



# Mimer SQL

## Packaging Guide for Windows

Version 9.1

Mimer SQL, Packaging Guide for Windows, Version 9.1, May 2004  
© Copyright Mimer Information Technology AB.

The contents of this manual may be printed in limited quantities for use at a Mimer SQL installation site. No parts of the manual may be reproduced for sale to a third party.  
Information in this document is subject to change without notice. All registered names, product names and trademarks of other companies mentioned in this documentation are used for identification purposes only and are acknowledged as property of the respective company. Companies, names and data used in examples herein are fictitious unless otherwise noted.

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

Mimer Web Sites:  
<http://developer.mimer.com>  
<http://www.mimer.com>

# Contents

---

<b>Chapter 1 Introduction .....</b>	<b>1</b>
<b>1.1 Document Overview .....</b>	<b>1</b>
<b>1.2 Definitions and Abbreviations .....</b>	<b>2</b>
<b>1.3 References .....</b>	<b>2</b>
<b>Chapter 2 Creating the Installation Package .....</b>	<b>3</b>
<b>2.1 Methods for Creating a Silent Mimer SQL Installation.....</b>	<b>3</b>
<b>2.2 Creating a Silent Mimer SQL Installation with InstallShield.....</b>	<b>4</b>
2.2.1 The Mimer Installation Files .....	4
2.2.2 Creating a Response File.....	4
2.2.3 Including Mimer SQL in Your Installation Script.....	5
2.2.3.1 Example Output .....	5
<b>2.3 Creating a Mimer ODBC Stand-alone Installation Package.....</b>	<b>6</b>
2.3.1 Including the Mimer ODBC Components .....	6
2.3.1.1 ODBC Stand-alone Installation Example .....	7
2.3.2 Removing the Mimer ODBC Components .....	7
2.3.2.1 ODBC Stand-alone Removal Example .....	8
2.3.3 External Dependencies.....	8
<b>2.4 Adding a Mimer SQL License Key .....</b>	<b>9</b>
<b>2.5 Defining a Database .....</b>	<b>9</b>
2.5.1 Defining a Remote Database .....	9
2.5.1.1 Example 1 .....	10
2.5.1.2 Example 2 .....	10
2.5.2 Defining and Configuring a Local Database.....	11
2.5.3 Error Handling .....	11
<b>2.6 Generating the System Databanks .....</b>	<b>12</b>
2.6.1 SDBGEN Parameters .....	12
<b>2.7 Starting the Database Server .....</b>	<b>13</b>
<b>2.8 More Useful Examples .....</b>	<b>13</b>
2.8.1 Example 1 .....	13
2.8.2 Example 2 .....	14
2.8.3 Example 3 .....	14
2.8.4 Example 4 .....	14

2.8.5 Example 5 .....	14
<b>Chapter 3 Example Configuration Program.....</b>	<b>15</b>
<b>3.1 Main Program.....</b>	<b>15</b>
3.1.1 I_SetupLicense.....	16
3.1.2 I_DefineDatabase.....	16
3.1.3 I_CreateSystemDatabanks .....	17
3.1.4 I_StartupDatabaseServer.....	17
3.1.5 I_CreateApplicationObjects.....	18
3.1.6 I_InstError.....	19
<b>Chapter 4 About Previous Mimer SQL Versions .....</b>	<b>21</b>
4.1 Upgrading a Mimer SQL Installation.....	21
4.2 Upgrading a Mimer SQL Database.....	22
<b>Chapter 5 Mimer Packaging Parameters.....</b>	<b>23</b>
<b>5.1 Mimer SQL Specific Parameters and Commands .....</b>	<b>23</b>
5.1.1 Data Source Parameters.....	24
5.1.2 Local Database Parameters.....	24
5.1.3 Local Database Server Commands .....	28
5.1.4 Remote Database Parameters.....	28
5.1.5 License Key Parameters .....	29
<b>Chapter 6 ODBC Supplementary Information.....</b>	<b>31</b>
6.1 SQLConfigDataSource .....	31
6.2 SQLConfigDriver.....	34
6.3 SQLInstallDriverEx .....	37
6.4 SQLInstallerError.....	43
6.5 SQLRemoveDriver .....	45
<b>Chapter 7 InstallShield Information .....</b>	<b>49</b>
<b>7.1 setup.exe .....</b>	<b>49</b>
7.1.1 setup.exe Error Codes .....	52
7.1.2 Setup Runtime Errors.....	54
7.1.3 Component Error Codes .....	54

# 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 9.1 or later. The following Windows operating systems are supported: 98, ME, NT, 2000, and XP.

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

## 1.1 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. There are two ways of doing this: using InstallShield or writing your own Setup program. See *Methods for Creating a Silent Mimer SQL Installation* on page 3 for more information.
- Add a Mimer SQL license key, see *Adding a Mimer SQL License Key* on page 9.
- Define a database, see *Defining a Database* on page 9.
- Generate the system databanks, see *Generating the System Databanks* on page 12.
- Start the database server, see *Starting the Database Server* on page 13.
- Create the database objects needed by the application, see *I\_CreateApplicationObjects* on page 18.

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

See *About Previous Mimer SQL Versions* on page 21, 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 version 7.x, and 8.x to version 9.1.

## 1.2 Definitions and Abbreviations

Definition	Description
ODBC Data Source	The database that the user wants to access and its associated operating system, DBMS and network protocol (if any).
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.
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.

## 1.3 References

- [1] InstallShield™ 6.3 Professional Edition, Getting Started
- [2] Mimer SQL Reference Manual
- [3] Mimer SQL System Management Handbook
- [4] ODBC 3.0 Programmer's Reference and SDK Guide volumes 1 and 2, Microsoft

# 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.

## 2.1 Methods for Creating a Silent Mimer SQL Installation

The first step when creating an installation package is to create a silent installation of Mimer SQL. This installation will, in the integrated solution, be activated by your application's installation program.

There are two ways of creating a silent Mimer SQL installation:

- Using InstallShield to create the installation, see *Creating a Silent Mimer SQL Installation with InstallShield* on page 4. This is the recommended way of creating a silent Mimer SQL installation.
- Writing your own Setup program and including Mimer ODBC components in it, see *Creating a Mimer ODBC Stand-alone Installation Package* on page 6.

## 2.2 Creating a Silent Mimer SQL Installation with InstallShield

Start by obtaining a distribution of Mimer SQL. Contact Mimer Information Technology AB to obtain a Mimer SQL distribution on CD for the Windows platform (NT/2000/XP or 98/ME) that suits your needs.

**Note:** The distribution can be either a full distribution, containing both client and server functionality, or just a client distribution.

The Mimer SQL version 9.1 installation uses the InstallShield 6.1 installation program. You can find more information about InstallShield at their web site <http://www.installshield.com>.

### 2.2.1 The Mimer Installation Files

An installation package is based on the Mimer SQL distribution files.

The program `setup.exe` performs the Mimer SQL installation.

The installation package should contain copies of the following files from the CD distribution:

```
data1.cab  
data1.hdr  
data2.cab  
ikernel.ex_  
layout.bin  
setup.exe  
setup.ini  
setup.inx
```

**Note:** You may not copy the `defkey9x.mcfg` or `defkeynt.mcfg` files.

### 2.2.2 Creating a Response File

A Mimer response file contains information about installation options. To create a response file, you must run the Mimer SQL `setup.exe` using the `-r` option.

**To create the response file:**

- 1 Locate `setup.exe` on the Mimer SQL distribution.
- 2 Run `setup.exe` with option `-r`, for example:

```
setup -r -f1"C:\My App\mimresp.iss"
```

The option `-r` causes some changes in the setup dialog box. The modified dialog box allows the same response file to be used both to install and to reinstall Mimer SQL. That is, running a silent setup will implicitly remove an older version of Mimer SQL when necessary. For information on the `-f1` option, see *Switches* on page 49.

The choices you make while installing are recorded in the response file. In the example above, the response file is placed in the directory `C:\My App`.

## 2.2.3 Including Mimer SQL in Your Installation Script

In your installation script, you must specify the Mimer SQL setup and response file you want to use.

**To add the information:**

**1** Run the following command:

```
setup -c"C:\My App\mimresp.log" -s  
      -f1"C:\My App\mimresp.iss" -f2"C:\My App\mimresp.log"
```

Note that in the example above:

```
-c"C:\My App\mimresp.log"
```

is the Mimer SQL specific switch which specifies the location and name of a reboot indicator log file and inserts a `RebootNeeded` line in the file.

```
-f2"C:\My App\mimresp.log"
```

specifies the location and name of the log file created by InstallShield Silent.

We recommend that the reboot indicator log file and the log file are the same file.

For more information on switches, see *Switches* on page 49.

### 2.2.3.1 Example Output

The following text is an example of the output in `mimresp.log` after the command above:

```
[InstallShield Silent]  
Version=v6.00.000  
File=Log File  
[ResponseResult]  
ResultCode=0  
RebootNeeded=0  
[Application]  
Name=Mimer SQL  
Version=9.1  
Company=Mimer  
Lang=0009
```

`ResultCode=0` indicates that the installation was successful. The setup runtime error codes are further described in *setup.exe* on page 49.

`RebootNeeded=0` indicates that it was not necessary to reboot after this installation.

If `RebootNeeded=1` had been returned, it would indicate that a reboot of Windows had been necessary in order to complete the Mimer installation.

See *InstallShield Information* on page 49 for more information on the options for creating the silent install file and invoking the silent setup.

In error situations there may be an entry "`ErrorText=...`" in the log file that provides additional information about the cause of the failure.

## 2.3 Creating a Mimer ODBC Stand-alone Installation Package

The Mimer ODBC stand-alone installation package is an alternative to the Mimer SQL client installation described above.

This method means that you write a Setup program that explicitly handles the Mimer ODBC components and calls a number of Microsoft ODBC functions for installing, uninstalling and upgrading.

The Mimer ODBC stand-alone installation has the following characteristics:

- Installs a complete Mimer ODBC driver.
- Installs minimal support for managing Mimer data sources and remote databases using the ODBC Administrator.
- Does not install or upgrade Microsoft ODBC or support for Embedded SQL and JDBC. Nor does it install any Mimer SQL tools or documentation.

When you choose this method, you should consider the following:

- Your Setup program must manage upgrading the Mimer ODBC components by removing the old components and then installing the new components.
- You are responsible for installing a suitable version of Microsoft ODBC and Microsoft libraries (Visual C++ run-time and Common Controls) required by the Mimer ODBC driver.

### 2.3.1 Including the Mimer ODBC Components

In your Setup program, you must include the following Mimer ODBC components:

- `Mimodbcw.dll`, the Mimer ODBC driver

- `Mimsetw.dll`, the Mimer ODBC data source and database administration.

You obtain these components by running a typical Mimer SQL installation and then copying the components from the directory in which you installed Mimer SQL.

In your Setup program, you install the components by using calls to `SQLInstallDriverEx` and `SQLConfigDriver`.

`SQLInstallDriverEx` adds information about the driver to the `ODBCINST.INI` entry in the system information and increments the driver's `UsageCount` by 1.

`SQLConfigDriver` completes the Mimer SQL installation by performing Mimer SQL specific functions such as update of Mimer SQL specific system information.

The arguments of these functions are further described in *ODBC Supplementary Information* on page 31.

### 2.3.1.1 ODBC Stand-alone Installation Example

The following C example will perform an ODBC stand-alone installation:

```
rc = SQLInstallDriverEx(
    "MIMER\0Driver=Mimodbcw.dll\0Setup=Mimsetw.dll\0"
    "SQLLevel=1\0"
    "FileUsage=0\0"
    "DriverODBCVer=03.51\0"
    "ConnectFunctions=YYY\0"
    "APILevel=2\0"
    "CTimeout=60\0\0",
    "C:\\Program Files\\MIMER91",
    pathout,
    255,
    NULL,
    ODBC_INSTALL_COMPLETE,
    &ucount);
if (!rc) {
    rc = SQLInstallerError(1, &errcod, errtxt, 1000, &errlen);
    printf("Error: %s\n", errtxt);
}
rc = SQLConfigDriver(NULL, ODBC_INSTALL_DRIVER, "MIMER",
    NULL, pathout, 255, &meslen);
if (!rc) {
    rc = SQLInstallerError(1, &errcod, errtxt, 1000, &errlen);
    printf("Error: %s\n", errtxt);
}
```

### 2.3.2 Removing the Mimer ODBC Components

You remove the Mimer ODBC components by using calls to `SQLRemoveDriver` and `SQLConfigDriver`.

`SQLRemoveDriver` deletes information about the driver from the `ODBCINST.INI` entry in the system information and decrements the driver's `UsageCount` by 1.

`SQLConfigDriver` completes the removal of the components by performing Mimer SQL specific functions such as updating Mimer SQL specific system information.

The arguments to these functions are further described in *ODBC Supplementary Information* on page 31.

### 2.3.2.1 ODBC Stand-alone Removal Example

The following C example will remove Mimer ODBC components:

```
rc = SQLConfigDriver(NULL, ODBC_REMOVE_DRIVER, "MIMER", NULL,
    pathout, 255, &meslen);
if (!rc) {
    rc = SQLInstallerError(1, &errcod, errtxt, 1000, &errlen);
    printf("Error: %s\n", errtxt);
}
rc = SQLRemoveDriver("MIMER", 0, &ucount);
if (!rc) {
    rc = SQLInstallerError(1, &errcod, errtxt, 1000, &errlen);
    printf("Error: %s\n", errtxt);
}
```

### 2.3.3 External Dependencies

**Note:** You are responsible for installing a suitable version of Microsoft ODBC and Microsoft libraries (Visual C++ run-time and Common Controls) required by the Mimer ODBC driver.

Microsoft ODBC version 3.51 is recommended for a stand-alone installation of Mimer ODBC version 9.1.

The following Microsoft DLLs are also required:

File name	Version	Description
Msvcrt.dll	6.0.8397.0	Visual C++ run-time library
Comctl32.dll	5.80.2614.3600	Common Controls

`Msvcrt.dll` is the Visual C++ run-time library distributed with Visual C++ 6.0.

`Comctl32.dll` can be installed by executing `50comupd.exe` which can be retrieved from Microsoft. The necessary files and instructions can be found at Microsoft's Download Center, <http://www.microsoft.com/downloads/details.aspx?FamilyID=cb2cf3a2-8025-4e8f-8511-9b476a8d35d2&DisplayLang=en>.

## 2.4 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 *Mimer Packaging Parameters* on page 23 and *ODBC Supplementary Information* on page 31 for more information about `SQLConfigDataSource`.

## 2.5 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 routine is documented in *ODBC Supplementary Information* on page 31 and the Mimer specific parameters are documented in *Mimer Packaging Parameters* on page 23.

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.

### 2.5.1 Defining a Remote Database

The following sections document examples of how to define a remote database. For more examples, see *Example Configuration Program* on page 15. For more information on the parameters used, see *Mimer Packaging Parameters* on page 23.

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.

### 2.5.1.1 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.

### 2.5.1.2 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",
    "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 SQL 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.

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");
```

## 2.5.2 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 2K pages in the bufferpool.

```
rc = SQLConfigDataSource(NULL, ODBC_ADD_SYS_DSN, "MIMER",  
    "DATABASE=MyLocalMimer\0"  
    "DIRECTORY=C:\DB6\0"  
    "Pages2K=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 2K pages will be set to 1000 and all other parameters will be set to their default values.

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.

## 2.5.3 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* on page 10, it is possible that a data source was created even though the database definition was not. See *I\_InstError* on page 19 for more information on error handling.

## 2.6 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.

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 2K 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.

### 2.6.1 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.

## 2.7 Starting the Database Server

By now you have installed Mimer SQL and, probably created a local database with associated system databanks. The next step is to start the local database server. On Windows 98 and ME this is not necessary if the database has been configured for Autostart.

The following code works on Windows 98, ME, NT, 2000 and XP, so you may select to use the same logic in all of these 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");
```

## 2.8 More Useful Examples

This section contains a few more useful example calls to SQLConfigDataSource. For more examples, see *Example Configuration Program* on page 15.

### 2.8.1 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");
```

## 2.8.2 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.

## 2.8.3 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");
```

## 2.8.4 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");
```

## 2.8.5 Example 5

This example demonstrates how to remove an ODBC data source.

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

# 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.

## 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_SetupLicense();
    I_DefineDatabase();
    I_CreateSystemDatabanks();
    I_StartupDatabaseServer();
    I_CreateApplicationObjects();
    exit(0);
}
```

### 3.1.1 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:");
    }
}
```

### 3.1.2 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:");
    }
}
```

### 3.1.3 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:");
    }
}
```

### 3.1.4 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:");
    }
}
```

### 3.1.5 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.

### 3.1.6 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\n",
              iError,
              fErrorCode,
              szErrorMsg);
    }
}
```

The routine `SQLInstallerError` is further described in *SQLInstallerError* on page 43 and also in the Microsoft ODBC documentation.



# Chapter 4

# About Previous Mimer SQL Versions

---

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

Information about packaging in earlier versions of Mimer SQL is found in the *Mimer Packaging Guide for version 8.1.2*.

On Windows, Mimer SQL 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.

## 4.1 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 8.2 to 9.1, 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.
- If the new installation is a change of minor version, for example from version 9.1.3 to 9.1.4, the system will run a maintenance setup and install all the components that were previously installed, plus any additional components selected by the silent setup.

This method of upgrading means that the installation program does not know where the installation directory is.

- If the new installation is a change of minor revision, but to an older version, only components previously not 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 9.1 installed and trying to install 8.2), the installation fails. To do this, the user must first uninstall the new version and then redo the old installation.

## 4.2 Upgrading a Mimer SQL Database

When upgrading between major versions, for example from 7.3 or 8.2 to 9.1, 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.

Mimer SQL version 9.1 supports a number of new data types and other features, that can easily be brought into use through the `ALTER TABLE` statement.

See the *Mimer SQL Reference Manual* for further information about `ALTER TABLE`.

# Chapter 5

# Mimer Packaging Parameters

---

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

## 5.1 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 24.
- Local database parameters, see *Local Database Parameters* on page 24.
- Local database server commands, see *Local Database Server Commands* on page 28.
- Remote database parameters, see *Remote Database Parameters* on page 28.
- License key parameters, see *License Key Parameters* on page 29.

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.

## 5.1.1 Data Source Parameters

Data Source Parameter	Explanation
DSN	The ODBC data source name <b>(required)</b> . 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 <b>(required)</b> .

## 5.1.2 Local Database Parameters

Local Database Parameter	Explanation
Database	The name of the Mimer SQL database <b>(required)</b> .
Description	A text describing the database.
Directory	The database home directory <b>(required)</b> .
Users	Maximum number of simultaneous connections allowed to the database server <b>(recommended)</b> .
DBCcheck	The type of check performed after an improper shutdown of the system. All Pages is strongly recommended. Data pages are checked in the background. 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 = 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 from the default.

Local Database Parameter	Explanation
RequestPriority	<p>The priority class of the kernel threads. Should be an integer in the range 0-6.</p> <p>0 = Idle            1 = Lowest            2 = BelowNormal            3 = Normal            4 = AboveNormal            5 = Highest            6 = TimeCritical</p> <p>Further information about priorities may be found in the Win32 documentation for the routine SetThreadPriority.</p>
BackgroundThreads	<p>An integer specifying the number of background threads.</p> <p>If there are many large transactions and/or shadowing is used, the number of threads may need to be increased.</p> <p>Actually, it is more important to give the background threads sufficient priority rather than increase the number of threads.</p> <p>The number of background threads determine how many databases are checked in parallel. Half of the threads are used for this.</p>
BackgroundPriority	<p>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.</p>
StartupType	<p>An integer value which is specified if the server is to be disabled, started manually or started automatically after a system reboot.</p> <p>0 = Autostart            1 = Manual start            2 = Disabled</p> <p>On Windows 98/ME, Autostart means that the database is started whenever accessed by an application.</p>
AutoRestart	<p>An integer value which is specified if the server is to be started manually or started automatically after a failure.</p> <p>0 = Manual restart            1 = Automatic restart</p>

Local Database Parameter	Explanation
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.
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.

<b>Local Database Parameter</b>	<b>Explanation</b>
RmGuid	<p>This is a unique identifier identifying the database server used for inter-operating 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>

### 5.1.3 Local Database Server Commands

Local Database Server Command	Explanation
Database	The name of the Mimer SQL database <b>(required)</b> .
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 <i>Defining a Database</i> on page 9.
Upgrade	The parameters to the command is the SYSADM password and the name of the Mimer SQL database.  This is further described in <i>Upgrading a Mimer SQL Installation</i> on page 21.
DbServer	The DbServer command accepts the following strings: START, STOP, ENABLE and DISABLE.

### 5.1.4 Remote Database Parameters

Remote Database Parameter	Explanation
Database	The name of the Mimer SQL database on the remote host <b>(required)</b> .
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 <b>(required)</b> .
Protocol	Specifies whether to use named pipes or TCP/IP. Should either be the string 'NamedPipes' or 'tcp'. 'tcp' is the default.

Remote Database Parameter	Explanation
Service	<p>The IP port number of the server for TCP/IP. The default is 1360.</p> <p>For named pipes, this is the name of the pipe that the database server is waiting for incoming connections on.</p> <p>The default for version 8 and version 9 servers is to listen on a pipe with the same name as the database. In previous versions, the default was MIMER. This means that you must know if the server is a version 7 or version 8 or 9 database server. The default for version 8 and 9 clients is to use the database name.</p>
Interface	Not used on the Windows platform.

### 5.1.5 License Key Parameters

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



# Chapter 6

# ODBC

# Supplementary

# Information

---

This chapter contains information about ODBC. You can also refer to the ODBC help files for complete information about the available interfaces for ODBC configuration.

## 6.1 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.

<i>*pfErrorCode</i>	<b>Error</b>	<b>Description</b>
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 <code>ConfigDSN</code> 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_MEM	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`.

## 6.2 SQLConfigDriver

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

### Summary

`SQLConfigDriver` loads the appropriate driver setup DLL and calls the `ConfigDriver` function.

### Syntax

```
BOOL SQLConfigDriver (
    HWND hwndParent,
    WORD fRequest,
    LPCSTR lpszDriver,
    LPCSTR lpszArgs,
    LPSTR lpszMsg,
    WORD cbMsgMax,
    WORD* pcbMsgOut);
```

### 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_CONFIG_DRIVER`: Changes the connection pooling timeout used by the driver.

`ODBC_INSTALL_DRIVER`: Installs a new driver.

`ODBC_REMOVE_DRIVER`: Removes an existing driver.

This option can also be driver-specific, in which case the `fRequest` for the first option must start from `ODBC_CONFIG_DRIVER_MAX+1`. The `fRequest` for any additional option must also start from a value greater than `ODBC_CONFIG_DRIVER_MAX+1`.

`lpszDriver` [Input]

The name of the driver as registered in the system information.

*lpszArgs* [Input]

A null-terminated string containing arguments for a driver-specific *fRequest*.

*lpszMsg* [Output]

A null-terminated string containing an output message from the driver setup.

*cbMsgMax* [Input]

Length of *lpszMsg*.

*pcbMsgOut* [Output]

Total number of bytes available to return in *lpszMsg*. If the number of bytes available to return is greater than or equal to *cbMsgMax*, the output message in *lpszMsg* is truncated to *cbMsgMax* minus the null-termination character. The *pcbMsgOut* argument can be a null pointer.

### Returns

The function returns TRUE if it is successful, FALSE if it fails.

### Diagnostics

When `SQLConfigDriver` 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>	<b>Error</b>	<b>Description</b>
ODBC_ERROR_GENERAL_ERR	General installer error	An error occurred for which there was no specific installer error.
ODBC_ERROR_INVALID_BUFFER_LEN	Invalid buffer length	The <i>lpszMsg</i> argument was invalid.
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_INSTALL_DRIVER ODBC_REMOVE_DRIVER  The <i>fRequest</i> argument was a driver-specific option that was less than or equal to ODBC_CONFIG_DRIVER_MAX.
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 <i>ConfigDriver</i> 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_MEM	Out of memory	The installer could not perform the function because of a lack of memory.

### Comments

*SQLConfigDriver* allows an application to call a driver's *ConfigDriver* routine without having to know the name and load the driver-specific setup DLL.

A setup program calls this function after the driver setup DLL has been installed. The calling program should be aware that this function may not be available for all drivers. In such a case, the calling program should continue without error.

### Driver-Specific Options

An application can request driver-specific features exposed by the driver by using the *fRequest* argument. The *fRequest* for the first option will be `ODBC_CONFIG_DRIVER_MAX+1`, and additional options will be incremented by 1 from that value.

Any arguments required by the driver for that function should be provided in a null-terminated string passed in the *lpzArgs* argument. Drivers providing such functionality should maintain a table of driver-specific options. The options should be fully documented in driver documentation. Application writers who make use of driver-specific options should be aware that this use will make the application less interoperable.

### Setting Connection Pooling Timeout

Connection pool timeout properties can be set when setting the configuration of the driver. `SQLConfigDriver` is called with an *fRequest* of `ODBC_CONFIG_DRIVER` and *lpzArgs* set to `CPTimeout`.

`CPTimeout` determines the amount of time that a connection can remain in the connection pool without being used. When the timeout expires, the connection is closed and removed from the pool. The default timeout is 60 seconds.

When `SQLConfigDriver` is called with *fRequest* set to `SQL_INSTALL_DRIVER` or `SQL_REMOVE_DRIVER`, the Driver Manager loads the appropriate driver setup DLL and calls the `ConfigDriver` function. When `SQLConfigDriver` is called with an *fRequest* of `ODBC_CONFIG_DRIVER`, all processing is performed in the ODBC installer, so the driver setup DLL does not need to be loaded.

### Messages

A driver setup routine can send a text message to an application as null-terminated strings in the *lpzMsg* buffer. The message will be truncated to `cbMsgMax` minus the null-termination character by the `ConfigDriver` function if it is greater than or equal to `cbMsgMax` characters.

## 6.3 SQLInstallDriverEx

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

### Summary

`SQLInstallDriverEx` adds information about the driver to the `ODBCINST.INI` entry in the system information and increments the driver's `UsageCount` by 1.

However, if a version of the driver already exists, but the *UsageCount* value for the driver does not exist, the new *UsageCount* value is set to 2.

This function does not actually copy any files. It is the responsibility of the calling program to copy the driver's files to the target directory properly.

## Syntax

```
BOOL SQLInstallDriverEx(
    LPCSTR lpszDriver,
    LPCSTR lpszPathIn,
    LPSTR lpszPathOut,
    WORD cbPathOutMax,
    WORD * pcbPathOut,
    WORD fRequest,
    LPDWORD lpdwUsageCount);
```

## Arguments

*lpszDriver* [Input]

The driver description (usually the name of the associated DBMS) presented to users instead of the physical driver name. The *lpszDriver* argument must contain a list of keyword-value pairs describing the driver.

For more information, see *Comments* on page 40.

*lpszPathIn* [Input]

Full path of the target directory of the installation, or a null pointer. If *lpszPathIn* is a null pointer, the drivers will be installed in the System directory.

*lpszPathOut* [Output]

Path of the target directory where the driver should be installed. If the driver has not previously been installed then *lpszPathOut* should be the same as *lpszPathIn*. If the driver was previously installed then *lpszPathOut* is the path of the previous installation.

*cbPathOutMax* [Input]

Length of *lpszPathOut*.

*pcbPathOut* [Output]

Total number of bytes (excluding the null-termination character) available to return in *lpszPathOut*. If the number of bytes available to return is greater than or equal to *cbPathOutMax* then the output path in *lpszPathOut* is truncated to *cbPathOutMax* minus the null-termination character. The *pcbPathOut* argument can be a null pointer.

*fRequest* [Input]

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

ODBC\_INSTALL\_INQUIRY: Inquire about where a driver can be installed.

ODBC\_INSTALL\_COMPLETE: Complete the installation request.

*lpdwUsageCount* [Output]

The usage count of the driver after this function has been called.

### Returns

The function returns TRUE if it is successful, FALSE if it fails.

### Diagnostics

When `SQLInstallDriverEx` 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>	<b>Error</b>	<b>Description</b>
ODBC_ERROR_GENERAL_ERR	General installer error	An error occurred for which there was no specific installer error.
ODBC_ERROR_INVALID_BUFF_LEN	Invalid buffer length	The <i>lpzPathOut</i> argument was not large enough to contain the output path. The buffer contains the truncated path. The <i>cbPathOutMax</i> argument was 0 and <i>fRequest</i> was ODBC_INSTALL_COMPLETE.
ODBC_ERROR_INVALID_REQUEST_TYPE	Invalid type of request	The <i>fRequest</i> argument was not one of the following: ODBC_INSTALL_INQUERY ODBC_INSTALL_COMPLETE
ODBC_ERROR_INVALID_KEYWORD_VALUE	Invalid keyword-value pairs	The <i>lpzDriver</i> argument contained a syntax error.
ODBC_ERROR_INVALID_PATH	Invalid install path	The <i>lpzPathIn</i> argument contained an invalid path.
ODBC_ERROR_LOAD_LIBRARY_FAILED	Could not load the driver or translator setup library	The driver setup library could not be loaded.
ODBC_ERROR_INVALID_PARAM_SEQUENCE	Invalid parameter sequence	The <i>lpzDriver</i> argument did not contain a list of keyword value pairs.
ODBC_ERROR_USAGE_UPDATE_FAILED	Could not increment or decrement the component usage count	The installer failed to increment the driver's usage count.

## Comments

The *lpszDriver* argument is a list of attributes in the form of keyword-value pairs. Each pair is terminated with a null byte and the entire list is terminated with a null byte (that is, two null bytes mark the end of the list).

The format of this list is:

```
driver-desc\0Driver=driver-DLL-filename\0 [Setup=setup-DLL-filename\0]
[driver-attr-keyword1=value1\0] [driver-attr-keyword2=value2\0]... \0
```

where \0 is a null byte and *driver-attr-keywordn* is any driver attribute keyword described in the table that follows:

<i>Keyword</i>	<b>Usage</b>
<i>APILevel</i>	<p>A number indicating the ODBC interface conformance level supported by the driver:</p> <p>0 = None            1 = Level 1 supported            2 = Level 2 supported</p> <p>This must be the same as the value returned for the <code>SQL_ODBC_INTERFACE_CONFORMANCE</code> option in <code>SQLGetInfo</code>.</p>
<i>CreateDSN</i>	<p>The name of one or more data sources to be created when the driver is installed.</p> <p>The system information must include one data source specification section for each data source listed with the <i>CreateDSN</i> keyword.</p> <p>These sections should not include the <i>Driver</i> keyword, since this is specified in the driver specification section, but must include enough information for the <code>ConfigDSN</code> function in the driver setup DLL to create a remote data source specification without displaying any dialog boxes. For the format of a data source specification section, see <i>Remote Database Parameters</i> on page 28.</p>

<i>Keyword</i>	<b>Usage</b>
<i>ConnectFunctions</i>	<p>A three-character string indicating whether the driver supports <code>SQLConnect</code>, <code>SQLDriverConnect</code>, and <code>SQLBrowseConnect</code>.</p> <p>If the driver supports <code>SQLConnect</code>, the first character is 'Y'; otherwise, it is 'N'.</p> <p>If the driver supports <code>SQLDriverConnect</code>, the second character is 'Y'; otherwise, it is 'N'.</p> <p>If the driver supports <code>SQLBrowseConnect</code>, the third character is 'Y'; otherwise, it is 'N'.</p> <p>For example, if a driver supports <code>SQLConnect</code> and <code>SQLDriverConnect</code>, but not <code>SQLBrowseConnect</code>, the three-character string is 'YYN'.</p>
<i>DriverODBCVer</i>	<p>A character string with the version of ODBC that the driver supports.</p> <p>The version is of the form <code>nn.nn</code>, where the first two digits are the major version and the next two digits are the minor version.</p> <p>For the version of ODBC described in this manual, the driver must return '03.00'. This must be the same as the value returned for the <code>SQL_DRIVER_ODBC_VER</code> option in <code>SQLGetInfo</code>.</p>
<i>FileExtns</i>	<p>For file-based drivers, a comma-separated list of extensions of the files the driver can use. See the <i>FileUsage</i> keyword below.</p>
<i>FileUsage</i>	<p>A number indicating how a file-based driver directly treats files in a data source.</p> <p>0 = The driver is not a file-based driver. For example, a Mimer SQL driver is a DBMS-based driver.</p> <p>1 = A file-based driver treats files in a data source as tables. For example, an Xbase driver treats each Xbase file as a table.</p> <p>2 = A file-based driver treats files in a data source as a catalog. For example, a Microsoft Access driver treats each Microsoft Access file as a complete database.</p>
<i>SQLLevel</i>	<p>A number indicating the SQL-92 grammar supported by the driver:</p> <p>0 = SQL-92 Entry  1 = FIPS127-2 Transitional  2 = SQL-92 Intermediate  3 = SQL-92 Full</p> <p>This must be the same as the value returned for the <code>SQL_SQL_CONFORMANCE</code> option in <code>SQLGetInfo</code>.</p>

The keywords must appear in the specified order.

### Mimer SQL Example

A typical Mimer SQL value for this argument is the following string (new lines added for readability):

```
MIMER\0
Driver=Mimodbcw.dll\0
Setup=Mimsetw.dll\0
SQLLevel=1\0
FileUsage=0\0
DriverODBCVer=03.51\0
ConnectFunctions=YYY\0
APILevel=2\0
CPTimeout=60\0\0
```

After `SQLInstallDriverEx` retrieves information about the driver from the `lpzDriver` argument, it adds the driver description to the [ODBC Drivers] section of the ODBCINST.INI entry in the system information. It then creates a section titled with the driver's description and adds the full paths of the driver DLL and the setup DLL. Finally, it returns the path of the target directory of the installation but does not copy the driver files to it. The calling program must actually copy the driver files to the target directory.

`SQLInstallDriverEx` increments the component usage count for the installed driver by 1. If a version of the driver already exists, but the component usage count for the driver does not exist, the new component usage count value is set to 2.

The application setup program is responsible for physically copying the driver file, and maintaining the file usage count. If the driver file has not previously been installed, the application setup program must copy the file in the `lpzPathIn` path, and create the file usage count. If the file has previously been installed, the setup program merely increments the file usage count, and returns the path of the prior installation in the `lpzPathOut` argument.

If an older version of the driver file was previously installed by the application, the driver should be uninstalled, then reinstalled, so that the driver component usage count is valid. `SQLConfigDriver` (with an `fRequest` of `ODBC_REMOVE_DRIVER`) should first be called, then `SQLRemoveDriver` should be called to decrement the component usage count.

`SQLInstallDriverEx` should then be called to reinstall the driver, incrementing the component usage count. The application setup program must physically replace the old file with the new file. The file usage count will remain the same, and any other application that used the older version file will now use the newer version.

**Note:** If the driver was previously installed, and `SQLInstallDriverEx` is called to install the driver in a different directory, the function will return `TRUE`, but `lpzPathOut` will include the directory where the driver was already installed. It will not include the directory entered in the `lpzDriver` argument.

The length of the path in *lpszPathOut* in `SQLInstallDriverEx` allows for a two-phase install process, so an application can determine what *cbPathOutMax* should be by calling `SQLInstallDriverEx` with an *fRequest* of `ODBC_INSTALL_INQUIRY` mode. This will return the total number of bytes available in the *pcbPathOut* buffer.

`SQLInstallDriverEx` can then be called with an *fRequest* of `ODBC_INSTALL_COMPLETE` and the *cbPathOutMax* argument set to the value in the *pcbPathOut* buffer, plus the null-termination character.

If you choose not to use the two-phase model for `SQLInstallDriverEx` then you must set *cbPathOutMax*, which defines the size of the storage for the path of the target directory, to the value `_MAX_PATH`, as defined in `STDLIB.H`, to prevent truncation.

When *fRequest* is `ODBC_INSTALL_COMPLETE`, `SQLInstallDriverEx` does not allow *lpszPathOut* to be `NULL` (or *cbPathOutMax* to be 0). If *fRequest* is `ODBC_INSTALL_COMPLETE`, `FALSE` is returned when the number of bytes available to return is greater than or equal to *cbPathOutMax*, with the result that truncation occurs.

After `SQLInstallDriverEx` has been called, and the application setup program has copied the driver file (if necessary), the driver setup DLL must call `SQLConfigDriver` to set the configuration for the driver.

## 6.4 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 lpszErrorMsg,  
    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* on page 44.

*lpszErrorMsg* [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 *lpszErrorMsg*.

If the number of bytes available to return is greater than or equal to *cbErrorMsgMax*, the error message text in *lpszErrorMsg* 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`, it returns `SQL_ERROR`, but does not post any error values.

If either the *lpszErrorMsg* argument is `NULL`, or 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).

## 6.5 SQLRemoveDriver

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

### Summary

`SQLRemoveDriver` changes or removes information about the driver from the `ODBCINST.INI` entry in the system information.

### Syntax

```
BOOL SQLRemoveDriver (
    LPCSTR lpszDriver,
    BOOL fRemoveDSN,
    LPDWORD lpdwUsageCount);
```

### Arguments

*lpszDriver* [Input]

The name of the driver as registered in the `ODBCINST.INI` key of the system information.

*fRemoveDSN* [Input]

The valid values are:

- **TRUE:**  
Remove DSNs associated with the driver specified in *lpszDriver*.
- **FALSE:**  
Do not remove DSNs associated with the driver specified in *lpszDriver*.

*lpdwUsageCount* [Output]

The usage count of the driver after this function has been called.

### 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 `SQLRemoveDriver` 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.

<code>*pfErrorCode</code>	<b>Error</b>	<b>Description</b>
<code>ODBC_ERROR_GENERAL_ERR</code>	General installer error	An error occurred for which there was no specific installer error.
<code>ODBC_ERROR_COMPONENT_NOT_FOUND</code>	Component not found in registry	The installer could not remove the driver information because it either did not exist in the registry or could not be found in the registry.
<code>ODBC_ERROR_INVALID_NAME</code>	Invalid driver or translator name	The <code>lpszDriver</code> argument was invalid.
<code>ODBC_ERROR_USAGE_UPDATE_FAILED</code>	Could not increment or decrement the component usage count	The installer failed to decrement the usage count of the driver.
<code>ODBC_ERROR_REQUEST_FAILED</code>	Request failed	The <code>fRemoveDSN</code> argument was <code>TRUE</code> ; however, one or more DSNs could not be removed. The call to <code>SQLConfigDriver</code> with the <code>ODBC_REMOVE_DRIVER</code> request failed.
<code>ODBC_ERROR_OUT_OF_MEM</code>	Out of memory	The installer could not perform the function because of a lack of memory.

## Comments

`SQLRemoveDriver` complements the `SQLInstallDriverEx` function, and updates the component usage count in the system information. This function should only be called from a setup application.

`SQLRemoveDriver` will decrement the component usage count value by 1. If the component usage count goes to 0 then the following will occur:

- The `SQLConfigDriver` function with the `ODBC_REMOVE_DRIVER` option will be called. If the `fRemoveDSN` option is set to `TRUE`, the `ConfigDSN` function calls `SQLRemoveDSNFromIni` to remove all the data sources associated with the driver specified in `lpszDriver`. If the `fRemoveDSN` option is set to `FALSE`, the data sources will not be deleted.

- The driver entry in the system information will be removed. The driver entry is in the following system information location, under the driver name:

```
HKEY_LOCAL_MACHINE
SOFTWARE
ODBC
ODBCINST.INI
```

`SQLRemoveDriver` does not actually remove any files. The calling program is responsible for deleting files, and maintaining the file usage count. Only after both the component usage count and the file usage count have reached zero is a file physically deleted. Some files in a component can be deleted, and others not deleted, depending upon whether the files are used by other applications that have incremented the file usage count.

`SQLRemoveDriver` is also called as part of an upgrade process. If an application detects that it has to perform an upgrade, and it has previously installed the driver, then the driver should be removed, then reinstalled.

`SQLRemoveDriver` should first be called to decrement the component usage count, then `SQLInstallDriverEx` should then be called to increment the component usage count. The application setup program must physically replace the old files with the new files.

The file usage count will remain the same, and other applications that use the older version files will now use the newer version.



# Chapter 7

# InstallShield 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.

## 7.1 setup.exe

setup.exe is the main setup executable; it performs setup initialization and launches the appropriate setup engine file on the target system.

Once you have built your setup, you can rename setup.exe to any valid file name, such as install.exe.

### Syntax

```
Setup [switches]
```

### Switches

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

#### **/c and -c –**

#### **Examples:**

```
/c<path\RebootFile>
```

or

```
-c<path\RebootFile>
```

This is a Mimer SQL specific switch which specifies location and name of a reboot indicator log file.

“RebootNeeded=1” is inserted into the file if reboot is required to complete the installation.

“RebootNeeded=0” is inserted into the file if reboot is not required.

It is recommended that the reboot indicator log file and the log file are the same file.

**/f and -f****Examples:**

```
/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 and -f1****Examples:**

```
/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 and -f2****Examples:**

```
/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****Examples:**

```
/r
```

or

```
-r
```

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

**/s or -s****Examples:**`/s``or``-s`

Runs InstallShield Silent to execute a silent setup.

**/z or -z****Examples:**`/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/fInstall.ins`.
- When using long path and file name 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.

Error messages are displayed in message boxes. Every error message has a number. These are InstallShield system error messages and there is no way to suppress them in your script.

## 7.1.1 setup.exe Error Codes

The following table explains setup.exe error codes:

Error No.	Message
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.

<b>Error No.</b>	<b>Message</b>
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 the uninstaller is in use. Please close the uninstaller and restart setup.
440	Setup has detected a possible infinite loop in the script with function function_name. Make sure you are handling the error return 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.

## 7.1.2 Setup Runtime Errors

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

Code	Description
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.

## 7.1.3 Component Error Codes

The following table describes the error codes returned by ComponentError:

Code	Description	Cause
-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.
-104	Specified component name is not valid.	The value passed in the second parameter of ComponentInitialize is not valid.

Code	Description	Cause
-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 re-initialized. To re-initialize 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 cannot 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 or the path to the target folder is invalid.
-117	Cannot read the specified file.	A data cab file or an uncompressed data file may be corrupt.

Code	Description	Cause
-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 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 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 re-initialized. If your script calls ComponentMoveData more than once, you must re-initialize it after each call by calling it again with a null string in its first parameter.

Code	Description	Cause
-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 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.

Code	Description	Cause
-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.
-620	Error with third-party shell.	Consult the documentation for the shell in question to determine whether it is 100% Explorer shell compatible.
-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.