# MySQL Proxy Guide

# MySQL Proxy Guide

**Abstract**

This is the MySQL Proxy extract from the MySQL Reference Manual. This document covers MySQL Proxy 0.8.2.

For release notes detailing the changes in each release of MySQL Proxy, see MySQL Proxy Release Notes.

Document generated on: 2013-01-18 (revision: 33983)

# Table of Contents

# Preface and Legal Notices

This is the MySQL Proxy extract from the MySQL Reference Manual. This document covers MySQL Proxy 0.8.2.

## Legal Notices

# Chapter 1. MySQL Proxy

The MySQL Proxy is an application that communicates over the network using the MySQL network protocol and provides communication between one or more MySQL servers and one or more MySQL clients. Because MySQL Proxy uses the MySQL network protocol, it can be used without modification with any MySQL-compatible client that uses the protocol. This includes the `mysql` command-line client, any clients that uses the MySQL client libraries, and any connector that supports the MySQL network protocol.

In the most basic configuration, MySQL Proxy simply interposes itself between the server and clients, passing queries from the clients to the MySQL Server and returning the responses from the MySQL Server to the appropriate client. In more advanced configurations, the MySQL Proxy can also monitor and alter the communication between the client and the server. Query interception enables you to add profiling, and interception of the exchanges is scriptable using the Lua scripting language.

By intercepting the queries from the client, the proxy can insert additional queries into the list of queries sent to the server, and remove the additional results when they are returned by the server. Using this functionality you can return the results from the original query to the client while adding informational statements to each query, for example, to monitor their execution time or progress, and separately log the results.

The proxy enables you to perform additional monitoring, filtering, or manipulation of queries without requiring you to make any modifications to the client and without the client even being aware that it is communicating with anything but a genuine MySQL server.

This documentation covers MySQL Proxy 0.8.2. And MySQL Proxy contains third-party code. For license information on third-party code, see Licenses for Third-Party Components.

> **Warning**
>
> MySQL Proxy is currently an Alpha release and should not be used within production environments.

> **Important**
>
> MySQL Proxy is compatible with MySQL 5.0 or later. Testing has not been performed with Version 4.1. Please provide feedback on your experiences using the MySQL Proxy Forum.

For release notes detailing the changes in each release of MySQL Proxy, see MySQL Proxy Release Notes.

# Chapter 2. MySQL Proxy Supported Platforms

MySQL Proxy is currently available as a precompiled binary for the following platforms:

- Linux (including Red Hat, Fedora, Debian, SuSE) and derivatives
- Mac OS X
- FreeBSD
- IBM AIX
- Sun Solaris
- Microsoft Windows (including Microsoft Windows XP, Microsoft Windows Vista, Microsoft Windows Server 2003, Microsoft Windows Server 2008)

  > **Note**
  >
  > You must have the .NET Framework 1.1 or higher installed.

Other Unix/Linux platforms not listed should be compatible by using the source package and building MySQL Proxy locally.

System requirements for the MySQL Proxy application are the same as the main MySQL server. Currently MySQL Proxy is compatible only with MySQL 5.0.1 and later. MySQL Proxy is provided as a standalone, statically linked binary. You need not have MySQL or Lua installed.

# Chapter 3. Installing MySQL Proxy

## Table of Contents



You have three choices for installing MySQL Proxy:

- Precompiled binaries are available for a number of different platforms. See Section 3.1, “Installing MySQL Proxy from a Binary Distribution”.
- You can install from the source code to build on an environment not supported by the binary distributions. See Section 3.2, “Installing MySQL Proxy from a Source Distribution”.
- The latest version of the MySQL Proxy source code is available through a development repository is the best way to stay up to date with the latest fixes and revisions. See Section 3.3, “Installing MySQL Proxy from the Bazaar Repository”.

## 3.1. Installing MySQL Proxy from a Binary Distribution

If you download a binary package, you must extract and copy the package contents to your desired installation directory. The package contains files required by MySQL Proxy, including additional Lua scripts and other components required for execution.

To install, unpack the archive into the desired directory, then modify your `PATH` environment variable so that you can use the `mysql-proxy` command directly:

```
shell> cd /usr/local
shell> tar zxf mysql-proxy-0.8.2-platform.tar.gz
shell> PATH=$PATH:/usr/local/mysql-proxy-0.8.2-platform/sbin
```

To update the path globally on a system, you might need administrator privileges to modify the appropriate `/etc/profile`, `/etc/bashrc`, or other system configuration file.

On Windows, you can update the `PATH` environment variable using this procedure:

1. On the Windows desktop, right-click the My Computer icon, and select Properties.
2. Next select the Advanced tab from the System Properties menu that appears, and click the Environment Variables button.
3. Under **System Variables**, select Path, then click the Edit button. The Edit System Variable dialogue should appear.

The Microsoft Visual C++ runtime libraries are a requirement for running MySQL Proxy as of version 0.8.2. Users that do not have these libraries must download and install the Microsoft Visual C++ 2008 Service Pack 1 Redistributable Package MFC Security Update. Use the following link to obtain the package:

```
http://www.microsoft.com/download/en/details.aspx?id=26368
```

## 3.2. Installing MySQL Proxy from a Source Distribution

You can download a source package and compile the MySQL Proxy yourself. To build from source, you must have the following prerequisite components installed:

- `libevent` 1.x or higher (1.3b or later is preferred).
- `lua` 5.1.x or higher.
- `glib2` 2.6.0 or higher.
- `pkg-config`.
- `libtool` 1.5 or higher.
- MySQL 5.0.x or higher developer files.

> **Note**
>
> On some operating systems, you might need to manually build the required components to get the latest version. If you have trouble compiling MySQL Proxy, consider using a binary distributions instead.

After verifying that the prerequisite components are installed, configure and build MySQL Proxy:

```
shell> tar zxf mysql-proxy-0.8.2.tar.gz
shell> cd mysql-proxy-0.8.2
shell> ./configure
shell> make
```

To test the build, use the `check` target to `make`:

```
shell> make check
```

The tests try to connect to `localhost` using the `root` user. To provide a password, set the `MYSQL_PASSWORD` environment variable:

```
shell> MYSQL_PASSWORD=root_pwd make check
```

You can install using the `install` target:

```
shell> make install
```

By default, `mysql-proxy` is installed into `/usr/local/sbin/mysql-proxy`. The Lua example scripts are installed into `/usr/local/share`.

## 3.3. Installing MySQL Proxy from the Bazaar Repository

The MySQL Proxy source is available through a public Bazaar repository and is the quickest way to get the latest releases and fixes.

A build from the Bazaar repository requires that the following prerequisite components be installed:

- Bazaar 1.10.0 or later.
- `libtool` 1.5 or higher.
- `autoconf` 2.56 or higher.
- `automake` 1.10 or higher.
- `libevent` 1.x or higher (1.3b or later is preferred).
- `lua` 5.1.x or higher.

- `glib2` 2.4.0 or higher.
- `pkg-config`.
- MySQL 5.0.x or higher developer files.

The `mysql-proxy` source is hosted on Launchpad. To check out a local copy of the Bazaar repository, use `bzr`:

```
shell> bzr branch lp:mysql-proxy
```

The preceding command downloads a complete version of the Bazaar repository for `mysql-proxy`. The main source files are located within the `trunk` subdirectory. The configuration scripts must be generated before you can configure and build `mysql-proxy`. The `autogen.sh` script generates the required configuration scripts for you:

```
shell> sh ./autogen.sh
```

The `autogen.sh` script creates the standard `configure` script, which you then use to configure and build with `make`:

```
shell> ./configure
shell> make
shell> make install
```

To create a standalone source distribution, identical to the source distribution available for download, use this command:

```
shell> make distcheck
```

The preceding command creates the file `mysql-proxy-0.8.2.tar.gz` (with the corresponding current version) within the current directory.

## 3.4. Setting Up MySQL Proxy as a Windows Service

The MySQL distribution on Windows includes the `mysql-proxy-svc.exe` command that enables a MySQL Proxy instance to be managed by the Windows service control manager. You can control the service, including automatically starting and stopping it during boot, reboot and shutdown, without separately running the MySQL Proxy application.

To set up a MySQL Proxy service, use the `sc` command to create a new service using the MySQL Proxy service command. Specify the MySQL Proxy options on the `sc` command line, and identify the service with a unique name. For example, to configure a new MySQL Proxy instance that will automatically start when your system boots, redirecting queries to the local MySQL server:

```
C:\> sc create "Proxy" DisplayName= "MySQL Proxy" start= "auto" »
  binPath= "C:\Program Files\MySQL\mysql-proxy-0.8.2\bin\mysql-proxy-svc.exe »
  --proxy-backend-addresses=127.0.0.1:3306"
```

> **Note**
>
> The space following the equal sign after each property is required; failure to include it results in an error.

The preceding command creates a new service called `Proxy`. You can start and stop the service using the `net start|stop` command with the service name. The service is not automatically started after it is created. To start the service:

```
C:\> net start proxy
```

```
The MySQL Proxy service is starting.
The MySQL Proxy service was started successfully.
```

You can specify additional command-line options to the `sc` command. You can also set up multiple MySQL Proxy services on the same machine (providing they are configured to listen on different ports and/or IP addresses.

You can delete a service that you have created:

```
C:\> sc delete proxy
```

For more information on creating services using `sc`, see How to create a Windows service by using Sc.exe.

# Chapter 4. MySQL Proxy Command Options

To start MySQL Proxy, you can run it directly from the command line:

```
shell> mysql-proxy
```

For most situations, you specify at least the host name or address and the port number of the backend MySQL server to which the MySQL Proxy should pass queries.

You can specify options to `mysql-proxy` either on the command line, or by using a configuration file and the `--defaults-file` [13] command-line option to specify the file location.

If you use a configuration file, format it as follows:

- Specify the options within a `[mysql-proxy]` configuration group. For example:

  ```
  [mysql-proxy]
  admin-address = host:port
  ```

- Specify all configuration options in the form of a configuration name and the value to set.
- For options that are a simple toggle on the command line (for example, `--proxy-skip-profiling` [17]), use `true` or `false`. For example, the following is invalid:

  ```
  [mysql-proxy]
  proxy-skip-profiling
  ```

  But this is valid:

  ```
  [mysql-proxy]
  proxy-skip-profiling = true
  ```

- Give the configuration file Unix permissions of `0660` (readable and writable by user and group, no access for others).

Failure to adhere to any of these requirements causes `mysql-proxy` to generate an error during startup.

The following tables list the supported configuration file and command-line options.

**Table 4.1. `mysql-proxy` Help Options**

| Format | Option File | Description |
|---|---|---|
| --help [11] | | Show help options |
| --help-admin [11] | | Show admin module options |
| --help-all [11] | | Show all help options |
| --help-proxy [11] | | Show proxy module options |

**Table 4.2. `mysql-proxy` Admin Options**

| Format | Option File | Description |
|---|---|---|
| --admin-address=host:port [11] | admin-address=host:port [11] | The admin module listening host and port |
| --admin-lua-script=file_name [12] | admin-lua-script=file_name [12] | Script to execute by the admin module |
| --admin-password=password [12] | admin-password=password [12] | Authentication password for admin module |

| Format | Option File | Description |
|---|---|---|
| --admin-username=user_name [12] | admin-username=user_name [12] | Authentication user name for admin module |
| --proxy-address=host:port [15] | proxy-address=host:port [15] | The listening proxy server host and port |

**Table 4.3. `mysql-proxy` Proxy Options**

| Format | Option File | Description | Removed |
|---|---|---|---|
| --no-proxy [15] | no-proxy [15] | Do not start the proxy module | |
| --proxy-backend-addresses=host:port [16] | proxy-backend-addresses=host:port [16] | The MySQL server host and port | |
| --proxy-fix-bug-25371 [17] | proxy-fix-bug-25371 [17] | Enable the fix for Bug #25371 for older libmysql versions | 0.8.1 |
| --proxy-lua-script=file_name [17] | proxy-lua-script=file_name [17] | Filename for Lua script for proxy operations | |
| --proxy-pool-no-change-user [17] | proxy-pool-no-change-user [17] | Do not use the protocol CHANGE_USER command to reset the connection when coming from the connection pool | |
| --proxy-read-only-backend-addresses=host:port [16] | proxy-read-only-backend-addresses=host:port [16] | The MySQL server host and port (read only) | |
| --proxy-skip-profiling [17] | proxy-skip-profiling [17] | Disable query profiling | |

**Table 4.4. `mysql-proxy` Applications Options**

| Format | Option File | Description |
|---|---|---|
| --basedir=dir_name [12] | basedir=dir_name [12] | The base directory prefix for paths in the configuration |
| --daemon [13] | daemon [13] | Start in daemon mode |
| --defaults-file=file_name [13] | | The configuration file to use |
| --event-threads=count [13] | event-threads=count [13] | The number of event-handling threads |
| --keepalive [13] | keepalive [13] | Try to restart the proxy if a crash occurs |
| --log-backtrace-on-crash [13] | log-backtrace-on-crash [13] | Try to invoke the debugger and generate a backtrace on crash |
| --log-file=file_name [13] | log-file=file_name [13] | The file where error messages are logged |
| --log-level=level [14] | log-level=level [14] | The logging level |
| --log-use-syslog [14] | log-use-syslog [14] | Log errors to syslog |
| --lua-cpath=dir_name [14] | lua-cpath=dir_name [14] | Set the LUA_CPATH |

| Format | Option File | Description |
|---|---|---|
| --lua-path=dir_name [14] | lua-path=dir_name [14] | Set the LUA_PATH |
| --max-open-files=count [14] | max-open-files=count [14] | The maximum number of open files to support |
| --pid-file=file_name [17] | pid-file=file_name [17] | File in which to store the process ID |
| --plugin-dir=dir_name [15] | plugin-dir=dir_name [15] | Directory containing plugin files |
| --plugins=plugin,... [15] | plugins=plugin,... [15] | List of plugins to load |
| --user=user_name [18] | user=user_name [18] | The user to use when running mysql-proxy |
| --version [18] | | Show version information |

Except as noted in the following details, all of the options can be used within the configuration file by supplying the option and the corresponding value. For example:

```
[mysql-proxy]
log-file = /var/log/mysql-proxy.log
log-level = message
```

- `--help` [11], `-h`

  | Command-Line Format | `--help` |
  |---|---|
  | | `-h` |

  Show available help options.

- `--help-admin` [11]

  | Command-Line Format | `--help-admin` |
  |---|---|

  Show options for the admin module.

- `--help-all` [11]

  | Command-Line Format | `--help-all` |
  |---|---|

  Show all help options.

- `--help-proxy` [11]

  | Command-Line Format | `--help-proxy` |
  |---|---|

  Show options for the proxy module.

- `--admin-address=`*`host`*`:`*`port`* [11]

  | Command-Line Format | `--admin-address=host:port` |
  |---|---|
  | Option-File Format | `admin-address=host:port` |
  | | **Permitted Values** |

| | | |
|---|---|---|
| | **Type** | `string` |
| | **Default** | `:4041` |

The host name (or IP address) and port for the administration port. The default is `localhost:4041`.

- `--admin-lua-script=`*`file_name`* [12]

| | | |
|---|---|---|
| **Command-Line Format** | `--admin-lua-script=file_name` | |
| **Option-File Format** | `admin-lua-script=file_name` | |
| | **Permitted Values** | |
| | **Type** | `file name` |
| | **Default** | |

The script to use for the proxy administration module.

- `--admin-password=`*`password`* [12]

| | | |
|---|---|---|
| **Command-Line Format** | `--admin-password=password` | |
| **Option-File Format** | `admin-password=password` | |
| | **Permitted Values** | |
| | **Type** | `string` |
| | **Default** | |

The password to use to authenticate users wanting to connect to the MySQL Proxy administration module. This module uses the MySQL protocol to request a user name and password for connections.

- `--admin-username=`*`user_name`* [12]

| | | |
|---|---|---|
| **Command-Line Format** | `--admin-username=user_name` | |
| **Option-File Format** | `admin-username=user_name` | |
| | **Permitted Values** | |
| | **Type** | `string` |
| | **Default** | `root` |

The user name to use to authenticate users wanting to connect to the MySQL Proxy administration module. This module uses the MySQL protocol to request a user name and password for connections. The default user name is `root`.

- `--basedir=`*`dir_name`* [12]

| | | |
|---|---|---|
| **Command-Line Format** | `--basedir=dir_name` | |
| **Option-File Format** | `basedir=dir_name` | |
| | **Permitted Values** | |
| | **Type** | `directory name` |

The base directory to use as a prefix for all other file name configuration options. The base name should be an absolute (not relative) directory. If you specify a relative directory, `mysql-proxy` generates an error during startup.

- `--daemon` [13]

| Command-Line Format | `--daemon` |
| --- | --- |
| Option-File Format | `daemon` |

Starts the proxy in daemon mode.

- `--defaults-file=file_name` [13]

| Command-Line Format | `--defaults-file=file_name` |
| --- | --- |

The file to read for configuration options. If not specified, MySQL Proxy takes options only from the command line.

- `--event-threads=count` [13]

| Command-Line Format | `--event-threads=count` | |
| --- | --- | --- |
| Option-File Format | `event-threads=count` | |
| | **Permitted Values** | |
| | **Type** | `numeric` |
| | **Default** | `1` |

The number of event threads to reserve to handle incoming requests.

- `--keepalive` [13]

| Command-Line Format | `--keepalive` |
| --- | --- |
| Option-File Format | `keepalive` |

Create a process surrounding the main `mysql-proxy` process that attempts to restart the main `mysql-proxy` process in the event of a crash or other failure.

> **Note**
>
> The `--keepalive` [13] option is not available on Microsoft Windows. When running as a service, `mysql-proxy` automatically restarts.

- `--log-backtrace-on-crash` [13]

| Command-Line Format | `--log-backtrace-on-crash` |
| --- | --- |
| Option-File Format | `log-backtrace-on-crash` |

Log a backtrace to the error log and try to initialize the debugger in the event of a failure.

- `--log-file=file_name` [13]

| Command-Line Format | `--log-file=file_name` | |
| --- | --- | --- |
| Option-File Format | `log-file=file_name` | |
| | **Permitted Values** | |
| | **Type** | `file name` |

The file to use to record log information. If this option is not given, `mysql-proxy` logs to the standard error output.

- `--log-level=level` [14]

| Command-Line Format | `--log-level=level` | |
|---|---|---|
| Option-File Format | `log-level=level` | |
| | **Permitted Values** | |
| | **Type** | `enumeration` |
| | **Valid Values** | `error`<br>`warning`<br>`info`<br>`message`<br>`debug` |

The log level to use when outputting error messages. Messages with that level (or lower) are output. For example, `message` level also outputs message with `info`, `warning`, and `error` levels.

- `--log-use-syslog` [14]

| Command-Line Format | `--log-use-syslog` |
|---|---|
| Option-File Format | `log-use-syslog` |

Log errors to the syslog (Unix/Linux only).

- `--lua-cpath=dir_name` [14]

| Command-Line Format | `--lua-cpath=dir_name` | |
|---|---|---|
| Option-File Format | `lua-cpath=dir_name` | |
| | **Permitted Values** | |
| | **Type** | `directory name` |

The `LUA_CPATH` to use when loading compiled modules or libraries for Lua scripts.

- `--lua-path=dir_name` [14]

| Command-Line Format | `--lua-path=dir_name` | |
|---|---|---|
| Option-File Format | `lua-path=dir_name` | |
| | **Permitted Values** | |
| | **Type** | `directory name` |

The `LUA_CPATH` to use when loading modules for Lua.

- `--max-open-files=count` [14]



| Command-Line Format | `--max-open-files=count` |
|---|---|

| Option-File Format | max-open-files=count | |
|---|---|---|
| | Permitted Values | |
| | Type | numeric |

The maximum number of open files and sockets supported by the `mysql-proxy` process. Certain scripts might require a higher value.

- `--no-proxy` [15]

| Command-Line Format | --no-proxy |
|---|---|
| Option-File Format | no-proxy |

Disable the proxy module.

- `--plugin-dir=dir_name` [15]

| Command-Line Format | --plugin-dir=dir_name | |
|---|---|---|
| Option-File Format | plugin-dir=dir_name | |
| | Permitted Values | |
| | Type | directory name |

The directory to use when loading plugins for `mysql-proxy`.

- `--plugins=plugin` [15]

| Command-Line Format | --plugins=plugin,... | |
|---|---|---|
| Option-File Format | plugins=plugin,... | |
| | Permitted Values | |
| | Type | string |

Loads a plugin.

When using this option on the command line, you can specify the option multiple times to specify multiple plugins. For example:

```
shell> mysql-proxy --plugins=proxy --plugins=admin
```

When using the option within the configuration file, you should separate multiple plugins by commas. The equivalent of the preceding example would be:

```
...
plugins=proxy,admin
```

- `--proxy-address=host:port` [15], `-P host:port`

| Command-Line Format | --proxy-address=host:port | |
|---|---|---|
| | -P host:port | |
| Option-File Format | proxy-address=host:port | |
| | Permitted Values | |
| | Type | string |

| | | |
|---|---|---|
| | **Default** | `:4040` |

The listening host name (or IP address) and port of the proxy server. The default is `:4040` (all IPs on port 4040).

- `--proxy-read-only-backend-addresses=`*`host`*`:`*`port`* [16], `-r` *`host`*`:`*`port`*

| | | |
|---|---|---|
| **Command-Line Format** | `--proxy-read-only-backend-addresses=host:port` | |
| | `-r host:port` | |
| **Option-File Format** | `proxy-read-only-backend-addresses=host:port` | |
| | **Permitted Values** | |
| | **Type** | `string` |

The listening host name (or IP address) and port of the proxy server for read-only connections. The default is for this information not to be set.

> **Note**
>
> Setting this value only configures the servers within the corresponding internal structure (see `proxy.global.backends` [24]). You can determine the backend type by checking the `type` field for each connection.
>
> You should therefore only use this option in combination with a script designed to make use of the different backend types.

When using this option on the command line, you can specify the option and the server multiple times to specify multiple backends. For example:

```
shell> mysql-proxy --proxy-read-only-backend-addresses=192.168.0.1:3306 --proxy-read-only-backend-addresses=
```

When using the option within the configuration file, you should separate multiple servers by commas. The equivalent of the preceding example would be:

```
...
proxy-read-only-backend-addresses = 192.168.0.1:3306,192.168.0.2:3306
```

- `--proxy-backend-addresses=`*`host`*`:`*`port`* [16], `-b` *`host`*`:`*`port`*

| | | |
|---|---|---|
| **Command-Line Format** | `--proxy-backend-addresses=host:port` | |
| | `-b host:port` | |
| **Option-File Format** | `proxy-backend-addresses=host:port` | |
| | **Permitted Values** | |
| | **Type** | `string` |
| | **Default** | `127.0.0.1:3306` |

The host name (or IP address) and port of the MySQL server to connect to. You can specify multiple backend servers by supplying multiple options. Clients are connected to each backend server in round-robin fashion. For example, if you specify two servers A and B, the first client connection will go to server A; the second client connection to server B and the third client connection to server A.

When using this option on the command line, you can specify the option and the server multiple times to specify multiple backends. For example:

```
shell> mysql-proxy --proxy-backend-addresses 192.168.0.1:3306 --proxy-backend-addresses 192.168.0.2:3306
```

When using the option within the configuration file, you should separate multiple servers by commas. The equivalent of the preceding example would be:

```
...
proxy-backend-addresses = 192.168.0.1:3306,192.168.0.2:3306
```

- `--proxy-pool-no-change-user` [17]

| | |
|---|---|
| **Command-Line Format** | `--proxy-pool-no-change-user` |
| **Option-File Format** | `proxy-pool-no-change-user` |

Disable use of the MySQL protocol `CHANGE_USER` command when reusing a connection from the pool of connections specified by the `proxy-backend-addresses` list.

- `--proxy-skip-profiling` [17]

| | |
|---|---|
| **Command-Line Format** | `--proxy-skip-profiling` |
| **Option-File Format** | `proxy-skip-profiling` |

Disable query profiling (statistics time tracking). The default is for tracking to be enabled.

- `--proxy-fix-bug-25371` [17]

| | |
|---|---|
| **Version Removed** | 0.8.1 |
| **Command-Line Format** | `--proxy-fix-bug-25371` |
| **Option-File Format** | `proxy-fix-bug-25371` |

Enable a workaround for an issue when connecting to a MySQL server later than 5.1.12 when using a MySQL client library of any earlier version.

This option was removed in `mysql-proxy` 0.8.1. Now, `mysql-proxy` returns an error message at the protocol level if it sees a `COM_CHANGE_USER` being sent to a server that has a version from 5.1.14 to 5.1.17.

- `--proxy-lua-script=`*`file_name`* [17], `-s` *`file_name`*

| | | |
|---|---|---|
| **Command-Line Format** | `--proxy-lua-script=file_name` | |
| | `-s file_name` | |
| **Option-File Format** | `proxy-lua-script=file_name` | |
| | **Permitted Values** | |
| | **Type** | `file name` |

The Lua script file to be loaded. Note that the script file is not physically loaded and parsed until a connection is made. Also note that the specified Lua script is reloaded for each connection; if the content of the Lua script changes while `mysql-proxy` is running, the updated content is automatically used when a new connection is made.

- `--pid-file=`*`file_name`* [17]

| | |
|---|---|
| **Command-Line Format** | `--pid-file=file_name` |

| Option-File Format | pid-file=file_name | |
|---|---|---|
| | Permitted Values | |
| | Type | file name |

The name of the file in which to store the process ID.

- `--user=user_name` [18]

| Command-Line Format | --user=user_name | |
|---|---|---|
| Option-File Format | user=user_name | |
| | Permitted Values | |
| | Type | string |

Run `mysql-proxy` as the specified `user`.

- `--version` [18], `-V`

| Command-Line Format | --version |
|---|---|
| | -V |

Show the version number.

The most common usage is as a simple proxy service (that is, without additional scripting). For basic proxy operation, you must specify at least one `proxy-backend-addresses` option to specify the MySQL server to connect to by default:

```
shell> mysql-proxy --proxy-backend-addresses=MySQL.example.com:3306
```

The default proxy port is `4040`, so you can connect to your MySQL server through the proxy by specifying the host name and port details:

```
shell> mysql --host=localhost --port=4040
```

If your server requires authentication information, this will be passed through natively without alteration by `mysql-proxy`, so you must also specify the required authentication information:

```
shell> mysql --host=localhost --port=4040 \
   --user=user_name --password=password
```

You can also connect to a read-only port (which filters out `UPDATE` and `INSERT` queries) by connecting to the read-only port. By default the host name is the default, and the port is `4042`, but you can alter the host/port information by using the `--proxy-read-only-backend-addresses` [16] command-line option.

For more detailed information on how to use these command-line options, and `mysql-proxy` in general in combination with Lua scripts, see Chapter 6, *Using MySQL Proxy*.

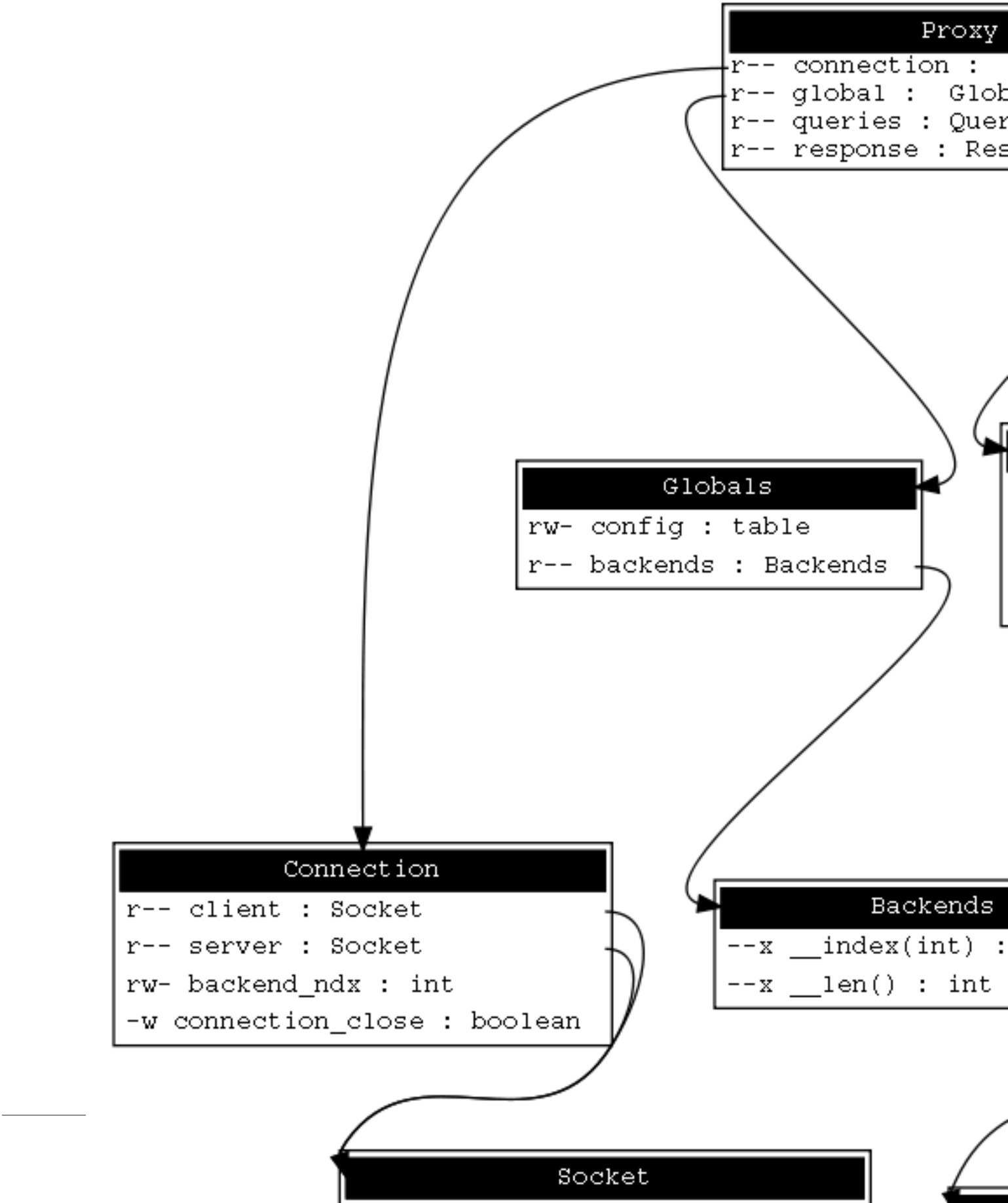
# Chapter 5. MySQL Proxy Scripting

## Table of Contents



You can control how MySQL Proxy manipulates and works with the queries and results that are passed on to the MySQL server through the use of the embedded Lua scripting language. You can find out more about the Lua programming language from the Lua Web site.

The following diagram shows an overview of the classes exposed by MySQL Proxy.


Proxy
r-- connection :
r-- global : Glob
r-- queries : Quer
r-- response : Res
Globals
rw- config : table
r-- backends : Backends
Connection
r-- client : Socket
r-- server : Socket
rw- backend_ndx : int
-w connection_close : boolean
Backends
--x __index(int) :
--x __len() : int
Socket

The primary interaction between MySQL Proxy and the server is provided by defining one or more functions through an Lua script. A number of functions are supported, according to different events and operations in the communication sequence between a client and one or more backend MySQL servers:

- `connect_server()`: This function is called each time a connection is made to MySQL Proxy from a client. You can use this function during load-balancing to intercept the original connection and decide which server the client should ultimately be attached to. If you do not define a special solution, a simple round-robin style distribution is used by default.
- `read_handshake()`: This function is called when the initial handshake information is returned by the server. You can capture the handshake information returned and provide additional checks before the authorization exchange takes place.
- `read_auth()`: This function is called when the authorization packet (user name, password, default database) are submitted by the client to the server for authentication.
- `read_auth_result()`: This function is called when the server returns an authorization packet to the client indicating whether the authorization succeeded.
- `read_query()`: This function is called each time a query is sent by the client to the server. You can use this to edit and manipulate the original query, including adding new queries before and after the original statement. You can also use this function to return information directly to the client, bypassing the server, which can be useful to filter unwanted queries or queries that exceed known limits.
- `read_query_result()`: This function is called each time a result is returned from the server, providing you have manually injected queries into the query queue. If you have not explicitly injected queries within the `read_query()` function, this function is not triggered. You can use this to edit the result set, or to remove or filter the result sets generated from additional queries you injected into the queue when using `read_query()`.

The following table describes the direction of information flow at the point when the function is triggered.

| Function | Supplied Information | Direction |
|---|---|---|
| `connect_server()` | None | Client to Server |
| `read_handshake()` | None | Server to Client |
| `read_auth()` | None | Client to Server |
| `read_auth_result()` | None | Server to Client |
| `read_query()` | Query | Client to Server |
| `read_query_result()` | Query result | Server to Client |

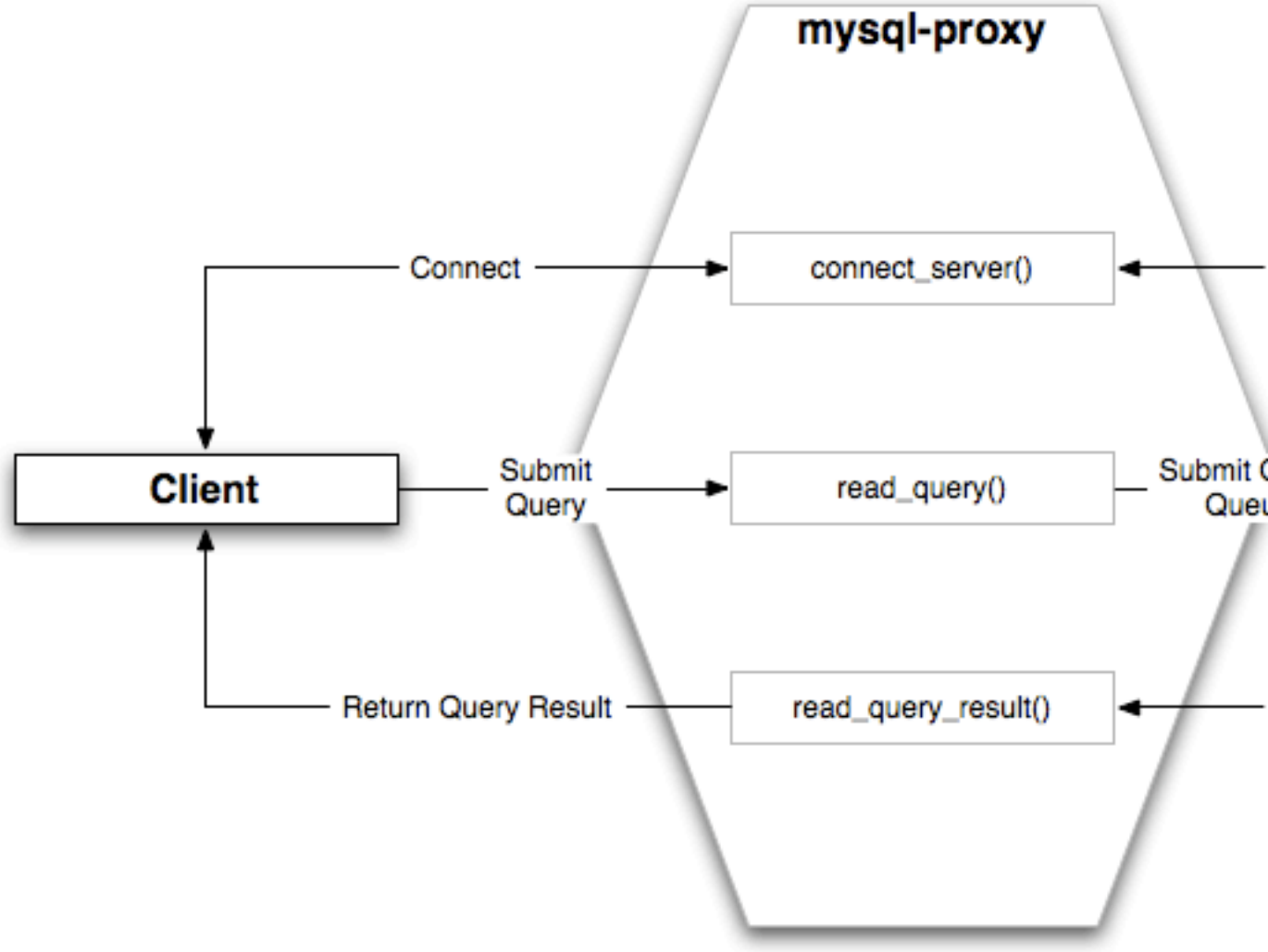
By default, all functions return a result that indicates whether the data should be passed on to the client or server (depending on the direction of the information being transferred). This return value can be overridden by explicitly returning a constant indicating that a particular response should be sent. For example, it is possible to construct result set information by hand within `read_query()` and to return the result set directly to the client without ever sending the original query to the server.

In addition to these functions, a number of built-in structures provide control over how MySQL Proxy forwards queries and returns the results by providing a simplified interface to elements such as the list of queries and the groups of result sets that are returned.

# 5.1. Proxy Scripting Sequence During Query Injection

The following figure gives an example of how the proxy might be used when injecting queries into the query queue. Because the proxy sits between the client and MySQL server, what the proxy sends to the

server, and the information that the proxy ultimately returns to the client, need not match or correlate. Once the client has connected to the proxy, the sequence shown in the following diagram occurs for each individual query sent by the client.




1. When the client submits one query to the proxy, the `read_query()` function within the proxy is triggered. The function adds the query to the query queue.
2. Once manipulation by `read_query()` has completed, the queries are submitted, sequentially, to the MySQL server.
3. The MySQL server returns the results from each query, one result set for each query submitted. The `read_query_result()` function is triggered for each result set, and each invocation can decide which result set to return to the client

For example, you can queue additional queries into the global query queue to be processed by the server. This can be used to add statistical information by adding queries before and after the original query, changing the original query:

```
SELECT * FROM City;
```

Into a sequence of queries:

```
SELECT NOW();
SELECT * FROM City;
SELECT NOW();
```

You can also modify the original statement; for example, to add `EXPLAIN` to each statement executed to get information on how the statement was processed, again altering our original SQL statement into a number of statements:

```
SELECT * FROM City;
EXPLAIN SELECT * FROM City;
```

In both of these examples, the client would have received more result sets than expected. Regardless of how you manipulate the incoming query and the returned result, the number of queries returned by the proxy must match the number of original queries sent by the client.

You could adjust the client to handle the multiple result sets sent by the proxy, but in most cases you will want the existence of the proxy to remain transparent. To ensure that the number of queries and result sets match, you can use the MySQL Proxy `read_query_result()` to extract the additional result set information and return only the result set the client originally requested back to the client. You can achieve this by giving each query that you add to the query queue a unique ID, then filter out queries that do not match the original query ID when processing them with `read_query_result()`.

## 5.2. Internal Structures

There are a number of internal structures within the scripting element of MySQL Proxy. The primary structure is `proxy` and this provides an interface to the many common structures used throughout the script, such as connection lists and configured backend servers. Other structures, such as the incoming packet from the client and result sets are only available within the context of one of the scriptable functions.

| Attribute | Description |
|---|---|
| `connection` | A structure containing the active client connections. For a list of attributes, see `proxy.connection` [23]. |
| `servers` | A structure containing the list of configured backend servers. For a list of attributes, see `proxy.global.backends` [24]. |
| `queries` | A structure containing the queue of queries that will be sent to the server during a single client query. For a list of attributes, see `proxy.queries` [24]. |
| `PROXY_VERSION` | The version number of MySQL Proxy, encoded in hex. You can use this to check that the version number supports a particular option from within the Lua script. Note that the value is encoded as a hex value, so to check the version is at least 0.5.1 you compare against `0x00501`. |

### `proxy.connection`

The `proxy.connection` object is read only, and provides information about the current connection, and is split into a `client` and `server` tables. This enables you to examine information about both the incoming client connections to the proxy (`client`), and to the backend servers (`server`).

| Attribute | Description |
|---|---|
| `client.default_db` | Default database requested by the client |
| `client.username` | User name used to authenticate |
| `client.scrambled_password` | The scrambled version of the password used to authenticate |

| Attribute | Description |
|---|---|
| client.dst.name | The combined address:port of the Proxy port used by this client (should match the --proxy-address [15] configuration parameter) |
| client.dst.address | The IP address of the of the Proxy port used by this client |
| client.dst.port | The port number of the of the Proxy port used by this client |
| client.src.name | The combined address:port of the client (originating) TCP/IP endpoint |
| client.src.address | The IP address of the client (originating) TCP/IP port |
| client.src.port | The port of the client (originating) TCP/IP endpoint |
| server.scramble_buffer | The scramble buffer used to scramble the password |
| server.mysqld_version | The MySQL version number of the server |
| server.thread_id | The ID of the thread handling the connection to the current server |
| server.dst.name | The combined address:port for the backend server for the current connection (i.e. the connection to the MySQL server) |
| server.dst.address | The address for the backend server |
| server.dst.port | The port for the backend server |
| server.src.name | The combined address:port for the TCP/IP endpoint used by the Proxy to connect to the backend server |
| server.src.address | The address of the endpoint for the proxy-side connection to the MySQL server |
| server.src.port | The port of the endpoint for the proxy-side connection to the MySQL server |

### proxy.global.backends

The proxy.global.backends table is partially writable and contains an array of all the configured backend servers and the server metadata (IP address, status, etc.). You can determine the array index of the current connection using proxy.connection["backend_ndx"] which is the index into this table of the backend server being used by the active connection.

The attributes for each entry within the proxy.global.backends table are shown in this table.

| Attribute | Description |
|---|---|
| dst.name | The combined address:port of the backend server. |
| dst.address | The IP address of the backend server. |
| dst.port | The port of the backend server. |
| connected_clients | The number of clients currently connected. |
| state | The status of the backend server. See Backend State/Type Constants [27]. |
| type | The type of the backend server. You can use this to identify whether the backed was configured as a standard read/write backend, or a read-only backend. You can compare this value to the proxy.BACKEND_TYPE_RW and proxy.BACKEND_TYPE_RO. |

### proxy.queries

The proxy.queries object is a queue representing the list of queries to be sent to the server. The queue is not populated automatically, but if you do not explicitly populate the queue, queries are

passed on to the backend server verbatim. Also, if you do not populate the query queue by hand, the `read_query_result()` function is not triggered.

The following methods are supported for populating the `proxy.queries` object.

| Function | Description |
|---|---|
| `append(id,packet, [options])` | Appends a query to the end of the query queue. The `id` is an integer identifier that you can use to recognize the query results when they are returned by the server. The packet should be a properly formatted query packet. The optional `options` should be a table containing the options specific to this packet. |
| `prepend(id,packet)` | Prepends a query to the query queue. The `id` is an identifier that you can use to recognize the query results when they are returned by the server. The packet should be a properly formatted query packet. |
| `reset()` | Empties the query queue. |
| `len()` | Returns the number of query packets in the queue. |

For example, you could append a query packet to the `proxy.queries` queue by using the `append()`:

```
proxy.queries:append(1,packet)
```

The optional third argument to `append()` should contain the options for the packet. To have access to the result set through the `read_query_result()` function, set the `resultset_is_needed` flag to `true`:

```
proxy.queries:append( 1, packet, { resultset_is_needed = true } )
```

If that flag is `false` (the default), proxy will:

- Send the result set to the client as soon as it is received
- Reduce memory usage (because the result set is not stored internally for processing)
- Reduce latency of returning results to the client
- Pass data from server to client unaltered

The default mode is therefore quicker and useful if you only want to monitor the queries sent, and the basic statistics.

To perform any kind of manipulation on the returned data, you must set the flag to `true`, which will:

- Store the result set so that it can be processed.
- Enable modification of the result set before it is returned to the client.
- Enable you to discard the result set instead of returning it to the client.

### `proxy.response`

The `proxy.response` structure is used when you want to return your own MySQL response, instead of forwarding a packet that you have received a backend server. The structure holds the response type information, an optional error message, and the result set (rows/columns) to return.

| Attribute | Description |
|---|---|
| `type` | The type of the response. The type must be either `MYSQLD_PACKET_OK` or `MYSQLD_PACKET_ERR`. If the `MYSQLD_PACKET_ERR`, you should |

| Attribute | Description |
|---|---|
| | set the value of the `mysql.response.errmsg` with a suitable error message. |
| `errmsg` | A string containing the error message that will be returned to the client. |
| `resultset` | A structure containing the result set information (columns and rows), identical to what would be returned when returning a results from a `SELECT` query. |

When using `proxy.response` you either set `proxy.response.type` to `proxy.MYSQLD_PACKET_OK` and then build `resultset` to contain the results to return, or set `proxy.response.type` to `proxy.MYSQLD_PACKET_ERR` and set the `proxy.response.errmsg` to a string with the error message. To send the completed result set or error message, you should return the `proxy.PROXY_SEND_RESULT` to trigger the return of the packet information.

An example of this can be seen in the `tutorial-resultset.lua` script within the MySQL Proxy package:

```
if string.lower(command) == "show" and string.lower(option) == "querycounter" then
        ---
        -- proxy.PROXY_SEND_RESULT requires
        --
        -- proxy.response.type to be either
        -- * proxy.MYSQLD_PACKET_OK or
        -- * proxy.MYSQLD_PACKET_ERR
        --
        -- for proxy.MYSQLD_PACKET_OK you need a resultset
        -- * fields
        -- * rows
        --
        -- for proxy.MYSQLD_PACKET_ERR
        -- * errmsg
        proxy.response.type = proxy.MYSQLD_PACKET_OK
        proxy.response.resultset = {
                fields = {
                        { type = proxy.MYSQL_TYPE_LONG, name = "global_query_counter", },
                        { type = proxy.MYSQL_TYPE_LONG, name = "query_counter", },
                },
                rows = {
                        { proxy.global.query_counter, query_counter }
                }
        }
        -- we have our result, send it back
        return proxy.PROXY_SEND_RESULT
elseif string.lower(command) == "show" and string.lower(option) == "myerror" then
        proxy.response.type = proxy.MYSQLD_PACKET_ERR
        proxy.response.errmsg = "my first error"
        return proxy.PROXY_SEND_RESULT
```

### `proxy.response.resultset`

The `proxy.response.resultset` structure should be populated with the rows and columns of data to return. The structure contains the information about the entire result set, with the individual elements of the data shown in the following table.

| Attribute | Description |
|---|---|
| `fields` | The definition of the columns being returned. This should be a dictionary structure with the `type` specifying the MySQL data type, and the `name` specifying the column name. Columns should be listed in the order of the column data that will be returned. |

| Attribute | Description |
|---|---|
| flags | A number of flags related to the result set. Valid flags include auto_commit (whether an automatic commit was triggered), no_good_index_used (the query executed without using an appropriate index), and no_index_used (the query executed without using any index). |
| rows | The actual row data. The information should be returned as an array of arrays. Each inner array should contain the column data, with the outer array making up the entire result set. |
| warning_count | The number of warnings for this result set. |
| affected_rows | The number of rows affected by the original statement. |
| insert_id | The last insert ID for an auto-incremented column in a table. |
| query_status | The status of the query operation. You can use the MYSQLD_PACKET_OK or MYSQLD_PACKET_ERR constants to populate this parameter. |

For an example showing how to use this structure, see proxy.response [25].

### Proxy Return State Constants

The following constants are used internally by the proxy to specify the response to send to the client or server. All constants are exposed as values within the main proxy table.

| Constant | Description |
|---|---|
| PROXY_SEND_QUERY | Causes the proxy to send the current contents of the queries queue to the server. |
| PROXY_SEND_RESULT | Causes the proxy to send a result set back to the client. |
| PROXY_IGNORE_RESULT | Causes the proxy to drop the result set (nothing is returned to the client). |

As constants, these entities are available without qualification in the Lua scripts. For example, at the end of the read_query_result() you might return PROXY_IGNORE_RESULT:

```
return proxy.PROXY_IGNORE_RESULT
```

### Packet State Constants

The following states describe the status of a network packet. These items are entries within the main proxy table.

| Constant | Description |
|---|---|
| MYSQLD_PACKET_OK | The packet is OK |
| MYSQLD_PACKET_ERR | The packet contains error information |
| MYSQLD_PACKET_RAW | The packet contains raw data |

### Backend State/Type Constants

The following constants are used either to define the status or type of the backend MySQL server to which the proxy is connected. These items are entries within the main proxy table.

| Constant | Description |
|---|---|
| BACKEND_STATE_UNKNOWN | The current status is unknown |

| Constant | Description |
|---|---|
| BACKEND_STATE_UP | The backend is known to be up (available) |
| BACKEND_STATE_DOWN | The backend is known to be down (unavailable) |
| BACKEND_TYPE_UNKNOWN | Backend type is unknown |
| BACKEND_TYPE_RW | Backend is available for read/write |
| BACKEND_TYPE_RO | Backend is available only for read-only use |

**Server Command Constants**

The following values are used in the packets exchanged between the client and server to identify the information in the rest of the packet. These items are entries within the main `proxy` table. The packet type is defined as the first character in the sent packet. For example, when intercepting packets from the client to edit or monitor a query, you would check that the first byte of the packet was of type `proxy.COM_QUERY`.

| Constant | Description |
|---|---|
| COM_SLEEP | Sleep |
| COM_QUIT | Quit |
| COM_INIT_DB | Initialize database |
| COM_QUERY | Query |
| COM_FIELD_LIST | Field List |
| COM_CREATE_DB | Create database |
| COM_DROP_DB | Drop database |
| COM_REFRESH | Refresh |
| COM_SHUTDOWN | Shutdown |
| COM_STATISTICS | Statistics |
| COM_PROCESS_INFO | Process List |
| COM_CONNECT | Connect |
| COM_PROCESS_KILL | Kill |
| COM_DEBUG | Debug |
| COM_PING | Ping |
| COM_TIME | Time |
| COM_DELAYED_INSERT | Delayed insert |
| COM_CHANGE_USER | Change user |
| COM_BINLOG_DUMP | Binlog dump |
| COM_TABLE_DUMP | Table dump |
| COM_CONNECT_OUT | Connect out |
| COM_REGISTER_SLAVE | Register slave |
| COM_STMT_PREPARE | Prepare server-side statement |
| COM_STMT_EXECUTE | Execute server-side statement |
| COM_STMT_SEND_LONG_DATA | Long data |
| COM_STMT_CLOSE | Close server-side statement |

| Constant | Description |
|---|---|
| COM_STMT_RESET | Reset statement |
| COM_SET_OPTION | Set option |
| COM_STMT_FETCH | Fetch statement |
| COM_DAEMON | Daemon (MySQL 5.1 only) |
| COM_ERROR | Error |

### MySQL Type Constants

These constants are used to identify the field types in the query result data returned to clients from the result of a query. These items are entries within the main `proxy` table.

| Constant | Field Type |
|---|---|
| MYSQL_TYPE_DECIMAL | Decimal |
| MYSQL_TYPE_NEWDECIMAL | Decimal (MySQL 5.0 or later) |
| MYSQL_TYPE_TINY | Tiny |
| MYSQL_TYPE_SHORT | Short |
| MYSQL_TYPE_LONG | Long |
| MYSQL_TYPE_FLOAT | Float |
| MYSQL_TYPE_DOUBLE | Double |
| MYSQL_TYPE_NULL | Null |
| MYSQL_TYPE_TIMESTAMP | Timestamp |
| MYSQL_TYPE_LONGLONG | Long long |
| MYSQL_TYPE_INT24 | Integer |
| MYSQL_TYPE_DATE | Date |
| MYSQL_TYPE_TIME | Time |
| MYSQL_TYPE_DATETIME | Datetime |
| MYSQL_TYPE_YEAR | Year |
| MYSQL_TYPE_NEWDATE | Date (MySQL 5.0 or later) |
| MYSQL_TYPE_ENUM | Enumeration |
| MYSQL_TYPE_SET | Set |
| MYSQL_TYPE_TINY_BLOB | Tiny Blob |
| MYSQL_TYPE_MEDIUM_BLOB | Medium Blob |
| MYSQL_TYPE_LONG_BLOB | Long Blob |
| MYSQL_TYPE_BLOB | Blob |
| MYSQL_TYPE_VAR_STRING | Varstring |
| MYSQL_TYPE_STRING | String |
| MYSQL_TYPE_TINY | Tiny (compatible with MYSQL_TYPE_CHAR) |
| MYSQL_TYPE_ENUM | Enumeration (compatible with MYSQL_TYPE_INTERVAL) |
| MYSQL_TYPE_GEOMETRY | Geometry |
| MYSQL_TYPE_BIT | Bit |

## 5.3. Capturing a Connection with connect_server()

When the proxy accepts a connection from a MySQL client, the connect_server() function is called.

There are no arguments to the function, but you can use and if necessary manipulate the information in the proxy.connection table, which is unique to each client session.

For example, if you have multiple backend servers, you can specify which server that connection should use by setting the value of proxy.connection.backend_ndx to a valid server number. The following code chooses between two servers based on whether the current time in minutes is odd or even:

```
function connect_server()
        print("--> a client really wants to talk to a server")
        if (tonumber(os.date("%M")) % 2 == 0) then
                proxy.connection.backend_ndx = 2
                print("Choosing backend 2")
        else
                proxy.connection.backend_ndx = 1
                print("Choosing backend 1")
        end
        print("Using " .. proxy.global.backends[proxy.connection.backend_ndx].dst.name)
end
```

This example also displays the IP address/port combination by accessing the information from the internal proxy.global.backends table.

## 5.4. Examining the Handshake with read_handshake()

Handshake information is sent by the server to the client after the initial connection (through connect_server()) has been made. The handshake information contains details about the MySQL version, the ID of the thread that will handle the connection information, and the IP address of the client and server. This information is exposed through the proxy.connection structure.

- proxy.connection.server.mysqld_version: The version of the MySQL server.
- proxy.connection.server.thread_id: The thread ID.
- proxy.connection.server.scramble_buffer: The password scramble buffer.
- proxy.connection.server.dst.name: The IP address of the server.
- proxy.connection.client.src.name: The IP address of the client.

For example, you can print out the handshake data and refuse clients by IP address with the following function:

```
function read_handshake()
        print("<-- let's send him some information about us")
        print("    mysqld-version: " .. proxy.connection.server.mysqld_version)
        print("    thread-id     : " .. proxy.connection.server.thread_id)
        print("    scramble-buf  : " .. string.format("%q",proxy.connection.server.scramble_buffer))
        print("    server-addr   : " .. proxy.connection.server.dst.name)
        print("    client-addr   : " .. proxy.connection.client.dst.name)
        if not proxy.connection.client.src.name:match("^127.0.0.1:") then
                proxy.response.type = proxy.MYSQLD_PACKET_ERR
                proxy.response.errmsg = "only local connects are allowed"
                print("we don't like this client");
                return proxy.PROXY_SEND_RESULT
        end
end
```

Note that you must return an error packet to the client by using proxy.PROXY_SEND_RESULT.

## 5.5. Examining the Authentication Credentials with read_auth()

The read_auth() function is triggered when an authentication handshake is initiated by the client. In the execution sequence, read_auth() occurs immediately after read_handshake(), so the server selection has already been made, but the connection and authorization information has not yet been provided to the backend server.

You can obtain the authentication information by examining the proxy.connection.client structure. For more information, see proxy.connection [23].

For example, you can print the user name and password supplied during authorization using:

```
function read_auth()
        print("    username      : " .. proxy.connection.client.username)
        print("    password      : " .. string.format("%q", proxy.connection.client.scrambled_password))
end
```

You can interrupt the authentication process within this function and return an error packet back to the client by constructing a new packet and returning proxy.PROXY_SEND_RESULT:

```
proxy.response.type = proxy.MYSQLD_PACKET_ERR
proxy.response.errmsg = "Logins are not allowed"
return proxy.PROXY_SEND_RESULT
```

## 5.6. Accessing Authentication Information with read_auth_result()

The return packet from the server during authentication is captured by read_auth_result(). The only argument to this function is the authentication packet returned by the server. As the packet is a raw MySQL network protocol packet, you must access the first byte to identify the packet type and contents. The MYSQLD_PACKET_ERR and MYSQLD_PACKET_OK constants can be used to identify whether the authentication was successful:

```
function read_auth_result(auth)
        local state = auth.packet:byte()
        if state == proxy.MYSQLD_PACKET_OK then
                print("<-- auth ok");
        elseif state == proxy.MYSQLD_PACKET_ERR then
                print("<-- auth failed");
        else
                print("<-- auth ... don't know: " .. string.format("%q", auth.packet));
        end
end
```

If a long-password capable client tries to authenticate to a server that supports long passwords, but the user password provided is actually short, read_auth_result() will be called twice. The first time, auth.packet:byte() will equal 254, indicating that the client should try again using the old password protocol. The second time time read_auth_result()/ is called, auth.packet:byte() will indicate whether the authentication actually succeeded.

## 5.7. Manipulating Queries with read_query()

The read_query() function is called once for each query submitted by the client and accepts a single argument, the query packet that was provided. To access the content of the packet, you must parse the packet contents manually.

For example, you can intercept a query packet and print out the contents using the following function definition:

```
function read_query( packet )
        if packet:byte() == proxy.COM_QUERY then
                print("we got a normal query: " .. packet:sub(2))
        end
end
```

This example checks the first byte of the packet to determine the type. If the type is COM_QUERY (see Server Command Constants [28]), we extract the query from the packet and print it. The structure of the packet type supplied is important. In the case of a COM_QUERY packet, the remaining contents of the packet are the text of the query string. In this example, no changes have been made to the query or the list of queries that will ultimately be sent to the MySQL server.

To modify a query, or add new queries, you must populate the query queue (proxy.queries), then execute the queries that you have placed into the queue. If you do not modify the original query or the queue, the query received from the client is sent to the MySQL server verbatim.

When adding queries to the queue, you should follow these guidelines:

- The packets inserted into the queue must be valid query packets. For each packet, you must set the initial byte to the packet type. If you are appending a query, you can append the query statement to the rest of the packet.
- Once you add a query to the queue, the queue is used as the source for queries sent to the server. If you add a query to the queue to add more information, you must also add the original query to the queue or it will not be executed.
- Once the queue has been populated, you must set the return value from read_query() to indicate whether the query queue should be sent to the server.
- When you add queries to the queue, you should add an ID. The ID you specify is returned with the result set so that you identify each query and corresponding result set. The ID has no other purpose than as an identifier for correlating the query and result set. When operating in a passive mode, during profiling for example, you identify the original query and the corresponding result set so that the results expected by the client can be returned correctly.
- Unless your client is designed to cope with more result sets than queries, you should ensure that the number of queries from the client match the number of results sets returned to the client. Using the unique ID and removing result sets you inserted will help.

Normally, the read_query() and read_query_result() function are used in conjunction with each other to inject additional queries and remove the additional result sets. However, read_query_result() is only called if you populate the query queue within read_query().

## 5.8. Manipulating Results with read_query_result()

The read_query_result() is called for each result set returned by the server only if you have manually injected queries into the query queue. If you have not manipulated the query queue, this function is not called. The function supports a single argument, the result packet, which provides a number of properties:

- id: The ID of the result set, which corresponds to the ID that was set when the query packet was submitted to the server when using append(id) on the query queue. You must have set the resultset_is_needed flag to append to intercept the result set before it is returned to the client. See proxy.queries [24].

- `query`: The text of the original query.
- `query_time`: The number of microseconds required to receive the first row of a result set since the query was sent to the server.
- `response_time`: The number of microseconds required to receive the last row of the result set since the query was sent to the server.
- `resultset`: The content of the result set data.

By accessing the result information from the MySQL server, you can extract the results that match the queries that you injected, return different result sets (for example, from a modified query), and even create your own result sets.

The following Lua script, for example, will output the query, followed by the query time and response time (that is, the time to execute the query and the time to return the data for the query) for each query sent to the server:

```
function read_query( packet )
        if packet:byte() == proxy.COM_QUERY then
                print("we got a normal query: " .. packet:sub(2))
                proxy.queries:append(1, packet )
                return proxy.PROXY_SEND_QUERY
        end
end
function read_query_result(inj)
        print("query-time: " .. (inj.query_time / 1000) .. "ms")
        print("response-time: " .. (inj.response_time / 1000) .. "ms")
end
```

You can access the rows of returned results from the result set by accessing the `rows` property of the `resultset` property of the result that is exposed through `read_query_result()`. For example, you can iterate over the results showing the first column from each row using this Lua fragment:

```
for row in inj.resultset.rows do
        print("injected query returned: " .. row[1])
end
```

Just like `read_query()`, `read_query_result()` can return different values for each result according to the result returned. If you have injected additional queries into the query queue, for example, remove the results returned from those additional queries and return only the results from the query originally submitted by the client.

The following example injects additional `SELECT NOW()` statements into the query queue, giving them a different ID to the ID of the original query. Within `read_query_result()`, if the ID for the injected queries is identified, we display the result row, and return the `proxy.PROXY_IGNORE_RESULT` from the function so that the result is not returned to the client. If the result is from any other query, we print out the query time information for the query and return the default, which passes on the result set unchanged. We could also have explicitly returned `proxy.PROXY_IGNORE_RESULT` to the MySQL client.

```
function read_query( packet )
        if packet:byte() == proxy.COM_QUERY then
                proxy.queries:append(2, string.char(proxy.COM_QUERY) .. "SELECT NOW()", {resultset_is_needed
                proxy.queries:append(1, packet, {resultset_is_needed = true})
                proxy.queries:append(2, string.char(proxy.COM_QUERY) .. "SELECT NOW()", {resultset_is_needed
                return proxy.PROXY_SEND_QUERY
        end
end
function read_query_result(inj)
        if inj.id == 2 then
                for row in inj.resultset.rows do
```

```
                        print("injected query returned: " .. row[1])
                end
                return proxy.PROXY_IGNORE_RESULT
        else
                print("query-time: " .. (inj.query_time / 1000) .. "ms")
                print("response-time: " .. (inj.response_time / 1000) .. "ms")
        end
end
```

For further examples, see Chapter 6, *Using MySQL Proxy*.

# Chapter 6. Using MySQL Proxy

## Table of Contents



There are a number of different ways to use MySQL Proxy. At the most basic level, you can allow MySQL Proxy to pass queries from clients to a single server. To use MySQL Proxy in this mode, you just have to specify on the command line the backend server to which the proxy should connect:

```
shell> mysql-proxy --proxy-backend-addresses=sakila:3306
```

If you specify multiple backend MySQL servers, the proxy connects each client to each server in a round-robin fashion. Suppose that you have two MySQL servers, A and B. The first client to connect is connected to server A, the second to server B, the third to server A. For example:

```
shell> mysql-proxy \
     --proxy-backend-addresses=narcissus:3306 \
     --proxy-backend-addresses=nostromo:3306
```

When you specify multiple servers in this way, the proxy automatically identifies when a MySQL server has become unavailable and marks it accordingly. New connections are automatically attached to a server that is available, and a warning is reported to the standard output from `mysql-proxy`:

```
network-mysqld.c.367: connect(nostromo:3306) failed: Connection refused
network-mysqld-proxy.c.2405: connecting to backend (nostromo:3306) failed, marking it as down for ...
```

Lua scripts enable a finer level of control, both over the connections and their distribution and how queries and result sets are processed. When using an Lua script, you must specify the name of the script on the command line using the `--proxy-lua-script` [17] option:

```
shell> mysql-proxy --proxy-lua-script=mc.lua --proxy-backend-addresses=sakila:3306
```

When you specify a script, the script is not executed until a connection is made. This means that faults with the script are not raised until the script is executed. Script faults will not affect the distribution of queries to backend MySQL servers.

> **Note**
>
> Because a script is not read until the connection is made, you can modify the contents of the Lua script file while the proxy is still running and the modified script is automatically used for the next connection. This ensures that MySQL Proxy remains available because it need not be restarted for the changes to take effect.

## 6.1. Using the Administration Interface

The `mysql-proxy` administration interface can be accessed using any MySQL client using the standard protocols. You can use the administration interface to gain information about the proxy server as a whole - standard connections to the proxy are isolated to operate as if you were connected directly to the backend MySQL server.

In `mysql-proxy` 0.8.0 and earlier, a rudimentary interface was built into the proxy. In later versions this was replaced so that you must specify an administration script to be used when users connect to the administration interface.

To use the administration interface, specify the user name and password required to connect to the admin service, using the `--admin-username` [12] and `--admin-password` [12] options. You must also specify the Lua script to be used as the interface to the administration service by using the `admin-lua-script` [12] script option to point to a Lua script.

For example, you can create a basic interface to the internal components of the `mysql-proxy` system using the following script, written by Diego Medina:

```
--[[
   Copyright 2008, 2010, Oracle and/or its affiliates. All rights reserved.

   This program is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation; version 2 of the License.
   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.
   You should have received a copy of the GNU General Public License
   along with this program; if not, write to the Free Software
   Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
--]]
-- admin.lua
--[[
    See http://www.chriscalender.com/?p=41
    (Thanks to Chris Calender)
    See http://datacharmer.blogspot.com/2009/01/mysql-proxy-is-back.html
    (Thanks Giuseppe Maxia)
--]]
function set_error(errmsg)
    proxy.response = {
        type = proxy.MYSQLD_PACKET_ERR,
        errmsg = errmsg or "error"
    }
end
function read_query(packet)
    if packet:byte() ~= proxy.COM_QUERY then
        set_error("[admin] we only handle text-based queries (COM_QUERY)")
        return proxy.PROXY_SEND_RESULT
    end
    local query = packet:sub(2)
    local rows = { }
    local fields = { }
    -- try to match the string up to the first non-alphanum
    local f_s, f_e, command = string.find(packet, "^%s*(%w+)", 2)
    local option
    if f_e then
            -- if that match, take the next sub-string as option
            f_s, f_e, option = string.find(packet, "^%s+(%w+)", f_e + 1)
    end
    -- we got our commands, execute it
    if command == "show" and option == "querycounter" then
            ---
            -- proxy.PROXY_SEND_RESULT requires
            --
            -- proxy.response.type to be either
            -- * proxy.MYSQLD_PACKET_OK or
            -- * proxy.MYSQLD_PACKET_ERR
            --
            -- for proxy.MYSQLD_PACKET_OK you need a resultset
            -- * fields
            -- * rows
            --
            -- for proxy.MYSQLD_PACKET_ERR
            -- * errmsg
            proxy.response.type = proxy.MYSQLD_PACKET_OK
```

```
            proxy.response.resultset = {
                    fields = {
                            { type = proxy.MYSQL_TYPE_LONG, name = "query_counter", },
                    },
                    rows = {
                            { proxy.global.query_counter }
                    }
            }
            -- we have our result, send it back
            return proxy.PROXY_SEND_RESULT
    elseif command == "show" and option == "myerror" then
            proxy.response.type = proxy.MYSQLD_PACKET_ERR
            proxy.response.errmsg = "my first error"
            return proxy.PROXY_SEND_RESULT

    elseif string.sub(packet, 2):lower() == 'select help' then
            return show_process_help()

    elseif string.sub(packet, 2):lower() == 'show proxy processlist' then
            return show_process_table()
    elseif query == "SELECT * FROM backends" then
        fields = {
            { name = "backend_ndx",
              type = proxy.MYSQL_TYPE_LONG },
            { name = "address",
              type = proxy.MYSQL_TYPE_STRING },
            { name = "state",
              type = proxy.MYSQL_TYPE_STRING },
            { name = "type",
              type = proxy.MYSQL_TYPE_STRING },
        }
        for i = 1, #proxy.global.backends do
            local b = proxy.global.backends[i]
            rows[#rows + 1] = {
                i, b.dst.name, b.state, b.type
            }
        end
    else
        set_error()
        return proxy.PROXY_SEND_RESULT
    end
    proxy.response = {
        type = proxy.MYSQLD_PACKET_OK,
        resultset = {
            fields = fields,
            rows = rows
        }
    }
    return proxy.PROXY_SEND_RESULT
end
function make_dataset (header, dataset)
    proxy.response.type = proxy.MYSQLD_PACKET_OK
    proxy.response.resultset = {
        fields = {},
        rows = {}
    }
    for i,v in pairs (header) do
        table.insert(proxy.response.resultset.fields, {type = proxy.MYSQL_TYPE_STRING, name = v})
    end
    for i,v in pairs (dataset) do
        table.insert(proxy.response.resultset.rows, v )
    end
    return proxy.PROXY_SEND_RESULT
end
function show_process_table()
    local dataset = {}
    local header = { 'Id', 'IP Address', 'Time' }
```

```
    local rows = {}
    for t_i, t_v in pairs (proxy.global.process) do
        for s_i, s_v in pairs ( t_v ) do
            table.insert(rows, { t_i, s_v.ip, os.date('%c',s_v.ts) })
        end
    end
    return make_dataset(header,rows)
end
function show_process_help()
    local dataset = {}
    local header = { 'command',  'description' }
    local rows = {
        {'SELECT HELP',                 'This command.'},
        {'SHOW PROXY PROCESSLIST',      'Show all connections and their true IP Address.'},
    }
    return make_dataset(header,rows)
end
function dump_process_table()
    proxy.global.initialize_process_table()
    print('current contents of process table')
    for t_i, t_v in pairs (proxy.global.process) do
        print ('session id: ', t_i)
        for s_i, s_v in pairs ( t_v ) do
            print ( '\t', s_i, s_v.ip, s_v.ts )
        end
    end
    print ('---END PROCESS TABLE---')
end
--[[    Help
we use a simple string-match to split commands are word-boundaries
mysql> show querycounter
is split into
command = "show"
option  = "querycounter"
spaces are ignored, the case has to be as is.
mysql> show myerror
returns a error-packet
--]]
```

The script works in combination with a main proxy script, `reporter.lua`:

```
--[[
   Copyright 2008, 2010, Oracle and/or its affiliates. All rights reserved.

   This program is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation; version 2 of the License.
   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.
   You should have received a copy of the GNU General Public License
   along with this program; if not, write to the Free Software
   Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
--]]
-- reporter.lua
--[[
    See http://www.chriscalender.com/?p=41
    (Thanks to Chris Calender)
    See http://datacharmer.blogspot.com/2009/01/mysql-proxy-is-back.html
    (Thanks Giuseppe Maxia)
--]]
proxy.global.query_counter = proxy.global.query_counter or 0
function proxy.global.initialize_process_table()
    if proxy.global.process == nil then
        proxy.global.process = {}
    end
```

```
    if proxy.global.process[proxy.connection.server.thread_id] == nil then
        proxy.global.process[proxy.connection.server.thread_id] = {}
    end
end
function read_auth_result( auth )
    local state = auth.packet:byte()
    if state == proxy.MYSQLD_PACKET_OK then
        proxy.global.initialize_process_table()
        table.insert( proxy.global.process[proxy.connection.server.thread_id],
            { ip = proxy.connection.client.src.name, ts = os.time() } )
    end
end
function disconnect_client()
    local connection_id = proxy.connection.server.thread_id
    if connection_id then
        -- client has disconnected, set this to nil
        proxy.global.process[connection_id] = nil
    end
end
---
-- read_query() can return a resultset
--
-- You can use read_query() to return a result-set.
--
-- @param packet the mysql-packet sent by the client
--
-- @return
--   * nothing to pass on the packet as is,
--   * proxy.PROXY_SEND_QUERY to send the queries from the proxy.queries queue
--   * proxy.PROXY_SEND_RESULT to send your own result-set
--
function read_query( packet )
        -- a new query came in in this connection
        -- using proxy.global.* to make it available to the admin plugin
        proxy.global.query_counter = proxy.global.query_counter + 1
end
```

To use the script, save the first script to a file (`admin.lua` in the following example) and the other to `reporter.lua`, then run `mysql-proxy` specifying the admin script and a backend MySQL server:

```
shell> mysql-proxy --admin-lua-script=admin.lua --admin-password=password \ »
     --admin-username=root --proxy-backend-addresses=127.0.0.1:3306 -proxy-lua-script=reporter.lua
```

In a different window, connect to the MySQL server through the proxy:

```
shell> mysql --user=root --password=password --port=4040
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1798669
Server version: 5.0.70-log Gentoo Linux mysql-5.0.70-r1
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```

In another different window, connect to the `mysql-proxy` admin service using the specified user name and password:

```
shell> mysql --user=root --password=password --port=4041 --host=localhost
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.0.99-agent-admin
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```

To monitor the status of the proxy, ask for a list of the current active processes:

```
mysql> show proxy processlist;
+---------+--------------------+--------------------------+
```

```
| Id      | IP Address          | Time                      |
+---------+---------------------+---------------------------+
| 1798669 | 192.168.0.112:52592 | Wed Jan 20 16:58:00 2010 |
+---------+---------------------+---------------------------+
1 row in set (0.00 sec)
mysql>
```

For more information on the example, see MySQL Proxy Admin Example.

# Chapter 7. MySQL Proxy FAQ

**Questions**

- 7.22: [44] So authentication with connection pooling has to be done at every connection? What is the authentication latency?
- 7.23: [44] If you have multiple databases on the same box, can you use proxy to connect to databases on default port 3306?
- 7.24: [44] What about caching the authorization information so clients connecting are given back-end connections that were established with identical authorization information, thus saving a few more round trips?
- 7.25: [44] Is there any big web site using MySQL Proxy? For what purpose and what transaction rate have they achieved?
- 7.26: [44] How does MySQL Proxy compare to DBSlayer?
- 7.27: [45] I tried using MySQL Proxy without any Lua script to try a round-robin type load balancing. In this case, if the first database in the list is down, MySQL Proxy would not connect the client to the second database in the list.
- 7.28: [45] Is it “safe” to use `LuaSocket` with proxy scripts?
- 7.29: [45] How different is MySQL Proxy from DBCP (Database connection pooling) for Apache in terms of connection pooling?
- 7.30: [45] MySQL Proxy can handle about 5000 connections, what is the limit on a MySQL server?
- 7.31: [45] Would the Java-only connection pooling solution work for multiple web servers? With this, I would assume that you can pool across many web servers at once?

**Questions and Answers**

**7.1: In load balancing, how can I separate reads from writes?**

There is no automatic separation of queries that perform reads or writes to the different backend servers. However, you can specify to `mysql-proxy` that one or more of the “backend” MySQL servers are read only.

```
shell> mysql-proxy \
--proxy-backend-addresses=10.0.1.2:3306 \
--proxy-read-only-backend-addresses=10.0.1.3:3306 &
```

**7.2: How do I use a socket with MySQL Proxy? Proxy change logs mention that support for UNIX sockets has been added.**

Specify the path to the socket:

```
--proxy-backend-addresses=/path/to/socket
```

**7.3: Can I use MySQL Proxy with all versions of MySQL?**

MySQL Proxy is designed to work with MySQL 5.0 or higher, and supports the MySQL network protocol for 5.0 and higher.

**7.4: Can I run MySQL Proxy as a daemon?**

Use the `--daemon` [13] option. To keep track of the process ID, the daemon can be started with the `--pid-file=file` [17] option to save the PID to a known file name. On version 0.5.x, the Proxy cannot be started natively as a daemon.

**7.5: Do proxy applications run on a separate server? If not, what is the overhead incurred by Proxy on the DB server side?**

You can run the proxy on the application server, on its own box, or on the DB-server depending on the use case.

**7.6: With load balancing, what happens to transactions? Are all queries sent to the same server?**

Without any special customization the whole connection is sent to the same server. That keeps the whole connection state intact.

**7.7: Is it possible to use MySQL Proxy with updating a Lucene index (or Solr) by making TCP calls to that server to update?**

Yes, but it is not advised for now.

**7.8: Is the system context switch expensive, how much overhead does the Lua script add?**

Lua is fast and the overhead should be small enough for most applications. The raw packet overhead is around 400 microseconds.

**7.9: How much latency does a proxy add to a connection?**

In the range of 400 microseconds per request.

**7.10: Do you have to make one large script and call it at proxy startup, can I change scripts without stopping and restarting (interrupting) the proxy?**

You can just change the script and the proxy will reload it when a client connects.

**7.11: If MySQL Proxy has to live on same machine as MySQL, are there any tuning considerations to ensure both perform optimally?**

MySQL Proxy can live on any box: application, database, or its own box. MySQL Proxy uses comparatively little CPU or RAM, with negligible additional requirements or overhead.

**7.12: I currently use SQL Relay for efficient connection pooling with a number of Apache processes connecting to a MySQL server. Can MySQL Proxy currently accomplish this? My goal is to minimize connection latency while keeping temporary tables available.**

Yes.

**7.13: Are these reserved function names (for example, `error_result()`) that get automatically called?**

Only functions and values starting with `proxy.*` are provided by the proxy. All others are user provided.

**7.14: As the script is re-read by MySQL Proxy, does it cache this or is it looking at the file system with each request?**

It looks for the script at client-connect and reads it if it has changed, otherwise it uses the cached version.

**7.15: Given that there is a `connect_server()` function, can a Lua script link up with multiple servers?**

MySQL Proxy provides some tutorials in the source package; one is `examples/tutorial-keepalive.lua`.

**7.16: Is the MySQL Proxy an API?**

No, MySQL Proxy is an application that forwards packets from a client to a server using the MySQL network protocol. The MySQL Proxy provides a API allowing you to change its behavior.

**7.17: The global namespace variable example with quotas does not persist after a reboot, is that correct?**

Yes. If you restart the proxy, you lose the results, unless you save them in a file.

**7.18: Can MySQL Proxy handle SSL connections?**

No, being the man-in-the-middle, Proxy cannot handle encrypted sessions because it cannot share the SSL information.

**7.19: Could MySQL Proxy be used to capture passwords?**

The MySQL network protocol does not allow passwords to be sent in cleartext, all you could capture is the encrypted version.

**7.20: Are there tools for isolating problems? How can someone figure out whether a problem is in the client, the database, or the proxy?**

You can set a debug script in the proxy, which is an exceptionally good tool for this purpose. You can see very clearly which component is causing the problem, if you set the right breakpoints.

**7.21: Is MySQL Proxy similar to what is provided by Java connection pools?**

Yes and no. Java connection pools are specific to Java applications, MySQL Proxy works with any client API that talks the MySQL network protocol. Also, connection pools do not provide any functionality for intelligently examining the network packets and modifying the contents.

**7.22: So authentication with connection pooling has to be done at every connection? What is the authentication latency?**

You can skip the round-trip and use the connection as it was added to the pool. As long as the application cleans up the temporary tables it used. The overhead is (as always) around 400 microseconds.

**7.23: If you have multiple databases on the same box, can you use proxy to connect to databases on default port 3306?**

Yes, MySQL Proxy can listen on any port, provided that none of the MySQL servers are listening on the same port.

**7.24: What about caching the authorization information so clients connecting are given back-end connections that were established with identical authorization information, thus saving a few more round trips?**

There is an `--proxy-pool-no-change-user` [17] option that provides this functionality.

**7.25: Is there any big web site using MySQL Proxy? For what purpose and what transaction rate have they achieved?**

Yes, gaiaonline. They have tested MySQL Proxy and seen it handle 2400 queries per second through the proxy.

**7.26: How does MySQL Proxy compare to DBSlayer?**

DBSlayer is a REST->MySQL tool, MySQL Proxy is transparent to your application. No change to the application is needed.

**7.27: I tried using MySQL Proxy without any Lua script to try a round-robin type load balancing. In this case, if the first database in the list is down, MySQL Proxy would not connect the client to the second database in the list.**

This issue is fixed in version 0.7.0.

**7.28: Is it "safe" to use `LuaSocket` with proxy scripts?**

You can, but it is not advised because it may block.

**7.29: How different is MySQL Proxy from DBCP (Database connection pooling) for Apache in terms of connection pooling?**

Connection Pooling is just one use case of the MySQL Proxy. You can use it for a lot more and it works in cases where you cannot use DBCP (for example, if you do not have Java).

**7.30: MySQL Proxy can handle about 5000 connections, what is the limit on a MySQL server?**

The server limit is given by the value of the `max_connections` system variable. The default value is version dependent.

**7.31: Would the Java-only connection pooling solution work for multiple web servers? With this, I would assume that you can pool across many web servers at once?**

Yes. But you can also start one proxy on each application server to get a similar behavior as you have it already.

# Appendix A. Third Party Licenses

## Table of Contents



Use of any of this software is governed by the terms of the licenses that follow.

### MySQL Proxy

- Section A.1, “GLib License (for MySQL Proxy)”
- Section A.2, “GNU Lesser General Public License Version 2.1, February 1999”
- Section A.3, “`libevent` License”
- Section A.4, “Libiconv License”
- Section A.5, “`libintl` License”
- Section A.6, “`LPeg` Library License”
- Section A.7, “Lua (liblua) License”
- Section A.8, “`LuaFileSystem` Library License”
- Section A.9, “PCRE License”

## A.1. GLib License (for MySQL Proxy)

The following software may be included in this product:

```
GLib

You are receiving a copy of the GLib library in both source
and object code in the following [proxy install dir]/lib/ and
[proxy install dir]/licenses/lgpl folders. The terms of the
Oracle license do NOT apply to the GLib library; it is licensed
under the following license, separately from the Oracle programs
you receive. If you do not wish to install this library, you may
create an "exclude" file and run tar with the X option, as in
the following example, but the Oracle program might not operate
properly or at all without the library:
  tar -xvfX <package-tar-file> <exclude-file>
where the exclude-file contains, e.g.:
  <package-name>/lib/libglib-2.0.so.0.1600.6
  <package-name>/lib/libglib-2.0.so.0
  ...

Example:
tar -xvfX mysql-proxy-0.8.1-solaris10-x86-64bit.tar.gz Exclude
```

```
Exclude File:
mysql-proxy-0.8.1-solaris10-x86-64bit/lib/libglib-2.0.so
mysql-proxy-0.8.1-solaris10-x86-64bit/lib/libglib-2.0.so.0
mysql-proxy-0.8.1-solaris10-x86-64bit/lib/libglib-2.0.so.0.1600.6
mysql-proxy-0.8.1-solaris10-x86-64bit/lib/libgmodule-2.0.so
mysql-proxy-0.8.1-solaris10-x86-64bit/lib/libgmodule-2.0.so.0
mysql-proxy-0.8.1-solaris10-x86-64bit/lib/libgmodule-2.0.so.0.1600.6
mysql-proxy-0.8.1-solaris10-x86-64bit/lib/libgthread-2.0.so
mysql-proxy-0.8.1-solaris10-x86-64bit/lib/libgthread-2.0.so.0
mysql-proxy-0.8.1-solaris10-x86-64bit/lib/libgthread-2.0.so.0.1600.6
mysql-proxy-0.8.1-solaris10-x86-64bit/licenses/lgpl/glib-2.16.6.tar.gz
```

This component is licensed under Section A.2, “GNU Lesser General Public License Version 2.1, February 1999”.

## A.2. GNU Lesser General Public License Version 2.1, February 1999

```
The following applies to all products licensed under the
GNU Lesser General Public License, Version 2.1: You may
not use the identified files except in compliance with
the GNU Lesser General Public License, Version 2.1 (the
"License"). You may obtain a copy of the License at
http://www.gnu.org/licenses/lgpl-2.1.html. A copy of the
license is also reproduced below. Unless required by
applicable law or agreed to in writing, software distributed
under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
or implied. See the License for the specific language governing
permissions and limitations under the License.

                  GNU LESSER GENERAL PUBLIC LICENSE
                       Version 2.1, February 1999

 Copyright (C) 1991, 1999 Free Software Foundation, Inc.
 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL.  It also counts
 as the successor of the GNU Library Public License, version 2, hence
 the version number 2.1.]

                            Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
Licenses are intended to guarantee your freedom to share and change
free software--to make sure the software is free for all its users.

  This license, the Lesser General Public License, applies to some
specially designated software packages--typically libraries--of the
Free Software Foundation and other authors who decide to use it.  You
can use it too, but we suggest you first think carefully about whether
this license or the ordinary General Public License is the better
strategy to use in any particular case, based on the explanations below.

  When we speak of free software, we are referring to freedom of use,
not price.  Our General Public Licenses are designed to make sure that
you have the freedom to distribute copies of free software (and charge
for this service if you wish); that you receive source code or can get
it if you want it; that you can change the software and use pieces of
it in new free programs; and that you are informed that you can do
these things.

  To protect your rights, we need to make restrictions that forbid
```

```
distributors to deny you these rights or to ask you to surrender these
rights.  These restrictions translate to certain responsibilities for
you if you distribute copies of the library or if you modify it.

  For example, if you distribute copies of the library, whether gratis
or for a fee, you must give the recipients all the rights that we gave
you.  You must make sure that they, too, receive or can get the source
code.  If you link other code with the library, you must provide
complete object files to the recipients, so that they can relink them
with the library after making changes to the library and recompiling
it.  And you must show them these terms so they know their rights.

  We protect your rights with a two-step method: (1) we copyright the
library, and (2) we offer you this license, which gives you legal
permission to copy, distribute and/or modify the library.

  To protect each distributor, we want to make it very clear that
there is no warranty for the free library.  Also, if the library is
modified by someone else and passed on, the recipients should know
that what they have is not the original version, so that the original
author's reputation will not be affected by problems that might be
introduced by others.

  Finally, software patents pose a constant threat to the existence of
any free program.  We wish to make sure that a company cannot
effectively restrict the users of a free program by obtaining a
restrictive license from a patent holder.  Therefore, we insist that
any patent license obtained for a version of the library must be
consistent with the full freedom of use specified in this license.

  Most GNU software, including some libraries, is covered by the
ordinary GNU General Public License.  This license, the GNU Lesser
General Public License, applies to certain designated libraries, and
is quite different from the ordinary General Public License.  We use
this license for certain libraries in order to permit linking those
libraries into non-free programs.

  When a program is linked with a library, whether statically or using
a shared library, the combination of the two is legally speaking a
combined work, a derivative of the original library.  The ordinary
General Public License therefore permits such linking only if the
entire combination fits its criteria of freedom.  The Lesser General
Public License permits more lax criteria for linking other code with
the library.

  We call this license the "Lesser" General Public License because it
does Less to protect the user's freedom than the ordinary General
Public License.  It also provides other free software developers Less
of an advantage over competing non-free programs.  These disadvantages
are the reason we use the ordinary General Public License for many
libraries.  However, the Lesser license provides advantages in certain
special circumstances.

  For example, on rare occasions, there may be a special need to
encourage the widest possible use of a certain library, so that it
becomes a de-facto standard.  To achieve this, non-free programs
must be allowed to use the library.  A more frequent case is that
a free library does the same job as widely used non-free libraries.
In this case, there is little to gain by limiting the free library
to free software only, so we use the Lesser General Public License.

  In other cases, permission to use a particular library in non-free
programs enables a greater number of people to use a large body of
free software.  For example, permission to use the GNU C Library in
non-free programs enables many more people to use the whole GNU
operating system, as well as its variant, the GNU/Linux operating
system.
```

```
  Although the Lesser General Public License is Less protective of the
users' freedom, it does ensure that the user of a program that is
linked with the Library has the freedom and the wherewithal to run
that program using a modified version of the Library.

  The precise terms and conditions for copying, distribution and
modification follow.  Pay close attention to the difference between a
"work based on the library" and a "work that uses the library".  The
former contains code derived from the library, whereas the latter must
be combined with the library in order to run.

                  GNU LESSER GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

  0. This License Agreement applies to any software library or other
program which contains a notice placed by the copyright holder or
other authorized party saying it may be distributed under the terms of
this Lesser General Public License (also called "this License").
Each licensee is addressed as "you".

  A "library" means a collection of software functions and/or data
prepared so as to be conveniently linked with application programs
(which use some of those functions and data) to form executables.

  The "Library", below, refers to any such software library or work
which has been distributed under these terms.  A "work based on the
Library" means either the Library or any derivative work under
copyright law: that is to say, a work containing the Library or a
portion of it, either verbatim or with modifications and/or translated
straightforwardly into another language.  (Hereinafter, translation is
included without limitation in the term "modification".)

  "Source code" for a work means the preferred form of the work for
making modifications to it.  For a library, complete source code means
all the source code for all modules it contains, plus any associated
interface definition files, plus the scripts used to control
compilation and installation of the library.

  Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running a program using the Library is not restricted, and output from
such a program is covered only if its contents constitute a work based
on the Library (independent of the use of the Library in a tool for
writing it).  Whether that is true depends on what the Library does
and what the program that uses the Library does.

  1. You may copy and distribute verbatim copies of the Library's
complete source code as you receive it, in any medium, provided that
you conspicuously and appropriately publish on each copy an
appropriate copyright notice and disclaimer of warranty; keep intact
all the notices that refer to this License and to the absence of any
warranty; and distribute a copy of this License along with the
Library.

  You may charge a fee for the physical act of transferring a copy,
and you may at your option offer warranty protection in exchange for a
fee.

  2. You may modify your copy or copies of the Library or any portion
of it, thus forming a work based on the Library, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

    a) The modified work must itself be a software library.

    b) You must cause the files modified to carry prominent notices
```

```
    stating that you changed the files and the date of any change.

    c) You must cause the whole of the work to be licensed at no
    charge to all third parties under the terms of this License.

    d) If a facility in the modified Library refers to a function or a
    table of data to be supplied by an application program that uses
    the facility, other than as an argument passed when the facility
    is invoked, then you must make a good faith effort to ensure that,
    in the event an application does not supply such function or
    table, the facility still operates, and performs whatever part of
    its purpose remains meaningful.

    (For example, a function in a library to compute square roots has
    a purpose that is entirely well-defined independent of the
    application.  Therefore, Subsection 2d requires that any
    application-supplied function or table used by this function must
    be optional: if the application does not supply it, the square
    root function must still compute square roots.)

These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Library,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Library, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote
it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Library.

In addition, mere aggregation of another work not based on the Library
with the Library (or with a work based on the Library) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

  3. You may opt to apply the terms of the ordinary GNU General Public
License instead of this License to a given copy of the Library.  To do
this, you must alter all the notices that refer to this License, so
that they refer to the ordinary GNU General Public License, version 2,
instead of to this License.  (If a newer version than version 2 of the
ordinary GNU General Public License has appeared, then you can specify
that version instead if you wish.)  Do not make any other change in
these notices.

  Once this change is made in a given copy, it is irreversible for
that copy, so the ordinary GNU General Public License applies to all
subsequent copies and derivative works made from that copy.

  This option is useful when you wish to copy part of the code of
the Library into a program that is not a library.

  4. You may copy and distribute the Library (or a portion or
derivative of it, under Section 2) in object code or executable form
under the terms of Sections 1 and 2 above provided that you accompany
it with the complete corresponding machine-readable source code, which
must be distributed under the terms of Sections 1 and 2 above on a
medium customarily used for software interchange.

  If distribution of object code is made by offering access to copy
from a designated place, then offering equivalent access to copy the
source code from the same place satisfies the requirement to
```

```
distribute the source code, even though third parties are not
compelled to copy the source along with the object code.

  5. A program that contains no derivative of any portion of the
Library, but is designed to work with the Library by being compiled or
linked with it, is called a "work that uses the Library".  Such a
work, in isolation, is not a derivative work of the Library, and
therefore falls outside the scope of this License.

  However, linking a "work that uses the Library" with the Library
creates an executable that is a derivative of the Library (because it
contains portions of the Library), rather than a "work that uses the
library".  The executable is therefore covered by this License.
Section 6 states terms for distribution of such executables.

  When a "work that uses the Library" uses material from a header file
that is part of the Library, the object code for the work may be a
derivative work of the Library even though the source code is not.
Whether this is true is especially significant if the work can be
linked without the Library, or if the work is itself a library.  The
threshold for this to be true is not precisely defined by law.

  If such an object file uses only numerical parameters, data
structure layouts and accessors, and small macros and small inline
functions (ten lines or less in length), then the use of the object
file is unrestricted, regardless of whether it is legally a derivative
work.  (Executables containing this object code plus portions of the
Library will still fall under Section 6.)

  Otherwise, if the work is a derivative of the Library, you may
distribute the object code for the work under the terms of Section 6.
Any executables containing that work also fall under Section 6,
whether or not they are linked directly with the Library itself.

  6. As an exception to the Sections above, you may also combine or
link a "work that uses the Library" with the Library to produce a
work containing portions of the Library, and distribute that work
under terms of your choice, provided that the terms permit
modification of the work for the customer's own use and reverse
engineering for debugging such modifications.

  You must give prominent notice with each copy of the work that the
Library is used in it and that the Library and its use are covered by
this License.  You must supply a copy of this License.  If the work
during execution displays copyright notices, you must include the
copyright notice for the Library among them, as well as a reference
directing the user to the copy of this License.  Also, you must do one
of these things:

    a) Accompany the work with the complete corresponding
    machine-readable source code for the Library including whatever
    changes were used in the work (which must be distributed under
    Sections 1 and 2 above); and, if the work is an executable linked
    with the Library, with the complete machine-readable "work that
    uses the Library", as object code and/or source code, so that the
    user can modify the Library and then relink to produce a modified
    executable containing the modified Library.  (It is understood
    that the user who changes the contents of definitions files in the
    Library will not necessarily be able to recompile the application
    to use the modified definitions.)

    b) Use a suitable shared library mechanism for linking with the
    Library.  A suitable mechanism is one that (1) uses at run time a
    copy of the library already present on the user's computer system,
    rather than copying library functions into the executable, and (2)
    will operate properly with a modified version of the library, if
    the user installs one, as long as the modified version is
```

```
    interface-compatible with the version that the work was made with.

    c) Accompany the work with a written offer, valid for at
    least three years, to give the same user the materials
    specified in Subsection 6a, above, for a charge no more
    than the cost of performing this distribution.

    d) If distribution of the work is made by offering access to copy
    from a designated place, offer equivalent access to copy the above
    specified materials from the same place.

    e) Verify that the user has already received a copy of these
    materials or that you have already sent this user a copy.

  For an executable, the required form of the "work that uses the
Library" must include any data and utility programs needed for
reproducing the executable from it.  However, as a special exception,
the materials to be distributed need not include anything that is
normally distributed (in either source or binary form) with the major
components (compiler, kernel, and so on) of the operating system on
which the executable runs, unless that component itself accompanies
the executable.

  It may happen that this requirement contradicts the license
restrictions of other proprietary libraries that do not normally
accompany the operating system.  Such a contradiction means you cannot
use both them and the Library together in an executable that you
distribute.

  7. You may place library facilities that are a work based on the
Library side-by-side in a single library together with other library
facilities not covered by this License, and distribute such a combined
library, provided that the separate distribution of the work based on
the Library and of the other library facilities is otherwise
permitted, and provided that you do these two things:

    a) Accompany the combined library with a copy of the same work
    based on the Library, uncombined with any other library
    facilities.  This must be distributed under the terms of the
    Sections above.

    b) Give prominent notice with the combined library of the fact
    that part of it is a work based on the Library, and explaining
    where to find the accompanying uncombined form of the same work.

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

  9. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Library or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Library (or any work based on the
Library), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Library or works based on it.

  10. Each time you redistribute the Library (or any work based on the
Library), the recipient automatically receives a license from the
original licensor to copy, distribute, link with or modify the Library
subject to these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
```

You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

```
  15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO
WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW.
EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR
OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY
KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE
LIBRARY IS WITH YOU.  SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME
THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN
WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY
AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU
FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR
CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE
LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A
FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF
SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH
DAMAGES.

                     END OF TERMS AND CONDITIONS

           How to Apply These Terms to Your New Libraries

  If you develop a new library, and you want it to be of the greatest
possible use to the public, we recommend making it free software that
everyone can redistribute and change.  You can do so by permitting
redistribution under these terms (or, alternatively, under the terms
of the ordinary General Public License).

  To apply these terms, attach the following notices to the library.
It is safest to attach them to the start of each source file to most
effectively convey the exclusion of warranty; and each file should
have at least the "copyright" line and a pointer to where the full
notice is found.

    <one line to give the library's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

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

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

    You should have received a copy of the GNU Lesser General Public
    License along with this library; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
    02110-1301  USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the library, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the
  library `Frob' (a library for tweaking knobs) written by James
  Random Hacker.

  <signature of Ty Coon>, 1 April 1990
  Ty Coon, President of Vice
```

```
That's all there is to it!
```

## A.3. libevent License

The following software may be included in this product:

```
libevent

Copyright (c) 2000-2007 Niels Provos <provos@citi.umich.edu>
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. 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.
3. The name of the author may not be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``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 AUTHOR 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

==
Parts developed by Adam Langley
==

==
log.c
Based on err.c, which was adapted from OpenBSD libc *err*warncode.

Copyright (c) 2005 Nick Mathewson
Copyright (c) 2000 Dug Song
Copyright (c) 1993 The Regents of the University of California.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. 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.
3. Neither the name of the University nor the names of its
   contributors may be used to endorse or promote products derived
   from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS 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 REGENTS
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.
==

==
min_heap.h

Copyright (c) 2006 Maxim Yegorushkin
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. 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.
3. The name of the author may not be used to endorse or promote
   products derived from this software without specific prior
   written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "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 AUTHOR 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.
==

==
win32.c

Copyright 2000-2002 Niels Provos
Copyright 2003 Michael A. Davis
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. 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.
3.  The name of the author may not be used to endorse or promote
    products derived from this software without specific prior
    written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "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 AUTHOR 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.
==
```

## A.4. Libiconv License

The following software may be included in this product:

```
Libiconv

You are receiving a copy of the GNU LIBICONV Library. The terms of the Oracle
license do NOT apply to the GNU LIBICONV Library; it is licensed under the
following license, separately from the Oracle programs you receive. If you do
not wish to install this program, you may delete [agent install
dir]/lib/libiconv.* and [agent install dir]/licenses/lgpl/iconv files.
```

This component is licensed under Section A.2, “GNU Lesser General Public License Version 2.1, February 1999”.

## A.5. `libintl` License

The following software may be included in this product:

libintl

```
Copyright (C) 1994 X Consortium

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE X CONSORTIUM
BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF
CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of the X Consortium shall not be
used in advertising or otherwise to promote the sale, use or other dealings in
this Software without prior written authorization from the X Consortium.

FSF changes to this file are in the public domain.
 .
Copyright 1996-2007 Free Software Foundation, Inc. Taken from GNU libtool, 2001

Originally by Gordon Matzigkeit <gord@gnu.ai.mit.edu>, 1996

This file is free software; the Free Software Foundation gives unlimited
permission to copy and/or distribute it, with or without modifications, as long
as this notice is preserved.
.
You are receiving a copy of the libintl library. The terms of the Oracle license
do NOT apply to the libintl library; it is licensed under the following license,
separately from the Oracle programs you receive. If you do not wish to install
this program, you may create an "exclude" file and run tar with the X option.
```

```
This component is licensed under Section A.2, “GNU Lesser General Public License Version 2.1, February 199
```

## A.6. LPeg Library License

The following software may be included in this product:

```
LPeg

Use of any of this software is governed by the terms of the license below:

Copyright © 2008 Lua.org, PUC-Rio.

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

## A.7. Lua (liblua) License

The following software may be included in this product:

```
Lua (liblua)

Copyright © 1994–2008 Lua.org, PUC-Rio.

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject
to the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN THE SOFTWARE.
```

## A.8. LuaFileSystem Library License

The following software may be included in this product:

```
LuaFileSystem
```

```
Copyright © 2003 Kepler Project.

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject
to the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR
ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF
CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

## A.9. PCRE License

The following software may be included in this product:

PCRE (Perl Compatible Regular Expressions) Library

```
PCRE LICENCE

PCRE is a library of functions to support regular expressions
whose syntax and semantics are as close as possible to those
of the Perl 5 language.

Release 7 of PCRE is distributed under the terms of the "BSD"
licence, as specified below. The documentation for PCRE,
supplied in the "doc" directory, is distributed under the same
terms as the software itself.

The basic library functions are written in C and are
freestanding. Also included in the distribution is a set
of C++ wrapper functions.

THE BASIC LIBRARY FUNCTIONS
---------------------------
Written by:       Philip Hazel
Email local part: ph10
Email domain:     cam.ac.uk

University of Cambridge Computing Service,
Cambridge, England. Phone: +44 1223 334714.

Copyright (c) 1997-2006 University of Cambridge
All rights reserved.

THE C++ WRAPPER FUNCTIONS
-------------------------
Contributed by:   Google Inc.

Copyright (c) 2006, Google Inc.
All rights reserved.

THE "BSD" LICENCE
-----------------

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.
* Neither the name of the University of Cambridge nor
  the name of Google Inc. nor the names of their contributors
  may be used to endorse or promote products derived from
  this software without specific prior written 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.

End
```