Database Independent Abstraction Layer for C

libdbi Driver Author's Guide

David A. Parker
Neon Goat Productions

david@neongoat.com

Markus Hoenicka

mhoenicka@users.sourceforge.net

Database Independent Abstraction Layer for C: libdbi Driver Author's Guide

by David A. Parker

by Markus Hoenicka

Document revision: $Id: driver-guide.sgml,v 1.8 2013/02/03 23:04:32 mhoenicka Exp $ Edition
Published $Date: 2013/02/03 23:04:32 $
Copyright © 2001-2013 David Parker, Neon Goat ProductionsMarkus Hoenicka

libdbi implements a database-independent abstraction layer in C, similar to the DBI/DBD layer in Perl. Writing one generic set of code, programmers can leverage the power of multiple databases and multiple simultaneous database connections by using this framework.

This guide explains the internal DBD interface for libdbi drivers, and provides a reference for all available driver helper functions.

Permission is granted to copy, distribute and/or modify this document under the terms of the *GNU Free Documentation License*, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in Appendix A.

Table of Contents

1. Introduction **1**
1.1. Description 1
1.2. libdbi Concepts and Terminology 1
1.3. Modifications and redistribution of libdbi 1
1.4. Contact Info 1
2. Driver Infrastructure **3**
2.1. Driver Capabilities 3
2.1.1. Setting driver capabilities 3
2.1.2. Required driver capabilities 3
2.1.3. Recommended driver capabilities 3
2.2. Database directories 4
2.2.1. Default Database Directories 4
2.2.2. Custom Database Directories 4
2.3. Driver data 4
2.3.1. Driver specific functions 4
2.3.2. Reserved words 5
3. Driver Functions **6**
3.1. Driver Infrastructure Functions 6
3.1.1. dbd_register_driver 6
3.1.2. dbd_initialize 6
3.1.3. dbd_connect 6
3.1.4. dbd_disconnect 7
3.1.5. dbd_geterror 7
3.1.6. dbd_get_socket 8
3.2. Internal Database Query Functions 8
3.2.1. dbd_goto_row 8
3.2.2. dbd_fetch_row 9
3.2.3. dbd_free_query 9
3.3. Public Database Query Functions 9
3.3.1. dbd_get_encoding 10
3.3.2. dbd_encoding_to_iana 10
3.3.3. dbd_encoding_from_iana 10
3.3.4. dbd_get_engine_version 11
3.3.5. dbd_list_dbs 11
3.3.6. dbd_list_tables 11
3.3.7. dbd_quote_string 12
3.3.8. dbd_conn_quote_string 13
3.3.9. dbd_quote_binary 13
3.3.10. dbd_query 14
3.3.11. dbd_query_null 14
3.3.12. dbd_select_db 15
3.3.13. dbd_get_seq_last 15

3.3.14. dbd_get_seq_next 15
3.3.15. dbd_ping 16
3.4. DBD Helper Functions 16
3.4.1. _dbd_result_create 16
3.4.2. _dbd_result_set_numfields 17
3.4.3. _dbd_result_add_field 17
3.4.4. _dbd_row_allocate 18
3.4.5. _dbd_row_finalize 18
3.4.6. _dbd_internal_error_handler 18
3.4.7. _dbd_result_create_from_stringarray 19
3.4.8. _dbd_register_driver_cap 19
3.4.9. _dbd_register_conn_cap 20
3.4.10. _dbd_parse_datetimex 20
3.4.11. _dbd_escape_chars 21
3.4.12. _dbd_encode_binary 21
3.4.13. _dbd_decode_binary 22
A. GNU Free Documentation License 23

Chapter 1. Introduction

1.1. Description

libdbi provides application developers with a database independent abstraction layer for C. It handles the database-specific implementations for each type of database, so that you can use the same exact code with any type of database server that libdbi supports. You can initiate and use multiple database connections simultaneously, regardless of the types of database servers you are connecting to. The plugin architecture allows for new database drivers to be easily added dynamically by a third party.

To aid the development of new database drivers, libdbi ships a template which contains everything you need to get started. Copy the `drivers/example` directory to your CVS version of the libdbi-drivers (http://sourceforge.net/projects/libdbi-drivers) project, rename it to the name of your database engine, and replace the string "example" by the name of your database engine in all files in that directory. This should get you pretty far. Check the name of the client library in the `Makefile.am` and the name of the client library headers in the driver source file. Then have a peek at the existing drivers, and implement the functions in the driver source template accordingly.

1.2. libdbi Concepts and Terminology

In this guide, the terms “author” and “programmer” are used interchangably, since the target audience is the software developer writing a driver for libdbi.

1.3. Modifications and redistribution of libdbi

libdbi is Copyright © 2001-2005, David Parker and Mark Tobenkin.

libdbi is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

1.4. Contact Info

Please email us with any bugs, ideas, feature requests, or questions. The libdbi website has the latest version of this documentation and the libdbi software, as well as a central database of third-party drivers.

- http://libdbi.sourceforge.net
- libdbi-users mailing list <libdbi-users@lists.sourceforge.net>

Chapter 2. Driver Infrastructure

This chapter briefly discusses some infrastructure features which you have to consider when implementing a database driver.

2.1. Driver Capabilities

Driver capabilities are essentially an array of key/value pairs which the drivers set when they're loaded. Both the libdbi framework and programs linked against libdbi can query these capabilities and adjust their behaviour accordingly.

2.1.1. Setting driver capabilities

The perfect place to set driver capabilities is in the dbd_initialize function which is called right after the driver is loaded by libdbi. To set capabilities, call the _dbd_register_driver_cap function for each of them:

```
void _dbd_register_driver_cap(dbi_driver_t *driver, const char *capname, int value);
```

Arguments

`driver`: the driver as passed to dbd_initialize.

`capname`: A string containing the name of the capability (i.e. the key).

`value`: The value of the capability.

2.1.2. Required driver capabilities

libdbi currently queries only one driver capability.

safe_dlclose

A nonzero value indicates that the driver can safely be unloaded from memory by calling dlclose(). A value of 0 (zero) indicates that the driver should not be unloaded when libdbi is shut down. Drivers must not be unloaded if they, or any library they are linked against, install exit handlers via atexit() as this would leave dangling pointers, causing segfaults on some platforms.

2.1.3. Recommended driver capabilities

Two driver capabilities may be of interest to programs using libdbi and should therefore be published by drivers:

transaction_support

A nonzero value indicates that the database engine supports transactions. Therefore functions to start, commit, or roll back transactions may be safely used.

savepoint_support

A nonzero value indicates that the database engine supports savepoints within transactions. Functions to set, release, and roll back to savepoints may be safely used.

2.2. Database directories

Many database engines use database directories which they manage themselves. Other engines, like sqlite, sqlite3, or firebird, allow the database user to store database files basically anywhere on the filesystem. Drivers of such engines should follow some guidelines to make database operations as transparent as possible across all drivers. In order to allow e.g. listing available databases, most applications should keep all databases managed by such a driver in the same directory.

2.2.1. Default Database Directories

libdbi-drivers uses a common root directory (`<localstatedir>/lib/libdbi`) containing homonymous subdirectories for each driver as a default database path. If nothing else is specified (see Section 2.2.2), databases will be created and accessed in these subdirectories. Using different subdirectories for each driver makes switching between drivers painless without having to re-create database files from scratch.

Users can change the compile-time default root directory using the `--with-dbi-dbdir` option when configuring libdbi-drivers.

2.2.2. Custom Database Directories

In order to allow applications to store databases outside of the default directory if desired, a driver should implement a <drivername>_dbdir option which overrides the default database path in a connection.

2.3. Driver data

Each driver has to declare two global string arrays which are queried by libdbi. They provide a list of driver-specific functions and a list of reserved words. Both string arrays must be declared even if they are empty.

2.3.1. Driver specific functions

libdbi was designed to provide access to a variety of database engines using a single common interface. However, database engines may have client library APIs weird enough to make them badly suited for libdbi. A driver may therefore have to provide additional functions which are specific to this driver. Also, it may be useful to provide access to database engine specific functions (i.e. functions of the database engine client library) if their usage is not covered by the libdbi interface.

The following line defines a string array with two representative function names. Please note that the string array must be terminated with a NULL string. This holds true even if the driver does not export any custom functions.

```
static const char *custom_functions[] = {"foo", "bar", NULL};
```

> **Note:** libdbi internally attempts to create pointers to the named functions. It is not considered an error if the symbol is missing, so it is safe to provide the names of functions which are not present in all versions of a client library. Creating pointers may also fail if functions are implemented as macros by the client library.

2.3.2. Reserved words

Database engines use different implementations of the SQL standard. Some language features of the SQL standard may not be supported, whereas some engines implement language features which are not part of the standard. In order to avoid conflicts between e.g. table or column names and "reserved words" (i.e. words which a specific SQL implementation considers part of the language), libdbi provides a function to find out at runtime whether or not a word is a reserved word. Each driver therefore has to provide such a list of reserved words. Again, the string array used to provide this list must be terminated by a NULL string:

```
static const char *reserved_words[] = {"foo", "bar", NULL};
```

Chapter 3. Driver Functions

3.1. Driver Infrastructure Functions

These functions are called by libdbi at startup and when the libdbi user establishes or takes down a database engine connection.

3.1.1. dbd_register_driver

```
void dbd_register_driver(const dbi_info_t **_driver_info, const char
***_custom_functions, const char ***_reserved_words);
```

This is the first function called after the driver module is loaded into memory. It passes back meta-information back to libdbi through the pointers passed as arguments.

Arguments

`_driver_info`: A pointer used to link to the driver's information struct.

`_custom_functions`: A pointer used to link to the driver's string array of custom database-specific functions.

`_reserved_words`: A pointer used to link to the driver's string array of reserved words.

3.1.2. dbd_initialize

```
int dbd_initialize(dbi_driver_t *driver);
```

Performs any database-specific server initialization. This is called right after dbd_register_driver().

Arguments

`driver`: The driver's pointer.

Returns

-1 on error, 0 on success. If -1 is returned, the driver will not be added to the list of available drivers.

3.1.3. dbd_connect

```
int dbd_connect(dbi_conn_t *conn);
```

Connects to the database, setting the connection's DB-specific connection handle and current database name. Connection parameters are already filled through the connection's option settings. The standard options that all drivers must recognize (if applicable) are: host, port, username, password, dbname, and encoding. Any driver-specific functions must be prefixed with the name of the driver and an underscore, such as "mysql_compression".

Arguments

`conn`: The target connection instance of the driver.

Returns

<0 on error, 0 on success.

3.1.4. dbd_disconnect

```
int dbd_disconnect(dbi_conn_t *conn);
```

Disconnects from the database server.

Arguments

`conn`: The target connection instance of the driver.

Returns

-1 on error, 0 on success.

3.1.5. dbd_geterror

```
int dbd_geterror(dbi_conn_t *conn, int *errno, char **errstr);
```

Retrieves and stores error information, in numeric and/or string format.

Arguments

`conn`: The target connection.

`errno`: The int variable to hold the error number.

`errstr`: The string to hold the error description. The driver is supposed to provide the string as allocated memory which is further managed by libdbi.

Returns

0 if there was an error, 1 if errno was filled, 2 if errstr was filled, 3 if both errno and errstr were filled.

3.1.6. dbd_get_socket

```
int dbd_get_socket(dbi_conn_t *conn);
```

Retrieves the socket of the client/server connection used by the database client library, if applicable.

Arguments

`conn`: The target connection.

Returns

The file descriptor of the socket if successful, -1 if there was an error. Drivers of database engines that do not use sockets should return 0.

3.2. Internal Database Query Functions

These functions are called by libdbi when the libdbi user runs queries and accesses their results. There are also a bunch of helper functions that deal with the character encodings as well as with string escaping and quoting.

3.2.1. dbd_goto_row

```
int dbd_goto_row(dbi_result_t *result, unsigned long long rowidx, unsigned long long
currowidx);
```

Jumps to the specifed row in the result set. Internal row counts start at 0. The current row number is passed to the driver to allow it to check whether a (presumably expensive) seek operation is required.

Arguments

`result`: The target result handle.

`row`: The target row number.

`row`: The current row number.

Returns

1 on success, 0 on error.

3.2.2. dbd_fetch_row

```
int dbd_fetch_row(dbi_result_t *result, unsigned long long rowidx);
```

Fetches the target row, retrieving one-time field information if necessary. Also see the _dbd_row_allocate and _dbd_row_finalize helper functions.

Arguments

`result`: The target result object.

`rowidx`: The number of the row to fetch. Internal row numbers start at zero.

Returns

0 on error, 1 on successful fetch.

3.2.3. dbd_free_query

```
int dbd_free_query(dbi_result_t *result);
```

Frees the target result handle.

Arguments

`result`: The target result handle.

Returns

0 on success.

3.3. Public Database Query Functions

3.3.1. dbd_get_encoding

```
const char *dbd_get_encoding(dbi_conn_t *conn);
```

Returns the character encoding used by the current connection.

Arguments

`conn`: The target connection.

Returns

A zero-terminated string containing the IANA name of the character encoding.

3.3.2. dbd_encoding_to_iana

```
const char *dbd_encoding_to_iana(const char *db_encoding);
```

Converts the database-engine-specific name of a character encoding to the corresponging IANA name.

Arguments

`db_encoding`: A pointer to a string containing the character encoding name.

Returns

A zero-terminated string containing the IANA name of the character encoding. If there is no equivalent IANA name, the original string will be returned.

3.3.3. dbd_encoding_from_iana

```
const char *dbd_encoding_from_iana(const char *iana_encoding);
```

Converts the IANA name of a character encoding to the corresponging database-engine-specific name.

Arguments

`iana_encoding`: A pointer to a string containing the character encoding name.

Returns

A zero-terminated string containing the database-engine-specific name of the character encoding. If there is no equivalent IANA name, the original string will be returned.

3.3.4. dbd_get_engine_version

```
char *dbd_get_engine_version(dbi_conn_t *conn, char *versionstring);
```

Returns the version string of the database engine that serves the current connection.

Arguments

`conn`: The current connection.

`versionstring`: A pointer to a string that can hold at least VERSIONSTRING_LENGTH bytes, including the trailing NULL byte. The function will write the version string to this buffer.

Returns

`versionstring` which now contains a zero-terminated string representing the database engine version. This string contains only digits and periods. Returns an empty string in case of an error.

3.3.5. dbd_list_dbs

```
dbi_result_t *dbd_list_dbs(dbi_conn_t *conn, const char *pattern);
```

Performs a query that retrieves the list of databases, with the database name as the first column in the result set. If *`pattern`* is non-NULL, only databases whose name match *`pattern`* are listed.

Arguments

`conn`: The target connection.

`pattern`: A SQL regular expression that limits the search, or NULL to list all tables.

Returns

A DBI result object, or NULL if an error occurs.

3.3.6. dbd_list_tables

```
dbi_result_t *dbd_list_tables(dbi_conn_t *conn, const char *db, const char *pattern);
```

Performs a query that retrieves the list of tables in the specified database, with the table name as the first column in the result set. If *pattern* is non-NULL, lists only the tables that match *pattern*.

Arguments

conn: The target connection.

db: The name of the database where tables should be looked for.

pattern: A SQL regular expression that limits the search, or NULL to list all tables.

Returns

A DBI result object, or NULL if an error occurs.

3.3.7. dbd_quote_string

```
size_t dbd_quote_string(dbi_driver_t *driver, const char *orig, char *dest);
```

Given a string, wrap quotes around that string and escape any characters that the database server needs escaped.

> **Note:** The use of this function in user programs is deprecated, but drivers must still implement it at the moment. If the quoting and escaping does not depend on the connection parameters, it is perfectly legal to let your implementation of dbd_conn_quote_string call this function (it is not possible to do it the other way). libdbi makes sure that both *orig* and *dest* are non-NULL before calling this function.

Arguments

driver: A pointer to the driver itself, which may be useful in weird cases.

orig: The string to quote and escape.

dest: The destination for the new string, which is already allocated as (strlen(orig)*2)+4+1. In the worst case, each character will need to be escaped, with two quote characters at both the beginning and end of the string, plus one for the terminating NULL.

Returns

The length of the new string.

3.3.8. dbd_conn_quote_string

```
size_t dbd_conn_quote_string(dbi_conn_t *conn, const char *orig, char *dest);
```

Given a string, wrap quotes around that string and escape any characters that the database server needs escaped.

> **Note:** The use of this function in user programs is preferred over dbd_quote_string. If the quoting and escaping does not depend on the connection parameters, it is perfectly legal to let your implementation of this function call dbd_quote_string. libdbi makes sure that both *orig* and *dest* are non-NULL before calling this function.

Arguments

conn: A pointer to the current connection.

orig: The string to quote and escape.

dest: The destination for the new string, which is already allocated as (strlen(orig)*2)+4+1. In the worst case, each character will need to be escaped, with two quote characters at both the beginning and end of the string, plus one for the terminating NULL.

Returns

The length of the new string.

3.3.9. dbd_quote_binary

```
size_t dbd_quote_binary(dbi_conn_t *conn, const char *orig, size_t from_length, char
**dest);
```

Given a binary string (which may contain NULL bytes and other non-printable characters), wrap quotes around that string and escape any characters that the database server needs escaped. If the function returns an error, **dest* is not a valid pointer to a string.

Arguments

`conn`: A pointer to the current connection.

`orig`: The string to quote and escape.

`from_length`: The length, in bytes, of the binary string.

`dest`: A pointer to the destination of the new zero-terminated string. The function allocates the required memory as required and updates the pointer that *`dest`* points to accordingly.

Returns

The length of the new string, or DBI_LENGTH_ERROR in case of an error.

3.3.10. dbd_query

```
dbi_result_t *dbd_query(dbi_conn_t *conn, const char *statement);
```

Performs a query and keeps track of meta-information about the query. Also see the _dbd_result_create helper function.

Arguments

`conn`: The target connection.

`statement`: The zero-terminated query string to execute.

Returns

A DBI result object, or NULL on error.

3.3.11. dbd_query_null

```
dbi_result_t *dbd_query_null(dbi_conn_t *conn, const unsigned char *statement, size_t
st_length);
```

Performs a query using a binary query string and keeps track of meta-information about the query. Also see the _dbd_result_create helper function.

Arguments

`conn`: The target connection.

`statement`: The query string to execute, which may contain NULL bytes and other non-printable characters.

`st_length`: The length of the binary query string.

Returns

A DBI result object, or NULL on error.

3.3.12. dbd_select_db

```
const char *dbd_select_db(dbi_conn_t *conn, const char* db);
```

Selects a new database on the server.

Arguments

`conn`: The target connection.

`db`: The name of the database to switch to.

Returns

The database name on success, NULL on error, or an empty string if the operation is not supported by the database server.

3.3.13. dbd_get_seq_last

```
unsigned long long dbd_get_seq_last(dbi_conn_t *conn, const char *sequence);
```

Returns the row ID generated by the last **INSERT** command.

Arguments

`conn`: The target connection.

`sequence`: The name of the sequence if the database engine requires this, or NULL if it is not required.

Returns

The row ID if successful, otherwise 0.

3.3.14. dbd_get_seq_next

```
unsigned long long dbd_get_seq_next(dbi_conn_t *conn, const char *sequence);
```

Increments the sequence counter by the preset increment, and returns the resulting row ID.

Arguments

`conn`: The target connection.

`sequence`: The name of the sequence if the database engine requires this, or NULL if it is not required.

Returns

The row ID if successful, otherwise 0. Also return 0 if the database engine does not implement this feature.

3.3.15. dbd_ping

```
int dbd_ping(dbi_conn_t *conn);
```

Checks whether the database connection is still alive.

Arguments

`conn`: The target connection.

Returns

1 if the connection is alive, otherwise 0. This function may be implemented such that it automatically attempts to reconnect if the connection went down. If the reconnect is successful, the function should also return 1.

3.4. DBD Helper Functions

libdbi implements a couple of functions which come in handy when implementing database engine drivers. Call them from your driver code if appropriate.

3.4.1. _dbd_result_create

```
dbi_result_t *_dbd_result_create(dbi_conn_t *conn, void *handle, unsigned long long
numrows_matched, unsigned long long numrows_affected);
```

Allocates a new dbi_result_t, filling the number of rows matched and affected, storing the database-specific result handle, and allocating room for rows to be stored.

Arguments

`conn`: The target connection.

`handle`: The database-specific result handle used internally by the driver.

`numrows_matched`: The number of rows matched by the query.

`numrows_affected`: The number of rows affected by the query.

Returns

A new DBI result object.

3.4.2. _dbd_result_set_numfields

```
void _dbd_result_set_numfields(dbi_result_t *result, unsigned int numfields);
```

Sets a result's number of fields and allocates memory for field information to be stored.

Arguments

`result`: The target result.

`numfields`: The number of fields in the result set.

3.4.3. _dbd_result_add_field

```
void _dbd_result_add_field(dbi_result_t *result, unsigned int idx, char *name,
unsigned short type, unsigned int attribs);
```

Stores information about the target field into the result set.

Arguments

`result`: The target result.

`idx`: The numeric field index.

`name`: The name of the field.

`type`: The datatype of the field.

`attribs`: The attributes of the field.

3.4.4. _dbd_row_allocate

```
dbi_row_t *_dbd_row_allocate(unsigned int numfields);
```

Allocates a new row, ready to be filled with data.

Arguments

`numfields`: The number of fields in the result set.

Returns

A new DBI row, or NULL on error.

3.4.5. _dbd_row_finalize

```
void _dbd_row_finalize(dbi_result_t *result, dbi_row_t *row, unsigned long long
rowidx);
```

Associates and stores the row with the result set, once the row's data has been filled.

Arguments

`result`: The target result set.

`row`: The target row object.

`rowidx`: The index of the row.

3.4.6. _dbd_internal_error_handler

```
void _dbd_internal_error_handler(dbi_conn_t *conn, const char *errmsg, const int
errno);
```

Saves error message information. libdbi makes this information available to the software to check the error status after each call to a libdbi API function. If an old error message string exists, it will be freed.

Arguments

`conn`: The target connection.

`errmsg`: The error message to store. This will be stdup'd by libdbi so it has its own copy. If NULL, libdbi will attempt to provide an appropriate message string.

`errno`: The error number to store. Use only the predefined (in `include/dbi/dbi.h`) constants DBI_ERROR_*. If the error number is DBI_ERROR_DBD, libdbi will replace the error number and message by calling the driver function `dbd_geterror` which retrieves the error code and message from the database client library. If errmsg is NULL and errno is any other of the predefined constants, libdbi will provide its own message string.

3.4.7. _dbd_result_create_from_stringarray

```
dbi_result_t *_dbd_result_create_from_stringarray(dbi_conn_t *conn, unsigned long long
numrows_matched, const char **stringarray);
```

Creates a result object from an array of strings which contains the data of a single field for each row.

Arguments

`conn`: The target connection.

`numrows_matched`: The number of rows contained in the *`stringarray`*.

`stringarray`: A pointer to an array of strings with *`numrows_matched`* members.

Returns

A result object, or NULL if there is an error.

3.4.8. _dbd_register_driver_cap

```
void _dbd_register_driver_cap(dbi_driver_t *driver, const char *capname, int value);
```

Adds a key-value pair to the list of driver capabilities.

Arguments

`driver`: The target driver.

`capname`: The key.

`value`: The value.

3.4.9. _dbd_register_conn_cap

```
void _dbd_register_conn_cap(dbi_conn_t *conn, const char *capname, int value);
```

Adds a key-value pair to the list of connection capabilities.

Arguments

`conn`: The target connection.

`capname`: The key.

`value`: The value.

3.4.10. _dbd_parse_datetimex

```
int _dbd_parse_datetimex(const char *raw, unsigned int attribs, dbi_datetimex *dtx);
```

Parses the input time, date, or datetime string into a dbi_datetimex, the latter of which groups a struct tm and a timezone offset.

Arguments

`raw`: A zero-terminated string containing a time, date, or datetime value. Accepted formats are YYYY-MM-DD for date values, HH:MM:SS for time values, and YYYY-MM-DD HH:MM:SS for datetime values. The separators must be present, but can be any character.

`attribs`: The field attributes of raw.

`dtx`: a dbi_datetimex structure to fill in.

Returns

Always zero.

3.4.11. _dbd_escape_chars

```
size_t _dbd_escape_chars(char *dest, const char *orig, size_t orig_size, const char
*toescape);
```

Escapes the characters contained in *`toescape`* in the string *`orig`* and puts the result into the allocated memory pointed to by *`dest`*. The size of *`dest`* must be at least (*`orig_size`**2)+5. The characters are escaped by preceding them with a backslash.

Arguments

`dest`: Pointer to allocated memory which will receive the escaped string.

`orig`: The string to escape.

`orig_size`: The length of the string to escape.

`toescape`: A string containing all characters that need escaping.

Returns

The length, in bytes, of the escaped string.

3.4.12. _dbd_encode_binary

```
size_t _dbd_encode_binary(const unsigned char *in, size_t n, unsigned char *out);
```

Encodes a binary string as a zero-terminated string which can be safely included in a SQL query. Use _dbd_decode_binary to decode the string again.

Arguments

`in`: Pointer to the binary string.

`n`: Length, in bytes, of the binary string *`in`*.

`out`: Pointer to allocated memory which will receive the escaped string. The size must be at least 2 +(257*`n`)/254 bytes.

Returns

The length, in bytes, of the escaped string.

3.4.13. _dbd_decode_binary

```
size_t _dbd_decode_binary(const unsigned char *in, unsigned char *out);
```

Decodes a zero-terminated string with escaped characters as created by _dbd_encode_binary.

Arguments

`in`: Pointer to the input string.

`out`: Pointer to allocated memory which will receive the unescaped string. The output string is always shorter than the input string, i.e. if the size of `out` is the same as the size of `in`, you're on the safe side. The implementation allows to decode the string in place, i.e. `out` may be the same as `in`.

Returns

The length, in bytes, of the unescaped binary string.

Appendix A. GNU Free Documentation License

Version 1.1, March 2000

Copyright (C) 2000 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has

been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document’s Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.