



MIMER

Packaging Guide for Windows

Version 8.1.2

Copyright © 1999 Sysdeco Mimer AB

MIMER Packaging Guide for Windows

November, 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.

| | | |
|----------|--|-----|
| 1 | Introduction | |
| 1.1 | Purpose..... | 1-1 |
| 1.2 | Definitions, Acronyms and Abbreviations | 1-1 |
| 1.3 | References..... | 1-2 |
| 1.4 | Document Overview..... | 1-2 |
| 2 | Creating the installation package | |
| 2.1 | Silent MIMER installation | 2-1 |
| 2.2 | Defining MIMER databases..... | 2-2 |
| 2.2.1 | Introduction | 2-2 |
| 2.2.2 | MIMER specific parameters..... | 2-2 |
| 2.2.3 | Defining a remote database | 2-6 |
| 2.2.4 | Defining a local database..... | 2-7 |
| 2.3 | Creating the System Databanks..... | 2-8 |
| 2.4 | Starting the database server..... | 2-9 |
| 2.5 | More useful examples | 2-9 |
| 3 | Example MIMER configuration program | |
| 3.1 | Main program..... | 3-1 |
| 3.2 | I_DefineDatabase..... | 3-1 |
| 3.3 | I_CreateSystemDatabanks..... | 3-2 |
| 3.4 | I_StartupDatabaseServer..... | 3-2 |
| 3.5 | I_CreateApplicationObjects..... | 3-2 |
| 3.6 | I_InstError..... | 3-3 |
| 4 | Information about older MIMER versions | |
| 4.1 | Upgrading from earlier versions..... | 4-1 |
| 4.2 | Version 7.3.3..... | 4-2 |
| 4.3 | Version 8.1.1..... | 4-3 |
| 5 | ODBC supplementary information | |
| 5.1 | SQLConfigDataSource..... | 5-1 |
| 5.2 | SQLInstallerError..... | 5-3 |
| 6 | InstallShield supplementary information | |
| 6.1 | Setup.exe..... | 6-1 |
| 6.1.1 | Setup.exe error codes | 6-2 |
| 6.1.2 | Setup runtime errors | 6-4 |

1 Introduction

1.1 Purpose

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

The document describes in detail how this is performed in the Windows environment with MIMER version 8.1.2 or later. In chapter 4 of this document there is information on how to package version 7.3.3 and 8.1.1 of MIMER.

The document is intended for Value Added Resellers (VARs) which package and sell MIMER together with their own application.

1.2 Definitions, Acronyms and Abbreviations

| <u>Definition</u> | <u>Description</u> |
|-------------------|--|
| Data source | The data that the user wants to access and its associated operating system, DBMS and network protocol (if any). |
| Driver | A routine library where the database specific ODBC functions are implemented. Drivers are specific to a single DBMS. |
| 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. |

| <u>Abbreviation</u> | <u>Short for</u> | <u>Description</u> |
|---------------------|-----------------------------------|--|
| API | Application Programming Interface | A documented set of general routines designed for some limited area of use. |
| 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. |
| PSM | Persistent Stored Modules | Term used by standard committees, such as ANSI and ISO, when they mean stored procedures. |

SQL Structured Query Query language for relational databases.
Language

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

1.3 **References**

[1] MIMER System Management Handbook

[2] ODBC 3.0 Programmer's Reference and SDK Guide volumes 1 and 2, Microsoft

[3] InstallShield 5.1 Professional Edition, Getting Started

1.4 **Document Overview**

The document describes, step by step, how to install MIMER together with an application. The steps required are

- Running a silent setup of MIMER
- Configuring a database and a data source
- Creating the database
- Starting the database server
- Creating the database objects needed by the application.

A special section deals with how MIMER's installation program handles other versions of MIMER on the system where the installation is taking place.

The document also discusses how to upgrade already existing MIMER systems from version 7.x to version 8.1.

2 Creating the installation package

The goal of this document is to explain the steps necessary to bundle the MIMER relational database management system with an application. This will allow an application developer to install both the application and MIMER with one integrated installation script.

This chapter describes the steps to follow to create the installation package.

2.1 Silent MIMER installation

The first step is to create a silent installation of MIMER. This installation will, in the integrated solution, be activated by the application's installation program.

Start by obtaining a distribution of MIMER. The distribution is typically divided into a number of directories called Disk1, Disk2 and so on. The directory called Disk1 contains a program called setup.exe which performs the MIMER installation.

The installation of MIMER version 7.3.3 and 8.1 uses the Installshield 5.1 installation program. You can find more information about Installshield at their web site <http://www.installshield.com>.

To install MIMER together with your application you need to create a template installation of MIMER by actually running an installation and recording the options used when installing MIMER in a response file.. It is then possible to perform a silent installation of MIMER.

To create the response file, run setup.exe located in the Disk1 folder of your MIMER installation with option -r:

```
setup -r -f1C:\MyApp\mimresp.iss
```

Perform a normal installation of MIMER. The choices you make are recorded in the response file (in the example the response file is placed in the directory C:\MyApp). The silent setup can be tested by the following command line:

```
setup -f1C:\MyApp\mimresp.iss -f2C:\MyApp\mimresp.log
```

The options for creating the silent install file and later invoking the silent setup are described in chapter 6.

2.2 Defining MIMER databases

Once MIMER has been installed, the next phase is to create a MIMER database.

2.2.1 Introduction

The interactive method of defining MIMER databases and their properties is by using the MIMER Administrator. The MIMER Administrator allows the user to create, modify or to remove local or remote MIMER databases as well as ODBC data sources.

To create and manage MIMER databases and data sources from a program the ODBC routine `SQLConfigDataSource` is used. The routine is documented in Section 5.1. `SQLConfigDataSource` may be used to perform the same tasks as you would normally do through the MIMER Administrator. It is possible to override any defaults for parameters and, if you like, to invoke the same dialogs as the MIMER Administrator uses to allow the user of your application to perform customization for their specific environment.

2.2.2 MIMER specific parameters

To set up and configure a MIMER database, the parameters 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 through the help facility in the MIMER Administrator. Many of the parameters are also described in the MIMER System Management Handbook.

The parameters are divided into the following groups:

- Data source parameters
- Local database parameters
- Local database commands
- Remote database parameters

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

Explanation

DSN

The ODBC data source name (**required**). It is recommended to use the same name as for the database

Description

The explanatory text describing the data source.

Database

The name of the MIMER database (**required**).

| <u>Local Database Parameters</u> | <u>Explanation</u> |
|----------------------------------|--|
| Database | The name of the MIMER database (required). |
| Description | The explanatory text describing the database. |
| Directory | The database home directory (required). |
| 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 is strongly recommended even though it may take longer to restart the database server after a system failure. 0 = Index Only 1 = All Pages |
| PriorityClass | The priority class of the server. Should be an integer in the range 0-3. 0 = Idle 1 = Normal 2 = High 3 = Realtime 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 raised. |
| 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. |

| | |
|--------------------|---|
| 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 |
| Pages2K | An integer specifying the number of 2K pages in the bufferpool. |
| Pages16K | An integer specifying the number of 16K pages in the bufferpool. |
| Pages64K | An integer specifying the number of 64K 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. |
| 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. |
| LocalComm | This parameter controls the type of local communication supported between applications and the database server. A string specifying <Shared Memory> or <Disabled> are currently supported. |
| 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 | <p>This is the name of the named pipe the database server use to listen for incoming requests. The default is a named pipe with the same name as the database server.</p> <p>As for TcpPort the strings <Name Server> or <Disabled> may also be used.</p> |
| RmGuid | <p>This is a unique identifier identifying the database server used for interoperating with Microsoft Distributed Transaction Coordinator.</p> <p>Do not specify RmGuid unless you are renaming a database server, in which case the original RmGuid should be kept.</p> |
| DumpPath | <p>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.</p> |

Local Database Server Commands

Explanation

| | |
|----------|--|
| Database | The name of the MIMER database (required). |
| SDBGEN | <p>The parameters to the command correspond to the command line arguments specified when using the system databank generation utility (SDBGEN).</p> <p>This is further described in section 2.3.</p> |
| DbServer | The DbServer command accepts the two strings START and STOP. |

Remote Database Parameters

Explanation

| | |
|-------------|--|
| Database | The name of the MIMER database on the remote host (required). |
| Description | The explanatory 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 database is created (required). |
| Protocol | Specifies whether to use named pipes or TCP/IP. Should either be the string 'NamedPipes' or 'tcp'. 'tcp' is 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 for a version 8 server is to listen on a pipe with the same name as the database. In previous versions, the default was MIMER. I.e. you must know if the database is a version 7 or version 8 database. The default for a version 8 client is to use the database name. |
| Interface | Not used on the Windows platform. |

2.2.3 Defining a remote database

The following C example will create the definition for a remote database on the host db.mimer.com and will communicate by using the TCP/IP protocol. The routine SQLConfigDataSource is described in detail in section 5.1. Note that this examples creates a system-wide data source name.

```
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 creates a new data source MyRemoteODBC and also a remote MIMER definition for the database MyRemoteMIMER. The operation is performed without displaying any dialogs as the first (hwndParent) parameter is set to NULL.

Alternately, the same thing could have been achieved with the following two calls instead:

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

and

```
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");
```

First, the data source is created and then, in a separate call, the remote MIMER definition is created. 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. If any parameters which are specific to a Local or Remote database definition are given, the system will create the appropriate database definition. In the second example the parameters NODE and PROTOCOL indicates that a remote database definition should be created.

Since the parameter is ODBC_ADD_SYS_DSN this means that any existing definition 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");
```

Also note that if the configuration is handled as a number of calls, then the error handling, from the installation programs point of view, can much more easily determine what went wrong. In the first example it is possible that a data source has been created even though the database definition was not.

The error handling is described in Section 3.6.

2.2.4 Defining a local database

To create a local database one or more local parameters are specified in a call to SQLConfigDataSource. The following C program will create a local database definition with 1000 2K pages in the bufferpool.

```
rc = SQLConfigDataSource(hWnd, ODBC_ADD_SYS_DSN,
    "MIMER",
    "DATABASE=MyLocalMIMER\0"
    "Pages2K=1000\0\0");
```

The changes made by the installation program can be viewed by the Registry Editor (regedit.exe) under the following keys:

HKEY_LOCAL_MACHINE:

```
Software\Sysdeco Mimer\MIMER\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.

The call will display a dialog box (as the first argument (hwndParent) is not null). The call would fail if the hwndParent had been NULL as no directory was specified and this parameter is required for a Local database definition. The number of 2K pages will be set to 1000 and all other parameters will be set to their default values.

2.3 Creating the System Databanks

It is possible to create the system databanks for a database with the SQLConfigDataSource routine. The same arguments are specified as can be given as command line arguments to SDBGEN:

```
SDBGEN      -pSYSADM-Password
            Database-name
            Sysdb-Size
            Transdb-FileName
            Transdb-Size
            Logdb-Filename
            Logdb-Size
            Sqldb-Filename
            Sqldb-Size
```

Any parameter where you want to use the default value can be specified as two double quotes (""). All parameters except database-name are optional.

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

The above call will start a system databank generation for the database MyLocal. The initial size for SYSDB will be 1000 and for TRANSDB it will be 2000 2K blocks. Note that the default filename for TRANSDB is used by specifying "" for the parameter. It is highly recommended to 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 from a recovery and performance point of view.

Whenever the -p option is used the program is run in silent mode. In this mode no dialogs are displayed and any missing directories are created automatically. If an error occurs such as disk space exhausted, a dialog is displayed where filenames and/or sizes may be changed. However, in this case the fields for SYSADM password are disabled.

If option -p is not given the values specified will be used as the default values in the system databank generation dialog.

Examples of valid command lines assuming a local database definition with the name DB6 exists:

- 1) SDBGEN DB6
- 2) SDBGEN -pSYSpsw DB6
- 3) SDBGEN -pSYSpsw DB6 "" "d:\my directory\TRANS.DBF" 200 e:\logdir\logdb.dbf

In case 1) the ordinary System databank generation is run for a database named DB6. In case 2) the program is run in silent mode using the defaults for filenames and sizes as decided by the SDBGEN program. The password SYSpsw is given to the system administrator. Finally, in case 3) the default size for SYSDB (") is used, but the filename for TRANSDB is set to d:\my directory\TRANS.DBF and the size to 200 2K pages. The filename for LOGDB is set to e:\logdir\logdb.dbf. Please note the use of quotes when blanks are included in the file name (in the directory path for the file TRANS.DBF).

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

2.4 Starting the database server

At this point MIMER has been installed and, usually, a local database with associated system databanks has been created. The local database server should be started next. On Windows 95/98 this is not necessary if the database has been configured for Autostart. The following code works on both Windows 95/98 and Windows NT so you may elect to use the same logic in both environments.

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");
```

2.5 More useful examples

Here are a few more useful example calls to SQLConfigDataSource:

This is the simplest possible call to create a system data source, a MIMER database, creating the system databank, and starting the database server in just one call:

```
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");
```

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.

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");
```


3 Example MIMER configuration program

The following example program is a console mode program which 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 silent install of MIMER has completed, i.e. MIMER needs to be installed at this point.

In the example program the routines are prefixed by I_. All other calls are either to C runtime library, ODBC, or WIN32 routines. The main program looks as follows:

3.1 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_DefineDatabase();
    I_CreateSystemDatabanks();
    I_StartupDatabaseServer();
    I_CreateApplicationObjects();
    exit(0);
}
```

3.2 I_DefineDatabase

The first example subroutine defines the ODBC data source and the MIMER database parameters:

```
void I_DefineDatabase()
{
    RETCODE 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 != SQL_SUCCESS &&
        rc != SQL_SUCCESS_WITH_INFO) {
        I_InstError("Error creating database definition:");
    }
}
```

3.3 I_CreateSystemDatabanks

This routine starts a process which runs the MIMER system databank generation program in silent mode.

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

3.4 I_StartupDatabaseServer

The following code starts the database server:

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

3.5 I_CreateApplicationObjects

This routine sets up the MIMER database environment needed by the application (error handling has been left out of this routine for clarity):

```
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 objects such as databanks, users, tables, and views needed to run the application.

3.6 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",
               iError,
               fErrorCode,
               szErrorMsg);
    }
}

```

The routine `SQLInstallerError` is further described in section 5.2 and also in the Microsoft ODBC documentation.

4 Information about older MIMER versions

This chapter describes the differences between version 7.3.3, 8.1.1 and version 8.1.2 (or later).

4.1 Upgrading from earlier versions

On Windows, MIMER does not support parallel installations of different versions of MIMER. In the following sections components refer to items such as the Database server, Database client, and Online documentation.

The logic used by the installation program is as follows:

- If the new installation is a change of major version (i.e. 7.3 to 8.1) the install script asks the user whether they want to uninstall the old version and install the new. If a silent install is made, the installation proceeds without asking.
- Only the components selected by the silent setup are installed.
- If the new installation is a change of minor version (i.e. 8.1.1 to 8.1.2) the system will uninstall the old version (8.1.1) and install all the components that were previously installed plus any additional components selected by the silent setup.
- The fact that an uninstallation is performed allows a change of installation directory without causing any disturbance.
- If the new installation is a change of minor revision, but to an older version, only components previously installed are added to the current installation.
- If the new version is a change of major version, but the new version is an older version (such as 8.2 installed and trying to install 8.1). the installation fails. To do this, the user must first uninstall the new version and then redo the old installation.

When upgrading between major versions (7.3 to 8.1) you must run a database upgrade utility. The utility program is called UP7TO8W. The program accepts a database name as argument. When a database is supplied the upgrade proceeds immediately. This may be part of your configuration program discussed in the previous chapter.

Remember that a proper backup should have been made before upgrading the database.

4.2 Version 7.3.3

Version 7.3.3 of MIMER did not have support for the following:

- The administration dialogs
- System databank generation support
- Start/stop database server
- SQLInstallerError

A number of parameters had different names and default values:

| <u>7.3 parameter</u> | <u>8.1 parameter</u> | <u>Comment</u> |
|----------------------|----------------------|---|
| Kernels | RequestThreads | In version 8 each request takes relatively much longer time (and gets much more work done!). As a consequence more threads are needed. |
| Shadows | BackgroundThreads | In version 8 the background threads perform more tasks than the shadow threads did in version 7. As a consequence more threads are needed. |
| Cleanup | | This area no longer needs to be configured in version 8. |
| KernelPriority | RequestPriority | Name change following on from the use of RequestThreads. |
| ShadowPriority | BackgroundPriority | In version 8 the background threads may need a slightly higher priority than the request threads in order to maintain a balance in the database server. This can become evident under very high system loads. |

4.3 Version 8.1.1

Version 8.1.1 of MIMER did not have support for the following options in SQLConfigDataSource:

- The administration dialogs
- System databank generation
- Start/stop database server
- SQLInstallerError

Instead these things had to be coded by hand. Contact your MIMER distributor if you would like sample programs for how to do this. SQLInstallerError did not return any error messages which indicated what had gone wrong.

5 ODBC supplementary information

Refer to the ODBC help files for complete information about the available interfaces for ODBC configuration.

5.1 SQLConfigDataSource

The following specification of SQLConfigDataSource is from the Microsoft ODBC SDK Reference.

Summary

SQLConfigDataSource adds, modifies, or deletes data sources.

Syntax

```
BOOL SQLConfigDataSource(
    HWND      hwndParent,
    WORD      fRequest,
    LPCSTR    lpszDriver,
    LPCSTR    lpszAttributes);
```

Arguments

hwndParent [Input]

Parent window handle. The function will not display any dialog boxes if the handle is null.

fRequest [Input]

Type of request. *fRequest* must contain one of the following values:

ODBC_ADD_DSN: Add a new user data source.

ODBC_CONFIG_DSN: Configure (modify) an existing user data source.

ODBC_REMOVE_DSN: Remove an existing user data source.

ODBC_ADD_SYS_DSN: Add a new system data source.

ODBC_CONFIG_SYS_DSN: Modify an existing system data source.

ODBC_REMOVE_SYS_DSN: Remove an existing system data source.

ODBC_REMOVE_DEFAULT_DSN: Remove the default data source specification section from the system information. It also removes the default driver specification section from the ODBCINST.INI entry in the system information. (This *fRequest* performs the same function as the deprecated **SQLRemoveDefaultDataSource** function.) When this option is specified, all of the other parameters in the call to **SQLConfigDataSource** should be NULLs; if they are not NULL, they will be ignored.

lpszDriver [Input]

Driver description (usually the name of the associated DBMS) presented to users instead of the physical driver name.

lpszAttributes [Input]

List of attributes in the form of keyword-value pairs.

Returns

The function returns TRUE if it is successful, FALSE if it fails. If no entry exists in the system information when this function is called, the function returns FALSE.

Diagnostics

When **SQLConfigDataSource** returns FALSE, an associated **pfErrorCode* value may be obtained by calling **SQLInstallerError**. The following table lists the **pfErrorCode* values that can be returned by **SQLInstallerError** and explains each one in the context of this function.

| <i>*pfErrorCode</i> | Error | Description |
|----------------------------------|--|---|
| ODBC_ERROR_GENERAL_ERR | General installer error | An error occurred for which there was no specific installer error. |
| ODBC_ERROR_INVALID_HWND | Invalid window handle. | The <i>hwndParent</i> argument was invalid. |
| ODBC_ERROR_INVALID_REQUEST_TYPE | Invalid type of request | The <i>fRequest</i> argument was not one of the following: ODBC_ADD_DSN ODBC_CONFIG_DSN ODBC_REMOVE_DSN ODBC_ADD_SYS_DSN ODBC_CONFIG_SYS_DSN ODBC_REMOVE_SYS_DSN ODBC_REMOVE_DEFAULT_DSN |
| ODBC_ERROR_INVALID_NAME | Invalid driver or translator name. | The <i>lpszDriver</i> argument was invalid. It could not be found in the registry. |
| ODBC_ERROR_INVALID_KEYWORD_VALUE | Invalid keyword value pairs. | The <i>lpszAttributes</i> argument contained a syntax error. |
| ODBC_ERROR_REQUEST_FAILED | Request failed. | The installer could not perform the operation requested by the <i>fRequest</i> argument. The call to ConfigDSN failed. |
| ODBC_ERROR_LOAD_LIBRARY_FAILED | Could not load the driver or translator setup library. | The driver setup library could not be loaded. |
| ODBC_ERROR_OUT_OF_MEMORY | Out of memory | The installer could not perform the function because of a lack of memory. |

Comments

SQLConfigDataSource uses the value of *lpszDriver* to read the full path of the setup DLL for the driver from the system information. It loads the DLL and calls **ConfigDSN** with the same arguments that were passed to it.

SQLConfigDataSource returns FALSE if it is unable to find or load the setup DLL, or if the user cancels the dialog box. Otherwise, it returns the status it received from **ConfigDSN**.

5.2 SQLInstallerError

The following specification of SQLInstallerError is from the Microsoft ODBC SDK Reference.

Summary

SQLInstallerError returns error or status information for the ODBC installer functions.

Syntax

```
RETCODE SQLInstallerError(
    WORD          iError,
    DWORD *      pfErrorCode,
    LPSTR        lpzErrorMsg,
    WORD         cbErrorMsgMax,
    WORD *       pcbErrorMsg);
```

Arguments

iError [Input]

Error record number. Valid numbers are from 1 to 8.

pfErrorCode [Output]

Installer error code. (For more information, see “Comments.”)

lpzErrorMsg [Output]

Pointer to storage for the error message text.

cbErrorMsgMax [Input]

Maximum length of the *szErrorMsg* buffer. This must be less than or equal to SQL_MAX_MESSAGE_LENGTH minus the null-termination character.

cbErrorMsgMax [Input]

Maximum length of the *szErrorMsg* buffer. This must be less than or equal to SQL_MAX_MESSAGE_LENGTH minus the null-termination character.

pcbErrorMsg [Output]

Pointer to the total number of bytes (excluding the null-termination character) available to return in *lpzErrorMsg*. If the number of bytes available to return is greater than or equal to *cbErrorMsgMax*, the error message text in *lpzErrorMsg* is truncated to *cbErrorMsgMax* minus the null-termination character bytes. The *pcbErrorMsg* argument can be a null pointer.

Returns

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NO_DATA, or SQL_ERROR.

Diagnostics

SQLInstallerError does not post error values for itself. **SQLInstallerError** returns `SQL_NO_DATA` when it is unable to retrieve any error information (in which case *pfErrorCode* is undefined). If **SQLInstallerError** cannot access error values for any reason that would normally return `SQL_ERROR`, **SQLInstallerError** returns `SQL_ERROR`, but does not post any error values. If either the *lpszErrorMsg* argument is `NULL`, the *cbErrorMsgMax* is less than or equal to 0, then this function returns `SQL_ERROR`. If the buffer for the error message is too short, **SQLInstallerError** returns `SQL_SUCCESS_WITH_INFO`, and returns the correct *pfErrorCode* value for **SQLInstallerError**.

To determine whether a truncation occurred in the error message, an application can compare the value in the *cbErrorMsgMax* argument to the actual length of the message text written to the *pcbErrorMsg* argument. If truncation does occur, the correct buffer length should be allocated for *lpszErrorMsg* and **SQLInstallerError** should be called again with the corresponding *iError* record.

Comments

An application calls **SQLInstallerError** when a previous call to the ODBC installer function returns `FALSE`. ODBC installer and driver or translator setup functions post zero or more errors only when the function fails (returns `FALSE`); therefore, an application calls **SQLInstallerError** only after an ODBC installer function fails.

The ODBC installer error queue is flushed each time a new installer function is called. Therefore, an application cannot expect to retrieve errors for functions other than from the last installer function call.

To retrieve multiple errors for a function call, an application calls **SQLInstallerError** multiple times.

When there is no additional information, **SQLInstallerError** returns `SQL_NO_DATA`, the *pfErrorCode* argument is undefined, the *pcbErrorMsg* argument equals 0, and the *lpszErrorMsg* argument contains a single null-termination character (unless the *cbErrorMsgMax* argument is equal to 0).

6 InstallShield supplementary information

This chapter contains relevant information about InstallShield errors that may occur. For complete information please refer to the InstallShield help files and/or reference manual.

6.1 Setup.exe

Setup.exe is the main setup executable; it performs setup initialization and launches the appropriate Setup engine file (`_instxxx.ex_`, where `xxx` indicates the target platform) to execute the setup script (Setup.ins) on the target system.

When you first create a setup project or run the Media Build Wizard with a new media name, Setup.exe and `_instxxx.ex_` are placed in the Disk1 folder under the default location. You must ship these files on Disk1 of your distribution media. Once you have built your setup, you can rename Setup.exe to any valid filename, such as Install.exe.

You can pass command line switches directly to Setup.exe, or you can place them in Setup.ini. For testing purposes, while you are using the InstallShield IDE you can pass command line switches to Setup.exe in the Setup Settings dialog box.

Syntax

Setup [switches]

Switches

These switches are optional. They are not case sensitive; upper- or lowercase letters can be used.

/f<path\CompiledScript> or -f<path\CompiledScript>

Specifies an alternate compiled script. Unless the compiled script (.ins file) also resides in the same directory as that of Setup.exe, the full path to the compiled script must be specified. `_setup.dll` must also reside in the same directory as your .ins file. For example, `setup -ftest.ins` will launch setup using Test.ins instead of Setup.ins.

/f1<path\ResponseFile> or -f1<path\ResponseFile>

Specifies an alternate location and name of the response file (.iss file). If this option is used when running InstallShield Silent, the response file is read from the folder/file specified by `<path\ResponseFile>`. If this option is used along with the `-r` option, the response file is written to the folder/file specified by `<path\ResponseFile>`. If an alternate compiled script is specified using the `-f` switch, the `-f1` switch entry must follow the `-f` switch entry.

/f2<path\LogFile> or -f2<path\LogFile>

Specifies an alternate location and name of the log file created by InstallShield Silent. By default, Setup.log log file is created and stored in the same directory as that of Setup.ins. If an alternate compiled script is specified using the -f switch, the -f2 switch entry must follow the -f switch entry.

/r or -r

Causes Setup.exe automatically to generate a silent setup file (.iss file), which is a record of the setup input, in the Windows folder.

/s or -s

Runs InstallShield Silent to execute a silent setup .

/z or -z

Prevents Setup.exe from checking the available memory during initialization. This switch is necessary when running a setup on a machine with more than 256 MB of disk space; if it is not used, Setup.exe reports insufficient memory and exits.

Comments

- Separate multiple command line switches with a space. But do not put a space inside a command line switch (for example, /r /fInstall.ins is valid, but not /r/f Install.ins).
- When using long path and filename expressions with switches, enclose the expressions in double quotation marks. The enclosing double quotes tell the operating system that spaces within the quotation marks are not to be treated as command line delimiters.

Errors

Setup.exe may produce error messages if it cannot start properly. In most cases you'll encounter these messages when a severe error occurs. Rarely will your end users see these messages.

When you get an error message, it appears in a message box. Every error message has a number. These are InstallShield system error messages and there is no way to suppress them in your script.

6.1.1 Setup.exe error codes

| <u>Error Number</u> | <u>Message</u> |
|---------------------|---|
| 101 | Setup is unable to find a hard disk location to store temporary files. Make at least 500KB of free disk space available and then try the setup again. |
| 102 | Setup is unable to find a compressed library file required to proceed with the setup. Check to make sure all required files are located with the Setup program. |
| 103 | Setup is unable to find _setup.dll, which is needed to complete the setup. Restart your system and try again. |

- 104 Setup is unable to find the language section in the Lang.dat file, or Lang.dat could not be found.
- 105 Setup.lid cannot be found or it has an invalid format.
- 106 The language dialog box cannot be created
- 107 Setup is unable to locate the script file <%s> which is needed to complete the setup.
- 110 Setup was started with a command line argument that contained an incomplete parameter bad_parameter.
- 111 Insufficient memory available to run Setup. Close all other applications to make more memory available and try to run Setup again.
- 112 Setup is unable to locate the layout file ‘...\Layout.bin’ which is needed to complete the setup.
- 201 Setup is unable to initialize the setup program (Install.exe).
- 202 Setup is unable to initialize the setup program.
- 301 Setup was unable to start up the setup program.
- 401 String variable is not large enough for string. Please check string declarations.
- 420 Setup is unable to copy the setup support file <filename> to a temporary location.
- 421 Setup is unable to copy the setup support file _user1.cab to a temporary location. Make more space available and try again.
- 422 Setup is unable to expand the setup support file <filename>.
- 423 Setup is unable to load the setup script file.
- 424 Setup has encountered an internal stack overflow error. Close all applications, restart the system and try the setup again.
- 425 Setup has encountered an incomplete return statement in the script. Check your script for unmatched return statements.
- 426 Setup is unable to find the setup script file: script_filename.
- 427 Setup is unable to load the setup script file: script_filename. The script may be from a previous version or corrupted.
- 432 Setup has detected that unInstallShield is in use. Please close unInstallShield and restart setup.
- 440 Setup has detected a possible infinite loop in the script with function function_name. Make sure your are handling the error returns codes properly.
- 502 Setup is unable to initialize the setup program. The script file may be bad.

| | |
|------|--|
| 701 | A division by zero error was detected in the script. Setup will continue. |
| 702 | An internal error has occurred. Insufficient memory to allocate buffer. |
| 703 | An internal read error has occurred on script_filename. Unable to load setup instructions. |
| 704 | Script_filename file has become corrupted. Unable to load setup instructions. |
| 30xx | Error messages ranging from 3000 to 3021 are internal memory-related error conditions. |

6.1.2 Setup runtime errors

When copying the installation files the following errors may occur. These error numbers will occur in the the ResponseCode in the response file.

| <u>Code</u> | <u>Description</u> |
|-------------|---|
| 0 | Success. |
| -1 | General error. |
| -2 | Invalid mode. |
| -3 | Required data not found in the Setup.iss file. |
| -4 | Not enough memory available. |
| -5 | File does not exist. |
| -6 | Cannot write to the response file. |
| -7 | Unable to write to the log file. |
| -8 | Invalid path to the InstallShield Silent response file. |
| -9 | Not a valid list type (string or number). |
| -10 | Data type is invalid. |
| -11 | Unknown error during setup. |
| -12 | Dialog boxes are out of order. |
| -51 | Cannot create the specified folder. |
| -52 | Cannot access the specified file or folder. |
| -53 | Invalid option selected. |

The following table describes the error codes returned by ComponentError:

| <u>Code</u> | <u>Description</u> | <u>Cause</u> |
|-------------|--|--|
| -101 | Cannot add component. | ComponentAddItem was unable to add a component to the script-created component set. |
| -102 | Specified component already exists. | ComponentAddItem was called twice with the same media name and component name. |
| -103 | Specified component cannot be selected or unselected. | ComponentSelectItem was called to select or deselect a component required by a currently selected component. Before attempting to select or deselect the specified component, deselect the currently selected component that requires the specified component. |
| -104 | Specified component name is not valid. | The value passed in the second parameter of ComponentInitialize is not valid. |
| -105 | Specified component cannot be found in the media. | An attempt was made to access a component that does not exist in the named media. This error occurs when a component name is specified incorrectly in call to a component function. Component names must be specified exactly as they appear in the Components Pane or in the call to ComponentAddItem. Case is sensitive. |
| -106 | Unable to decompress a file. | An internal error occurred. Contact technical support. |
| -107 | Disk ID specified in call to ComponentMoveData is not valid. | ComponentMoveData has already been called to transfer files and has not been reinitialized. To reinitialize ComponentMoveData, call that function with a null string in the first parameter. |
| -108 | Out of disk space. | The target disk or directory has insufficient free space; the disk space can not be determined because TARGETDIR is invalid; or a script-defined folder of a component has not been set. |
| -109 | EnterDisk function called failed. | Internal error occurred. Contact technical support. |
| -112 | Specified file cannot be found. | To determine which file is missing, check the value returned by ComponentError in the parameter svFile. |
| -113 | Specified file cannot be opened as read-only. | The file Data1.cab (or one of the other data cab files) is missing or corrupted; or an uncompressed data file is missing from a CD-ROM, Data As Files build. |

| | | |
|------|---|--|
| -114 | Specified file cannot be opened as read/write. | Unable to append to split file. Contact technical support. |
| -115 | Specified file cannot be opened as write. | An attempt was made to overwrite a locked file belonging to a file group that does not have the Potentially Locked or Shared property set to Yes; an attempt was made to install a file with a long file name (or to a folder with a long path name) in a 16-bit setup; or the path to the target folder is invalid. |
| -116 | Invalid file specification made in function call. | The value of the parameter szFile in a call to ComponentFileInfo is invalid (for example, a null string). |
| -117 | Cannot read the specified file. | A data cab file or an uncompressed data file may be corrupt. |
| -118 | Attempted operation not allowed with script-created component sets. | A script-created component set name was passed to a component function (for example, ComponentFileInfo), that operates only on file media. |
| -119 | Unable to self-register a file properly. | This error has many possible causes. For details, refer to article Q101538 in the InstallShield Knowledge Base. |
| -120 | Unable to update a shared file in ComponentMoveData. | Internal error. Contact technical support. |
| -121 | Unable to write to a file. | Internal error. Contact technical support. |
| -123 | Unable to find a file group. | The specified file group could not be found. The name of the missing file group is returned by ComponentError in the parameter svFileGroup. |
| -125 | The list specified in the component function is not valid. | When calling ComponentFileEnum, ComponentFileInfo, ComponentListItems, or ComponentSetupTypeEnum, verify that the list you are passing to the function is valid. |
| -126 | Attempted operation not allowed with file media library. | A file media name was passed to a component function (for example, ComponentAddItem), that operates only on script-created component sets. |
| -127 | Media is already initialized. | ComponentInitialize was called to initialize a media that was already initialized. |
| -128 | The specified file media library was not generated by the InstallShield Media Build Wizard. | The file Data1.cab is corrupt, or the file specified in a call to ComponentInitialize is not an InstallShield-generated cabinet file. |

| | | |
|------|--|--|
| -132 | The specified media cannot be found. | The media has been declared but it not associated with any components. Make sure that either script-created components or file media components are associated with the media. |
| -133 | An error occurred with the specified media. | ComponentMoveData has already been called to transfer files and has not been reinitialized. If your script calls ComponentMoveData more than once, you must reinitialize it after each call by calling it again with a null string in its first parameter. |
| -136 | Unable to allocate memory. | Insufficient memory is available to the setup. Display a message to the end user to close down all other applications or to cancel the setup, reboot the system, and restart the setup. |
| -137 | Specified option is not valid. | An invalid option has been specified for a component function, for example, by passing INCLUDE_SUBDIR in the fourth parameter of ComponentFileInfo or by specifying only a file group when both a file group and file name are required. |
| -139 | Specified password does not match. | The specified password does not match the password stored in the specified file media library or component. |
| -141 | Specified password cannot be found. | ComponentValidate was called to validate a component or a file media library for which no password has been set. |
| -142 | The media or the component password was not validated. | ComponentValidate was not called to validate the components and/or the file media library before transferring those components with ComponentMoveData. |
| -145 | Target path for the component cannot be found. | The target directory for a script-defined folder has not been set or is invalid. |
| -147 | Invalid value passed to a component-related function. | One of the values passed to a component function is invalid. This error can be caused, for example, by passing an empty string in the second parameter of ComponentAddItem. |

| | | |
|------|---|--|
| -148 | Data cannot be read from the internet. | This error occurs when using InstallFromTheWeb in conjunction with InstallShield5. InstallShield is unable to read the data from the Internet because the files are corrupt or the Internet connection has been lost and cannot be reestablished by InstallFromTheWeb. |
| -149 | Internet has been disconnected. | This error occurs when using InstallFromTheWeb in conjunction with InstallShield5. The Internet connection has been lost and cannot be reestablished by InstallFromTheWeb. |
| -150 | Cabinet file generated by an older version of InstallShield5. | Verify that the project was built with your most recent version of InstallShield5. Verify that you are not using mismatched cabinet files generated by different versions of InstallShield5. |
| -623 | Error renaming a file. | An attempt was made to transfer an executable file (an .exe or .com file) over a locked file without setting the Potentially Locked property to Yes. |