mysqlsla v2 Reports

This document explains how to create and customize mysqlsla v2 standard reports. Standard reports are the human-readable reports that mysqlsla prints after parsing and filtering a log. These reports are formatted according to a report format which tells mysqlsla what to print, where and how.

A report format is associated with each type of MySQL log: slow, general, binary, udl and msl. mysqlsla has, internally, a default report format for each log type. For all but udl logs, these default report formats are usually sufficient (they cover every meta-property available).

However, you can override the default report format for any log type by explicitly setting the report format with the **report-format** option.

mysqlsla v2 Reports Table of Contents

Stanard Report Format Template

« Top

A report format is defined in a simple text file. The report format itself must follow a template in order to allow mysqlsla to easily parse it. The template is:

```
(extra command line options)
HEADER
(header line format)
(header line values)
REPORT
report line format
report line values
```

The first lines are extra command line options and are optional. Any command line option is valid and must be entered exactly as it would be on the real command line. These command line options override those actually given on the command line. Blank lines are allowed and ignored for readability.

With or without any extra command line options, the next line must be the exact literal string at the very start of the line: HEADER. This tells mysqlsla that all the lines that follow until REPORT are header lines which are printed only once at the beginning of the report. Here is where you begin putting **header and report format/value lines** as described in the next section. Blank lines are allowed but not ignored: a blank line here because a blank line in the header.

With or without any header lines (but always with the HEADER line present), the next line must be the exact literal string at the very start of the line: REPORT. This tells mysqlsla that all the rest of the lines are per-query report lines. Here again you put **header and report format/value lines**. Blank lines are allowed but not ignored: a blank line here becomes a blank line in the report.

mysqlsla will die with an error if it parses no report format lines (lines after REPORT). (Unless the **silent** option is being used, then mysqlsla does not parse the report format file at all.)

In general, mysqlsla tries to make the report format WYSIWYG, although it is by no means a complete and comprehensive reporting system. I think, at least, it is flexible enough to allow for some creativity. It is surely far superior to the v1 branch which had zero flexibility.

Header and Report Format/Value Lines

« Top

The most important lines in a report format are the header and report format/value lines. These pairs of lines are like Perl picture/argument lines or sprintf(FORMAT, ARGUMENTS). Since both header and report lines can have the exact same values (or "arguments" if you prefer that term), I will speak only of report lines.

The first line in the pair is the format. It is identical to an **sprintf() format** because it *is* an sprintf() format. mysqlsla simply passes the format line to sprintf(). Therefore, practically everything you know about sprintf() formats is true for mysqlsla report format lines, too.

The second line in the pair lists the values to print according to the format. (The values are interpolated into the format.) These values must be separated by a single space and put on one single line (there is no way to escape and break the line).

The values are simple names, not code-style variables. These are the available value names:

- Any **meta-property name** except those noted as "Only for meta-property filter" (except db)
- Any abbreviated **command line option** name: lt for log-type, etc. (see the **op code**)
- Any grand total meta-property name
- 'query' for the abstracted form of the current query
- 'total_queries' for the total number of queries
- 'total_unique_queries' for the total number of unique (abstracted) queries
- 'total_unique_users' for the total number of unique 'user@host IP'
- 'users' to produce the special total users **summary value**
- 'explain' to produce the special EXPLAIN **summary value**
- 'logs' to produce the special list of logs parsed **summary value**

If you use a value in a standard report which does not exist, nothing terrible will happen. mysqlsla will simply return a 0 (zero) or blank string.

Each value can be appended with a **code** in the form: value:code. Codes are the subject of the next section.

Finally, one important point remains to be noted here: be sure to use the correct conversion in the format for each value. For example, do not use an integer conversion (%d) for a string value, otherwise mysqlsla will

print many errors.

Codes

« Top

Technically any value can be "coded," but it only makes sense to code certain values. If you code a string value with **short** for example, the result is undefined (mysqlsla will probably print a lot of errors). In general, the codes are straightforward and there are only four.

micro

« Top

The micro code is used with microsecond values and requires an %s converstion in the format line. The value is shortened (if possible) and labeled according to its percision: 'value s' for values >= 1 second; 'value ms' for values 1-999 milliseconds; 'value µs' for values 1-999 microseconds. For microsecond values, the default µs label can be changed with the **microsecond-symbol** option.

See **Microsecond Support for MySQL Slow Logs** for a quick overview of microsecond values (1 s vs. 1 ms vs. 1 µs).

short

« Top

The short code is like the micro code except that it is used with big integer values; it also requires an %s converstion in the format line. The value is shorted and labeled according to its size:

- 'value' for values 0-999
- 'value.nnk' for values 1,000-999,999 (where n is always 2 decimal places of percision)
- 'value.nnM' for values 1,000,000-999,999,999
- 'value.nnG' for values 1,000,000,000-999,999,999,999
- 'value.nnT' for values 1,000,000,000,000-999,999,999,999,999
- 'value.nnP' for values 1,000,000,000,000,000-999,999,999,999,999,999 (1-999.99 petabytes)
- 'value.nnE' for values 1,000,000,000,000,000,000-999,999,999,999,999,999,999 (1-999.99 exabytes)

cap

« Top

The cap code capitalizes SQL syntax. It is used with the query or sample value to make the SQL easier to read. For the query value, it "un-flattens" the query which was flattened (made all lowercase) by the abstraction process. The same can be done to correct or normalize the SQL syntax of the sample value.

The capitalized words are all of the **MySQL reserved words** plus: SUM, MIN, MAX, COUNT, AGAINST, SQL_NO_CACHE, RESET, MASTER, SLAVE, CONCAT, UNIX_TIMESTAMP, ROUND, RAND.

mysqlsla does not understand backtick (`) escaping. In fact, it removes all backticks. Therefore, if a SQL statement has a terribly named table called `join`, mysqlsla will render the table name as JOIN thereby confusing everyone.

op

« Top

The final code is not a formatting code like the previous three. The op code tells mysqlsla that value is the name of a **command line option** and it should therefore print the value given to that command line option. If no value was given because the command line option was not used, a 0 (zero) or blank will be printed. Or, some options are set with their defaults values even if not explicitly used (sort, for example). For such options their default values will be printed.

value must reference the abbreviated name of the option if it has one. For example: db for databases, lt for log-type, rf for standard-report-format, mf for meta-filter, sf for statement-filter *but* grep for grep, top for top, host for host.

This code was implemented to avoid name collisions between command line options and SQL statement meta-properties. At present there are no collisions but since mysqlsla can parse **user-defined logs** it is possible that a udl would introduce a meta-property with the same name as a command line option.

Special Lines and Values

« Top

Conditional Lines

« Top

Conditional lines begin with a single quesiton mark (?). They are only printed if all values exist. A value can be zero (or an empty string) and still exist. Non-existent values are usually command line options that were not used or meta-properties that were not available from the log or not calucated for some reason. The question mark is automatically removed from the line if it is printed.

Valueless Lines

« Top

Valueless lines are created by using a single underscore (_) for the value line of the format line which has no values yet should still be printed. Do not simply leave the value line blank; this will terribly confuse mysqlsla.

Summary Values

« Top

Currently there are only three summary values: users, explain and logs.

users refers to the list of all unique "user@host IP" which are associated with the query. The users summary prints these users as well as what percentage each user accounts for each occurrence of the query and what percentage each user accounts for all queries.

explain refers to the output from EXPLAIN if the **explain** option was used and the query was able to be EXPLAINed. Otherwise, the summary will print why they query could not be EXPLAINed.

logs refers simply to the list of log files given to mysqlsla to parse. The summary prints all given log files even those that could not be parsed for some reason (if the file did not exist for example). (Technically, this summary just prints ARGV.)

Walk-Through Example

« Top

Now that you are a certified mysqlsla v2 report format hacker, having read all the above documentation, here is a rapid walkthrough of the default report format for slow logs. Line numbers have been added for reference. First, brief yourself with the format. After, we will walk through it "aloud."

```
01  -nthp
02
03  HEADER
04  Report for %s logs: %s
05  lt:op logs
06  %d queries total, %d unique
07  total_queries, total_unique_queries
08  Sorted by '%s'
09  sort:op
10  Grand Totals: Time %s s, Lock %s s, Rows sent %s, Rows Examined %s
11  gt_t:short gt_l:short gt_rs:short gt_re:short
12
13  REPORT
14
15  ____________________________________________________________________ %03d ___
16  sort_rank
17  Count          : %s  (%.2f%%)
18  c_sum:short c_sum_p
19  Time           : %s total, %s avg, %s to %s max  (%.2f%%)
20  t_sum:micro t_avg:micro t_min:micro t_max:micro t_sum_p
21  ? %3s%% of Time : %s total, %s avg, %s to %s max
22  nthp:op t_sum_nthp:micro t_avg_nthp:micro t_min_nthp:micro t_max_nthp:micro
23  ? Distribution : %s
24  t_dist
25  Lock Time      : %s total, %s avg, %s to %s max  (%.2f%%)
26  l_sum:micro l_avg:micro l_min:micro l_max:micro l_sum_p
27  ? %3s%% of Lock : %s total, %s avg, %s to %s max
28  nthp:op l_sum_nthp:micro l_avg_nthp:micro l_min_nthp:micro l_max_nthp:micro
29  Rows sent      : %s avg, %s to %s max  (%.2f%%)
30  rs_avg:short rs_min:short rs_max:short rs_sum_p
31  Rows examined : %s avg, %s to %s max  (%.2f%%)
32  re_avg:short re_min:short re_max:short re_sum_p
33  Database       : %s
34  db
35  Users          : %s
36  users
37  ?EXPLAIN        : %s
38  explain
39
40  Query abstract:
41  _
42  %s
43  query:cap
44
45  Query sample:
46  _
47  %s
48  sample
49
```

Line 01 and all lines until HEADER are extra command line options. The **nth-percent** option is given to enable, if possible, conditional lines 21 and 27. Line 02 is ignored because blank lines are ignored until HEADER.

HEADER is read at line 03, signaling that the lines that follow until REPORT are print-once header lines. Lines 04 and 05 are read as one header line. lt refers to the command line option **log-type** because of the **op code**; whatever value was given to log-type will be printed. logs is a **summary value** which will result in @ARGV being printed: all the log files that mysqlsla was told to parse. Lines 06 and 07 are read as one header line. total_queries and total_unique_queries are normal values. Lines 08 and 09 are read as one header line. sort refers to the command line option **sort** because of the **op code**; whatever value was given to sort will be printed (or sort's default value). Lines 10 and 11 are read as one header line. The four values, gt_t, gt_l, gt_rs and gt_re, all refer to grand totals: the log-wide sum of the respective meta-properties. Each value is **short coded** so it will be printed in short format: 12.73M instead of 12730021. Line 12 is read as one blank header line.

REPORT is read at line 13, signaling that all the following lines are report lines. Line 14 is read a one blank report line. Lines 15 and 16 are read as one report line. sort_rank is a normal value (from the unique queries hash). Lines 17 and 18 are read as one report line. c_sum short is **short coded** as we have already seen so we will not mention this coding anymore. Both c_sum and c_sum_p are normal values (from the unique queries hash).

Lines 19 and 20 are read as one report line. The first four values, t_sum, t_avg, t_min and t_max, are **micro coded** so their values will be formatted and printed in terms of either microseconds (µs unless changed by the **microsecond-symbol** option), milliseconds (ms) or seconds (s). The fifth and final value, t_sum_p, is a normal value.

Lines 21 and 22 are read as one **conditional report line** becasue line 21 begins with a question mark (?). If one or more value from line 22 does not exist, then line 21 is not printed. The first value, nthp, is **op coded** as we have already seen so we will not mention it further. If mysqlsla was not ran with the **nth-percent** option, line 21 will immediately be skipped. However, the nth-percent option was given on line 01, therefore at least this variable will have a value. However, the next four micro coded values, t_sum_nthp, t_avg_nthp, t_min_nthp and t_max_nthp, may not exist and therefore may still cause line 21 to be skipped. These values will not exist if the query appeared only once in the log (c_sum == 1, lines 17/18) or if it had less than **nthp-min-values** for this meta-property (t).

Lines 23 and 24 are read as one conditional report line because line 23 begins with a question mark. If mysqlsla was not ran with the **dist** option, line 24 will not be printed. If it was, the array of time (t) distributions (_dist) will be printed.

For brevity's sake, lines 25/26 and 27/28 are the same as 19/20 and 21/22 except that the meta-property is lock time (l) instead of slow query time (t).

Lines 29/30 and 31/32 are the same execpt that the former reports the rows sent (rs) meta-property and the later rows examined (re). Each have 3 short coded regular values and 1 regular value at the end (rs_sum_p and re_sum_p). Nothing special.

Lines 33 and 34 are read as one report line. Line 33 simple prints the database associated with the query if any. Very often, this value is blank.

Lines 35 and 36 are read as one report line. users is a special **summary value** that will cause the log-wide summary of all unique users to be printed along with what percentage each is associated with the specific query being report and with all queries (log-wide). Due to a current **limitation**, this later percentage is blank when replaying a **replay**.

Lines 37 and 38 are read as one conditional report line. explain is a special summary value that will cause the full output of EXPLAIN for the query to be printed if it is possible to EXPLAIN the query (if the query has a database and is a SELECT statement). Otherwise, any error or reason for why the query could not be EXPLAIN is printed (assuming, first, the line itself is printed due to its conditional nature).

Line 39 is a blank line. Lines 40 and 41 are read as one **valueless line** because line 40 has no conversions (%s, %d, %.2f, etc.) and line 41 is only a single underscore (_). Line 40 will be printed as-is with nothing added.

Lines 42 and 43 are read as one report line. query is a regular value referring to the abstracted form of the current query. It is **cap coded** so mysqlsla will capitalize all the major SQL synatx words which were flattened due to the abstraction process.

Line 44 is a blank line and lines 45 through 49 are nothing new: 45 and 46 are a **valueless line**, 47 and 48 cause a real sample of the query to be printed (not the abstracted form but a sample straight from the log), and the standard format file ends on line 49 with another blank line.

Why Not Perl Formats

« Top

Yes, I know the whole reporting system could be implemented using native Perl formats. Surely, it would allow much more powerful operations, but that power comes with complexity. I know Perl formats, you know Perl formats, but you and I are not the only two people using mysqlsla. A lot of good hackers do not hack in Perl; they have chosen Python or some other language so native Perl formats would be more a cursing than a blessing.

I think, however, we can all agree on printf()-style formats which is what mysqlsla essentially implements in a grandiose manner. Unless you code only in Lisp or Fortran, I cannot imagine anyone of even the slightest technical inclination not knowing at least the basic printf() formats.

I am interested in knowing if there is an easy ready-made solution I have egregiously overlooked? If you know of one, please **tell me**. It, however, cannot rely on any non-standard Perl module or external script or program.