
libdrizle-redux Documentation

Release v6.2.0

Feb 10, 2019

Contents

1	Introduction	1
1.1	Differences from Libdrizzle	1
1.2	Differences from libmysqlclient	1
2	Licensing	3
2.1	Documentation Content	3
2.2	About the Drizzle logo	3
2.3	Libdrizzle Redux License	3
2.4	Trademarks	4
3	Installing Libdrizzle Redux	5
3.1	Debian	5
3.2	Redhat	6
4	Compiling Libdrizzle Redux	7
4.1	Building Libdrizzle Redux	7
4.2	Running the Test Suite	7
4.3	Building For OSX (clang and gcc)	8
4.4	Building code coverage	8
4.5	Linking Your Application	9
5	Libdrizzle API	11
5.1	Constants	11
5.2	Library Functions	28
5.3	Connection Functions	28
5.4	Query Functions	38
5.5	Prepared Statements	46
5.6	Binlog Functions	54
6	Code Examples	59
6.1	Buffered Results	59
6.2	Unbuffered Results	60
6.3	Prepared Statements	62
6.4	Event Callback	63
6.5	Binlog Retrieval	64

Drizzle Redux is a project which aims to breath new life into the libdrizzle C connector. It is designed to allow you to connect to and query a MySQL database server using a simple API.

The connector is 3-clause BSD licensed so it can statically and dynamically link with almost any other open source or commercial software.

1.1 Differences from Libdrizzle

- The server-side functionality has been removed, it no longer acts as both a client and server API.
- The Drizzle prototype library functions have been removed. It now only talks to MySQL compatible servers.
- API functions have been simplified. In Libdrizzle there was a big confusion over whether the application or library should be doing the allocation and freeing of objects. It is now less ambiguous.
- New binlog API added. The library can now connect as a slave or mysqlbinlog client and retrieve binlog events.

There are many more new features to come.

1.2 Differences from libmysqlclient

- API is slightly different
- Currently missing server-side prepared statement support and protocol compression

These missing features are to be added.

2.1 Documentation Content



Libdrizzle Redux Documentation is licensed under a [Creative Commons Share Alike 3.0 license](#).

2.2 About the Drizzle logo

The Drizzle logo was created by Zak Greant under the [Creative Commons Share Alike 3.0 license](#).

The logo is available in SVG format on [Wikipedia](#).

2.3 Libdrizzle Redux License

Libdrizzle Redux is licensed under the [BSD 3-Clause License](#).

```
Copyright (c) 2012, Drizzle Developer Group
All rights reserved.
```

```
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
```

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- * The names of its contributors may be used to endorse or promote products derived from this software without specific prior written

(continues on next page)

(continued from previous page)

permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

2.4 Trademarks

MySQL is a registered trademark of Oracle and/or its affiliates

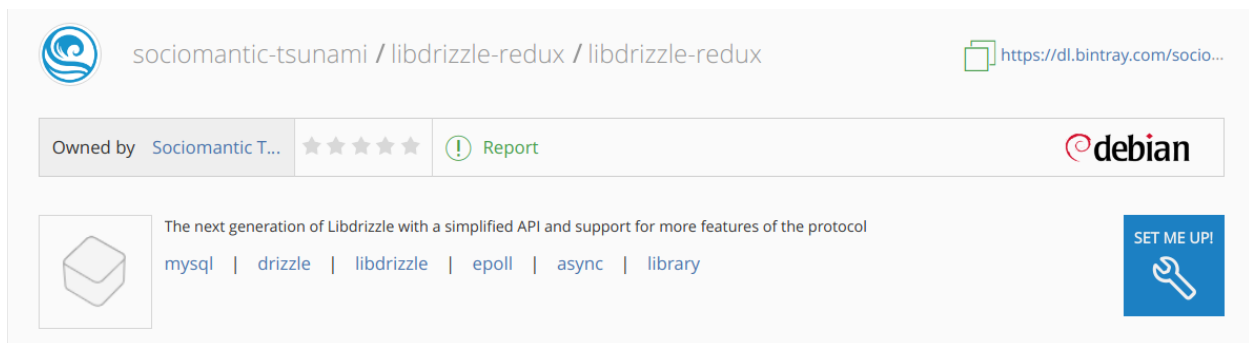
CHAPTER 3

Installing Libdrizzle Redux

3.1 Debian

For Debian based systems running **ubuntu xenial** deb packages are available in the project's **apt** repository at [bintray](#).

To add the **apt** repository to your system follow the instructions given when clicking on the **[set up me]!** button.



The screenshot shows the Bintray package page for `sociomantic-tsunami / libdrizzle-redux / libdrizzle-redux`. The page includes a header with the package name and a link to the Bintray repository. Below the header, there is a section for the package details, including the owner (Sociomantic T...), a star rating, and a 'Report' button. The package description states: 'The next generation of Libdrizzle with a simplified API and support for more features of the protocol'. It lists the components: `mysql`, `drizzle`, `libdrizzle`, `epoll`, `async`, and `library`. A blue button labeled 'SET ME UP!' with a wrench icon is visible in the bottom right corner.

The parameter `{distribution}` should be `xenial` while `{components}` can be `release` and/or `prerelease`. E.g:

```
https://dl.bintray.com/sociomantic-tsunami/libdrizzle-redux xenial release prerelease
```

Then run:

```
sudo apt update
sudo apt install libdrizzle-redux6 (for versions on the v6.x branch)
```

3.2 Redhat

Distribution packages are not available for Redhat based systems, but rpm packages can be generated by running `make rpm`.

Compiling Libdrizzle Redux

4.1 Building Libdrizzle Redux

To build libdrizzle-redux run the following commands:

```
mkdir build && cd build
autoreconf -fi ..
../configure
make
make install
```

Please check the [RELEASE NOTES](#) for a list of dependencies specific to the version of the library you are trying to compile.

4.2 Running the Test Suite

Libdrizzle has a unit test suite, it needs a running MySQL server which has a user that can create databases, tables and can connect as a MySQL slave.

The test suite uses system environment variables to find the MySQL server:

- `MYSQL_SERVER` - The hostname of the MySQL server (default localhost)
- `MYSQL_PORT` - The port number of the MySQL server (default 3306)
- `MYSQL_USER` - The username for the MySQL connection (default empty)
- `MYSQL_PASSWORD` - The password for the MySQL username (default empty)
- `MYSQL_SCHEMA` - The default database for the MySQL connection (default empty)

The test suite can then be run using `make check` or `make distcheck` for testing a source distribution rather than the straight git branch.

To test with valgrind you can run the following:

```
``TESTS_ENVIRONMENT="./libtool --mode=execute valgrind --error-exitcode=1 --leak-  
→check=yes --track-fds=yes --malloc-fill=A5 --free-fill=DE" make check``
```

4.3 Building For OSX (clang and gcc)

You can compile the source code with the `clang` compiler provided by **Xcode Command Line Tools**. Alternatively you can use [Homebrew](#) to install a specific `gcc` or `clang` compiler. Regardless of the choice of compiler, you will need to install **Xcode** and the **Xcode Command Line Tools**.

Compatible compilers:

Compiler	Version
GNU gcc	>= 4.5
LLVM clang	>= 3.3
Apple LLVM clang ²	>= 6.1

1. Install the dependencies specified in the [RELEASE NOTES](#) of the latest minor release.
2. Ensure **OpenSSL** headers are linked by creating a symlink:

```
ln -sf "$(brew --prefix openssl)/include/openssl" /usr/local/include/openssl
```

or pass the OpenSSL directory to configure using `--with-openssl`:

```
# create and cd to build directory and run autoreconf  
../configure --with-openssl=$(brew --prefix openssl)
```

3. Optionally set the C and C++ compiler before running `configure`, e.g.:

```
# create and cd to build directory and run autoreconf  
CC=gcc-4.9 CXX=g++-4.9 ../configure  
make
```

4.4 Building code coverage

Compiling the library with code coverage is supported for GCC and LLVM based compilers on linux and OSX. The coverage report tool `lcov` is a common requirement used to build the html coverage report. However the tool that generates the coverage depends on the compiler and build system and must in some cases be specified using the `configure` option `--with-cov-tool=`. If the coverage tool is added to **PATH** only the name of the executable is required, else the full path should be specified.

The general steps to build code coverage is:

```
autoreconf -fi  
./configure --with-coverage-support=[compile,capture,report] [--with-cov-tool=]  
make check-code-coverage
```

Note that coverage files generated by GCC compilers cannot be processed by the coverage tool available in the **XCode Command Line Tool Package** and visa versa.

² The version listed for Apple LLVM is the compiler used in the OS X builds on Travis CI. However earlier versions should be compatible as long as they support C++11 features, i.e. Apple LLVM 5.0, Xcode 5.0 and later.

4.4.1 Coverage tool for different compiler and build system

GCC (Linux, OSX)

With GCC compilers the default coverage tool is `gcov`. Thus, if `gcov` is in **PATH** it is not required to set `--with-cov-tool=` unless an alternative version of `gcov` should be used.

Clang from LLVM (Linux, OSX)

In this case LLVM's own coverage tool called `llvm-cov` must be used to process the coverage files. If `llvm-cov` is in **PATH** it is not required to set `--with-cov-tool=` unless an alternative version of `llvm-cov` should be used.

Clang from XCode Developer Tools Package (OSX)

On OSX the **XCode Command Line Tool Package** provides `llvm-cov`. However it doesn't provide the `lcov` report tool so it must be installed additionally using e.g. **Homebrew** or **Macports**

Check that the **XCode Command Line Tool Package** are installed on the system:

```
xcode-select -p
```

If not then install the package by running:

```
xcode-select --install
```

Find the `llvm-cov` executable by running:

```
`xcrun -f llvm-lcov`
```

Then either add the bin directory to **\$PATH** or specify the full name when running configure using `--with-cov-tool=`xcrun -f llvm-lcov``

4.5 Linking Your Application

Ensure the library is in your library and include paths. For releases prior to version `v6.0.2` linking your app against `libdrizzle-redux` requires the flag `-ldrizzle-redux`:

```
g++ app.c -oapp -ldrizzle-redux6 -lssl -lcrypto -pthread
```

From version `v6.0.3` and later the API level of the library is appended to the installed library name¹. This is also reflected in the install path for development headers which now follows the pattern:

```
/<include-prefix>/libdrizzle-redux[MAJOR_VERSION]/libdrizzle-redux
```

Thus, linking against `libdrizzle-redux v6.0.3` requires the flag `-ldrizzle-redux6` and if headers are included to add `-I/<prefix>/libdrizzle-redux6`, e.g.:

```
g++ app.c -oapp -I/usr/include/libdrizzle-redux6 -ldrizzle-redux6 -lssl -lcrypto -
↳ pthread
```

Another option is to link against `libdrizzle-redux` using the full name of the dynamic library, e.g.:

¹ v6.0.2 added the major version to the package name and the library file but the release is deprecated since the linking did not work correctly.

```
g++ app.c -oapp -I/usr/include/libdrizzle-redux6 -l:libdrizzle-redux6.so.13 -lssl -  
↳lcrypto -pthread
```

A tool called **libdrizzle-redux_config** is included to also assist with this.

5.1 Constants

5.1.1 Introduction

Libdrizzle Redux contains a number of constants, most of what are in the form of ENUMs. All ENUMs are typedef'd so no need to use the 'enum' keyword.

5.1.2 Library

drizzle_verbose_t

An ENUM of the verbosity for the library

DRIZZLE_VERBOSE_NEVER

Completely silent

DRIZZLE_VERBOSE_FATAL

Fatal errors only

DRIZZLE_VERBOSE_ERROR

All errors

DRIZZLE_VERBOSE_INFO

Information messages and errors

DRIZZLE_VERBOSE_DEBUG

Debugging messages and errors

DRIZZLE_VERBOSE_CRAZY

Everything

5.1.3 Global constants

Constants available to the client and internally

DRIZZLE_DEFAULT_TCP_HOST Default socket tcp connection host	"localhost"
DRIZZLE_DEFAULT_TCP_PORT Default socket tcp connection port	3306
DRIZZLE_MYSQL_TCP_PORT Unused	3306
DRIZZLE_MYSQL_TCP_SERVICE Unused	"mysql"
DRIZZLE_DRIZZLE_TCP_PORT Unused	4427
DRIZZLE_DEFAULT_TCP_SERVICE Unused	"mysql"
DRIZZLE_DRIZZLE_TCP_SERVICE Unused	"drizzle"
DRIZZLE_DEFAULT_UDS Default path for the Unix Domain Socket	"/tmp/mysql.sock"
DRIZZLE_DEFAULT_BACKLOG Default number of pending connections on the listening queue	64
DRIZZLE_MAX_ERROR_SIZE Maximum length of a drizzle error message	2048
DRIZZLE_MAX_USER_SIZE Maximum length for the database user name	64
DRIZZLE_MAX_PASSWORD_SIZE Maximum length for the database password	32
DRIZZLE_MAX_DB_SIZE Maximum length for the database name	64
DRIZZLE_MAX_INFO_SIZE Maximum length of a <i>drizzle_result_st</i> info or error message	2048
DRIZZLE_MAX_SQLSTATE_SIZE Maximum length a MySQL SQLSTATE code	5
DRIZZLE_MAX_CATALOG_SIZE Maximum length of the catalog name on a <i>drizzle_column_st</i>	128
DRIZZLE_MAX_TABLE_SIZE Maximum length of the table name on a <i>drizzle_column_st</i>	128
DRIZZLE_MAX_COLUMN_NAME_SIZE Maximum length of a <i>drizzle_column_st</i> column name	2048
DRIZZLE_MAX_DEFAULT_VALUE_SIZE Maximum size of the default value for a column	2048
DRIZZLE_MAX_PACKET_SIZE Maximum packet size for the connection	UINT32_MAX

DRIZZLE_MAX_BUFFER_SIZE	1024*1024*1024
Maximum size of the allocated buffer on a <i>drizzle_st</i>	
DRIZZLE_DEFAULT_BUFFER_SIZE	1024*1024
Default size of the allocated buffer on a <i>drizzle_st</i>	
DRIZZLE_BUFFER_COPY_THRESHOLD	8192
Unused	
DRIZZLE_MAX_SERVER_VERSION_SIZE	32
Maximum length of the server version string	
DRIZZLE_MAX_SERVER_EXTRA_SIZE	32
Maximum size of additional data sent after server handshake	
DRIZZLE_MAX_SCRAMBLE_SIZE	20
Maximum size of the buffer used during authentication if password scrambling is enabled	
DRIZZLE_STATE_STACK_SIZE	8
Maximum number of states saved on the stack	
DRIZZLE_ROW_GROW_SIZE	8192
The number of rows to read at a time when buffering a result	
DRIZZLE_DEFAULT_SOCKET_TIMEOUT	10
The default time in seconds to wait before a setsockopt call times out	
DRIZZLE_DEFAULT_SOCKET_SEND_SIZE	DRIZZLE_DEFAULT_BUFFER_SIZE
The default size of the socket send buffer	
DRIZZLE_DEFAULT_SOCKET_RECV_SIZE	DRIZZLE_DEFAULT_BUFFER_SIZE
The default size of the socket receive buffer	
DRIZZLE_MYSQL_PASSWORD_HASH	41
Unused	
DRIZZLE_BINLOG_CRC32_LEN	4
Size of the CRC32 checksum appended to each binlog event	
DRIZZLE_BINLOG_CHECKSUM_VERSION	"5.6.1"
From this version and higher automatic checksums is on	
DRIZZLE_BINLOG_MAGIC	"xFEx62x69x6E"
The 4-byte header of a binary log file	

5.1.4 Return

drizzle_return_t

Function return status ENUM

DRIZZLE_RETURN_OK

Return is OK

DRIZZLE_RETURN_IO_WAIT

Waiting on IO

DRIZZLE_RETURN_PAUSE

DRIZZLE_RETURN_ROW_BREAK

Row break because row is larger than packet size

DRIZZLE_RETURN_MEMORY

Memory allocation error

DRIZZLE_RETURN_ERRNO

OS error code

DRIZZLE_RETURN_INTERNAL_ERROR

Internal error during handshake

DRIZZLE_RETURN_GETADDRINFO

Domain lookup failure

DRIZZLE_RETURN_NOT_READY

Client is not connected to server

DRIZZLE_RETURN_BAD_PACKET_NUMBER

Packets are out of sequence

DRIZZLE_RETURN_BAD_HANDSHAKE_PACKET

Bad packet received during handshake

DRIZZLE_RETURN_BAD_PACKET

Bad packet received (unused)

DRIZZLE_RETURN_PROTOCOL_NOT_SUPPORTED

Attempt to connect to a version of MySQL less than 4.1

DRIZZLE_RETURN_UNEXPECTED_DATA

Unexpected data in the receive buffer

DRIZZLE_RETURN_NO_SCRAMBLE

No password scramble received (usually if server is expecting an auth plugin but client didn't use one)

DRIZZLE_RETURN_AUTH_FAILED

Authentication failure

DRIZZLE_RETURN_NULL_SIZE

Internal status

DRIZZLE_RETURN_ERROR_CODE

Error code received from MySQL server

DRIZZLE_RETURN_TOO_MANY_COLUMNS

Unused

DRIZZLE_RETURN_ROW_END

Internal status

DRIZZLE_RETURN_LOST_CONNECTION

Connection failure

DRIZZLE_RETURN_COULD_NOT_CONNECT

Could not connect to server

DRIZZLE_RETURN_NO_ACTIVE_CONNECTIONS

Waiting on a connection which doesn't exist (this shouldn't happen)

DRIZZLE_RETURN_HANDSHAKE_FAILED

Handshake failure

DRIZZLE_RETURN_TIMEOUT

Timeout during connection

DRIZZLE_RETURN_INVALID_ARGUMENT

Bad arguments supplied to a function

DRIZZLE_RETURN_SSL_ERROR

An error occurred during SSL handshake

DRIZZLE_RETURN_EOF

No more data to retrieve

DRIZZLE_RETURN_STMT_ERROR

A prepared statement error has occurred

DRIZZLE_RETURN_BINLOG_CRC

A checksum error has occurred in a MySQL 5.6 binlog

DRIZZLE_RETURN_TRUNCATED

The result has been truncated

DRIZZLE_RETURN_INVALID_CONVERSION

The data type cannot be converted into the requested type

DRIZZLE_RETURN_NOT_FOUND

The requested column was not found

5.1.5 Connection

drizzle_charset_t

An ENUM of the possible character set with collation ID

DRIZZLE_CHARSET_BIG5_CHINESE_CI**DRIZZLE_CHARSET_LATIN2_CZECH_CS****DRIZZLE_CHARSET_DEC8_SWEDISH_CI****DRIZZLE_CHARSET_CP850_GENERAL_CI****DRIZZLE_CHARSET_LATIN1_GERMAN1_CI****DRIZZLE_CHARSET_HP8_ENGLISH_CI****DRIZZLE_CHARSET_KOI8R_GENERAL_CI****DRIZZLE_CHARSET_LATIN1_SWEDISH_CI****DRIZZLE_CHARSET_LATIN2_GENERAL_CI****DRIZZLE_CHARSET_SWE7_SWEDISH_CI****DRIZZLE_CHARSET_ASCII_GENERAL_CI****DRIZZLE_CHARSET_UJIS_JAPANESE_CI****DRIZZLE_CHARSET_SJIS_JAPANESE_CI****DRIZZLE_CHARSET_CP1251_BULGARIAN_CI****DRIZZLE_CHARSET_LATIN1_DANISH_CI****DRIZZLE_CHARSET_HEBREW_GENERAL_CI****DRIZZLE_CHARSET_TIS620_THAI_CI****DRIZZLE_CHARSET_EUCKR_KOREAN_CI****DRIZZLE_CHARSET_LATIN7_ESTONIAN_CS****DRIZZLE_CHARSET_LATIN2_HUNGARIAN_CI****DRIZZLE_CHARSET_KOI8U_GENERAL_CI**

DRIZZLE_CHARSET_CP1251_UKRAINIAN_CI
DRIZZLE_CHARSET_GB2312_CHINESE_CI
DRIZZLE_CHARSET_GREEK_GENERAL_CI
DRIZZLE_CHARSET_CP1250_GENERAL_CI
DRIZZLE_CHARSET_LATIN2_CROATIAN_CI
DRIZZLE_CHARSET_GBK_CHINESE_CI
DRIZZLE_CHARSET_CP1257_LITHUANIAN_CI
DRIZZLE_CHARSET_LATIN5_TURKISH_CI
DRIZZLE_CHARSET_LATIN1_GERMAN2_CI
DRIZZLE_CHARSET_ARMSCII8_GENERAL_CI
DRIZZLE_CHARSET_UTF8_GENERAL_CI
DRIZZLE_CHARSET_CP1250_CZECH_CS
DRIZZLE_CHARSET_UCS2_GENERAL_CI
DRIZZLE_CHARSET_CP866_GENERAL_CI
DRIZZLE_CHARSET_KEYBCS2_GENERAL_CI
DRIZZLE_CHARSET_MACCE_GENERAL_CI
DRIZZLE_CHARSET_MACROMAN_GENERAL_CI
DRIZZLE_CHARSET_CP852_GENERAL_CI
DRIZZLE_CHARSET_LATIN7_GENERAL_CI
DRIZZLE_CHARSET_LATIN7_GENERAL_CS
DRIZZLE_CHARSET_MACCE_BIN
DRIZZLE_CHARSET_CP1250_CROATIAN_CI
DRIZZLE_CHARSET_UTF8MB4_GENERAL_CI
DRIZZLE_CHARSET_UTF8MB4_BIN
DRIZZLE_CHARSET_LATIN1_BIN
DRIZZLE_CHARSET_LATIN1_GENERAL_CI
DRIZZLE_CHARSET_LATIN1_GENERAL_CS
DRIZZLE_CHARSET_CP1251_BIN
DRIZZLE_CHARSET_CP1251_GENERAL_CI
DRIZZLE_CHARSET_CP1251_GENERAL_CS
DRIZZLE_CHARSET_MACROMAN_BIN
DRIZZLE_CHARSET_UTF16_GENERAL_CI
DRIZZLE_CHARSET_UTF16_BIN
DRIZZLE_CHARSET_CP1256_GENERAL_CI
DRIZZLE_CHARSET_CP1257_BIN
DRIZZLE_CHARSET_CP1257_GENERAL_CI

DRIZZLE_CHARSET_UTF32_GENERAL_CI
DRIZZLE_CHARSET_UTF32_BIN
DRIZZLE_CHARSET_BINARY
DRIZZLE_CHARSET_ARMSCII8_BIN
DRIZZLE_CHARSET_ASCII_BIN
DRIZZLE_CHARSET_CP1250_BIN
DRIZZLE_CHARSET_CP1256_BIN
DRIZZLE_CHARSET_CP866_BIN
DRIZZLE_CHARSET_DEC8_BIN
DRIZZLE_CHARSET_GREEK_BIN
DRIZZLE_CHARSET_HEBREW_BIN
DRIZZLE_CHARSET_HP8_BIN
DRIZZLE_CHARSET_KEYBCS2_BIN
DRIZZLE_CHARSET_KOI8R_BIN
DRIZZLE_CHARSET_KOI8U_BIN
DRIZZLE_CHARSET_LATIN2_BIN
DRIZZLE_CHARSET_LATIN5_BIN
DRIZZLE_CHARSET_LATIN7_BIN
DRIZZLE_CHARSET_CP850_BIN
DRIZZLE_CHARSET_CP852_BIN
DRIZZLE_CHARSET_SWE7_BIN
DRIZZLE_CHARSET_UTF8_BIN
DRIZZLE_CHARSET_BIG5_BIN
DRIZZLE_CHARSET_EUCKR_BIN
DRIZZLE_CHARSET_GB2312_BIN
DRIZZLE_CHARSET_GBK_BIN
DRIZZLE_CHARSET_SJIS_BIN
DRIZZLE_CHARSET_TIS620_BIN
DRIZZLE_CHARSET_UCS2_BIN
DRIZZLE_CHARSET_UJIS_BIN
DRIZZLE_CHARSET_GEOSTD8_GENERAL_CI
DRIZZLE_CHARSET_GEOSTD8_BIN
DRIZZLE_CHARSET_LATIN1_SPANISH_CI
DRIZZLE_CHARSET_CP932_JAPANESE_CI
DRIZZLE_CHARSET_CP932_BIN
DRIZZLE_CHARSET_EUCJPMS_JAPANESE_CI

DRIZZLE_CHARSET_EUCJPMS_BIN
DRIZZLE_CHARSET_CP1250_POLISH_CI
DRIZZLE_CHARSET_UTF16_UNICODE_CI
DRIZZLE_CHARSET_UTF16_ICELANDIC_CI
DRIZZLE_CHARSET_UTF16_LATVIAN_CI
DRIZZLE_CHARSET_UTF16_ROMANIAN_CI
DRIZZLE_CHARSET_UTF16_SLOVENIAN_CI
DRIZZLE_CHARSET_UTF16_POLISH_CI
DRIZZLE_CHARSET_UTF16_ESTONIAN_CI
DRIZZLE_CHARSET_UTF16_SPANISH_CI
DRIZZLE_CHARSET_UTF16_SWEDISH_CI
DRIZZLE_CHARSET_UTF16_TURKISH_CI
DRIZZLE_CHARSET_UTF16_CZECH_CI
DRIZZLE_CHARSET_UTF16_DANISH_CI
DRIZZLE_CHARSET_UTF16_LITHUANIAN_CI
DRIZZLE_CHARSET_UTF16_SLOVAK_CI
DRIZZLE_CHARSET_UTF16_SPANISH2_CI
DRIZZLE_CHARSET_UTF16_ROMAN_CI
DRIZZLE_CHARSET_UTF16_PERSIAN_CI
DRIZZLE_CHARSET_UTF16_ESPERANTO_CI
DRIZZLE_CHARSET_UTF16_HUNGARIAN_CI
DRIZZLE_CHARSET_UTF16_SINHALA_CI
DRIZZLE_CHARSET_UCS2_UNICODE_CI
DRIZZLE_CHARSET_UCS2_ICELANDIC_CI
DRIZZLE_CHARSET_UCS2_LATVIAN_CI
DRIZZLE_CHARSET_UCS2_ROMANIAN_CI
DRIZZLE_CHARSET_UCS2_SLOVENIAN_CI
DRIZZLE_CHARSET_UCS2_POLISH_CI
DRIZZLE_CHARSET_UCS2_ESTONIAN_CI
DRIZZLE_CHARSET_UCS2_SPANISH_CI
DRIZZLE_CHARSET_UCS2_SWEDISH_CI
DRIZZLE_CHARSET_UCS2_TURKISH_CI
DRIZZLE_CHARSET_UCS2_CZECH_CI
DRIZZLE_CHARSET_UCS2_DANISH_CI
DRIZZLE_CHARSET_UCS2_LITHUANIAN_CI
DRIZZLE_CHARSET_UCS2_SLOVAK_CI

DRIZZLE_CHARSET_UCS2_SPANISH2_CI
DRIZZLE_CHARSET_UCS2_ROMAN_CI
DRIZZLE_CHARSET_UCS2_PERSIAN_CI
DRIZZLE_CHARSET_UCS2_ESPERANTO_CI
DRIZZLE_CHARSET_UCS2_HUNGARIAN_CI
DRIZZLE_CHARSET_UCS2_SINHALA_CI
DRIZZLE_CHARSET_UCS2_GENERAL_MYSQL500_CI
DRIZZLE_CHARSET_UTF32_UNICODE_CI
DRIZZLE_CHARSET_UTF32_ICELANDIC_CI
DRIZZLE_CHARSET_UTF32_LATVIAN_CI
DRIZZLE_CHARSET_UTF32_ROMANIAN_CI
DRIZZLE_CHARSET_UTF32_SLOVENIAN_CI
DRIZZLE_CHARSET_UTF32_POLISH_CI
DRIZZLE_CHARSET_UTF32_ESTONIAN_CI
DRIZZLE_CHARSET_UTF32_SPANISH_CI
DRIZZLE_CHARSET_UTF32_SWEDISH_CI
DRIZZLE_CHARSET_UTF32_TURKISH_CI
DRIZZLE_CHARSET_UTF32_CZECH_CI
DRIZZLE_CHARSET_UTF32_DANISH_CI
DRIZZLE_CHARSET_UTF32_LITHUANIAN_CI
DRIZZLE_CHARSET_UTF32_SLOVAK_CI
DRIZZLE_CHARSET_UTF32_SPANISH2_CI
DRIZZLE_CHARSET_UTF32_ROMAN_CI
DRIZZLE_CHARSET_UTF32_PERSIAN_CI
DRIZZLE_CHARSET_UTF32_ESPERANTO_CI
DRIZZLE_CHARSET_UTF32_HUNGARIAN_CI
DRIZZLE_CHARSET_UTF32_SINHALA_CI
DRIZZLE_CHARSET_UTF8_UNICODE_CI
DRIZZLE_CHARSET_UTF8_ICELANDIC_CI
DRIZZLE_CHARSET_UTF8_LATVIAN_CI
DRIZZLE_CHARSET_UTF8_ROMANIAN_CI
DRIZZLE_CHARSET_UTF8_SLOVENIAN_CI
DRIZZLE_CHARSET_UTF8_POLISH_CI
DRIZZLE_CHARSET_UTF8_ESTONIAN_CI
DRIZZLE_CHARSET_UTF8_SPANISH_CI
DRIZZLE_CHARSET_UTF8_SWEDISH_CI

DRIZZLE_CHARSET_UTF8_TURKISH_CI
DRIZZLE_CHARSET_UTF8_CZECH_CI
DRIZZLE_CHARSET_UTF8_DANISH_CI
DRIZZLE_CHARSET_UTF8_LITHUANIAN_CI
DRIZZLE_CHARSET_UTF8_SLOVAK_CI
DRIZZLE_CHARSET_UTF8_SPANISH2_CI
DRIZZLE_CHARSET_UTF8_ROMAN_CI
DRIZZLE_CHARSET_UTF8_PERSIAN_CI
DRIZZLE_CHARSET_UTF8_ESPERANTO_CI
DRIZZLE_CHARSET_UTF8_HUNGARIAN_CI
DRIZZLE_CHARSET_UTF8_SINHALA_CI
DRIZZLE_CHARSET_UTF8_GENERAL_MYSQL500_CI
DRIZZLE_CHARSET_UTF8MB4_UNICODE_CI
DRIZZLE_CHARSET_UTF8MB4_ICELANDIC_CI
DRIZZLE_CHARSET_UTF8MB4_LATVIAN_CI
DRIZZLE_CHARSET_UTF8MB4_ROMANIAN_CI
DRIZZLE_CHARSET_UTF8MB4_SLOVENIAN_CI
DRIZZLE_CHARSET_UTF8MB4_POLISH_CI
DRIZZLE_CHARSET_UTF8MB4_ESTONIAN_CI
DRIZZLE_CHARSET_UTF8MB4_SPANISH_CI
DRIZZLE_CHARSET_UTF8MB4_SWEDISH_CI
DRIZZLE_CHARSET_UTF8MB4_TURKISH_CI
DRIZZLE_CHARSET_UTF8MB4_CZECH_CI
DRIZZLE_CHARSET_UTF8MB4_DANISH_CI
DRIZZLE_CHARSET_UTF8MB4_LITHUANIAN_CI
DRIZZLE_CHARSET_UTF8MB4_SLOVAK_CI
DRIZZLE_CHARSET_UTF8MB4_SPANISH2_CI
DRIZZLE_CHARSET_UTF8MB4_ROMAN_CI
DRIZZLE_CHARSET_UTF8MB4_PERSIAN_CI
DRIZZLE_CHARSET_UTF8MB4_ESPERANTO_CI
DRIZZLE_CHARSET_UTF8MB4_HUNGARIAN_CI
DRIZZLE_CHARSET_UTF8MB4_SINHALA_CI

drizzle_status_t

An ENUM of connection statuses intended to be used in a bit field

DRIZZLE_CON_STATUS_NONE

No status set

DRIZZLE_CON_STATUS_IN_TRANS

In a transaction

DRIZZLE_CON_STATUS_AUTOCOMMIT

Autocommit is enabled

DRIZZLE_CON_STATUS_MORE_RESULTS_EXISTS

There are more result sets available

DRIZZLE_CON_STATUS_QUERY_NO_GOOD_INDEX_USED

No good index couldn't be used

DRIZZLE_CON_STATUS_QUERY_NO_INDEX_USED

No index was used

DRIZZLE_CON_STATUS_CURSOR_EXISTS

A cursor is available

DRIZZLE_CON_STATUS_LAST_ROW_SENT

The last row has been sent to the client

DRIZZLE_CON_STATUS_DB_DROPPED

The database has been dropped

DRIZZLE_CON_STATUS_NO_BACKSLASH_ESCAPES

NO_BACKSLASH_ESCAPES SQL mode set

DRIZZLE_CON_STATUS_QUERY_WAS_SLOW

Query hit the slow query timeout

drizzle_capabilities_t

An ENUM of connection capabilities intended to be used in a bit field

DRIZZLE_CAPABILITIES_NONE

No capabilities set

DRIZZLE_CAPABILITIES_LONG_PASSWORD

Long password support

DRIZZLE_CAPABILITIES_FOUND_ROWS

FOUND_ROWS support

DRIZZLE_CAPABILITIES_LONG_FLAG

Get all column flags

DRIZZLE_CAPABILITIES_IGNORE_SPACE

Ignore spaces before open brackets

DRIZZLE_CAPABILITIES_CONNECT_WITH_DB

A database can be specified upon connect

DRIZZLE_CAPABILITIES_NO_SCHEMA

Disable access to database.table.column way of accessing things

DRIZZLE_CAPABILITIES_COMPRESS

Enable compression protocol

DRIZZLE_CAPABILITIES_ODBC

An ODBC client

DRIZZLE_CAPABILITIES_LOCAL_FILES

Enables LOAD DATA LOCAL

DRIZZLE_CAPABILITIES_PROTOCOL_41

MySQL 4.1 and higher protocol

DRIZZLE_CAPABILITIES_INTERACTIVE

An interactive client

DRIZZLE_CAPABILITIES_SSL

Use SSL

DRIZZLE_CAPABILITIES_IGNORE_SIGPIPE

Ignore sigpipe

DRIZZLE_CAPABILITIES_TRANSACTIONS

Client understands transactions

DRIZZLE_CAPABILITIES_RESERVED

Unused

DRIZZLE_CAPABILITIES_SECURE_CONNECTION

MySQL 4.1 and higher authentication

DRIZZLE_CAPABILITIES_MULTI_STATEMENTS

Enable multiple statement support

DRIZZLE_CAPABILITIES_MULTI_RESULTS

Enable multiple result sets

DRIZZLE_CAPABILITIES_PS_MULTI_RESULTS

DRIZZLE_CAPABILITIES_PLUGIN_AUTH

Enable plugin authentication

DRIZZLE_CAPABILITIES_SSL_VERIFY_SERVER_CERT

Verify SSL cert

DRIZZLE_CAPABILITIES_REMEMBER_OPTIONS

DRIZZLE_CAPABILITIES_CLIENT

Enables the following:

*DRIZZLE_CAPABILITIES_LONG_PASSWORD, DRIZZLE_CAPABILITIES_FOUND_ROWS,
DRIZZLE_CAPABILITIES_LONG_FLAG, DRIZZLE_CAPABILITIES_CONNECT_WITH_DB,
DRIZZLE_CAPABILITIES_PLUGIN_AUTH, DRIZZLE_CAPABILITIES_TRANSACTIONS,
DRIZZLE_CAPABILITIES_PROTOCOL_41, DRIZZLE_CAPABILITIES_SECURE_CONNECTION*

drizzle_ssl_state_t

An enum of SSL States

DRIZZLE_SSL_STATE_NONE

SSL connection is not initialized

DRIZZLE_SSL_STATE_HANDSHAKE_COMPLETE

SSL connection is established

drizzle_socket_owner_t

Owner of socket connection

DRIZZLE_SOCKET_OWNER_NATIVE

DRIZZLE_SOCKET_OWNER_CLIENT

(deprecated) drizzle_socket_owner

typedef of *drizzle_socket_owner_t*

drizzle_socket_option_t

An ENUM of socket connection options

DRIZZLE_SOCKET_OPTION_KEEPIDLE

The socket connection timeout

DRIZZLE_SOCKET_OPTION_KEEPCNT

Number of probes before dropping connection

DRIZZLE_SOCKET_OPTION_KEEPINTVL

TCP interval between probes

DRIZZLE_SOCKET_OPTION_TIMEOUT

TCP Keep-alive timeout

(deprecated) drizzle_socket_option

typedef of *drizzle_socket_option_t*

5.1.6 Query

drizzle_field_t

Field data (an alias for `char*`)

drizzle_row_t

Row data (an array of *drizzle_field_t*)

drizzle_column_type_t

An ENUM of column types

DRIZZLE_COLUMN_TYPE_DECIMAL

An old style decimal type

DRIZZLE_COLUMN_TYPE_TINY

A tiny int

DRIZZLE_COLUMN_TYPE_SHORT

A short int

DRIZZLE_COLUMN_TYPE_LONG

A long int

DRIZZLE_COLUMN_TYPE_FLOAT

A float

DRIZZLE_COLUMN_TYPE_DOUBLE

A double

DRIZZLE_COLUMN_TYPE_NULL

A NULL

DRIZZLE_COLUMN_TYPE_TIMESTAMP

A timestamp

DRIZZLE_COLUMN_TYPE_LONGLONG

A bigint

DRIZZLE_COLUMN_TYPE_INT24**DRIZZLE_COLUMN_TYPE_DATE****DRIZZLE_COLUMN_TYPE_TIME****DRIZZLE_COLUMN_TYPE_DATETIME**

DRIZZLE_COLUMN_TYPE_YEAR
DRIZZLE_COLUMN_TYPE_NEWDATE
DRIZZLE_COLUMN_TYPE_VARCHAR
DRIZZLE_COLUMN_TYPE_BIT
DRIZZLE_COLUMN_TYPE_NEWDECIMAL
DRIZZLE_COLUMN_TYPE_ENUM
DRIZZLE_COLUMN_TYPE_SET
DRIZZLE_COLUMN_TYPE_TINY_BLOB
DRIZZLE_COLUMN_TYPE_MEDIUM_BLOB
DRIZZLE_COLUMN_TYPE_LONG_BLOB
DRIZZLE_COLUMN_TYPE_BLOB
DRIZZLE_COLUMN_TYPE_VAR_STRING
Text column type
DRIZZLE_COLUMN_TYPE_STRING
DRIZZLE_COLUMN_TYPE_GEOMETRY

drizzle_column_options_t

DRIZZLE_COLUMN_UNUSED

drizzle_column_flags_t

An ENUM of column flags intended to be used in a bit field

DRIZZLE_COLUMN_FLAGS_NONE
No flags set

DRIZZLE_COLUMN_FLAGS_NOT_NULL
Column is not NULL

DRIZZLE_COLUMN_FLAGS_PRI_KEY
Column is a primary key

DRIZZLE_COLUMN_FLAGS_UNIQUE_KEY
Column is a unique key

DRIZZLE_COLUMN_FLAGS_MULTIPLE_KEY
Column is part of a multi-part key

DRIZZLE_COLUMN_FLAGS_BLOB
Column is a blob

DRIZZLE_COLUMN_FLAGS_UNSIGNED
Column is unsigned

DRIZZLE_COLUMN_FLAGS_ZEROFILL
Column has a zerofill

DRIZZLE_COLUMN_FLAGS_BINARY

DRIZZLE_COLUMN_FLAGS_ENUM
Column is an ENUM

DRIZZLE_COLUMN_FLAGS_AUTO_INCREMENT

Column has auto increment

DRIZZLE_COLUMN_FLAGS_TIMESTAMP

Column in a timestamp

DRIZZLE_COLUMN_FLAGS_SET

Column is a SET data type

DRIZZLE_COLUMN_FLAGS_NO_DEFAULT_VALUE

Column has no default value

DRIZZLE_COLUMN_FLAGS_ON_UPDATE_NOW

Column has on update now timestamp

DRIZZLE_COLUMN_FLAGS_PART_KEY

Column is part of a key

DRIZZLE_COLUMN_FLAGS_NUM

Column is a number

Note: Group and num are the same flag

DRIZZLE_COLUMN_FLAGS_GROUP

Note: Group and num are the same flag

DRIZZLE_COLUMN_FLAGS_UNIQUE

DRIZZLE_COLUMN_FLAGS_BINCMP

DRIZZLE_COLUMN_FLAGS_GET_FIXED_FIELDS

DRIZZLE_COLUMN_FLAGS_IN_PART_FUNC

DRIZZLE_COLUMN_FLAGS_IN_ADD_INDEX

DRIZZLE_COLUMN_FLAGS_RENAMED

drizzle_result_options_t

An ENUM used to the indicate the state of a result

DRIZZLE_RESULT_NONE

DRIZZLE_RESULT_SKIP_COLUMN

DRIZZLE_RESULT_BUFFER_COLUMN

DRIZZLE_RESULT_BUFFER_ROW

DRIZZLE_RESULT_EOF_PACKET

DRIZZLE_RESULT_ROW_BREAK

DRIZZLE_RESULT_BINARY_ROWS

5.1.7 Prepared Statement

drizzle_stmt_state_t

An internal state for prepared statements

5.1.8 Binlog

drizzle_binlog_event_types_t

An ENUM of binlog event types

DRIZZLE_EVENT_TYPE_UNKNOWN

An unknown event

DRIZZLE_EVENT_TYPE_START

A binlog start event

DRIZZLE_EVENT_TYPE_QUERY

A MySQL query for SBR

DRIZZLE_EVENT_TYPE_STOP

Binlog end event

DRIZZLE_EVENT_TYPE_ROTATE

Binlog file rotate event

DRIZZLE_EVENT_TYPE_INTVAR

Insert ID event

DRIZZLE_EVENT_TYPE_LOAD

Load data from file event

DRIZZLE_EVENT_TYPE_CREATE_FILE

Create file event

DRIZZLE_EVENT_TYPE_APPEND_BLOCK

Append block data to a file

DRIZZLE_EVENT_TYPE_EXEC_LOAD

Exec load event

DRIZZLE_EVENT_TYPE_DELETE_FILE

Delete file event

DRIZZLE_EVENT_TYPE_NEW_LOAD

New load data from file event

DRIZZLE_EVENT_TYPE_RAND

Seeds for RAND() functions

DRIZZLE_EVENT_TYPE_USER_VAR

A user variable

DRIZZLE_EVENT_TYPE_FORMAT_DESCRIPTION

A description of the binlog file (a replacement for DRIZZLE_EVENT_TYPE_START in MySQL 5.0 onwards)

DRIZZLE_EVENT_TYPE_XID

XA Transaction ID

DRIZZLE_EVENT_TYPE_BEGIN_LOAD_QUERY

Truncate file and save block data

DRIZZLE_EVENT_TYPE_EXECUTE_LOAD_QUERY

Execute load query event

DRIZZLE_EVENT_TYPE_TABLE_MAP

A table map event for RBR

DRIZZLE_EVENT_TYPE_OBSOLETE_WRITE_ROWS

RBR Write rows event for MySQL 5.1 pre-release

DRIZZLE_EVENT_TYPE_OBSOLETE_UPDATE_ROWS

RBR Update rows event for MySQL 5.1 pre-release

DRIZZLE_EVENT_TYPE_OBSOLETE_DELETE_ROWS

RBR Delete rows event for MySQL 5.1 pre-release

DRIZZLE_EVENT_TYPE_V1_WRITE_ROWS

RBR Write rows event

DRIZZLE_EVENT_TYPE_V1_UPDATE_ROWS

RBR Update rows event

DRIZZLE_EVENT_TYPE_V1_DELETE_ROWS

RBR Delete rows event

DRIZZLE_EVENT_TYPE_INCIDENT

Replication incident message

DRIZZLE_EVENT_TYPE_HEARTBEAT

Replication heartbeat event

DRIZZLE_EVENT_TYPE_IGNOREABLE

DRIZZLE_EVENT_TYPE_ROWS_QUERY

DRIZZLE_EVENT_TYPE_V2_WRITE_ROWS

A MySQL 5.6 RBR Write rows event

DRIZZLE_EVENT_TYPE_V2_UPDATE_ROWS

A MySQL 5.6 RBR Update rows event

DRIZZLE_EVENT_TYPE_V2_DELETE_ROWS

A MySQL 5.6 RBR Delete rows event

DRIZZLE_EVENT_TYPE_GTID

DRIZZLE_EVENT_TYPE_ANONYMOUS_GTID

DRIZZLE_EVENT_TYPE_PREVIOUS_GTIDS

drizzle_binlog_event_positions_t

DRIZZLE_EVENT_POSITION_TIMESTAMP

DRIZZLE_EVENT_POSITION_TYPE

DRIZZLE_EVENT_POSITION_SERVERID

DRIZZLE_EVENT_POSITION_LENGTH

DRIZZLE_EVENT_POSITION_NEXT

DRIZZLE_EVENT_POSITION_FLAGS

DRIZZLE_EVENT_POSITION_EXTRA_FLAGS

5.2 Library Functions

5.2.1 Introduction

This section outlines the functions related to the Libdrizzle Redux library

5.2.2 Functions

void **drizzle_library_init** (void)

Setup of the SSL and Windows connection libraries. Should be run before any Libdrizzle Redux function. Only required if using SSL or Windows.

void **drizzle_library_deinit** (void)

Deinitialize the library. Only required for Windows

const char* **drizzle_version** (void)

Gives the version string for the Libdrizzle Redux library

Returns A string of the library version

const char* **drizzle_bugreport** (void)

Gives the URL for reporting library bugs

Returns A string containing the bug report URL

const char* **drizzle_verbose_name** (*drizzle_verbose_t verbose*)

Gives the verbosity name for a given verbosity type

Returns A string containing the verbosity name

5.3 Connection Functions

5.3.1 Introduction

This section outlines the connection functions

5.3.2 Structs

drizzle_st

The internal drizzle connection object struct

drizzle_options_st

The internal structure containing connection options

5.3.3 Functions

*drizzle_st** **drizzle_create** (const char **host*, in_port_t *port*, const char **user*, const char **password*, const char **db*, *drizzle_options_st* **options*)

Creates a connection connection object. If a path beginning with / is given as the host the library will connect as a UDS socket. Otherwise a TCP/IP connection is made.

Note: a connection does not happen until the first query or an explicit `drizzle_connect()` call is made

Parameters

- **host** – The socket path, hostname or IP of the server
- **port** – The port number of the server (if TCP/IP)
- **user** – The username of the server
- **password** – The password of the server
- **db** – The default DB to connect to on the server
- **options** – A pointer to a `drizzle_options_st` created using `drizzle_options_create()` or NULL

Returns A newly allocated and setup connection object

int **drizzle_fd** (const `drizzle_st` *con)

Get file descriptor for connection.

Parameters

- **con** – Connection structure previously initialized with `drizzle_create()`.

Returns File descriptor of connection, or -1 if not active.

int **drizzle_timeout** (const `drizzle_st` *con)

Gets the current connection timeout set in the connection object

Parameters

- **con** – A connection object

Returns The current timeout

void **drizzle_set_timeout** (`drizzle_st` *con, int timeout)

Sets the connection timeout for the connection object

Parameters

- **con** – A connection object
- **timeout** – The new timeout to set

`drizzle_verbosity_t` **drizzle_verbosity** (const `drizzle_st` *con)

Gets the verbosity level set in the connection object

Parameters

- **con** – A connection object

Returns The verbosity level from `drizzle_verbosity_t`

void **drizzle_set_verbosity** (`drizzle_st` *con, `drizzle_verbosity_t` verbose)

Sets the verbosity level for the connection object

Parameters

- **con** – A connection object
- **verbose** – The verbosity level from `drizzle_verbosity_t`

void **drizzle_set_log_fn** (*drizzle_st* *con, *drizzle_log_fn* *function, void *context)

Sets a callback function for log handling

Parameters

- **con** – A connection object
- **function** – The function to use in the format of *drizzle_log_fn()*
- **context** – A pointer to data to pass to the log function

void **drizzle_set_event_watch_fn** (*drizzle_st* *con, *drizzle_event_watch_fn* *function, void *context)

Set a custom I/O event watcher function for a drizzle structure

Parameters

- **con** – Drizzle structure previously initialized with *drizzle_create()*.
- **function** – Function to call when there is an I/O event, in the form of *drizzle_event_watch_fn()*
- **context** – Argument to pass into the callback function.

drizzle_return_t **drizzle_set_events** (*drizzle_st* *con, short events)

Set events to be watched for a connection.

Parameters

- **con** – Connection structure previously initialized with *drizzle_create()*.
- **events** – Bitfield of poll() events to watch.

Returns Standard drizzle return value.

drizzle_return_t **drizzle_set_revents** (*drizzle_st* *con, short revents)

Set events that are ready for a connection. This is used with the external event callbacks. See *drizzle_set_event_watch_fn()*.

Parameters

- **con** – Connection structure previously initialized with *drizzle_create()*.
- **revents** – Bitfield of poll() events that were detected.

Returns Standard drizzle return value.

const char* **drizzle_error** (const *drizzle_st* *con)

Get the last error from a connection

Parameters

- **con** – A connection object

Returns A string containing the error message

int **drizzle_errno** (const *drizzle_st* *con)

Get the last OS error code from a connection

Parameters

- **con** – A connection object

Returns The OS error code

uint16_t **drizzle_error_code** (const *drizzle_st* *con)

Gets the last error code from a connection

Parameters

- **con** – A connection object

Returns The server error code

const char* **drizzle_sqlstate** (const *drizzle_st* *con)
Gets the last sqlstate from a connection

Parameters

- **con** – A connection object

Returns A string containing the sqlstate

drizzle_options_st ***drizzle_options_create** (void)
Create a new connection options object

Returns The new connection options object

void **drizzle_options_destroy** (*drizzle_options_st* *options)
Destroys a connection options object

Parameters

- **options** – The options object to be destroyed

void **drizzle_socket_set_options** (*drizzle_options_st* *options, int wait_timeout, int keepidle,
int keepcnt, int keepintvl)
Sets several options for the socket connection

Parameters

- **options** – An initialized options structure
- **wait_timeout** – The timeout (in seconds) for setsockopt calls with option values: SO_SNDTIMEO, SO_RCVTIMEO, SO_LINGER
- **keepidle** – The time (in seconds) the connection needs to remain idle before TCP starts sending keepalive probes
- **keepcnt** – The maximum number of keepalive probes TCP should send before dropping the connection.
- **keepintvl** – The time (in seconds) between individual keepalive probes

void **drizzle_socket_set_option** (*drizzle_st* *con, *drizzle_socket_option_t* option, int value)
Sets the value of a socket option.

Note: The available options to set are:

DRIZZLE_SOCKET_OPTION_TIMEOUT : The timeout (in seconds) for setsockopt calls with option values: SO_SNDTIMEO, SO_RCVTIMEO, SO_LINGER

DRIZZLE_SOCKET_OPTION_KEEPIDLE : The time (in seconds) the connection needs to remain idle before TCP starts sending keepalive probes

DRIZZLE_SOCKET_OPTION_KEEPCNT : The maximum number of keepalive probes TCP should send before dropping the connection.

DRIZZLE_SOCKET_OPTION_KEEPINTVL : The time (in seconds) between individual keepalive probes

Parameters

- **con** – Connection structure previously initialized with *drizzle_create()*.
- **option** – the option to set the value for

- **value** – the value to set

int **drizzle_socket_get_option** (*drizzle_st* *con, *drizzle_socket_option_t* option)

Gets the value of a socket option. See *drizzle_socket_set_options()* for a description of the available options

Parameters

- **con** – Connection structure previously initialized with *drizzle_create()*.
- **option** – option to get the value for

Returns The value of the option, or -1 if the specified option doesn't exist

void **drizzle_options_set_non_blocking** (*drizzle_options_st* *options, bool state)

Sets/unsets non-blocking connect option

Parameters

- **options** – The options object to modify
- **state** – Set option to true/false

bool **drizzle_options_get_non_blocking** (*drizzle_options_st* *options)

Gets the non-blocking connect option

Parameters

- **options** – The options object to get the value from

Returns The state of the non-blocking option

void **drizzle_options_set_raw_scramble** (*drizzle_options_st* *options, bool state)

Sets/unsets the raw scramble connect option

Parameters

- **options** – The options object to modify
- **state** – Set to true/false

bool **drizzle_options_get_raw_scramble** (*drizzle_options_st* *options)

Gets the raw scramble connect option

Parameters

- **options** – The options object to get the value from

Returns The state of the raw scramble option

void **drizzle_options_set_found_rows** (*drizzle_options_st* *options, bool state)

Sets/unsets the found rows connect option

Parameters

- **options** – The options object to modify
- **state** – Set to true/false

bool **drizzle_options_get_found_rows** (*drizzle_options_st* *options)

Gets the found rows connect option

Parameters

- **options** – The options object to get the value from

Returns The state of the found rows option

void **drizzle_options_set_interactive** (*drizzle_options_st* *options, bool state)
Sets/unsets the interactive connect option

Parameters

- **options** – The options object to modify
- **state** – Set to true/false

bool **drizzle_options_get_interactive** (*drizzle_options_st* *options)
Gets the interactive connect option

Parameters

- **options** – The options object to get the value from

Returns The state of the interactive option

void **drizzle_options_set_multi_statements** (*drizzle_options_st* *options, bool state)
Sets/unsets the multi-statements connect option

Parameters

- **options** – The options object to modify

Parma state Set to true/false

bool **drizzle_options_get_multi_statements** (*drizzle_options_st* *options)
Gets the multi-statements connect option

Parameters

- **options** – The options object to get the value from

Returns The state of the multi-statements option

void **drizzle_options_set_auth_plugin** (*drizzle_options_st* *options, bool state)
Sets/unsets the auth plugin connect option

Parameters

- **options** – The options object to modify
- **state** – Set to true/false

bool **drizzle_options_get_auth_plugin** (*drizzle_options_st* *options)
Gets the auth plugin connect option

Parameters

- **options** – The options object to get the value from

Returns The state of the auth plugin option

void **drizzle_options_set_socket_owner** (*drizzle_options_st* *options, *drizzle_socket_owner_t* owner)

Sets the owner of the socket connection

Parameters

- **options** – The options object to modify
- **owner** – The owner of the socket connection

drizzle_socket_owner_t **drizzle_options_get_socket_owner** (*drizzle_options_st* *options)
Gets the owner of the socket connection

Parameters

- **options** – The options object to get the value from

Returns The owner of the socket connection

const char* **drizzle_host** (const *drizzle_st* *con)

Gets the host name from a TCP/IP connection

Parameters

- **con** – A connection object

Returns A string containing the host name or NULL for a UDS connection

in_port_t **drizzle_port** (const *drizzle_st* *con)

Gets the port number from a TCP/IP connection

Parameters

- **con** – A connection object

Returns The port number or 0 for a UDS connection

const char* **drizzle_user** (const *drizzle_st* *con)

Gets the user name used at connection time

Parameters

- **con** – A connection object

Returns A string containing the user name

const char* **drizzle_db** (const *drizzle_st* *con)

Gets the default database used at connection time

Parameters

- **con** – A connection object

Returns A string containing the DB name

void ***drizzle_context** (const *drizzle_st* *con)

Get application context pointer for a connection.

Parameters

- **con** – Connection structure previously initialized with *drizzle_create()*.

Returns Application context with this connection.

void **drizzle_set_context** (*drizzle_st* *con, void *context)

Set application context pointer for a connection.

Parameters

- **con** – Connection structure previously initialized with *drizzle_create()*.
- **context** – Application context to use with this connection.

void **drizzle_set_context_free_fn** (*drizzle_st* *con, drizzle_context_free_fn *function)

Set callback function when the context pointer should be freed.

Parameters

- **con** – Connection structure previously initialized with *drizzle_create()*.
- **function** – Function to call to clean up connection context.

uint8_t **drizzle_protocol_version** (const *drizzle_st* *con)

Gets the protocol version used for a connection

Parameters

- **con** – A connection object

Returns The protocol version

const char* **drizzle_server_version** (const *drizzle_st* *con)

Gets the server version string for a connection

Parameters

- **con** – A connection object

Returns A string containing the server version

uint32_t **drizzle_server_version_number** (const *drizzle_st* *con)

Gets the server version number for a connection

Parameters

- **con** – A connection object

Returns An integer containing the server version number

uint32_t **drizzle_thread_id** (const *drizzle_st* *con)

Gets the server thread ID for a connection

Parameters

- **con** – A connection object

Returns The server thread ID

const unsigned char ***drizzle_scramble** (const *drizzle_st* *con)

Get scramble buffer for a connection.

Parameters

- **con** – Connection structure previously initialized with *drizzle_create()*.

Returns Scramble buffer for connection.

drizzle_capabilities_t **drizzle_capabilities** (const *drizzle_st* *con)

Gets the server capabilities for a connection

Parameters

- **con** – A connection object

Returns A bit field of capabilities

drizzle_charset_t **drizzle_charset** (const *drizzle_st* *con)

Gets the character set ID for the connection

Parameters

- **con** – A connection object

Returns The character set used

drizzle_status_t **drizzle_status** (const *drizzle_st* *con)

Gets the status of the connection

Parameters

- **con** – A connection object

Returns The status of the connection

`uint32_t drizzle_max_packet_size (const drizzle_st *con)`

Gets the max packet size for a connection

Parameters

- **con** – A connection object

Returns The max packet size for the connection

`drizzle_return_t drizzle_connect (drizzle_st *con)`

Open connection to the specified server

Parameters

- **con** – A connection object

Returns A *drizzle_return_t* status. *DRIZZLE_RETURN_OK* upon success

`drizzle_return_t drizzle_wait (drizzle_st *con)`

Wait for I/O on connections.

Parameters

- **con** – Drizzle structure previously initialized with *drizzle_create()*.

Returns Standard drizzle return value.

`drizzle_st *drizzle_ready (drizzle_st *con)`

Get next connection that is ready for I/O.

Parameters

- **con** – Drizzle structure previously initialized with *drizzle_create()*.

Returns Connection that is ready for I/O, or NULL if there are none.

`drizzle_return_t drizzle_close (drizzle_st *con)`

Gracefully disconnect from a server (leaves the connection object available for a reconnect)

Parameters

- **con** – A connection object

Returns A *drizzle_return_t* response for the quit command sent to the server

`drizzle_return_t drizzle_quit (drizzle_st *con)`

Gracefully disconnect from a server and free the connection object

Parameters

- **con** – A connection object

Returns A *drizzle_return_t* response for the quit command sent to the server

`drizzle_return_t drizzle_select_db (drizzle_st *con, const char *db)`

Change the current default database

Parameters

- **con** – A connection object
- **db** – The new default database

Returns A *drizzle_return_t* response

`drizzle_result_st* drizzle_shutdown (drizzle_st *con, drizzle_return_t *ret_ptr)`

Send a shutdown command to the server

Parameters

- **con** – A connection object
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into

Returns A newly allocated result object

*drizzle_result_st** **drizzle_kill** (*drizzle_st* *con, uint32_t connection_id, *drizzle_return_t* *ret_ptr)

Sends a query kill command to the server

Parameters

- **con** – A connection object
- **connection_id** – The connection ID to kill a query from
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into

Returns A newly allocated result object

*drizzle_result_st** **drizzle_ping** (*drizzle_st* *con, *drizzle_return_t* *ret_ptr)

Sends a ping to the server

Parameters

- **con** – A connection object
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into

Returns A newly allocated result object

const char ***drizzle_strerror** (const *drizzle_return_t* ret)

Get detailed error description

Parameters

- **ret** – A libdrizzle return value

Returns description of libdrizzle error

5.3.4 Callback Functions

These are templates to be used when creating callback functions for the Libdrizzle Redux library.

void **drizzle_log_fn** (const char *log_buffer, *drizzle_verbose_t* verbose, void *context)

The format of a callback function for log handling

Parameters

- **log_buffer** – The log message passed to the function
- **verbose** – The verbosity level of the message
- **context** – A pointer to data set in *drizzle_set_log_fn()*

drizzle_return_t **drizzle_event_watch_fn** (*drizzle_st* *con, short events, void *context)

The format of a function to register or deregister interest in file descriptor events

Parameters

- **con** – Connection that has changed the events it is interested in. Use *drizzle_fd()* to get the file descriptor.
- **events** – A bit mask of POLLIN | POLLOUT, specifying if the connection is waiting for read or write events.
- **context** – Application context pointer registered with *drizzle_set_event_watch_fn()*

Returns `DRIZZLE_RETURN_OK` if successful.

5.4 Query Functions

5.4.1 Introduction

This section outlines the query and result functions

5.4.2 Structs

drizzle_query_st

The internal query object struct

drizzle_result_st

The internal result object struct

drizzle_column_st

The internal column object struct

5.4.3 Functions

drizzle_return_t **drizzle_set_ssl** (*drizzle_st* *con, const char *key, const char *cert, const char *ca, const char *capath, const char *cipher)

Sets the SSL data

Parameters

- **con** – A connection object
- **key** – The path to a key file
- **cert** – The path to a certificate file
- **ca** – The path to a certificate authority file
- **capath** – The path to a directory that contains trusted CA certificate files
- **cipher** – A list of allowed ciphers for SSL encryption

Returns A return status code, `DRIZZLE_RETURN_OK` upon success

*drizzle_result_st** **drizzle_query** (*drizzle_st* *con, const char *query, size_t size, *drizzle_return_t* *ret_ptr)

Executes a query and returns a newly allocated result struct

Parameters

- **con** – A connection object
- **query** – The query to execute
- **size** – The length of the query string, if set to 0 then `strlen()` is used to calculate the length
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into

Returns A newly allocated result object

`ssize_t drizzle_escape_str(drizzle_st *con, char **to, const char *from, const size_t from_size, bool is_pattern)`

Escape a string for an **SQL** query, optionally for pattern matching.

This function escapes the following characters:

```
'\0' (0x00), '\ ' (0x27), '"' (0x22), '\b' (0x08), '\n' (0x0A),
'\r' (0x0D), '\t' (0x09), '\z' (0x26), '\\ ' (0x5C).
```

In case **is_pattern** is set to `true`, `'%'` (0x25) and `'_'` (0x5F) will be escaped as well.

The **to** parameter is allocated by the function and needs to be freed by the application when finished with.

Parameters

- **con** – a connection object
- **to** – the destination string
- **from** – the source string
- **from_size** – the length of the source string
- **is_pattern** – whether to escape % and _. If set to `true`, they will be escaped, so the string can be used in a **LIKE** clause for example

Returns the length of the **to** string or `-1` upon error due to empty parameters or overflow

`ssize_t drizzle_escape_string(drizzle_st *con, char **to, const const char *from, const size_t from_size)`

Function wrapper which calls `drizzle_escape_str()` with `is_pattern=false`.

`void drizzle_result_free(drizzle_result_st *result)`

Frees a result object

Parameters

- **result** – the result set to free

`void drizzle_result_free_all(drizzle_st *con)`

Frees all result objects for a given connection object

Parameters

- **con** – A connection object

`drizzle_st* drizzle_result_drizzle_con(drizzle_result_st *result)`

Gets the connection object from a given result object

Parameters

- **result** – A result object

Returns The connection object associated to the result object

`bool drizzle_result_eof(drizzle_result_st *result)`

Tests to see if an EOF packet has been hit

Parameters

- **result** – A result object

Returns true on EOF or false

`const char* drizzle_result_message(drizzle_result_st *result)`

Get error or information message from result set

Parameters

- **result** – A result object

Returns The message to be returned

uint16_t **drizzle_result_error_code** (*drizzle_result_st *result*)

Gets the error code from a result set

Parameters

- **result** – A result object

Returns The error code

const char* **drizzle_result_sqlstate** (*drizzle_result_st *result*)

Gets the SQL state from a result set

Parameters

- **result** – A result object

Returns The SQL state string

uint16_t **drizzle_result_warning_count** (*drizzle_result_st *result*)

Gets the warning count from a result set

Parameters

- **result** – A result object

Returns The warning count

uint64_t **drizzle_result_insert_id** (*drizzle_result_st *result*)

Gets the insert ID for an auto_increment column in a result set

Note: With a MySQL server this returns the first ID with multiple inserts in a query.

Parameters

- **result** – A result object

Returns The insert ID

uint64_t **drizzle_result_affected_rows** (*drizzle_result_st *result*)

Gets the affected row count from a result set

Parameters

- **result** – A result object

Returns The affected row count

uint16_t **drizzle_result_column_count** (*drizzle_result_st *result*)

Gets the column count from a result set

Parameters

- **result** – A result object

Returns The column count

uint64_t **drizzle_result_row_count** (*drizzle_result_st *result*)

Gets the row count from a result set buffered with *drizzle_result_buffer()*

Parameters

- **result** – A result object

Returns The row count

*drizzle_result_st** **drizzle_result_read** (*drizzle_st* *con, *drizzle_return_t* *ret_ptr)

Reads the next result in a multi-result return

Parameters

- **con** – A connection object
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into

Returns The result struct for the new object

drizzle_return_t **drizzle_result_buffer** (*drizzle_result_st* *result)

Buffers a result set

Parameters

- **result** – A result object

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

size_t **drizzle_result_row_size** (*drizzle_result_st* *result)

Get result row packet size in bytes.

Parameters

- **result** – Caller allocated structure.

Returns size in bytes else 0

*drizzle_result_st** **drizzle_column_drizzle_result** (*drizzle_column_st* *column)

Gets a result set for a given column object

Parameters

- **column** – A column object

Returns A result object

const char* **drizzle_column_catalog** (*drizzle_column_st* *column)

Gets the catalog name for a given column

Parameters

- **column** – A column object

Returns The catalog name

const char* **drizzle_column_db** (*drizzle_column_st* *column)

Gets the database name for a given column

Parameters

- **column** – A column object

Returns The database name

const char* **drizzle_column_table** (*drizzle_column_st* *column)

Get the table name (or table alias) for a given column

Parameters

- **column** – A column object

Returns The table name

const char* **drizzle_column_orig_table** (*drizzle_column_st* *column)
Gets the original table name (if an alias has been used) for a given column

Parameters

- **column** – A column object

Returns The original table name

const char* **drizzle_column_name** (*drizzle_column_st* *column)
Gets the column name (or column alias) for a given column

Parameters

- **column** – A column object

Returns The column name

const char* **drizzle_column_orig_name** (*drizzle_column_st* *column)
Gets the original column name (if an alias has been used) for a given column

Parameters

- **column** – A column object

Returns The original column name

drizzle_charset_t **drizzle_column_charset** (*drizzle_column_st* *column)
Gets the character set ID for a given column

Parameters

- **column** – A column object

Returns The character set ID

uint32_t **drizzle_column_size** (*drizzle_column_st* *column)
Gets the size of a given column

Parameters

- **column** – A column object

Returns The column size

size_t **drizzle_column_max_size** (*drizzle_column_st* *column)
Gets the maximum size of a given column

Parameters

- **column** – A column object

Returns The maximum size

drizzle_column_type_t **drizzle_column_type** (*drizzle_column_st* *column)
Gets the type of data for the column

Parameters

- **column** – A column object

Returns The column type

const char* **drizzle_column_type_str** (*drizzle_column_type_t* type)
Get a column type as string

Parameters

- **type** – The table column type

Returns The type of the column in human readable format

drizzle_column_flags_t **drizzle_column_flags** (*drizzle_column_st* *column)

Gets the flags for a given column

Parameters

- **column** – A column object

Returns The column flags

uint8_t **drizzle_column_decimals** (*drizzle_column_st* *column)

Gets the number of decimal places for a given column

Parameters

- **column** – A column object

Returns The number of decimal places

const unsigned char* **drizzle_column_default_value** (*drizzle_column_st* *column, size_t *size)

Gets the default value for a given column

Parameters

- **column** – A column object

Returns A string containing the default value

drizzle_return_t **drizzle_column_skip** (*drizzle_result_st* *result)

Skips the next column in a result set when using *drizzle_column_read()* to get the column data

Parameters

- **result** – A result object

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

drizzle_return_t **drizzle_column_skip_all** (*drizzle_result_st* *result)

Skips all columns in a result set when using *drizzle_column_read()* to get the column data

Parameters

- **result** – pointer to the structure to read from.

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

void **drizzle_column_free** (*drizzle_column_st* *column)

Frees a column when using *drizzle_column_read()* to get the column data

Parameters

- **column** – The column to be freed

*drizzle_column_st** **drizzle_column_read** (*drizzle_result_st* *result, *drizzle_return_t* *ret_ptr)

Reads a column from network buffer

Parameters

- **result** – A result object
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into

Returns A newly allocated column

drizzle_return_t **drizzle_column_buffer** (*drizzle_result_st* *result)

Buffers all the columns for a result set

Parameters

- **result** – A result object

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

*drizzle_column_st** **drizzle_column_next** (*drizzle_result_st* *result)

Gets the next column in a buffered column result set

Parameters

- **result** – A result object

Returns A column object

*drizzle_column_st** **drizzle_column_prev** (*drizzle_result_st* *result)

Gets the previous column in a buffered column result set

Parameters

- **result** – A result object
- **column** – The column number

Returns A column object

void **drizzle_column_seek** (*drizzle_result_st* *result, uint16_t column)

Seeks to a given column in a buffered column result set

Parameters

- **result** – A result object
- **column** – The column number

*drizzle_column_st** **drizzle_column_index** (*drizzle_result_st* *result, uint16_t column)

Gets a given column in a column buffered result set

Parameters

- **result** – A result object
- **column** – The column number

Returns A column object

uint16_t **drizzle_column_current** (*drizzle_result_st* *result)

Gets the column number in a buffered or unbuffered column result set

Parameters

- **result** – A result object:

Returns The column number

uint64_t **drizzle_row_read** (*drizzle_result_st* *result, *drizzle_return_t* *ret_ptr)

Reads the next row header and returns the row number for unbuffered row reads. Use *drizzle_field_read()* or *drizzle_field_buffer()* to get the field data after this call.

Parameters

- **result** – A result object
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into

Returns The row number

drizzle_row_t **drizzle_row_buffer** (*drizzle_result_st* *result, *drizzle_return_t* *ret_ptr)

Read and buffer one entire row, must be freed with *drizzle_row_free()*

Parameters

- **result** – A result object
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into

Returns The newly allocated row buffer

void **drizzle_row_free** (*drizzle_result_st* *result, *drizzle_row_t* row)

Free a buffered row read

Parameters

- **result** – A result object
- **row** – The row data to be freed

size_t* **drizzle_row_field_sizes** (*drizzle_result_st* *result)

Gets an array of the field sizes for buffered rows

Parameters

- **result** – A result object

Returns An array of row sizes

drizzle_row_t **drizzle_row_next** (*drizzle_result_st* *result)

Gets the next row in a buffered result set

Parameters

- **result** – A result object

Returns The row data

drizzle_row_t **drizzle_row_prev** (*drizzle_result_st* *result)

Gets the previous row in a buffered result set

Parameters

- **result** – A result object

Returns The row data

void **drizzle_row_seek** (*drizzle_result_st* *result, uint64_t row)

Seeks to a given row in a buffered result set

Parameters

- **result** – A result object
- **row** – The row number to seek to

drizzle_row_t **drizzle_row_index** (*drizzle_result_st* *result, uint64_t row)

Gets a row at the given index in a buffered result set

Parameters

- **result** – A result object
- **row** – The row number to get

Returns The row data

uint64_t **drizzle_row_current** (*drizzle_result_st* *result)

Gets the current row number

Parameters

- **result** – A result object

Returns The row number

drizzle_field_t **drizzle_field_read** (*drizzle_result_st* *result, size_t *offset, size_t *size, size_t *total, *drizzle_return_t* *ret_ptr)

Reads the next field from the network buffer. Useful for large blobs without buffering the entire blob.

Parameters

- **result** – A result object
- **offset** – The offset position of the blob for this read, to be written to by the function
- **size** – The size of the read, to be written to by the function
- **total** – The total size of the field, to be written to by the function
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into

Returns The field data

drizzle_field_t **drizzle_field_buffer** (*drizzle_result_st* *result, size_t *total, *drizzle_return_t* *ret_ptr)

Read and buffer the entire field for an unbuffered row read.

Parameters

- **result** – A result object
- **total** – The total size of the field, to be written to by the function
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into

Returns The field data

void **drizzle_field_free** (*drizzle_field_t* field)

Frees field data for unbuffered row reads

Parameters

- **field** – The field data to free

bool **drizzle_success** (*drizzle_return_t* ret)

Check if a drizzle function call succeeded

Parameters

- **ret** – result code

Returns true on success, false otherwise

bool **drizzle_failed** (*drizzle_return_t* ret)

Check if a drizzle function call failed

Parameters

- **ret** – result code

Returns true on fail, false otherwise

5.5 Prepared Statements

5.5.1 Introduction

This section outlines the prepared statement functionality

5.5.2 Structs

drizzle_stmt_st

The internal struct containing the prepared statement object

drizzle_datetime_st

The internal struct for passing a date/time to/from the prepared statement API

5.5.3 Functions

*drizzle_stmt_st** **drizzle_stmt_prepare** (*drizzle_st* *con, const char *statement, size_t size, *drizzle_return_t* *ret_ptr)

Prepare a new statement

Parameters

- **con** – A connection object
- **statement** – The prepared statement with question marks (“?”) for the elements to be provided as parameters
- **size** – The length of the statement
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into

Returns A newly allocated and prepared statement object (or NULL on error)

drizzle_return_t **drizzle_stmt_set_tiny** (*drizzle_stmt_st* *stmt, uint16_t param_num, uint8_t value, bool is_unsigned)

Sets a parameter of a prepared statement to a tinyint value

Parameters

- **stmt** – A prepared statement object
- **param_num** – The parameter number to set (starting at 0)
- **value** – The value to set the parameter
- **is_unsigned** – Set to true if the parameter is unsigned

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

drizzle_return_t **drizzle_stmt_set_short** (*drizzle_stmt_st* *stmt, uint16_t param_num, uint16_t value, bool is_unsigned)

Sets a parameter of a prepared statement to a short int value

Parameters

- **stmt** – A prepared statement object
- **param_num** – The parameter number to set (starting at 0)
- **value** – The value to set the parameter
- **is_unsigned** – Set to true if the parameter is unsigned

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

drizzle_return_t **drizzle_stmt_set_int** (*drizzle_stmt_st* *stmt, uint16_t param_num, uint32_t value, bool is_unsigned)

Sets a parameter of a prepared statement to an int value

Parameters

- **stmt** – A prepared statement object

- **param_num** – The parameter number to set (starting at 0)
- **value** – The value to set the parameter
- **is_unsigned** – Set to true if the parameter is unsigned

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

drizzle_return_t **drizzle_stmt_set_bigint** (*drizzle_stmt_st* *stmt, uint16_t param_num, uint64_t value, bool is_unsigned)

Sets a parameter of a prepared statement to a bigint value

Parameters

- **stmt** – A prepared statement object
- **param_num** – The parameter number to set (starting at 0)
- **value** – The value to set the parameter
- **is_unsigned** – Set to true if the parameter is unsigned

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

drizzle_return_t **drizzle_stmt_set_double** (*drizzle_stmt_st* *stmt, uint16_t param_num, double value)

Sets a parameter of a prepared statement to a double value

Parameters

- **stmt** – A prepared statement object
- **param_num** – The parameter number to set (starting at 0)
- **value** – The value to set the parameter

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

drizzle_return_t **drizzle_stmt_set_float** (*drizzle_stmt_st* *stmt, uint16_t param_num, float value)

Sets a parameter of a prepared statement to a float value

Parameters

- **stmt** – A prepared statement object
- **param_num** – The parameter number to set (starting at 0)
- **value** – The value to set the parameter

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

drizzle_return_t **drizzle_stmt_set_string** (*drizzle_stmt_st* *stmt, uint16_t param_num, char *value, size_t length)

Sets a parameter of a prepared statement to a string value

Parameters

- **stmt** – A prepared statement object
- **param_num** – The parameter number to set (starting at 0)
- **value** – The value to set the parameter
- **length** – The length of the value data

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

drizzle_return_t **drizzle_stmt_set_null** (*drizzle_stmt_st* *stmt, uint16_t param_num)

Sets a parameter of a prepared statement to a NULL value

Parameters

- **stmt** – A prepared statement object
- **param_num** – The parameter number to set (starting at 0)

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

drizzle_return_t **drizzle_stmt_set_time** (*drizzle_stmt_st* *stmt, uint16_t param_num, uint32_t days, uint8_t hours, uint8_t minutes, uint8_t seconds, uint32_t microseconds, bool is_negative)

Sets a parameter of a prepared statement to a time value

Parameters

- **stmt** – A prepared statement object
- **param_num** – The parameter number to set (starting at 0)
- **days** – The number of days for the time
- **hours** – The number of hours for the time
- **minutes** – The number of minutes for the time
- **seconds** – The number of seconds for the time
- **microseconds** – The number of microseconds for the time
- **is_negative** – Flag for the sign of the time value

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

drizzle_return_t **drizzle_stmt_set_timestamp** (*drizzle_stmt_st* *stmt, uint16_t param_num, uint16_t year, uint8_t month, uint8_t day, uint8_t hours, uint8_t minutes, uint8_t seconds, uint32_t microseconds)

Sets a parameter of a prepared statement to a datetime/timestamp value

Parameters

- **stmt** – A prepared statement object
- **param_num** – The parameter number to set (starting at 0)
- **year** – The year number for the timestamp
- **month** – The month number for the timestamp
- **day** – The day number for the timestamp
- **hours** – The hour number for the timestamp
- **minutes** – The minute number for the timestamp
- **seconds** – The minute number for the timestamp
- **microseconds** – The minute number for the timestamp

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

drizzle_return_t **drizzle_stmt_execute** (*drizzle_stmt_st* *stmt)

Executes a prepared statement

Parameters

- **stmt** – The prepared statement object

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

drizzle_return_t **drizzle_stmt_send_long_data** (*drizzle_stmt_st* *stmt, uint16_t param_num, unsigned char *data, size_t len)

Send long binary data packet

Parameters

- **stmt** – The prepared statement object
- **param_num** – The parameter number this data is for
- **data** – A pointer to the data
- **len** – The length of the data

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

drizzle_return_t **drizzle_stmt_reset** (*drizzle_stmt_st* *stmt)

Reset a statement to the prepared state

Parameters

- **stmt** – The prepared statement object

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

drizzle_return_t **drizzle_stmt_fetch** (*drizzle_stmt_st* *stmt)

Fetch a row from the result set, can be used with buffered or unbuffered result sets

Parameters

- **stmt** – The prepared statement object

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

drizzle_return_t **drizzle_stmt_buffer** (*drizzle_stmt_st* *stmt)

Buffer the entire result set

Parameters

- **stmt** – The prepared statement object

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

bool **drizzle_stmt_get_is_null** (*drizzle_stmt_st* *stmt, uint16_t column_number, *drizzle_return_t* *ret_ptr)

Check if a column for a fetched row is set to NULL

Parameters

- **stmt** – The prepared statement object
- **column_number** – The column number to get (starting at 0)
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into

Returns True if NULL

bool **drizzle_stmt_get_is_null_from_name** (*drizzle_stmt_st* *stmt, const char *column_name, *drizzle_return_t* *ret_ptr)

Check if a column for a fetched row is set to NULL using a column name

Parameters

- **stmt** – The prepared statement object
- **column_name** – The column name to get
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into, *DRIZZLE_RETURN_NOT_FOUND* if the column name cannot be found

Returns True if NULL

```
bool drizzle_stmt_get_is_unsigned(drizzle_stmt_st *stmt, uint16_t column_number, drizzle_return_t *ret_ptr)
```

Check if a column for a fetched row is unsigned

Parameters

- **stmt** – The prepared statement object
- **column_number** – The column number to get (starting at 0)
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into

Returns True if unsigned

```
bool drizzle_stmt_get_is_unsigned_from_name(drizzle_stmt_st *stmt, const char *column_name, drizzle_return_t *ret_ptr)
```

Check if a column for a fetched row is unsigned using a column name

Parameters

- **stmt** – The prepared statement object
- **column_name** – The column name to get
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into, *DRIZZLE_RETURN_NOT_FOUND* if the column name cannot be found

Returns True if unsigned

```
const char *drizzle_stmt_get_string(drizzle_stmt_st *stmt, uint16_t column_number, size_t *len, drizzle_return_t *ret_ptr)
```

Get the string value for a column of a fetched row (int types are automatically converted)

Parameters

- **stmt** – The prepared statement object
- **column_number** – The column number to get (starting at 0)
- **len** – A pointer to a *size_t* to store the result length into
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into

Returns A pointer to the string value

```
const char *drizzle_stmt_get_string_from_name(drizzle_stmt_st *stmt, const char *column_name, size_t *len, drizzle_return_t *ret_ptr)
```

Get the string value for a column of a fetched row (int types are automatically converted) using a column name

Parameters

- **stmt** – The prepared statement object
- **column_name** – The column name to get
- **len** – A pointer to a *size_t* to store the result length into
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into, *DRIZZLE_RETURN_NOT_FOUND* if the column name cannot be found

Returns A pointer to the string value

```
uint32_t drizzle_stmt_get_int(drizzle_stmt_st *stmt, uint16_t column_number, drizzle_return_t *ret_ptr)
```

Get the int value for a column of a fetched row

Parameters

- **stmt** – The prepared statement object
- **column_number** – The column number to get (starting at 0)
- **ret_ptr** – A pointer to a `drizzle_return_t` to store the return status into `DRIZZLE_RETURN_TRUNCATED` if a truncation has occurred

Returns The int value

```
uint32_t drizzle_stmt_get_int_from_name (drizzle_stmt_st *stmt, const char *column_name, drizzle_return_t *ret_ptr)
```

Get the int value for a column of a fetched row using a column name

Parameters

- **stmt** – The prepared statement object
- **column_name** – The column name to get
- **ret_ptr** – A pointer to a `drizzle_return_t` to store the return status into `DRIZZLE_RETURN_TRUNCATED` if a truncation has occurred, `DRIZZLE_RETURN_NOT_FOUND` if the column name cannot be found

Returns The int value

```
uint64_t drizzle_stmt_get_bigint (drizzle_stmt_st *stmt, uint16_t column_number, drizzle_return_t *ret_ptr)
```

Get the bigint value for a column of a fetched row

Parameters

- **stmt** – The prepared statement object
- **column_number** – The column number to get (starting at 0)
- **ret_ptr** – A pointer to a `drizzle_return_t` to store the return status into `DRIZZLE_RETURN_TRUNCATED` if a truncation has occurred

Returns The bigint value

```
uint64_t drizzle_stmt_get_bigint_from_name (drizzle_stmt_st *stmt, const char *column_name, drizzle_return_t *ret_ptr)
```

Get the bigint value for a column of a fetched row using a column name

Parameters

- **stmt** – The prepared statement object
- **column_name** – The column name to get
- **ret_ptr** – A pointer to a `drizzle_return_t` to store the return status into `DRIZZLE_RETURN_TRUNCATED` if a truncation has occurred, `DRIZZLE_RETURN_NOT_FOUND` if the column name cannot be found

Returns The bigint value

```
double drizzle_stmt_get_double (drizzle_stmt_st *stmt, uint16_t column_number, drizzle_return_t *ret_ptr)
```

Get the double value for a column of a fetched row

Parameters

- **stmt** – The prepared statement object
- **column_number** – The column number to get (starting at 0)

- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into *DRIZZLE_RETURN_TRUNCATED* if a truncation has occurred

Returns The double value

double **drizzle_stmt_get_double_from_name**(*drizzle_stmt_st* *stmt, const char *column_name, *drizzle_return_t* *ret_ptr)

Get the double value for a column of a fetched row from a column name

Parameters

- **stmt** – The prepared statement object
- **column_name** – The column name to get
- **ret_ptr** – A pointer to a *drizzle_return_t* to store the return status into *DRIZZLE_RETURN_TRUNCATED* if a truncation has occurred, *DRIZZLE_RETURN_NOT_FOUND* if the column name cannot be found

Returns The double value

drizzle_return_t **drizzle_stmt_close**(*drizzle_stmt_st* *stmt)

Close and free a prepared statement

Parameters

- **stmt** – The prepared statement object

Returns A return status code, *DRIZZLE_RETURN_OK* upon success

uint16_t **drizzle_stmt_column_count**(*drizzle_stmt_st* *stmt)

Gets the column count for a result set which has been executed using *drizzle_stmt_execute()*

Parameters

- **stmt** – The prepared statement object

Returns The column count

uint64_t **drizzle_stmt_affected_rows**(*drizzle_stmt_st* *stmt)

Gets the affected rows count for a result set which has been executed using *drizzle_stmt_execute()*

Parameters

- **stmt** – The prepared statement object

Returns The column count

uint64_t **drizzle_stmt_insert_id**(*drizzle_stmt_st* *stmt)

Gets the insert ID for a result set which has been executed using *drizzle_stmt_execute()*

Parameters

- **stmt** – The prepared statement object

Returns The insert ID

uint16_t **drizzle_stmt_param_count**(*drizzle_stmt_st* *stmt)

Gets the number of parameters expected for a result set that has been prepared with *drizzle_stmt_prepare()*

Parameters

- **stmt** – The prepared statement object

Returns The number of parameters

uint64_t **drizzle_stmt_row_count** (*drizzle_stmt_st* **stmt*)

Gets the row count for a statement buffered with *drizzle_stmt_buffer()*

On error it returns UINT64_MAX;

Parameters

- **stmt** – The prepared statement object

Returns The row count

5.6 Binlog Functions

5.6.1 Introduction

Libdrizzle Redux contains functions which give it the capabilities to connect as a MySQL slave or a mysqlbinlog type client and retrieve the events.

Warning: You should start a binlog retrieval on a new connection only. Running on a connection that has already executed queries has an undefined (usually bad) behaviour.

The binlog functions use a callback API so that a function in the user application will be called whenever there is a new event to retrieve.

5.6.2 Structs

drizzle_binlog_st

The internal struct containing the binlog stream information

drizzle_binlog_event_st

The internal struct containing the binlog event header and data

5.6.3 Callback Functions

There are two callback functions. The first is called whenever a new event is available to retrieve. The second is triggered whenever an error (or EOF) occurs.

void (**drizzle_binlog_fn**) (*drizzle_binlog_event_st* **event*, void **context*)

This defines the function that will be supplied to accept binlog events

Warning: Event data needs to be copied/processed before exiting the function, it will be erased before the next callback.

Parameters

- **event** – A pointer to the event struct
- **context** – A user defined pointer supplied in *drizzle_binlog_init()*

void (**drizzle_binlog_error_fn**) (*drizzle_return_t* *error*, *drizzle_st* **con*, void **context*)

This defines the function that will be supplied to accept binlog errors

Parameters

- **error** – The *drizzle_return_t* for the error (or *DRIZZLE_RETURN_EOF* when all events have been retrieved)
- **con** – The connection object the error occurred on
- **context** – A user defined pointer supplied in *drizzle_binlog_init()*

5.6.4 Functions

drizzle_binlog_st ***drizzle_binlog_init**(*drizzle_st* *con, *drizzle_binlog_fn* *binlog_fn, *drizzle_binlog_error_fn* *error_fn, void *context, bool verify_checksums)

Initializes a binlog object for the connection and sets the event callback functions

Parameters

- **con** – The connection the binlog retrieval will be on
- **binlog_fn** – The function callback defined in (*drizzle_binlog_fn*)()
- **error_fn** – The function callback defined in (*drizzle_binlog_error_fn*)()
- **context** – A pointer to user data which will be used for the callback functions
- **verify_checksums** – Set to true if MySQL 5.6 and higher checksums should be verified

void **drizzle_binlog_free**(*drizzle_binlog_st* *binlog)

Frees a binlog object created with *drizzle_binlog_init()*

Parameters

- **binlog** – The binlog object to be freed

drizzle_return_t **drizzle_binlog_start**(*drizzle_binlog_st* *binlog, uint32_t server_id, const char *file, uint32_t start_position)

Start the binlog transaction. Set the server_id to 0 to disconnect automatically at the end of the last log.

Parameters

- **binlog** – A binlog object created using *drizzle_binlog_init()*
- **server_id** – A unique server ID (or 0) to connect to the MySQL server with
- **file** – The start binlog file, can be empty to start at the first known file
- **start_position** – The position of the binlog file to start at, a value of less than 4 is set to 4 due to the binlog header taking the first 4 bytes

Returns A Drizzle return type. *DRIZZLE_RETURN_OK* upon success.

uint32_t **drizzle_binlog_event_timestamp**(*drizzle_binlog_event_st* *event)

Get the timestamp for the event received by the event callback

Parameters

- **event** – The event from the binlog stream

Returns The timestamp for the binlog event

drizzle_binlog_event_types_t **drizzle_binlog_event_type**(*drizzle_binlog_event_st* *event)

Get the event type for the event received by the event callback

Parameters

- **event** – The event from the binlog stream

Returns The timestamp for the binlog event

uint32_t **drizzle_binlog_event_server_id** (*drizzle_binlog_event_st *event*)

Get the server_id for the event received by the event callback

Parameters

- **event** – The event from the binlog stream

Returns The server_id for the binlog event

uint32_t **drizzle_binlog_event_length** (*drizzle_binlog_event_st *event*)

Get the length of the event data received by the event callback

Parameters

- **event** – The event from binlog stream

Returns The event data length

uint32_t **drizzle_binlog_event_next_pos** (*drizzle_binlog_event_st *event*)

Get the next event position from the event received by the event callback

Parameters

- **event** – The event from the binlog stream

Returns The next event position

uint16_t **drizzle_binlog_event_flags** (*drizzle_binlog_event_st *event*)

Get the flags for the event received by the event callback

Parameters

- **event** – The event from the binlog stream

Returns The event flags

uint16_t **drizzle_binlog_event_extra_flags** (*drizzle_binlog_event_st *event*)

Get the extra flags for the event received by the event callback

Parameters

- **event** – The event from the binlog stream

Returns The extra event flags

const unsigned char* **drizzle_binlog_event_data** (*drizzle_binlog_event_st *event*)

Get the event data for the event received by the event callback

Parameters

- **event** – The event from the binlog stream

Returns A pointer to the event data

const unsigned char* **drizzle_binlog_event_raw_data** (*drizzle_binlog_event_st *event*)

Get the raw event data (including header) for the event received by the event callback

Parameters

- **event** – The event from the binlog stream

Returns A pointer to the raw event data

uint32_t **drizzle_binlog_event_raw_length** (*drizzle_binlog_event_st *event*)

Get the length of the raw event data (including header) for the event received by the event callback

Parameters

- **event** – The event from the binlog stream

Returns The length of the raw event data

```
const char *drizzle_binlog_event_type_str(drizzle_binlog_event_types_t event_type)
```

Get the event type for the binlog event as string

Parameters

- **event_type** – A binlog event type

Returns The event type of the binlog event as string

```
void drizzle_binlog_get_filename(drizzle_st *con, char **filename, uint32_t *end_position,  
                                int file_index)
```

Get the name and size of a binlog file in bytes

Queries the database for a list of binlog files and copies the **filename** to the passed buffer

If the **file_index** is invalid or no binlog files exist **filename** will contain an empty string. A valid **file_index** is in the range [-1 to (number of binlog files -1)] The **end_position** will hold the size of the binlog file and can be used to start reading from the end of the binlog file when passed to *drizzle_binlog_start()*

The filename parameter is allocated by the function and needs to be freed by the application when finished with. This is the case, regardless of the return status of the function.

Parameters

- **con** – Drizzle structure previously initialized with *drizzle_create()*
- **filename** – Buffer to copy filename to
- **end_position** – Variable to save the size of the binlog file into
- **file_index** – Index of the binlog to retrieve.

Returns

Standard drizzle return value:

- DRIZZLE_RETURN_OK the filename was retrieved successfully.
- DRIZZLE_RETURN_INVALID_ARGUMENT: invalid argument(s)
- DRIZZLE_RETURN_NOT_FOUND: no binlog files were available

6.1 Buffered Results

6.1.1 Introduction

In this example `drizzle_query()` is used to send a select query to a MySQL server. The whole result set is then retrieved and stored in memory using `drizzle_result_buffer()`.

The number of columns is retrieved using `drizzle_result_column_count()`. Each row is iterated through by calling `drizzle_row_next()` which returns an array containing string of the row data. We know how many elements are in this array due to the earlier call to `drizzle_result_column_count()`. The data from each element in the row is finally echoed to the console.

To end the query the result set is freed using `drizzle_result_free()`

6.1.2 Code

```
#include <libdrizzle-redux/libdrizzle.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    (void) argc;
    (void) argv;
    drizzle_st *con;
    drizzle_return_t ret;
    drizzle_result_st *result;
    drizzle_row_t row;
    int num_fields;

    con = drizzle_create("localhost", 3306, "user", "pass", "test", 0);
```

(continues on next page)

(continued from previous page)

```

if (con == NULL)
{
    printf("Drizzle connection object creation error\n");
    return EXIT_FAILURE;
}
ret = drizzle_connect(con);
if (ret != DRIZZLE_RETURN_OK)
{
    printf("Drizzle connection failure\n");
    return EXIT_FAILURE;
}

result= drizzle_query(con, "select * from libdrizzle.t1", 0, &ret);
if (ret != DRIZZLE_RETURN_OK)
{
    printf("Select failure\n");
    return EXIT_FAILURE;
}
drizzle_result_buffer(result);
num_fields= drizzle_result_column_count(result);

printf("%d fields\n", num_fields);
while ((row = drizzle_row_next(result)))
{
    printf("Data: ");
    for (uint16_t col=0; col < num_fields; col++)
    {
        printf("%s", row[col]);
    }
    printf("\n");
}

drizzle_result_free(result);

drizzle_quit(con);
return EXIT_SUCCESS;
}

```

6.2 Unbuffered Results

6.2.1 Introduction

In this example `drizzle_query()` is used to send a select query to a MySQL server. The first thing that is sent back in the results is a list of columns, so this list needs to be retrieved. The simplest way of doing this is to buffer the column data using `drizzle_column_buffer()`.

The number of columns is retrieved using `drizzle_result_column_count()`. Each row is iterated through by calling `drizzle_row_buffer()` which buffers and returns an array containing string of the row data. We know how many elements are in this array due to the earlier call to `drizzle_result_column_count()`. The data from each element in the row is finally echoed to the console. The row data is freed using `drizzle_row_free()`.

To end the query the result set is freed using `drizzle_result_free()`

6.2.2 Code

```
#include <libdrizzle-redux/libdrizzle.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    (void) argc;
    (void) argv;
    drizzle_st *con;
    drizzle_return_t ret;
    drizzle_result_st *result;
    drizzle_row_t row;
    int num_fields;

    con = drizzle_create("localhost", 3306, "root", "", "libdrizzle", 0);
    if (con == NULL)
    {
        printf("Drizzle connection object creation error\n");
        return EXIT_FAILURE;
    }
    ret = drizzle_connect(con);
    if (ret != DRIZZLE_RETURN_OK)
    {
        printf("Drizzle connection failure\n");
        return EXIT_FAILURE;
    }

    result= drizzle_query(con, "select * from libdrizzle.t1", 0, &ret);
    if (ret != DRIZZLE_RETURN_OK)
    {
        printf("Select failure\n");
        return EXIT_FAILURE;
    }

    if (drizzle_column_buffer(result) != DRIZZLE_RETURN_OK)
    {
        printf("Column buffer failure\n");
        return EXIT_FAILURE;
    }
    num_fields= drizzle_result_column_count(result);

    printf("%d fields\n", num_fields);
    while(1)
    {
        row= drizzle_row_buffer(result, &ret);
        if (ret != DRIZZLE_RETURN_OK)
        {
            printf("Row retrieval error\n");
            break;
        }
        if (row == NULL)
        {
            // EOF
            break;
        }
    }
}
```

(continues on next page)

(continued from previous page)

```

    printf("Data: ");
    for (uint16_t col=0; col < num_fields; col++)
    {
        printf("%s", row[col]);
    }
    printf("\n");
    drizzle_row_free(result, row);
}

drizzle_result_free(result);

drizzle_quit(con);
return EXIT_SUCCESS;
}

```

6.3 Prepared Statements

6.3.1 Introduction

This code example has been simplified to make it easier to read, the connection and error handling code has been removed.

In this example a basic select query has been defined which has one element to replace using the ‘?’ character. The statement is prepared using `drizzle_stmt_prepare()` and we can get the number of parameters the server is expecting with `drizzle_stmt_param_count()`. In this example we know that there is only one parameter required so we send one INT type parameter using `drizzle_stmt_set_int()` stating that this is parameter 0 and a signed value.

Once the parameters have been provided the statement is executed using `drizzle_stmt_execute()` and the results buffered using `drizzle_stmt_buffer()`. Once buffered the row count can be obtained using `drizzle_stmt_row_count()`.

Finally we get the result data. A call to `drizzle_stmt_fetch()` gets the next row from either the network or the buffer (the buffer in this case). The int data is retrieved using `drizzle_stmt_get_int()`, a call for each column in the row (in example the table only has one column) is made using the `drizzle_stmt_get_` functions.

When we are done the statement is closed and cleaned up using `drizzle_stmt_close()`. It can also be reused with `drizzle_stmt_reset()` or executed again with `drizzle_stmt_execute()` (this is useful for inserts).

6.3.2 Code

```

drizzle_stmt_st *stmt;
const char *query= "select * from libdrizzle.t1 where a > ?";
stmt= drizzle_stmt_prepare(con, query, strlen(query), &ret);

printf("Params: %" PRIu16 "\n", drizzle_stmt_param_count(stmt));

uint32_t val= 1;
ret = drizzle_stmt_set_int(stmt, 0, val, false);

ret = drizzle_stmt_execute(stmt);

```

(continues on next page)

(continued from previous page)

```
ret = drizzle_stmt_buffer(stmt);

printf("Rows found: %" PRIu64 "\n", drizzle_stmt_row_count(stmt));
while (drizzle_stmt_fetch(stmt) != DRIZZLE_RETURN_ROW_END)
{
    uint32_t res_val;
    res_val= drizzle_stmt_get_int(stmt, 0, &ret);
    printf("Got value: %" PRIu32 "\n", *res_val);
}
ret = drizzle_stmt_close(stmt);
```

6.4 Event Callback

6.4.1 Introduction

In this example *drizzle_event_watch_fn()* is used to set a custom I/O event watcher function for a drizzle structure. It is used to integrate libdrizzle-redux with a custom event loop. The callback will be invoked to register or deregister interest in events for a connection. When the events are triggered, *drizzle_set_revents()* should be called to indicate which events are ready. The event loop should stop waiting for these events, as libdrizzle-redux will call the callback again if it is still interested. To resume processing, the libdrizzle-redux function that returned *DRIZZLE_RETURN_IO_WAIT* should be called again.

The custom callback must have a signature as shown:

```
drizzle_return_t (drizzle_event_watch_fn)(drizzle_con_st *con, short events, void_
↪ *context);
```

6.4.2 Code

```
#include <libdrizzle-5.1/libdrizzle.h>
#include <libdrizzle-5.1/constants.h>

extern drizzle_return_t drizzle_event_callback(drizzle_st *con, short events,
void *context);

extern drizzle_return_t drizzle_event_callback(drizzle_st *con, short events,
void *context)
{
    // The context defined for the drizzle connection
    int* drizzle_cxt = (int*) drizzle_context(con);

    // The context defined for the callback function
    int* callback_cxt = (int*) context;

    printf("Called drizzle_event_callback\n");

    return DRIZZLE_RETURN_OK;
}

int main(int argc, char *argv[])
```

(continues on next page)

(continued from previous page)

```

{
    (void) argc;
    (void) argv;

    int cxt_a = 1;
    int cxt_b = 0;

    drizzle_st *con = drizzle_create("localhost", 3306, "root", "", "libdrizzle", 0);
    if (con == NULL)
    {
        printf("Drizzle connection object creation error\n");
        return EXIT_FAILURE;
    }

    // Set drizzle dummy context
    drizzle_set_context(con, (void*)&cxt_a);

    // Set user defined callback function event_watch_fn
    drizzle_set_event_watch_fn(con, drizzle_event_callback, (void*)&cxt_b);

    drizzle_return_t driz_ret= drizzle_connect(con);

    if (ret != DRIZZLE_RETURN_OK)
    {
        printf("Drizzle connection failure\n");
        return EXIT_FAILURE;
    }

    driz_ret= drizzle_quit(con);

    return EXIT_SUCCESS;
}

```

6.5 Binlog Retrieval

6.5.1 Introduction

This example shows how the binlog api can be used to retrieve the events from a mysql binlog.

First a *drizzle_binlog_st* is created by calling *drizzle_binlog_init()* with an already initialized *drizzle_st* connection. Two callbacks are specified; *binlog_event*, which is called every time a binlog event is retrieved and *binlog_error* which is called when an error occurs while parsing binlog-data. In addition it is possible to pass a user context to be used for the callback functions. When *NULL* is passed, as in this example, the context is unspecified. Calling *drizzle_binlog_start()* starts the parsing of the binlog. The second argument is the id of the server to connect to. It corresponds to the *server_id* variable defined in the **MySQL** config file. When 0 is passed as *server_id* the connection is disconnected automatically at the end of the last logfile.

In the callback function *binlog_event* several properties of the binlog event are retrieved from the *drizzle_binlog_event_st* object. E.g. the type of binlog event, cf. *drizzle_binlog_event_types_t*, is retrieved by calling *drizzle_binlog_event_type()*.

The binlog api offers retrieval of information which is common for all binlog event types. To retrieve information specific to each type, please refer to the [MySQL documentation](#)

When all binlog events have been parsed, `binlog_error` is called with a return status `DRIZZLE_RETURN_EOF` indicating that end of the last binlog logfile has been reached.

To compile the example, save the code and run:

```
g++ binlog_example.cxx -ldrizzle-redux -lpthread -o binlog_example
```

6.5.2 Code

```
#include <libdrizzle-redux/libdrizzle.h>
#include <stdio.h>
#include <stdlib.h>
#include <inttypes.h>

/**
 * @file binlog_example.cxx
 */

void binlog_error(drizzle_return_t ret, drizzle_st *connection, void *context)
{
    (void) context;
    if (ret != DRIZZLE_RETURN_EOF)
    {
        printf("Error retrieving binlog: %s\n", drizzle_error(connection));
    }
}

void binlog_event(drizzle_binlog_event_st *event, void *context)
{
    (void) context;
    printf("Timestamp: %"PRIu32"\n", drizzle_binlog_event_timestamp(event));
    printf("Type: %"PRIu8"\n", drizzle_binlog_event_type(event));
    printf("Server-id: %"PRIu32"\n", drizzle_binlog_event_server_id(event));
    printf("Next-pos: %"PRIu32"\n", drizzle_binlog_event_next_pos(event));
    uint length= drizzle_binlog_event_length(event);
    printf("Length: %"PRIu32"\n", length);
    const unsigned char *data= drizzle_binlog_event_data(event);
    printf("Data: 0x");
    for (uint i=0; i<length; i++)
        printf("%02X ", data[i]);
    printf("\n\n");
}

int main(int argc, char *argv[])
{
    (void) argc;
    (void) argv;
    drizzle_st *con;
    drizzle_return_t ret;
    drizzle_binlog_st *binlog;

    // Should be changed to the specifics of the MySQL installation
    con = drizzle_create("localhost", 3306, "root", "", "", 0);
    if (con == NULL)
    {
        printf("Drizzle connection object creation error\n");
    }
}
```

(continues on next page)

(continued from previous page)

```
    return EXIT_FAILURE;
}
ret = drizzle_connect(con);
if (ret != DRIZZLE_RETURN_OK)
{
    printf("Drizzle connection failure\n");
    return EXIT_FAILURE;
}

binlog= drizzle_binlog_init(con, binlog_event, binlog_error, NULL, true);
ret= drizzle_binlog_start(binlog, 0, "", 0);
if (ret != DRIZZLE_RETURN_EOF)
{
    printf("Drizzle binlog start failure\n");
    return EXIT_FAILURE;
}

drizzle_quit(con);
return EXIT_SUCCESS;
}
```

Symbols

(drizzle_binlog_error_fn) (*C function*), 54
(drizzle_binlog_fn) (*C function*), 54

D

drizzle_binlog_event_data (*C function*), 56
drizzle_binlog_event_extra_flags (*C function*), 56
drizzle_binlog_event_flags (*C function*), 56
drizzle_binlog_event_length (*C function*), 56
drizzle_binlog_event_next_pos (*C function*), 56
drizzle_binlog_event_positions_t (*C type*), 27
drizzle_binlog_event_raw_data (*C function*), 56
drizzle_binlog_event_raw_length (*C function*), 56
drizzle_binlog_event_server_id (*C function*), 56
drizzle_binlog_event_st (*C type*), 54
drizzle_binlog_event_timestamp (*C function*), 55
drizzle_binlog_event_type (*C function*), 55
drizzle_binlog_event_type_str (*C function*), 57
drizzle_binlog_event_types_t (*C type*), 26
drizzle_binlog_free (*C function*), 55
drizzle_binlog_get_filename (*C function*), 57
drizzle_binlog_init (*C function*), 55
drizzle_binlog_st (*C type*), 54
drizzle_binlog_start (*C function*), 55
drizzle_bugreport (*C function*), 28
drizzle_capabilities (*C function*), 35
DRIZZLE_CAPABILITIES_CLIENT (*built-in variable*), 22
DRIZZLE_CAPABILITIES_COMPRESS (*built-in variable*), 21
DRIZZLE_CAPABILITIES_CONNECT_WITH_DB (*built-in variable*), 21
DRIZZLE_CAPABILITIES_FOUND_ROWS (*built-in variable*), 21
DRIZZLE_CAPABILITIES_IGNORE_SIGPIPE (*built-in variable*), 22
DRIZZLE_CAPABILITIES_IGNORE_SPACE (*built-in variable*), 21
DRIZZLE_CAPABILITIES_INTERACTIVE (*built-in variable*), 22
DRIZZLE_CAPABILITIES_LOCAL_FILES (*built-in variable*), 21
DRIZZLE_CAPABILITIES_LONG_FLAG (*built-in variable*), 21
DRIZZLE_CAPABILITIES_LONG_PASSWORD (*built-in variable*), 21
DRIZZLE_CAPABILITIES_MULTI_RESULTS (*built-in variable*), 22
DRIZZLE_CAPABILITIES_MULTI_STATEMENTS (*built-in variable*), 22
DRIZZLE_CAPABILITIES_NO_SCHEMA (*built-in variable*), 21
DRIZZLE_CAPABILITIES_NONE (*built-in variable*), 21
DRIZZLE_CAPABILITIES_ODBC (*built-in variable*), 21
DRIZZLE_CAPABILITIES_PLUGIN_AUTH (*built-in variable*), 22
DRIZZLE_CAPABILITIES_PROTOCOL_41 (*built-in variable*), 21
DRIZZLE_CAPABILITIES_PS_MULTI_RESULTS (*built-in variable*), 22
DRIZZLE_CAPABILITIES_REMEMBER_OPTIONS (*built-in variable*), 22
DRIZZLE_CAPABILITIES_RESERVED (*built-in variable*), 22
DRIZZLE_CAPABILITIES_SECURE_CONNECTION (*built-in variable*), 22
DRIZZLE_CAPABILITIES_SSL (*built-in variable*), 22
DRIZZLE_CAPABILITIES_SSL_VERIFY_SERVER_CERT

(*built-in variable*), 22

drizzle_capabilities_t (*C type*), 21

DRIZZLE_CAPABILITIES_TRANSACTIONS (*built-in variable*), 22

drizzle_charset (*C function*), 35

DRIZZLE_CHARSET_ARMSCII8_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_ARMSCII8_GENERAL_CI (*built-in variable*), 16

DRIZZLE_CHARSET_ASCII_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_ASCII_GENERAL_CI (*built-in variable*), 15

DRIZZLE_CHARSET_BIG5_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_BIG5_CHINESE_CI (*built-in variable*), 15

DRIZZLE_CHARSET_BINARY (*built-in variable*), 17

DRIZZLE_CHARSET_CP1250_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_CP1250_CROATIAN_CI (*built-in variable*), 16

DRIZZLE_CHARSET_CP1250_CZECH_CS (*built-in variable*), 16

DRIZZLE_CHARSET_CP1250_GENERAL_CI (*built-in variable*), 16

DRIZZLE_CHARSET_CP1250_POLISH_CI (*built-in variable*), 18

DRIZZLE_CHARSET_CP1251_BIN (*built-in variable*), 16

DRIZZLE_CHARSET_CP1251_BULGARIAN_CI (*built-in variable*), 15

DRIZZLE_CHARSET_CP1251_GENERAL_CI (*built-in variable*), 16

DRIZZLE_CHARSET_CP1251_GENERAL_CS (*built-in variable*), 16

DRIZZLE_CHARSET_CP1251_UKRAINIAN_CI (*built-in variable*), 16

DRIZZLE_CHARSET_CP1256_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_CP1256_GENERAL_CI (*built-in variable*), 16

DRIZZLE_CHARSET_CP1257_BIN (*built-in variable*), 16

DRIZZLE_CHARSET_CP1257_GENERAL_CI (*built-in variable*), 16

DRIZZLE_CHARSET_CP1257_LITHUANIAN_CI (*built-in variable*), 16

DRIZZLE_CHARSET_CP850_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_CP850_GENERAL_CI (*built-in variable*), 15

DRIZZLE_CHARSET_CP852_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_CP852_GENERAL_CI (*built-in variable*), 16

DRIZZLE_CHARSET_CP866_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_CP866_GENERAL_CI (*built-in variable*), 16

DRIZZLE_CHARSET_CP932_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_CP932_JAPANESE_CI (*built-in variable*), 17

DRIZZLE_CHARSET_DEC8_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_DEC8_SWEDISH_CI (*built-in variable*), 15

DRIZZLE_CHARSET_EUCJPMS_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_EUCJPMS_JAPANESE_CI (*built-in variable*), 17

DRIZZLE_CHARSET_EUCKR_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_EUCKR_KOREAN_CI (*built-in variable*), 15

DRIZZLE_CHARSET_GB2312_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_GB2312_CHINESE_CI (*built-in variable*), 16

DRIZZLE_CHARSET_GBK_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_GBK_CHINESE_CI (*built-in variable*), 16

DRIZZLE_CHARSET_GEOSTD8_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_GEOSTD8_GENERAL_CI (*built-in variable*), 17

DRIZZLE_CHARSET_GREEK_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_GREEK_GENERAL_CI (*built-in variable*), 16

DRIZZLE_CHARSET_HEBREW_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_HEBREW_GENERAL_CI (*built-in variable*), 15

DRIZZLE_CHARSET_HP8_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_HP8_ENGLISH_CI (*built-in variable*), 15

DRIZZLE_CHARSET_KEYBCS2_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_KEYBCS2_GENERAL_CI (*built-in variable*), 16

DRIZZLE_CHARSET_KOI8R_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_KOI8R_GENERAL_CI (*built-in variable*), 15

DRIZZLE_CHARSET_KOI8U_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_KOI8U_GENERAL_CI (*built-in variable*), 15
 DRIZZLE_CHARSET_LATIN1_BIN (*built-in variable*), 16
 DRIZZLE_CHARSET_LATIN1_DANISH_CI (*built-in variable*), 15
 DRIZZLE_CHARSET_LATIN1_GENERAL_CI (*built-in variable*), 16
 DRIZZLE_CHARSET_LATIN1_GENERAL_CS (*built-in variable*), 16
 DRIZZLE_CHARSET_LATIN1_GERMAN1_CI (*built-in variable*), 15
 DRIZZLE_CHARSET_LATIN1_GERMAN2_CI (*built-in variable*), 16
 DRIZZLE_CHARSET_LATIN1_SPANISH_CI (*built-in variable*), 17
 DRIZZLE_CHARSET_LATIN1_SWEDISH_CI (*built-in variable*), 15
 DRIZZLE_CHARSET_LATIN2_BIN (*built-in variable*), 17
 DRIZZLE_CHARSET_LATIN2_CROATIAN_CI (*built-in variable*), 16
 DRIZZLE_CHARSET_LATIN2_CZECH_CS (*built-in variable*), 15
 DRIZZLE_CHARSET_LATIN2_GENERAL_CI (*built-in variable*), 15
 DRIZZLE_CHARSET_LATIN2_HUNGARIAN_CI (*built-in variable*), 15
 DRIZZLE_CHARSET_LATIN5_BIN (*built-in variable*), 17
 DRIZZLE_CHARSET_LATIN5_TURKISH_CI (*built-in variable*), 16
 DRIZZLE_CHARSET_LATIN7_BIN (*built-in variable*), 17
 DRIZZLE_CHARSET_LATIN7_ESTONIAN_CS (*built-in variable*), 15
 DRIZZLE_CHARSET_LATIN7_GENERAL_CI (*built-in variable*), 16
 DRIZZLE_CHARSET_LATIN7_GENERAL_CS (*built-in variable*), 16
 DRIZZLE_CHARSET_MACCE_BIN (*built-in variable*), 16
 DRIZZLE_CHARSET_MACCE_GENERAL_CI (*built-in variable*), 16
 DRIZZLE_CHARSET_MACROMAN_BIN (*built-in variable*), 16
 DRIZZLE_CHARSET_MACROMAN_GENERAL_CI (*built-in variable*), 16
 DRIZZLE_CHARSET_SJIS_BIN (*built-in variable*), 17
 DRIZZLE_CHARSET_SJIS_JAPANESE_CI (*built-in variable*), 15
 DRIZZLE_CHARSET_SWE7_BIN (*built-in variable*), 17
 DRIZZLE_CHARSET_SWE7_SWEDISH_CI (*built-in variable*), 15
 drizzle_charset_t (*C type*), 15
 DRIZZLE_CHARSET_TIS620_BIN (*built-in variable*), 17
 DRIZZLE_CHARSET_TIS620_THAI_CI (*built-in variable*), 15
 DRIZZLE_CHARSET_UCS2_BIN (*built-in variable*), 17
 DRIZZLE_CHARSET_UCS2_CZECH_CI (*built-in variable*), 18
 DRIZZLE_CHARSET_UCS2_DANISH_CI (*built-in variable*), 18
 DRIZZLE_CHARSET_UCS2_ESPERANTO_CI (*built-in variable*), 19
 DRIZZLE_CHARSET_UCS2_ESTONIAN_CI (*built-in variable*), 18
 DRIZZLE_CHARSET_UCS2_GENERAL_CI (*built-in variable*), 16
 DRIZZLE_CHARSET_UCS2_GENERAL_MYSQL500_CI (*built-in variable*), 19
 DRIZZLE_CHARSET_UCS2_HUNGARIAN_CI (*built-in variable*), 19
 DRIZZLE_CHARSET_UCS2_ICELANDIC_CI (*built-in variable*), 18
 DRIZZLE_CHARSET_UCS2_LATVIAN_CI (*built-in variable*), 18
 DRIZZLE_CHARSET_UCS2_LITHUANIAN_CI (*built-in variable*), 18
 DRIZZLE_CHARSET_UCS2_PERSIAN_CI (*built-in variable*), 19
 DRIZZLE_CHARSET_UCS2_POLISH_CI (*built-in variable*), 18
 DRIZZLE_CHARSET_UCS2_ROMAN_CI (*built-in variable*), 19
 DRIZZLE_CHARSET_UCS2_ROMANIAN_CI (*built-in variable*), 18
 DRIZZLE_CHARSET_UCS2_SINHALA_CI (*built-in variable*), 19
 DRIZZLE_CHARSET_UCS2_SLOVAK_CI (*built-in variable*), 18
 DRIZZLE_CHARSET_UCS2_SLOVENIAN_CI (*built-in variable*), 18
 DRIZZLE_CHARSET_UCS2_SPANISH2_CI (*built-in variable*), 18
 DRIZZLE_CHARSET_UCS2_SPANISH_CI (*built-in variable*), 18
 DRIZZLE_CHARSET_UCS2_SWEDISH_CI (*built-in variable*), 18
 DRIZZLE_CHARSET_UCS2_TURKISH_CI (*built-in variable*), 18
 DRIZZLE_CHARSET_UCS2_UNICODE_CI (*built-in variable*), 18
 DRIZZLE_CHARSET_UJIS_BIN (*built-in variable*),

17

DRIZZLE_CHARSET_UJIS_JAPANESE_CI (*built-in variable*), 15

DRIZZLE_CHARSET_UTF16_BIN (*built-in variable*), 16

DRIZZLE_CHARSET_UTF16_CZECH_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_DANISH_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_ESPERANTO_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_ESTONIAN_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_GENERAL_CI (*built-in variable*), 16

DRIZZLE_CHARSET_UTF16_HUNGARIAN_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_ICELANDIC_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_LATVIAN_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_LITHUANIAN_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_PERSIAN_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_POLISH_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_ROMAN_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_ROMANIAN_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_SINHALA_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_SLOVAK_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_SLOVENIAN_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_SPANISH2_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_SPANISH_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_SWEDISH_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_TURKISH_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF16_UNICODE_CI (*built-in variable*), 18

DRIZZLE_CHARSET_UTF32_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_UTF32_CZECH_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_DANISH_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_ESPERANTO_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_ESTONIAN_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_GENERAL_CI (*built-in variable*), 16

DRIZZLE_CHARSET_UTF32_HUNGARIAN_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_ICELANDIC_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_LATVIAN_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_LITHUANIAN_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_PERSIAN_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_POLISH_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_ROMAN_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_ROMANIAN_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_SINHALA_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_SLOVAK_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_SLOVENIAN_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_SPANISH2_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_SPANISH_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_SWEDISH_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_TURKISH_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF32_UNICODE_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF8_BIN (*built-in variable*), 17

DRIZZLE_CHARSET_UTF8_CZECH_CI (*built-in variable*), 20

DRIZZLE_CHARSET_UTF8_DANISH_CI (*built-in variable*), 20

DRIZZLE_CHARSET_UTF8_ESPERANTO_CI (*built-in variable*), 20

DRIZZLE_CHARSET_UTF8_ESTONIAN_CI (*built-in variable*), 19

DRIZZLE_CHARSET_UTF8_GENERAL_CI (*built-in variable*), 16

DRIZZLE_CHARSET_UTF8_GENERAL_MYSQL500_CI (*built-in variable*), 20

DRIZZLE_CHARSET_UTF8_HUNGARIAN_CI (*built-in variable*), 20

DRIZZLE_CHARSET_UTF8_ICELANDIC_CI (*built-*

<i>in variable</i>), 19	<i>variable</i>), 20
DRIZZLE_CHARSET_UTF8_LATVIAN_CI (<i>built-in variable</i>), 19	DRIZZLE_CHARSET_UTF8MB4_ROMANIAN_CI (<i>built-in variable</i>), 20
DRIZZLE_CHARSET_UTF8_LITHUANIAN_CI (<i>built-in variable</i>), 20	DRIZZLE_CHARSET_UTF8MB4_SINHALA_CI (<i>built-in variable</i>), 20
DRIZZLE_CHARSET_UTF8_PERSIAN_CI (<i>built-in variable</i>), 20	DRIZZLE_CHARSET_UTF8MB4_SLOVAK_CI (<i>built-in variable</i>), 20
DRIZZLE_CHARSET_UTF8_POLISH_CI (<i>built-in variable</i>), 19	DRIZZLE_CHARSET_UTF8MB4_SLOVENIAN_CI (<i>built-in variable</i>), 20
DRIZZLE_CHARSET_UTF8_ROMAN_CI (<i>built-in variable</i>), 20	DRIZZLE_CHARSET_UTF8MB4_SPANISH2_CI (<i>built-in variable</i>), 20
DRIZZLE_CHARSET_UTF8_ROMANIAN_CI (<i>built-in variable</i>), 19	DRIZZLE_CHARSET_UTF8MB4_SPANISH_CI (<i>built-in variable</i>), 20
DRIZZLE_CHARSET_UTF8_SINHALA_CI (<i>built-in variable</i>), 20	DRIZZLE_CHARSET_UTF8MB4_SWEDISH_CI (<i>built-in variable</i>), 20
DRIZZLE_CHARSET_UTF8_SLOVAK_CI (<i>built-in variable</i>), 20	DRIZZLE_CHARSET_UTF8MB4_TURKISH_CI (<i>built-in variable</i>), 20
DRIZZLE_CHARSET_UTF8_SLOVENIAN_CI (<i>built-in variable</i>), 19	DRIZZLE_CHARSET_UTF8MB4_UNICODE_CI (<i>built-in variable</i>), 20
DRIZZLE_CHARSET_UTF8_SPANISH2_CI (<i>built-in variable</i>), 20	drizzle_close (<i>C function</i>), 36
DRIZZLE_CHARSET_UTF8_SPANISH_CI (<i>built-in variable</i>), 19	drizzle_column_buffer (<i>C function</i>), 43
DRIZZLE_CHARSET_UTF8_SWEDISH_CI (<i>built-in variable</i>), 19	drizzle_column_catalog (<i>C function</i>), 41
DRIZZLE_CHARSET_UTF8_TURKISH_CI (<i>built-in variable</i>), 19	drizzle_column_charset (<i>C function</i>), 42
DRIZZLE_CHARSET_UTF8_UNICODE_CI (<i>built-in variable</i>), 19	drizzle_column_current (<i>C function</i>), 44
DRIZZLE_CHARSET_UTF8MB4_BIN (<i>built-in variable</i>), 16	drizzle_column_db (<i>C function</i>), 41
DRIZZLE_CHARSET_UTF8MB4_CZECH_CI (<i>built-in variable</i>), 20	drizzle_column_decimals (<i>C function</i>), 43
DRIZZLE_CHARSET_UTF8MB4_DANISH_CI (<i>built-in variable</i>), 20	drizzle_column_default_value (<i>C function</i>), 43
DRIZZLE_CHARSET_UTF8MB4_ESPERANTO_CI (<i>built-in variable</i>), 20	drizzle_column_drizzle_result (<i>C function</i>), 41
DRIZZLE_CHARSET_UTF8MB4_ESTONIAN_CI (<i>built-in variable</i>), 20	drizzle_column_flags (<i>C function</i>), 43
DRIZZLE_CHARSET_UTF8MB4_GENERAL_CI (<i>built-in variable</i>), 16	DRIZZLE_COLUMN_FLAGS_AUTO_INCREMENT (<i>built-in variable</i>), 24
DRIZZLE_CHARSET_UTF8MB4_HUNGARIAN_CI (<i>built-in variable</i>), 20	DRIZZLE_COLUMN_FLAGS_BINARY (<i>built-in variable</i>), 24
DRIZZLE_CHARSET_UTF8MB4_ICELANDIC_CI (<i>built-in variable</i>), 20	DRIZZLE_COLUMN_FLAGS_BINCMP (<i>built-in variable</i>), 25
DRIZZLE_CHARSET_UTF8MB4_LATVIAN_CI (<i>built-in variable</i>), 20	DRIZZLE_COLUMN_FLAGS_BLOB (<i>built-in variable</i>), 24
DRIZZLE_CHARSET_UTF8MB4_LITHUANIAN_CI (<i>built-in variable</i>), 20	DRIZZLE_COLUMN_FLAGS_ENUM (<i>built-in variable</i>), 24
DRIZZLE_CHARSET_UTF8MB4_PERSIAN_CI (<i>built-in variable</i>), 20	DRIZZLE_COLUMN_FLAGS_GET_FIXED_FIELDS (<i>built-in variable</i>), 25
DRIZZLE_CHARSET_UTF8MB4_POLISH_CI (<i>built-in variable</i>), 20	DRIZZLE_COLUMN_FLAGS_GROUP (<i>built-in variable</i>), 25
DRIZZLE_CHARSET_UTF8MB4_ROMAN_CI (<i>built-in</i>	DRIZZLE_COLUMN_FLAGS_IN_ADD_INDEX (<i>built-in variable</i>), 25

24
DRIZZLE_COLUMN_FLAGS_NOT_NULL (*built-in variable*), 24
DRIZZLE_COLUMN_FLAGS_NUM (*built-in variable*), 25
DRIZZLE_COLUMN_FLAGS_ON_UPDATE_NOW (*built-in variable*), 25
DRIZZLE_COLUMN_FLAGS_PART_KEY (*built-in variable*), 25
DRIZZLE_COLUMN_FLAGS_PRI_KEY (*built-in variable*), 24
DRIZZLE_COLUMN_FLAGS_RENAMED (*built-in variable*), 25
DRIZZLE_COLUMN_FLAGS_SET (*built-in variable*), 25
drizzle_column_flags_t (*C type*), 24
DRIZZLE_COLUMN_FLAGS_TIMESTAMP (*built-in variable*), 25
DRIZZLE_COLUMN_FLAGS_UNIQUE (*built-in variable*), 25
DRIZZLE_COLUMN_FLAGS_UNIQUE_KEY (*built-in variable*), 24
DRIZZLE_COLUMN_FLAGS_UNSIGNED (*built-in variable*), 24
DRIZZLE_COLUMN_FLAGS_ZEROFILL (*built-in variable*), 24
drizzle_column_free (*C function*), 43
drizzle_column_index (*C function*), 44
drizzle_column_max_size (*C function*), 42
drizzle_column_name (*C function*), 42
drizzle_column_next (*C function*), 44
drizzle_column_options_t (*C type*), 24
drizzle_column_orig_name (*C function*), 42
drizzle_column_orig_table (*C function*), 41
drizzle_column_prev (*C function*), 44
drizzle_column_read (*C function*), 43
drizzle_column_seek (*C function*), 44
drizzle_column_size (*C function*), 42
drizzle_column_skip (*C function*), 43
drizzle_column_skip_all (*C function*), 43
drizzle_column_st (*C type*), 38
drizzle_column_table (*C function*), 41
drizzle_column_type (*C function*), 42
DRIZZLE_COLUMN_TYPE_BIT (*built-in variable*), 24
DRIZZLE_COLUMN_TYPE_BLOB (*built-in variable*), 24
DRIZZLE_COLUMN_TYPE_DATE (*built-in variable*), 23
DRIZZLE_COLUMN_TYPE_DATETIME (*built-in variable*), 23
DRIZZLE_COLUMN_TYPE_DECIMAL (*built-in variable*), 23
DRIZZLE_COLUMN_TYPE_DOUBLE (*built-in variable*), 23
DRIZZLE_COLUMN_TYPE_ENUM (*built-in variable*), 24
DRIZZLE_COLUMN_TYPE_FLOAT (*built-in variable*), 23
DRIZZLE_COLUMN_TYPE_GEOMETRY (*built-in variable*), 24
DRIZZLE_COLUMN_TYPE_INT24 (*built-in variable*), 23
DRIZZLE_COLUMN_TYPE_LONG (*built-in variable*), 23
DRIZZLE_COLUMN_TYPE_LONG_BLOB (*built-in variable*), 24
DRIZZLE_COLUMN_TYPE_LONGLONG (*built-in variable*), 23
DRIZZLE_COLUMN_TYPE_MEDIUM_BLOB (*built-in variable*), 24
DRIZZLE_COLUMN_TYPE_NEWDATE (*built-in variable*), 24
DRIZZLE_COLUMN_TYPE_NEWDECIMAL (*built-in variable*), 24
DRIZZLE_COLUMN_TYPE_NULL (*built-in variable*), 23
DRIZZLE_COLUMN_TYPE_SET (*built-in variable*), 24
DRIZZLE_COLUMN_TYPE_SHORT (*built-in variable*), 23
drizzle_column_type_str (*C function*), 42
DRIZZLE_COLUMN_TYPE_STRING (*built-in variable*), 24
drizzle_column_type_t (*C type*), 23
DRIZZLE_COLUMN_TYPE_TIME (*built-in variable*), 23
DRIZZLE_COLUMN_TYPE_TIMESTAMP (*built-in variable*), 23
DRIZZLE_COLUMN_TYPE_TINY (*built-in variable*), 23
DRIZZLE_COLUMN_TYPE_TINY_BLOB (*built-in variable*), 24
DRIZZLE_COLUMN_TYPE_VAR_STRING (*built-in variable*), 24
DRIZZLE_COLUMN_TYPE_VARCHAR (*built-in variable*), 24
DRIZZLE_COLUMN_TYPE_YEAR (*built-in variable*), 23
DRIZZLE_COLUMN_UNUSED (*built-in variable*), 24
DRIZZLE_CON_STATUS_AUTOCOMMIT (*built-in variable*), 21
DRIZZLE_CON_STATUS_CURSOR_EXISTS (*built-in variable*), 21
DRIZZLE_CON_STATUS_DB_DROPPED (*built-in variable*), 21
DRIZZLE_CON_STATUS_IN_TRANS (*built-in variable*), 20
DRIZZLE_CON_STATUS_LAST_ROW_SENT (*built-in variable*), 21

DRIZZLE_CON_STATUS_MORE_RESULTS_EXISTS (built-in variable), 21
 DRIZZLE_CON_STATUS_NO_BACKSLASH_ESCAPES (built-in variable), 21
 DRIZZLE_CON_STATUS_NONE (built-in variable), 20
 DRIZZLE_CON_STATUS_QUERY_NO_GOOD_INDEX_USED (built-in variable), 21
 DRIZZLE_CON_STATUS_QUERY_NO_INDEX_USED (built-in variable), 21
 DRIZZLE_CON_STATUS_QUERY_WAS_SLOW (built-in variable), 21
 drizzle_connect (C function), 36
 drizzle_context (C function), 34
 drizzle_create (C function), 28
 drizzle_datetime_st (C type), 47
 drizzle_db (C function), 34
 drizzle_errno (C function), 30
 drizzle_error (C function), 30
 drizzle_error_code (C function), 30
 drizzle_escape_str (C function), 38
 drizzle_escape_string (C function), 39
 DRIZZLE_EVENT_POSITION_EXTRA_FLAGS (built-in variable), 27
 DRIZZLE_EVENT_POSITION_FLAGS (built-in variable), 27
 DRIZZLE_EVENT_POSITION_LENGTH (built-in variable), 27
 DRIZZLE_EVENT_POSITION_NEXT (built-in variable), 27
 DRIZZLE_EVENT_POSITION_SERVERID (built-in variable), 27
 DRIZZLE_EVENT_POSITION_TIMESTAMP (built-in variable), 27
 DRIZZLE_EVENT_POSITION_TYPE (built-in variable), 27
 DRIZZLE_EVENT_TYPE_ANONYMOUS_GTID (built-in variable), 27
 DRIZZLE_EVENT_TYPE_APPEND_BLOCK (built-in variable), 26
 DRIZZLE_EVENT_TYPE_BEGIN_LOAD_QUERY (built-in variable), 26
 DRIZZLE_EVENT_TYPE_CREATE_FILE (built-in variable), 26
 DRIZZLE_EVENT_TYPE_DELETE_FILE (built-in variable), 26
 DRIZZLE_EVENT_TYPE_EXEC_LOAD (built-in variable), 26
 DRIZZLE_EVENT_TYPE_EXECUTE_LOAD_QUERY (built-in variable), 26
 DRIZZLE_EVENT_TYPE_FORMAT_DESCRIPTION (built-in variable), 26
 DRIZZLE_EVENT_TYPE_GTID (built-in variable), 27
 DRIZZLE_EVENT_TYPE_HEARTBEAT (built-in variable), 27
 DRIZZLE_EVENT_TYPE_IGNOREABLE (built-in variable), 27
 DRIZZLE_EVENT_TYPE_INCIDENT (built-in variable), 27
 DRIZZLE_EVENT_TYPE_INTVAR (built-in variable), 26
 DRIZZLE_EVENT_TYPE_LOAD (built-in variable), 26
 DRIZZLE_EVENT_TYPE_NEW_LOAD (built-in variable), 26
 DRIZZLE_EVENT_TYPE_OBSOLETE_DELETE_ROWS (built-in variable), 27
 DRIZZLE_EVENT_TYPE_OBSOLETE_UPDATE_ROWS (built-in variable), 27
 DRIZZLE_EVENT_TYPE_OBSOLETE_WRITE_ROWS (built-in variable), 26
 DRIZZLE_EVENT_TYPE_PREVIOUS_GTIDS (built-in variable), 27
 DRIZZLE_EVENT_TYPE_QUERY (built-in variable), 26
 DRIZZLE_EVENT_TYPE_RAND (built-in variable), 26
 DRIZZLE_EVENT_TYPE_ROTATE (built-in variable), 26
 DRIZZLE_EVENT_TYPE_ROWS_QUERY (built-in variable), 27
 DRIZZLE_EVENT_TYPE_START (built-in variable), 26
 DRIZZLE_EVENT_TYPE_STOP (built-in variable), 26
 DRIZZLE_EVENT_TYPE_TABLE_MAP (built-in variable), 26
 DRIZZLE_EVENT_TYPE_UNKNOWN (built-in variable), 26
 DRIZZLE_EVENT_TYPE_USER_VAR (built-in variable), 26
 DRIZZLE_EVENT_TYPE_V1_DELETE_ROWS (built-in variable), 27
 DRIZZLE_EVENT_TYPE_V1_UPDATE_ROWS (built-in variable), 27
 DRIZZLE_EVENT_TYPE_V1_WRITE_ROWS (built-in variable), 27
 DRIZZLE_EVENT_TYPE_V2_DELETE_ROWS (built-in variable), 27
 DRIZZLE_EVENT_TYPE_V2_UPDATE_ROWS (built-in variable), 27
 DRIZZLE_EVENT_TYPE_V2_WRITE_ROWS (built-in variable), 27
 DRIZZLE_EVENT_TYPE_XID (built-in variable), 26
 drizzle_event_watch_fn (C function), 37
 drizzle_failed (C function), 46
 drizzle_fd (C function), 29
 drizzle_field_buffer (C function), 46
 drizzle_field_free (C function), 46
 drizzle_field_read (C function), 46
 drizzle_field_t (C type), 23
 drizzle_host (C function), 34

`drizzle_kill` (*C function*), 37
`drizzle_library_deinit` (*C function*), 28
`drizzle_library_init` (*C function*), 28
`drizzle_log_fn` (*C function*), 37
`drizzle_max_packet_size` (*C function*), 35
`drizzle_options_create` (*C function*), 31
`drizzle_options_destroy` (*C function*), 31
`drizzle_options_get_auth_plugin` (*C function*), 33
`drizzle_options_get_found_rows` (*C function*), 32
`drizzle_options_get_interactive` (*C function*), 33
`drizzle_options_get_multi_statements` (*C function*), 33
`drizzle_options_get_non_blocking` (*C function*), 32
`drizzle_options_get_raw_scramble` (*C function*), 32
`drizzle_options_get_socket_owner` (*C function*), 33
`drizzle_options_set_auth_plugin` (*C function*), 33
`drizzle_options_set_found_rows` (*C function*), 32
`drizzle_options_set_interactive` (*C function*), 32
`drizzle_options_set_multi_statements` (*C function*), 33
`drizzle_options_set_non_blocking` (*C function*), 32
`drizzle_options_set_raw_scramble` (*C function*), 32
`drizzle_options_set_socket_owner` (*C function*), 33
`drizzle_options_st` (*C type*), 28
`drizzle_ping` (*C function*), 37
`drizzle_port` (*C function*), 34
`drizzle_protocol_version` (*C function*), 34
`drizzle_query` (*C function*), 38
`drizzle_query_st` (*C type*), 38
`drizzle_quit` (*C function*), 36
`drizzle_ready` (*C function*), 36
`drizzle_result_affected_rows` (*C function*), 40
`DRIZZLE_RESULT_BINARY_ROWS` (*built-in variable*), 25
`drizzle_result_buffer` (*C function*), 41
`DRIZZLE_RESULT_BUFFER_COLUMN` (*built-in variable*), 25
`DRIZZLE_RESULT_BUFFER_ROW` (*built-in variable*), 25
`drizzle_result_column_count` (*C function*), 40
`drizzle_result_drizzle_con` (*C function*), 39
`drizzle_result_eof` (*C function*), 39
`DRIZZLE_RESULT_EOF_PACKET` (*built-in variable*), 25
`drizzle_result_error_code` (*C function*), 40
`drizzle_result_free` (*C function*), 39
`drizzle_result_free_all` (*C function*), 39
`drizzle_result_insert_id` (*C function*), 40
`drizzle_result_message` (*C function*), 39
`DRIZZLE_RESULT_NONE` (*built-in variable*), 25
`drizzle_result_options_t` (*C type*), 25
`drizzle_result_read` (*C function*), 41
`DRIZZLE_RESULT_ROW_BREAK` (*built-in variable*), 25
`drizzle_result_row_count` (*C function*), 40
`drizzle_result_row_size` (*C function*), 41
`DRIZZLE_RESULT_SKIP_COLUMN` (*built-in variable*), 25
`drizzle_result_sqlstate` (*C function*), 40
`drizzle_result_st` (*C type*), 38
`drizzle_result_warning_count` (*C function*), 40
`DRIZZLE_RETURN_AUTH_FAILED` (*built-in variable*), 14
`DRIZZLE_RETURN_BAD_HANDSHAKE_PACKET` (*built-in variable*), 14
`DRIZZLE_RETURN_BAD_PACKET` (*built-in variable*), 14
`DRIZZLE_RETURN_BAD_PACKET_NUMBER` (*built-in variable*), 14
`DRIZZLE_RETURN_BINLOG_CRC` (*built-in variable*), 15
`DRIZZLE_RETURN_COULD_NOT_CONNECT` (*built-in variable*), 14
`DRIZZLE_RETURN_EOF` (*built-in variable*), 15
`DRIZZLE_RETURN_ERRNO` (*built-in variable*), 14
`DRIZZLE_RETURN_ERROR_CODE` (*built-in variable*), 14
`DRIZZLE_RETURN_GETADDRINFO` (*built-in variable*), 14
`DRIZZLE_RETURN_HANDSHAKE_FAILED` (*built-in variable*), 14
`DRIZZLE_RETURN_INTERNAL_ERROR` (*built-in variable*), 14
`DRIZZLE_RETURN_INVALID_ARGUMENT` (*built-in variable*), 14
`DRIZZLE_RETURN_INVALID_CONVERSION` (*built-in variable*), 15
`DRIZZLE_RETURN_IO_WAIT` (*built-in variable*), 13
`DRIZZLE_RETURN_LOST_CONNECTION` (*built-in variable*), 14
`DRIZZLE_RETURN_MEMORY` (*built-in variable*), 13
`DRIZZLE_RETURN_NO_ACTIVE_CONNECTIONS` (*built-in variable*), 14
`DRIZZLE_RETURN_NO_SCRAMBLE` (*built-in variable*), 14

- able*), 14
- DRIZZLE_RETURN_NOT_FOUND (*built-in variable*), 15
- DRIZZLE_RETURN_NOT_READY (*built-in variable*), 14
- DRIZZLE_RETURN_NULL_SIZE (*built-in variable*), 14
- DRIZZLE_RETURN_OK (*built-in variable*), 13
- DRIZZLE_RETURN_PAUSE (*built-in variable*), 13
- DRIZZLE_RETURN_PROTOCOL_NOT_SUPPORTED (*built-in variable*), 14
- DRIZZLE_RETURN_ROW_BREAK (*built-in variable*), 13
- DRIZZLE_RETURN_ROW_END (*built-in variable*), 14
- DRIZZLE_RETURN_SSL_ERROR (*built-in variable*), 14
- DRIZZLE_RETURN_STMT_ERROR (*built-in variable*), 15
- drizzle_return_t (*C type*), 13
- DRIZZLE_RETURN_TIMEOUT (*built-in variable*), 14
- DRIZZLE_RETURN_TOO_MANY_COLUMNS (*built-in variable*), 14
- DRIZZLE_RETURN_TRUNCATED (*built-in variable*), 15
- DRIZZLE_RETURN_UNEXPECTED_DATA (*built-in variable*), 14
- drizzle_row_buffer (*C function*), 44
- drizzle_row_current (*C function*), 45
- drizzle_row_field_sizes (*C function*), 45
- drizzle_row_free (*C function*), 45
- drizzle_row_index (*C function*), 45
- drizzle_row_next (*C function*), 45
- drizzle_row_prev (*C function*), 45
- drizzle_row_read (*C function*), 44
- drizzle_row_seek (*C function*), 45
- drizzle_row_t (*C type*), 23
- drizzle_scramble (*C function*), 35
- drizzle_select_db (*C function*), 36
- drizzle_server_version (*C function*), 35
- drizzle_server_version_number (*C function*), 35
- drizzle_set_context (*C function*), 34
- drizzle_set_context_free_fn (*C function*), 34
- drizzle_set_event_watch_fn (*C function*), 30
- drizzle_set_events (*C function*), 30
- drizzle_set_log_fn (*C function*), 29
- drizzle_set_revents (*C function*), 30
- drizzle_set_ssl (*C function*), 38
- drizzle_set_timeout (*C function*), 29
- drizzle_set_verbose (*C function*), 29
- drizzle_shutdown (*C function*), 36
- drizzle_socket_get_option (*C function*), 32
- DRIZZLE_SOCKET_OPTION_KEEPCNT (*built-in variable*), 23
- DRIZZLE_SOCKET_OPTION_KEEPIDL (built-in variable), 23
- DRIZZLE_SOCKET_OPTION_KEEPIDL (built-in variable), 23
- drizzle_socket_option_t (*C type*), 22
- DRIZZLE_SOCKET_OPTION_TIMEOUT (built-in variable), 23
- DRIZZLE_SOCKET_OWNER_CLIENT (built-in variable), 22
- DRIZZLE_SOCKET_OWNER_NATIVE (built-in variable), 22
- drizzle_socket_owner_t (*C type*), 22
- drizzle_socket_set_option (*C function*), 31
- drizzle_socket_set_options (*C function*), 31
- drizzle_sqlstate (*C function*), 31
- DRIZZLE_SSL_STATE_HANDSHAKE_COMPLETE (built-in variable), 22
- DRIZZLE_SSL_STATE_NONE (built-in variable), 22
- drizzle_ssl_state_t (*C type*), 22
- drizzle_st (*C type*), 28
- drizzle_status (*C function*), 35
- drizzle_status_t (*C type*), 20
- drizzle_stmt_affected_rows (*C function*), 53
- drizzle_stmt_buffer (*C function*), 50
- drizzle_stmt_close (*C function*), 53
- drizzle_stmt_column_count (*C function*), 53
- drizzle_stmt_execute (*C function*), 49
- drizzle_stmt_fetch (*C function*), 50
- drizzle_stmt_get_bigint (*C function*), 52
- drizzle_stmt_get_bigint_from_name (*C function*), 52
- drizzle_stmt_get_double (*C function*), 52
- drizzle_stmt_get_double_from_name (*C function*), 53
- drizzle_stmt_get_int (*C function*), 51
- drizzle_stmt_get_int_from_name (*C function*), 52
- drizzle_stmt_get_is_null (*C function*), 50
- drizzle_stmt_get_is_null_from_name (*C function*), 50
- drizzle_stmt_get_is_unsigned (*C function*), 51
- drizzle_stmt_get_is_unsigned_from_name (*C function*), 51
- drizzle_stmt_get_string (*C function*), 51
- drizzle_stmt_get_string_from_name (*C function*), 51
- drizzle_stmt_insert_id (*C function*), 53
- drizzle_stmt_param_count (*C function*), 53
- drizzle_stmt_prepare (*C function*), 47
- drizzle_stmt_reset (*C function*), 50
- drizzle_stmt_row_count (*C function*), 53
- drizzle_stmt_send_long_data (*C function*), 49
- drizzle_stmt_set_bigint (*C function*), 48

`drizzle_stmt_set_double` (*C function*), 48
`drizzle_stmt_set_float` (*C function*), 48
`drizzle_stmt_set_int` (*C function*), 47
`drizzle_stmt_set_null` (*C function*), 48
`drizzle_stmt_set_short` (*C function*), 47
`drizzle_stmt_set_string` (*C function*), 48
`drizzle_stmt_set_time` (*C function*), 49
`drizzle_stmt_set_timestamp` (*C function*), 49
`drizzle_stmt_set_tiny` (*C function*), 47
`drizzle_stmt_st` (*C type*), 47
`drizzle_stmt_state_t` (*C type*), 25
`drizzle_strerror` (*C function*), 37
`drizzle_success` (*C function*), 46
`drizzle_thread_id` (*C function*), 35
`drizzle_timeout` (*C function*), 29
`drizzle_user` (*C function*), 34
`drizzle_verbose` (*C function*), 29
`DRIZZLE_VERBOSE_CRAZY` (*built-in variable*), 11
`DRIZZLE_VERBOSE_DEBUG` (*built-in variable*), 11
`DRIZZLE_VERBOSE_ERROR` (*built-in variable*), 11
`DRIZZLE_VERBOSE_FATAL` (*built-in variable*), 11
`DRIZZLE_VERBOSE_INFO` (*built-in variable*), 11
`drizzle_verbose_name` (*C function*), 28
`DRIZZLE_VERBOSE_NEVER` (*built-in variable*), 11
`drizzle_verbose_t` (*C type*), 11
`drizzle_version` (*C function*), 28
`drizzle_wait` (*C function*), 36