERROR) {
            break;
        }
        printf("%d: Error code = %d, Message = %s\n",
            iError,
            fErrorCode,
            szErrorMsg);
    }
}
```

The routine `SQLInstallerError` is further described in the Microsoft ODBC documentation.

Chapter 4

About Previous Mimer SQL Versions

This chapter describes how to upgrade from earlier versions of Mimer SQL.

For information about packaging in earlier versions of Mimer SQL, please refer to the corresponding version of *Mimer Packaging Guide*.

On Windows, Mimer does not support parallel installations of different versions of Mimer SQL. In the following sections, the term components refers to items such as the database server, database client, and online documentation.

Upgrading a Mimer SQL Installation

The logic used by the installation program is as follows:

- If the new installation is a change of major version, for example from version 10.0 to 11.0, the install script asks the user whether they want to uninstall the old version and install the new. If a silent install is carried out, the installation proceeds without asking.
- Only the components selected by the silent setup are installed. Typically, this is a full installation.
- If the new installation is a change of minor version, for example from version 11.0.5 to 11.0.7, the system will upgrade the existing installation with any changed components.

  This method of upgrading means that the installation program does not know where the installation directory is.
- If the version to be installed is older than the version already installed (such as 11.0.7 is installed and trying to install 11.0.6), the installation fails. To do this, the user must first uninstall the already installed version and then redo the installation of the older version.

Upgrading a Mimer SQL Database

When upgrading between major versions, for example from 10.1 to 11.0, existing databases must be upgraded using a database upgrade utility.

The interactive method of upgrading Mimer SQL databases is by using the Mimer Administrator. Upgrade is one of the menu alternatives when right-clicking a local database.

You can also upgrade a database from a program using the ODBC routine `SQLConfigDataSource`.

The following C example will upgrade the `MyLocalMimer` database where the system administrator password is `SYSPSW`:

```
rc = SQLConfigDataSource(NULL,ODBC_CONFIG_SYS_DSN,"MIMER",
     "DATABASE=MyLocalMimer\0"
     "UPGRADE=-pSYSPSW MyLocalMimer\0\0");
```

Always remember to carry out a proper backup before upgrading a database.

Chapter 5

Mimer Configuration Parameters

This chapter explains the Mimer SQL specific parameters and commands, used in `SQLConfigDataSource`.

Mimer SQL Specific Parameters and Commands

To set up and configure a Mimer SQL database, the parameters and commands below may be specified in a call to `SQLConfigDataSource`. If a parameter is not specified, the system uses an appropriate default.

You can find additional information about these parameters and commands through the help facility in the Mimer Administrator. Many of the parameters are also described in the *Mimer SQL System Management Handbook*.

The parameters and commands are divided into the following groups:

- Data source parameters, see *Data Source Parameters* on page *22*.
- Local database parameters, see *Local Database Parameters* on page *22*.
- Local database server commands, see *Local Database Server Parameters* on page *26*.
- Remote database parameters, see *Remote Database Parameters* on page *26*.
- License key parameters, see *License Key Parameters* on page *27*.

To create both a data source and a remote definition, you specify parameters from the two groups together. It is not possible to combine remote parameters with local parameters or commands.

Data Source Parameters

Data Source Parameter	Explanation
DSN	The ODBC data source name **(required)**. It is recommended to use the same name as for the database
Description	A text describing the data source.
Database	The name of the Mimer SQL database **(required)**.

Local Database Parameters

Local Database Parameter	Explanation
Database	The name of the Mimer SQL database **(required)**.
Description	A text describing the database.
Directory	The database home directory **(required)**. The database server will use the list of directories when performing lookup of a databank file with no device and directory specification in its name.
Users	Maximum number of simultaneous connections allowed to the database server **(recommended)**.
DBCheck	The type of check performed after an improper shutdown of the system. All Pages (1) or Immediate restart with complete check (4) is recommended. Data pages are checked in the background. 0 = Index Only 1 = All Pages 2 = Immediate start with no check 3 = Immediate start with index check 4 = Immediate start with complete check
PriorityClass	The priority class of the server. Should be an integer in the range 0-3. 0 = Idle 1 = Normal 2 = High 3 = Real-time Further information about priorities may be found in the Win32 documentation for the routine `SetPriorityClass`.
RequestThreads	An integer specifying the number of request threads in the database server. Each request thread can handle one concurrent application request. If there are many long requests this parameter may have to be increased form the default.

Local Database Parameter	Explanation
RequestPriority	The priority class of the kernel threads. Should be an integer in the range 0-6. 0 = Idle 1 = Lowest 2 = BelowNormal 3 = Normal 4 = AboveNormal 5 = Highest 6 = TimeCritical Further information about priorities may be found in the Win32 documentation for the routine `SetThreadPriority`.
BackgroundThreads	An integer specifying the number of background threads. If there are many large transactions and/or shadowing is used, the number of threads may need to be increased. Actually, it is more important to give the background threads sufficient priority rather than increase the number of threads. The number of background threads determine how many databases are checked in parallel. Half of the threads are used for this.
BackgroundPriority	The priority class of the background threads. Should be an integer in the range 0-6. See `RequestPriority` for the specific values. It is recommended that this parameter is equal to or one higher than the value for `RequestPriority`.
StartupType	An integer value which is specified if the server is to be disabled, started manually or started automatically after a system reboot. 0 = Autostart 1 = Manual start 2 = Disabled
AutoRestart	An integer value which is specified if the server is to be started manually or started automatically after a failure. 0 = Manual restart 1 = Automatic restart
Pages4K	An integer specifying the number of 4K pages in the bufferpool.
Pages32K	An integer specifying the number of 32K pages in the bufferpool.

Local Database Parameter	Explanation
Pages128K	An integer specifying the number of 128K pages in the bufferpool.
SQLPool	The initial size of the SQL-pool in bytes. The SQL-pool contains information about users logged in, compiled SQL statements and so on.
MaxSQLPool	If you want to limit the amount of memory the database server allocates, this parameter specifies the maximum number of bytes that the server allocates for the SQL-pool. The maximum size of the SQL-pool is 16 GB.
ActTrans	Maximum number of transactions that can be active in the database server including background threads processing.
Databanks	Maximum number of allowed databanks.
Tables	Maximum number of open tables allowed.
CommBuffSize	This is the size of local communication buffers (specified in bytes). If communication packages exceed this size, several calls are made to the database server. This may increase overhead. The default size is 64K. Any buffer size specified is rounded up to the nearest 64K boundary.
TcpPort	This can be either a port number, such as 1360 or one of the strings `<Name Server>` or `<Disabled>`. If `<Name Server>` is used, incoming TCP connections are handled by a separate TCP server process and then handed over to the correct database server.
NamedPipe	This is the name of the named pipe the database server uses to listen for incoming requests. The default is a named pipe with the same name as the database server. As for TcpPort the strings `<Name Server>` or `<Disabled>` may also be used.
RmGuid	This is a unique identifier identifying the database server used for inter-operating with Microsoft Distributed Transaction Coordinator. Do not specify RmGuid unless you are renaming a database server, in which case the original RmGuid should be kept.

Local Database Parameter	Explanation
DumpPath	This is a path to the directory under which database server dump directories and files will be placed. Typically these are placed below the database home directory.
DealyedCommit	Delayed transaction commit options. 0 = Default off 1 = Default on 2 = Disabled
DelayedCommitTimeout	If delayed commit is active, this parameter determines the number of milliseconds the system waits until a transaction is automatically secured on disc. If zero is specified there is no time limit.
GroupCommitTimeout	How many milliseconds to wait for other transactions to commit before proceeding with first transaction. If another transaction arrives within the timeout period if will be grouped with existing transactions before they are committed together with a single I/O rather. This improves overall performance but the delay prolongs commit times on a system with low load. Default is one millisecond.
MiniDump	Determines what is written to a bufferpool dump file when a database server fails or when the command `mimcontrol -A` is used. 0 = Full dump 1 = Minidump
MemLock	Whether bufferpool is locked in memory or not. If lock pages is used there has to be consecutive memory for the allocation. This may only be possible after a reboot as the memory becomes more fragmented after a while. 0 = Do not lock pages in memory 1 = Lock pages
NetworkEncryption	Controls if remote connections to clients require encryption or not. 0 = Never encrypted 1 = Encrypted if supported by client, e.g. if client is 11.0 or later 2 = Require encryption (will not allow clients from 10.1 or earlier to connect.)

Local Database Server Parameters

Local Database Server Command	Explanation
Database	The name of the Mimer SQL database **(required)**.
SDBGEN	The parameters to the command correspond to the command line arguments specified when using the system databank generation utility (SDBGEN). This is further described in *Generating the System Databanks* on page *11*.
Upgrade	The parameters to the command is the SYSADM password and the name of the Mimer SQL database. This is further described in *Upgrading a Mimer SQL Installation* on page *19*.
DbServer	The DbServer command accepts the following strings: START, STOP, ENABLE and DISABLE.

Remote Database Parameters

Remote Database Parameter	Explanation
Database	The name of the Mimer SQL database on the remote host **(required)**.
Description	A text describing the database.
Node	The name of the computer where the database is running. If this keyword is present, the definition for a remote Mimer SQL database is created **(required)**.
Protocol	Specifies whether to use named pipes or TCP/IP. Should either be the string `NamedPipes` or `tcp`. `tcp` is the default.
Service	The IP port number of the server for TCP/IP. The default is `1360`. For named pipes, this is the name of the pipe that the database server is waiting for incoming connections on. The default is to listen on a pipe with the same name as the database.
Interface	Not used on the Windows platform.

License Key Parameters

License Key Parameter	Explanation
LicenseKey	A string specifying a Mimer SQL license key.
InstallationNo	An integer specifying the installation number associated with a Mimer SQL license key. The installation number is specified when adding or removing a license key.
Description	A text describing the license key.