

# L3 TECHNOLOGIES

---

## Paradox File Driver

Copyright © 1998, L3 Technologies, Ltd. All rights reserved.  
<http://www.l3.u-net.com>

# CONTENTS

<b>INTRODUCTION.....</b>	<b>3</b>
<b>LICENSE AGREEMENT .....</b>	<b>4</b>
<b>INSTALLATION &amp; CONFIGURATION.....</b>	<b>7</b>
<b>REGISTERING &amp; UNLOCKING THE PARADOX FILE DRIVER .....</b>	<b>7</b>
<b>LICENSING THE BORLAND DATABASE ENGINE .....</b>	<b>7</b>
<b>GENERAL CHARACTERISTICS .....</b>	<b>8</b>
<b>FILE SPECIFICATIONS/MAXIMUMS.....</b>	<b>8</b>
<b>SUPPORTED DATA TYPES .....</b>	<b>9</b>
PARADOX DATA TYPES .....	9
BLOB TYPES .....	9
<b>DRIVER STRINGS AND SEND FUNCTIONS.....</b>	<b>11</b>
ABOUT .....	11
ALIAS .....	11
BDETRACE.....	11
CLIENTNAME.....	11
DBHANDLE .....	11
DBOPENMODE.....	12
DBSHAREMODE.....	12
INIFILE .....	12
LOCALINIT.....	12
POSITIONSIZE .....	12
PRIVATEDIR .....	12
QBE .....	13
SEQCURSOR .....	13
SQL .....	13
STRICT .....	13
TABLECURSOR .....	13
WORKDIR.....	13
<b>SQL SUPPORT.....</b>	<b>14</b>
CURSORS .....	14
<b>ERROR HANDLING.....</b>	<b>16</b>
<b>UTILITIES AND EXAMPLES .....</b>	<b>17</b>
PDXFIX .....	17
DBINFO.....	17
PDX2TXD.....	17
CWPD .....	17
<b>UNSUPPORTED OR MODIFIED FUNCTIONS &amp; ATTRIBUTES.....</b>	<b>18</b>
<b>LIMITATIONS.....</b>	<b>19</b>
<b>ACKNOWLEDGEMENTS.....</b>	<b>20</b>

## Introduction

The Paradox File Driver is one of the most sophisticated Clarion File Drivers yet developed.

The Paradox File Driver provides full native access to Paradox tables from Clarion for Windows applications. Paradox table access is achieved using the Borland Database Engine (BDE). This is the same engine that is used in the Paradox Database, so full compatibility is ensured.

A copy of the BDE Redistributables is included. If you develop and distribute software using the Paradox File Driver, you must redistribute this same copy of the BDE subject to certain restrictions (see Licensing the Borland Database Engine).

The Paradox File Driver is written using the Clarion File Driver Kit, ensuring complete compatibility with Clarion for Windows. In addition the driver incorporates new technology that enables applications to dynamically create, open and access Paradox tables without defining the file structure at compile time. This enables developers to produce general purpose applications that can operate on any Paradox table.

Full support for Local SQL is provided through the use of the Prop:SQL file property. Support for Query By Example (QBE) is provided by a Prop:QBE property. In addition a special form of file declaration, is supported which allows the efficient use of SQL style cursors in applications.

An extended error handling mechanism is incorporated, which provides access to the BDE error information as well as posting the appropriate Clarion error code. In addition there are a number of driver specific errors that are used to pinpoint specific error conditions during application development.

The Paradox Driver offers an unparalleled range of features and functionality which ease the development of applications previously impossible with conventional file drivers.

## **License Agreement**

This license agreement is a legal agreement between you (either an individual or a single entity) and L3 Technologies Ltd. for the software product identified above, which includes computer software and may include associated media, printed materials, and "online" or electronic documentation ("SOFTWARE PRODUCT"). By installing, copying, or otherwise using the SOFTWARE PRODUCT, you agree to be bound by the terms of this license agreement. If you do not agree to the terms of this license agreement, do not install or use the SOFTWARE PRODUCT; you may, however, return it to your place of purchase for a refund.

### **SOFTWARE PRODUCT LICENSE**

The SOFTWARE PRODUCT is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The SOFTWARE PRODUCT is licensed, not sold.

L3 Technologies Ltd. grants the following rights to the licensed, registered user of the SOFTWARE product, subject to all of the conditions in this license agreement:

You may install one copy of this SOFTWARE PRODUCT for use. You may make archive copies of the original distribution files for the sole purpose of backing up the SOFTWARE PRODUCT and safeguarding your investment. The SOFTWARE PRODUCT may be used by any number of people, and may be freely moved from one computer or location to another so long as there is no possibility of it being used by more than one person at a time. The SOFTWARE PRODUCT is licensed as a single product, except as provided below, its component parts may not be separated for use by more than one person at a time.

The SOFTWARE PRODUCT may include files intended for distribution by you to users of the programs you create. These files are specifically designated as "Redistributable" in the "readme" file for the SOFTWARE PRODUCT ("REDISTRIBUTABLES"). You are authorized to reproduce and distribute exact copies of the REDISTRIBUTABLES for the sole purpose of executing application programs that you have developed using the SOFTWARE PRODUCT. Under no circumstances may you distribute any copies of the REDISTRIBUTABLES separately.

You may not alter or remove any copyright or trademark notices or identifying screens contained in any of the libraries, source code or REDISTRIBUTABLES comprising the SOFTWARE PRODUCT.

Regardless of any modifications that you make, you may not distribute any files (particularly source code and other non-executable files) except REDISTRIBUTABLES.

You will remain solely responsible to anyone receiving your programs for support, service, upgrades, technical or other assistance, and such recipients will have no right to contact L3 Technologies for such services or assistance.

You will indemnify and hold L3 Technologies, and its suppliers harmless from and against any claims or liabilities arising out of the use, reproduction, or distribution of your programs.

If you have purchased an upgrade version of the SOFTWARE PRODUCT, it constitutes a single product with the SOFTWARE PRODUCT that you upgraded.

The upgrade and the SOFTWARE PRODUCT cannot both be available for use by two different people at the same time, and cannot be transferred separately.

#### **TITLE**

All L3 Technologies intellectual property rights, libraries, source code, REDISTRIBUTABLES and other files comprising the SOFTWARE PRODUCT remain the exclusive property of L3 Technologies Ltd. or its suppliers.

#### **TRANSFER OF LICENSE**

You may permanently transfer all of your rights under this license agreement, provided the recipient agrees to the terms of this license agreement, and you transfer all software, media and documentation comprising the SOFTWARE PRODUCT, and you transfer or destroy all copies in any form.

#### **TERMINATION OF LICENSE**

Without prejudice to any other rights, L3 Technologies may terminate this license agreement if you fail to comply with the terms and conditions of this license agreement. In such event, you must destroy all copies of the SOFTWARE PRODUCT and all of its component parts.

#### **LIMITED WARRANTY**

L3 Technologies Ltd. warrants the physical media and physical documentation provided by L3 Technologies Ltd. to be free from defects in materials and workmanship for a period not exceeding 30 days from the original purchase date. If L3 Technologies Ltd. receives notification within the warranty period of defects in materials or workmanship, and determines that such notification is correct, L3 Technologies Ltd. will replace the defective media or documentation.

#### **DISCLAIMER OF WARRANTY**

L3 Technologies Ltd. expressly disclaims any warranty for the SOFTWARE PRODUCT. The SOFTWARE PRODUCT and any related documentation is provided "as is" without warranty, representation or condition of any kind, either express or implied, including, without limitation, any implied warranty or condition of merchantability, fitness for a particular purpose, or noninfringement.

Specifically, L3 Technologies Ltd. makes no representation or warranty that the SOFTWARE PRODUCT, or any components of the SOFTWARE PRODUCT, including any related documentation are "error free" or meet any user's particular standards requirements or needs. In all events, any implied warranty, representation, condition or other term is limited to the physical media or documentation and is limited to the 30-day duration of the limited warranty.

The entire risk arising out of use or performance of the SOFTWARE PRODUCT remains with you.

In no event shall L3 Technologies Ltd. or its suppliers be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information or data, or any other pecuniary loss) arising out of the use of or inability to use this SOFTWARE PRODUCT, even if L3 Technologies Ltd. has been advised of the possibility of such damages.

L3 Technologies Ltd. is not responsible for, and does not make any representation, warranty, or condition concerning product, media, software, or documentation not manufactured by L3 Technologies Ltd. such as third parties' programs which are developed using L3 Technologies software or which include L3 Technologies programs, library code or files.

**HIGH RISK ACTIVITIES**

The SOFTWARE PRODUCT is not fault-tolerant and is not designed, manufactured or intended for use or resale as on-line control equipment in hazardous environments requiring fail-safe performance, such as (but not limited to) the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines, or weapons systems, in which the failure of the Software could lead directly to death, personal injury, or severe physical or environmental damage ("High Risk Activities"). L3 Technologies Ltd. and its suppliers specifically disclaim any express or implied warranty of fitness for High Risk Activities.

**MISCELLANEOUS**

This license agreement is governed by the laws of the United Kingdom.

## Installation & Configuration

The Paradox File Driver SETUP program will install the driver files, utilities, examples and documentation into your Clarion directory structure. A program group will be created to provide easy access to the documentation and utilities.

Before the Paradox File Driver can be accessed from the Clarion IDE, it is necessary to install it in the Database Driver Registry. To do this select the **Database Driver Registry...** item on the **Setup** menu, click on the **Add** button and choose the appropriate 16-bit Paradox File Driver DLL for the version of Clarion you are running:

<b>Clarion for Windows 2.0</b>	CW2PDX16.DLL
<b>Clarion 4.0</b>	C4PDX.DLL

It may be necessary to close the IDE and restart in order for the changes to take effect.

A number of environmental options may be configured using the configuration utility installed with the Borland Database Engine (BDE). The utility may be named BDECFG.EXE, BDECFG32.EXE or BDEADMIN.EXE depending on the version of the BDE you have installed. Consult the on-line help for this utility for additional information.

## Registering & Unlocking The Paradox File Driver

By unlocking the Paradox File Driver you will disable the splash screen and enable the driver to be distributed as part of your application. Before you can unlock the driver you must purchase a key from one of the dealers listed on the L3 Technologies web site (<http://www.l3.u-net.com>). You must then use the L3UNLOCK.EXE application supplied with the driver to apply the key. Simply load the application from Explorer or the File Manager and follow the on-screen instructions.

## Licensing the Borland Database Engine

Any applications developed using the Paradox File Driver must include the Borland Database Engine (BDE) in the distribution.

The Borland Database Engine (BDE) supplied with the Paradox Driver must be shipped together with the Paradox Table Repair utility (PDXFIX16.EXE or PDXFIX32.EXE) and/or the DBInfo utility (DBINFO16.EXE or DBINFO32.EXE) also supplied. This is to satisfy the Borland License Agreement, which states that the BDE may only be shipped with applications that are developed with a Borland development tool. The Paradox Table Repair utility and DBInfo are developed with Borland C++.

Full details of precisely which files need to be redistributed with your applications are included in the accompanying README.TXT file.

## General Characteristics

The Paradox File Driver supports Paradox versions 3.5 - 5.0. The default data file extension is .DB.

Keys and indexes exist separately from the main data file, keys are supported only if the table has a primary key. Primary keys have a .PX extension, whereas indexes and keys result in a pair of files with the extensions .X?? and .Y?? If the table has a primary key, all keys and indexes are maintained, otherwise only non-maintained indexes are supported.

Paradox supports only fixed length records. Memos are not supported, however BLOBs are supported for Paradox 4 and above. BLOBs are stored in a separate file with the .MB extension if they exceed the size of the blob leader in the data file.

## File Specifications/Maximums

File Size:	Limited only by disk space
Records per File:	4G
Record Size:	1350-32750 depending on type and version
Field Size:	255 (BLOBS 4GB)
Fields per Record:	255
Keys/Indexes per File:	255
Key Size:	255
Memo fields per File:	0
Memo Field Size:	0
BLOB fields per File:	255
BLOB Size:	4GB
Open Data Files:	Operating system dependent

## Supported Data Types

Some Clarion data types represent more than one Paradox type, for this reason it is necessary to augment the field definition to specify the Paradox type. Any field type may be augmented, though it is only necessary to augment fields used as BLOB leaders, type TimeStamp or AutoInc.

Augmented field definitions take the form:

```
FieldLabel ClarionType,NAME('FieldLabel->ParadoxType')
```

Where ClarionType is an appropriate supported Clarion data type, and ParadoxType is the letter normally used within Paradox to designate a given type. The following tables detail the Clarion data type equivalencies for Paradox types.

### Paradox Data Types

Alphanumeric	A	STRING(n)
Short	S	SHORT
Numeric	N	REAL
Money	\$	REAL, NAME('Field->\$') BFLOAT8
Timestamp	@	REAL, NAME('Field->@')
BCD	#	DECIMAL
Date	D	DATE
Time	T	Time
AutoInc	+	LONG, NAME('Field->+')
Logical	L	BYTE

### BLOB Types

Bytes	Y	STRING(0-240), NAME('Blob->Yn')
Memo	M	STRING(0-240), NAME('Blob->Mn')
Binary	B	STRING(0-240), NAME('Blob->Bn')
Formatted Memo	F	STRING(0-240), NAME('Blob->Fn')
OLE	O	STRING(0-240), NAME('Blob->On')
Graphic	G	STRING(0-240), NAME('Blob->G')

The Paradox Money data type is supported as Clarion type REAL for backward compatibility with the earlier Clarion File Driver for Paradox, though BFLOAT8 is the default type mapping.

Auto-incremented longs are read-only, the driver averts any attempts to update the value. The BDE would give an error on any update attempt, the driver circumvents this process to allow CLEAR() to be called for a record without restriction.

When viewed using the Clarion Database Browser, some fields with NULL values will display as a series of '#' characters.

Timestamps are implemented as REAL but may be decoded using the following group structure that also allows timestamps to be compared:

```
TimeStamp      GROUP,TYPE
T              GROUP
year           SHORT      ! -9999 - 9999
month          BYTE       !    1 - 12
day            BYTE       !    1 - 31
hours          BYTE       !    0 - 23
minutes        BYTE       !    0 - 59
milsecs        USHORT     !    0 - 59999
              END
R              REAL, OVER(T)
              END
```

## Driver Strings and SEND functions

All send commands listed below may be used as driver strings (the second parameter in the DRIVER attribute on a FILE declaration) with the exception of 'DBHANDLE', 'POSITIONSIZE', 'PXTYPE', 'SEQCURSOR' and 'TABLECURSOR'. Driver strings may be specified in one of two ways, comma separated:

```
f FILE, DRIVER('Paradox', 'ALIAS=MyTables, CLIENTNAME=PSW')
...
```

Preceded by '/':

```
f FILE, DRIVER('Paradox', '/ALIAS=MyTables /CLIENTNAME=PSW')
...
```

### ABOUT

```
SEND(f, 'ABOUT')
```

The ABOUT command displays the drivers' 'About' screen. This lists the name and versions of the driver, and the details of the registered developer supplied when unlocking the driver. If the contact details are prefixed with mailto: (for an email address) or http: (for a web site address) then this will present an active link which, when clicked, will load the appropriate application using the ShellExecute() API. The L3 button acts as an active link to the L3 Technologies web site.

### ALIAS

```
s" = SEND(f, 'ALIAS=dbalias')          f{PROP:Alias} = 'dbalias'
s" = SEND(f, 'ALIAS')                  s" = f{PROP:Alias}
```

The ALIAS command or property is used to reference the Paradox database alias. The alias must be specified before the table is opened. The Paradox database alias is used to group sets of related tables into a specific location. The alias is a configuration setting that is made using the IDAPICFG.EXE program.

### BDETRACE

```
s" = SEND(f, 'BDETRACE=error.log')    f{PROP:BDETrace} = 'error.log'
```

The BDETRACE command or property is used to specify a trace file into which BDE trace information is written. This command has no effect unless the debugging version of the IDAPI interface is enabled. Consult the IDAPI help for advice on installing the debugging interface.

### CLIENTNAME

```
s" = SEND(f, 'CLIENTNAME=client')     f{PROP:ClientName} = 'client'
s" = SEND(f, 'CLIENTNAME')            s" = f{PROP:ClientName}
```

The CLIENTNAME command or property is used to reference the client name for the current session. The client name must be specified before the table is opened. The client name is used for documentary purposes only (e.g. identifying the session that has locked a file).

### DBHANDLE

```
h# = SEND(f, 'DBHANDLE')              h# = f{PROP:DbHandle}
```

The DBHANDLE command or property is used to obtain the table's database handle which may then be used when calling the IDAPI functions directly from Clarion. Care must be taken when calling the IDAPI directly, since this can affect file processing by the driver.

### **DBOPENMODE**

```
s" = SEND(f, 'DBOPENMODE=0')          f{PROP:DbOpenMode} = 0
i# = SEND(f, 'DBOPENMODE')            i# = f{PROP:DbOpenMode}
```

The DBOPENMODE command or property is used to reference the database open mode for the current session. The mode must be specified before the table is opened. The open mode for the database determines the access rights of any FILE opened on the database. If the database is opened in read-only mode, FILES will have only read-only access. The open mode values are as follows:

0	Read/Write (default)
1	Read-only

### **DBSHAREMODE**

```
s" = SEND(f, 'DBSHAREMODE=0')          f{PROP:DbShareMode} = 0
i# = SEND(f, 'DBSHAREMODE')            i# = f{PROP:DbShareMode}
```

The DBSHAREMODE command or property is used to reference the database share mode for the current session. The mode must be specified before the table is opened. The share mode for the database determines the access rights of any FILE opened on the database. The share mode values are as follows:

0	Shared (default)
1	Exclusive

### **INIFILE**

```
s" = SEND(f, 'INIFILE=IDAPI.CFG')      f{PROP:IniFile} = 'IDAPI.CFG'
s" = SEND(f, 'INIFILE')                 s" = f{PROP:IniFile}
```

The INIFILE command or property is used to determine the configuration used by the Borland Database Engine. The configuration file must be specified before the table is opened.

### **LOCALINIT**

```
s" = SEND(f, 'LOCALINIT=1')            f{PROP:LocalInit} = TRUE
s" = SEND(f, 'LOCALINIT')               s" = f{PROP:LocalInit}
```

The LOCALINIT command or property is used to force local initialisation, if a value of TRUE (1) is specified, the default is FALSE (0).

### **POSITIONSIZE**

```
i# = SEND(f, 'POSITIONSIZE')           i# = f{PROP:PositionSize}
```

The POSITIONSIZE command or property is used to determine the size of the position string required for the current sequence.

### **PRIVATEDIR**

```
s" = SEND(f, 'PRIVATEDIR=C:\PRIV')     f{PROP:PrivateDir} = 'C:\PRIVATE'
s" = SEND(f, 'PRIVATEDIR')             s" = f{PROP:PrivateDir}
```

The PRIVATEDIR command or property is used to set or get the private directory for the current session. The private directory must be specified before the file is opened.

### **QBE**

```
s" = SEND(f, 'QBE=???')          f{PROP:QBE} = '???'
s" = SEND(f, 'QBE')              s" = f{PROP:QBE}
```

The QBE command or property is used to submit a query or determine whether a previously submitted query used QBE. If a previously committed query was submitted using QBE the result is TRUE (1), otherwise it is FALSE (0).

### **SEQCURSOR**

```
s" = SEND(f, 'SEQCURSOR')        s" = f{PROP:SeqCursor}
```

The SEQCURSOR command or property is used to obtain the table's sequential access handle. The sequential access handle is determined by any SET() call and is used internally to process subsequent calls to NEXT(), PREVIOUS(), EOF() or BOF(). This handle may be used when calling the IDAPI functions directly from Clarion. Care must be taken when calling the IDAPI directly, since this can affect file processing by the driver.

### **SQL**

```
s" = SEND(f, 'SQL=SELECT * FROM T') f{PROP:SQL} = 'SELECT * FROM T'
s" = SEND(f, 'SQL')                s" = f{PROP:SQL}
```

The SQL command or property is used to submit a query or determine the text of a previously submitted query. The SQL command may be used as a driver string on a file with no fields or keys to define a 'cursor'. A cursor provides a means of processing a table or tables according to some query defined at run-time. The results may be obtained using property syntax (see meta- commands).

### **STRICT**

```
s" = SEND(f, 'STRICT=1')         f{PROP:Strict} = TRUE
s" = SEND(f, 'STRICT')          s" = f{PROP:Strict}
```

The STRICT command or property is used to enforce stricter checking to ensure that a file definition completely matches the table. Usually this is not necessary but it may be helpful during development and testing.

### **TABLECURSOR**

```
s" = SEND(f, 'TABLECURSOR')     s" = f{PROP:TableCursor}
```

The TABLECURSOR command or property is used to obtain the table's cursor handle. The cursor handle is used for all access to the table and is synchronised with the sequential cursor. This handle may be used when calling the IDAPI functions directly from Clarion. Care must be taken when calling the IDAPI directly, since this can affect file processing by the driver.

### **WORKDIR**

```
s" = SEND(f, 'WORKDIR=C:\WORK')  f{PROP:WorkDir} = 'C:\WORK'
s" = SEND(f, 'WORKDIR')          s" = f{PROP:WorkDir}
```

The WORKDIR command or property is used to set or get the work directory for the current session. The work directory must be specified before the file is opened.

## SQL Support

The Paradox File Driver supports SQL commands using PROP:SQL or using a special form of FILE called a cursor (described below). If the SQL statement returns a result set, then a cursor must be used to retrieve the results.

The Paradox File Driver supports the IDAPI Local SQL dialect, this provides support for SELECT, INSERT, UPDATE and DELETE statements with some limitations.

Documentation on the IDAPI Local SQL dialect is included with Borland development tools (e.g. Borland C++ Programmer's Guide (Version 5), chapter 47.)

### Cursors

The Paradox File Driver implements cursors using a special form of FILE declaration. A cursor traverses the result set of an SQL query. Furthermore, the query can be determined at run-time since a cursor can execute any query. A cursor is also used to process the results of a query using QBE.

A cursor is implemented using an empty file declaration. The query may be supplied as a driver string, or using the SEND() command or the PROP:SQL or PROP:QBE property:

```
!*** Cursor1 traverses table T
Cursor1      FILE,DRIVER('Paradox', 'SQL=SELECT * FROM T')
              RECORD
              END
              END

!*** Cursor2 must be determined using PROP:SQL or SEND('SQL=???')
Cursor2      FILE,DRIVER('Paradox')
              RECORD
              END
              END

CODE
OPEN(Cursor1)

Cursor2{PROP:SQL} = 'SELECT * FROM T'
OPEN(Cursor2)
```

Once the cursor is opened it may be traversed using the usual sequential file commands, SET(), NEXT() and PREVIOUS(). In addition any other file commands appropriate for sequential processing (e.g. POSITION(), RESET()) may be used. Note that cursors do not have keys, the order and content of the result set must be determined by the query.

A cursor is updateable if the result set refers to the original table data rather than a temporary table. The BDE will create a temporary table for the result set of a query satisfying any of the following conditions:

- It involves more than one table
- It has an ORDER BY clause
- It uses aggregates
- The WHERE clause does not consist only of simple comparisons of columns to constant values combined by AND or OR operators

If the result set is temporary, the cursor will be treated as read-only. Any attempt to update the cursor will result in an 'Access Denied' error.

Care must be taken when updating cursors on tables for which a primary key exists. If the primary key value is changed, the sequence in which records are returned in the result set will be affected. This can result in sequential loops (using SET() ... NEXT()) returning records in an unexpected order, processing records repeatedly or not at all, and possibly failing to terminate. As a general rule, do not modify primary key values.

The structure and contents of the record buffer must be determined using 'Meta File Properties', these are described in the accompanying document METAPROP.PDF.

## Error Handling

The Paradox File Driver attempts to map any BDE errors to the appropriate Clarion error code. When it is not possible to do this, the driver will return an error 90. This indicates that driver specific error information is available via the FILEERROR() and FILEERRORCODE() functions. The FILEERROR() and FILEERRORCODE() functions may be called whenever any error is posted by the driver, to determine the IDAPI error.

Error codes in the range 2000h-2100h are returned to indicate a driver specific error, usually resulting from an illegal file structure. Error codes in the range 2100h-3F00h (and above) are IDAPI errors. The string returned by the FILEERROR() function gives a description of these errors.

During development it is advisable to check for errors in the above ranges after any OPEN() or CREATE() commands, to ensure that the file is properly formed.

Under certain circumstances the driver may display a 'Validation Error' dialog with a message like:

```
Error (n) validating registration data
```

This dialog is displayed when the driver fails when checking the registered user data or driver requirements at load time. The error number indicates the precise problem which may be one of the following:

- 0 - No error, however the driver evaluation has expired. You may download a later version or register the driver you have using the L3UNLOCK.EXE application.
- 1 - A consistency check on the registered user data failed. This indicates a corrupt driver that may have been tampered with.
- 2 - The key used to register the driver is incorrect. Should never occur.
- 3 - The Clarion IDE is not loaded. Unregistered copies of the driver require the Clarion IDE to be loaded.
- 4 - The .LIC file that was installed with the driver cannot be found. Unregistered copies of the driver require the .LIC file to be present on the PATH.
- 5 - An error occurred processing the .LIC file. This indicates a corrupt .LIC file that may have been tampered with.
- 6 - An error occurred validating the driver against the .LIC file. This may indicate a version mismatch between the .LIC file and the driver.

## Utilities and Examples

There are two hand-coded examples that illustrate the features of the Paradox Driver and the Meta Properties.

### **PDXFIX**

PdxFix is a simple utility, developed using Borland C++ 5.0, which may be used to recover corrupted Paradox tables. Before using PdxFix, you should make a complete backup of all data files and indexes.

Full source for the PdxFix utility is supplied to allow for customisation and improvement.

Either DBInfo or PdxFix must be shipped with any applications developed using the Paradox File Driver in order to comply with the licensing conditions for the Borland Database Engine (see Licensing the Borland Database Engine).

### **DBInfo**

DBInfo is a simple diagnostic utility, developed using Borland C++ 5.0, which displays information about the BDE environment which may be of use in diagnosing problems with applications developed using the Paradox File Driver.

Full source for the DBInfo utility is supplied to allow for customisation and improvement.

Either DBInfo or PdxFix must be shipped with any applications developed using the Paradox File Driver in order to comply with the licensing conditions for the Borland Database Engine (see Licensing the Borland Database Engine).

### **PDX2TXD**

This utility will generate a TXD file from one or more Paradox tables and has advantages over importing tables directly into the dictionary:

- More than one file may be imported at a time
- Properly augmented file descriptions are generated
- Handles BLOB fields correctly

### **CWPD**

This example allows dynamic loading of any paradox table. It also allows queries using local SQL or QBE on a table and browsing of the result set.

## Unsupported or Modified Functions & Attributes

MEMO's are not supported, consequently the NOMEMO() command is not supported. NOMEMO() does not apply to BLOBs since BLOBs are loaded on request only.

The BUFFER() command is not supported.

Paradox tables do not support variable length records, consequently the ADD(file, len), APPEND(file, len) and PUT(file, len) commands are not supported.

Transactions are not supported, consequently the LOGOUT(), COMMIT() and ROLLBACK() commands are not supported.

Optimistic concurrency is not currently supported, consequently WATCH() is not supported.

Filtered dynamic indexes are not currently supported, consequently BUILD(file, fields, filter) is not supported.

The STREAM() and FLUSH() commands are ignored.

The behaviour of the DUPLICATE() function has been modified since duplicates can only be detected on a primary key, hence this is the only key that will actually be tested by any call to duplicate.

The APPEND() and ADD() commands have identical functionality, i.e. the APPEND() command will update keys if any are defined.

The value returned by the POINTER() command is not stable. This means that the pointer for a given record may change as the file is modified. This is because Paradox tables are physically reorganised as records are added or deleted.

The value returned by POSITION() is stable only if the file has a primary key.

The DUP attribute for keys is ignored. It is implied for all keys except the PRIMARY key.

The OPT attribute is ignored.

The RECLAIM attribute is ignored, though executing the PACK command may reduce fragmentation and increase efficiency. BUILD() will achieve similar results.

Paradox allows only one key or index on a given field combination. During creation Paradox gives precedence to maintained indexes.

When building a dynamic index using BUILD(f,k,s), the driver will attempt to reuse an existing key if one exists with the same components. If a key exists with the same components that is not compatible (e.g. the key is sorted differently) then an error is returned.

## Limitations

There are a number of limitations in this version of the Paradox File Driver which are listed below. The limitations may be removed in future versions of the driver.

- There is currently no language support within the Paradox File Driver, however the Borland Database Engine configuration utility (BDECFG.EXE) supplied with the BDE may be used to specify an appropriate language driver for collation and sorting.
- Updating a primary key field for a cursor on a table with a primary key can destroy sequencing with unexpected results.
- PROP:SQL can only issue a SELECT statement for a cursor file structure.
- There is no support for specifying security options, referential integrity or validity checks when creating a table. However any applications using the Paradox File Driver will honour any such options or checks established for the table (e.g. using the Borland Database Desktop).
- Tables containing BCD fields cannot currently be browsed with the Clarion for Windows browser. This appears to be a limitation in the browser.

The following limitations are imposed by the current version of the Meta Property Extensions:

- Cursors and dynamically created or loaded files cannot be binded.

The following limitations have no known workaround:

- The Paradox BCD type supports up to 32 digits, whereas the Clarion BCD type has a maximum of 31 digits. This means that where a Paradox table contains 32 digit BCD values, they may be truncated if manipulated by Clarion code.

## **Acknowledgements**

All trademarks are trademarks or registered trademarks of their respective owners.