

---

# libdrizle-redux Documentation

*Release stable*

Feb 10, 2019



---

## Contents

---

|          |                                            |           |
|----------|--------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                        | <b>1</b>  |
| 1.1      | Differences from Libdrizzle . . . . .      | 1         |
| 1.2      | Differences from libmysqlclient . . . . .  | 1         |
| <b>2</b> | <b>Licensing</b>                           | <b>3</b>  |
| 2.1      | Documentation Content . . . . .            | 3         |
| 2.2      | About the Drizzle logo . . . . .           | 3         |
| 2.3      | Libdrizzle Redux License . . . . .         | 3         |
| 2.4      | Trademarks . . . . .                       | 4         |
| <b>3</b> | <b>Installing Libdrizzle Redux</b>         | <b>5</b>  |
| 3.1      | Debian . . . . .                           | 5         |
| 3.2      | Redhat . . . . .                           | 6         |
| <b>4</b> | <b>Compiling Libdrizzle Redux</b>          | <b>7</b>  |
| 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         |
| <b>5</b> | <b>Libdrizzle API</b>                      | <b>11</b> |
| 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        |
| <b>6</b> | <b>Code Examples</b>                       | <b>59</b> |
| 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

---

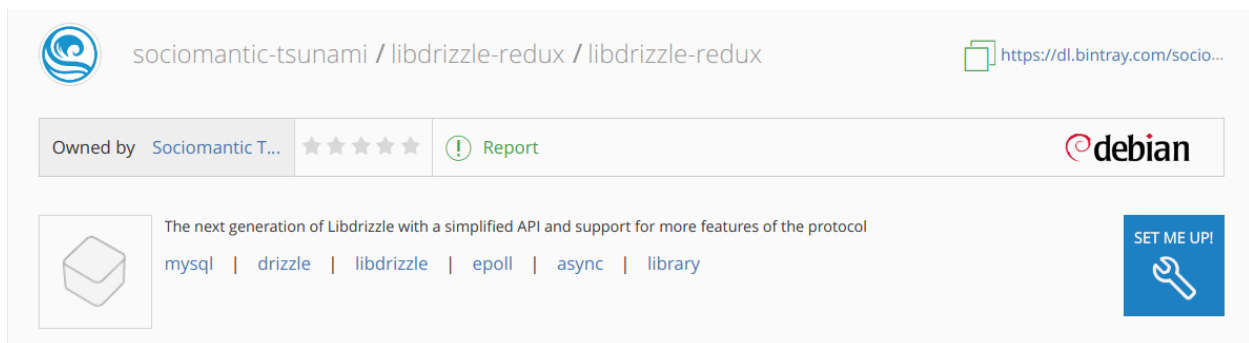
### Installing Libdrizzle Redux

---

#### 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 <sup>2</sup> | >= 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.

---

<sup>2</sup> 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 name<sup>1</sup>. 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.:

<sup>1</sup> 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

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



|                                                                                         |                                    |
|-----------------------------------------------------------------------------------------|------------------------------------|
| <b>DRIZZLE_MAX_BUFFER_SIZE</b>                                                          | <b>1024*1024*1024</b>              |
| Maximum size of the allocated buffer on a <i>drizzle_st</i>                             |                                    |
| <b>DRIZZLE_DEFAULT_BUFFER_SIZE</b>                                                      | <b>1024*1024</b>                   |
| Default size of the allocated buffer on a <i>drizzle_st</i>                             |                                    |
| <b>DRIZZLE_BUFFER_COPY_THRESHOLD</b>                                                    | <b>8192</b>                        |
| Unused                                                                                  |                                    |
| <b>DRIZZLE_MAX_SERVER_VERSION_SIZE</b>                                                  | <b>32</b>                          |
| Maximum length of the server version string                                             |                                    |
| <b>DRIZZLE_MAX_SERVER_EXTRA_SIZE</b>                                                    | <b>32</b>                          |
| Maximum size of additional data sent after server handshake                             |                                    |
| <b>DRIZZLE_MAX_SCRAMBLE_SIZE</b>                                                        | <b>20</b>                          |
| Maximum size of the buffer used during authentication if password scrambling is enabled |                                    |
| <b>DRIZZLE_STATE_STACK_SIZE</b>                                                         | <b>8</b>                           |
| Maximum number of states saved on the stack                                             |                                    |
| <b>DRIZZLE_ROW_GROW_SIZE</b>                                                            | <b>8192</b>                        |
| The number of rows to read at a time when buffering a result                            |                                    |
| <b>DRIZZLE_DEFAULT_SOCKET_TIMEOUT</b>                                                   | <b>10</b>                          |
| The default time in seconds to wait before a setsockopt call times out                  |                                    |
| <b>DRIZZLE_DEFAULT_SOCKET_SEND_SIZE</b>                                                 | <b>DRIZZLE_DEFAULT_BUFFER_SIZE</b> |
| The default size of the socket send buffer                                              |                                    |
| <b>DRIZZLE_DEFAULT_SOCKET_RECV_SIZE</b>                                                 | <b>DRIZZLE_DEFAULT_BUFFER_SIZE</b> |
| The default size of the socket receive buffer                                           |                                    |
| <b>DRIZZLE_MYSQL_PASSWORD_HASH</b>                                                      | <b>41</b>                          |
| Unused                                                                                  |                                    |
| <b>DRIZZLE_BINLOG_CRC32_LEN</b>                                                         | <b>4</b>                           |
| Size of the CRC32 checksum appended to each binlog event                                |                                    |
| <b>DRIZZLE_BINLOG_CHECKSUM_VERSION</b>                                                  | <b>"5.6.1"</b>                     |
| From this version and higher automatic checksums is on                                  |                                    |
| <b>DRIZZLE_BINLOG_MAGIC</b>                                                             | <b>"xFeX62x69x6E"</b>              |
| 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                                                          |                                                                  | (built-in variable), 19 |
| DRIZZLE_CHARSET_UJIS_JAPANESE_CI (built-in variable), 15    | DRIZZLE_CHARSET_UTF32_ESTONIAN_CI (built-in variable), 19        |                         |
| DRIZZLE_CHARSET_UTF16_BIN (built-in variable), 16           | DRIZZLE_CHARSET_UTF32_GENERAL_CI (built-in variable), 16         |                         |
| DRIZZLE_CHARSET_UTF16_CZECH_CI (built-in variable), 18      | DRIZZLE_CHARSET_UTF32_HUNGARIAN_CI (built-in variable), 19       |                         |
| DRIZZLE_CHARSET_UTF16_DANISH_CI (built-in variable), 18     | DRIZZLE_CHARSET_UTF32_ICELANDIC_CI (built-in variable), 19       |                         |
| DRIZZLE_CHARSET_UTF16_ESPERANTO_CI (built-in variable), 18  | DRIZZLE_CHARSET_UTF32_LATVIAN_CI (built-in variable), 19         |                         |
| DRIZZLE_CHARSET_UTF16_ESTONIAN_CI (built-in variable), 18   | DRIZZLE_CHARSET_UTF32_LITHUANIAN_CI (built-in variable), 19      |                         |
| DRIZZLE_CHARSET_UTF16_GENERAL_CI (built-in variable), 16    | DRIZZLE_CHARSET_UTF32_PERSIAN_CI (built-in variable), 19         |                         |
| DRIZZLE_CHARSET_UTF16_HUNGARIAN_CI (built-in variable), 18  | DRIZZLE_CHARSET_UTF32_POLISH_CI (built-in variable), 19          |                         |
| DRIZZLE_CHARSET_UTF16_ICELANDIC_CI (built-in variable), 18  | DRIZZLE_CHARSET_UTF32_ROMAN_CI (built-in variable), 19           |                         |
| DRIZZLE_CHARSET_UTF16_LATVIAN_CI (built-in variable), 18    | DRIZZLE_CHARSET_UTF32_ROMANIAN_CI (built-in variable), 19        |                         |
| DRIZZLE_CHARSET_UTF16_LITHUANIAN_CI (built-in variable), 18 | DRIZZLE_CHARSET_UTF32_SINHALA_CI (built-in variable), 19         |                         |
| DRIZZLE_CHARSET_UTF16_PERSIAN_CI (built-in variable), 18    | DRIZZLE_CHARSET_UTF32_SLOVAK_CI (built-in variable), 19          |                         |
| DRIZZLE_CHARSET_UTF16_POLISH_CI (built-in variable), 18     | DRIZZLE_CHARSET_UTF32_SLOVENIAN_CI (built-in variable), 19       |                         |
| DRIZZLE_CHARSET_UTF16_ROMAN_CI (built-in variable), 18      | DRIZZLE_CHARSET_UTF32_SPANISH2_CI (built-in variable), 19        |                         |
| DRIZZLE_CHARSET_UTF16_ROMANIAN_CI (built-in variable), 18   | DRIZZLE_CHARSET_UTF32_SPANISH_CI (built-in variable), 19         |                         |
| DRIZZLE_CHARSET_UTF16_SINHALA_CI (built-in variable), 18    | DRIZZLE_CHARSET_UTF32_SWEDISH_CI (built-in variable), 19         |                         |
| DRIZZLE_CHARSET_UTF16_SLOVAK_CI (built-in variable), 18     | DRIZZLE_CHARSET_UTF32_TURKISH_CI (built-in variable), 19         |                         |
| DRIZZLE_CHARSET_UTF16_SLOVENIAN_CI (built-in variable), 18  | DRIZZLE_CHARSET_UTF32_UNICODE_CI (built-in variable), 19         |                         |
| DRIZZLE_CHARSET_UTF16_SPANISH2_CI (built-in variable), 18   | DRIZZLE_CHARSET_UTF8_BIN (built-in variable), 17                 |                         |
| DRIZZLE_CHARSET_UTF16_SPANISH_CI (built-in variable), 18    | DRIZZLE_CHARSET_UTF8_CZECH_CI (built-in variable), 20            |                         |
| DRIZZLE_CHARSET_UTF16_SWEDISH_CI (built-in variable), 18    | DRIZZLE_CHARSET_UTF8_DANISH_CI (built-in variable), 20           |                         |
| DRIZZLE_CHARSET_UTF16_TURKISH_CI (built-in variable), 18    | DRIZZLE_CHARSET_UTF8_ESPERANTO_CI (built-in variable), 20        |                         |
| DRIZZLE_CHARSET_UTF16_UNICODE_CI (built-in variable), 18    | DRIZZLE_CHARSET_UTF8_ESTONIAN_CI (built-in variable), 19         |                         |
| DRIZZLE_CHARSET_UTF32_BIN (built-in variable), 17           | DRIZZLE_CHARSET_UTF8_GENERAL_CI (built-in variable), 16          |                         |
| DRIZZLE_CHARSET_UTF32_CZECH_CI (built-in variable), 19      | DRIZZLE_CHARSET_UTF8_GENERAL_MYSQL500_CI (built-in variable), 20 |                         |
| DRIZZLE_CHARSET_UTF32_DANISH_CI (built-in variable), 19     | DRIZZLE_CHARSET_UTF8_HUNGARIAN_CI (built-in variable), 20        |                         |
| DRIZZLE_CHARSET_UTF32_ESPERANTO_CI                          | 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