SQL Diagnostic Manager for MySQL

SQL Diagnostic Manager for MySQL 8.9.x

Exported on 09/15/2022

Table of Contents

Go to Version 8.8.x[1]

Go to Version 8.7.x[2]

SQL DM for MySQL is a monitor and advisor application for MySQL database administrations.

It provides you with tools to manage more database servers, to tune your current database infrastructure, and to aid you in finding and fixing problems with your database applications before they develop into more serious issues or costly outages.

Review the release notes for SQL Diagnostic Manager for MySQL, to get a peek of the most recent release:

- Access the SQL Diagnostic Manager for MySQL Release Notes(see page 28)

Or simply, start with the configuration of your product Get Started[3].[4]

[1] https://wiki.idera.com/display/SQLDMMYSQL88
[2] https://wiki.idera.com/display/SQLDMMYSQL87
[3] http://wiki.idera.com/x/pwAgvgE
[4] http://wiki.idera.com/x/pwAgvgE

1 Access the SQL Diagnostic Manager for MySQL Release Notes

To get a glimpse into the newest features, fixed issues, and known issues in this SQL Diagnostic Manager for MySQL release, review the following sections of the Release Notes:

- See New Features[5] in this release
- Review Fixed Issues[6] included in this release
- Review Previous Features and Fixed Issues[7]
- See Known Issues[8]

1.1 New Features and Fixed Issues

SQL Diagnostic Manager for MySQL provides the following new features and fixed issues:

1.1.1 8.9.4 New features

There are no New Features reported for this version.

1.1.2 8.9.3 New features

- SQL DM for MySQL shows a notification in the UI when it is unable to send an Alert message due to wrong settings.

1.1.3 8.9.2 New Features

- SQL DM for MySQL now can analyze audit logging in Aurora Server.

1.1.4 8.9 New Features

- SQL DM for MySQL includes a query load profiling to Query Analysis along with a new interface for Query Details to make it easier to analyze query performance over time.
- Introduced new APIs for updating user passwords and LDAP bind user password.
- This version includes several Enterprise security improvements:
    - Restrict access to individual custom dashboards and the ability to create new dashboards to newly created users.
    - Capability to hide literals in SQL statements,
    - Ability to remove default admin user, and rename the default Admin name.
    - Improved audit log filtering, adding and excluding filter.
    - Configurable SNMP trap format.

1.1.5 8.9.4 Fixed Issues

- SQL DM for MySQL now runs properly in Debian 10 and 11.

[5] http://wiki.idera.com/x/aAEvvgE
[6] http://wiki.idera.com/x/aAEvvgE
[7] http://wiki.idera.com/x/aQEvvgE
[8] http://wiki.idera.com/x/agEvvgE

1.1.6 8.9.3 Fixed Issues

- SQL DM for MySQL now connects to MySQL using SSL encryption with caching_sha2_password authentication plugin.
- On Chrome browser, the data is automatically filled for server names, tags, and other filters if the Save Password is turned ON.
- SQL DM for MySQL validates the email address properly, and it allows saving multiple emails using space and comma separator.

1.1.7 8.9.2 Fixed Issues

- Google Cloud Server logs are getting analyzed properly.
- Starting with version 8.9.2, modified UDOs will display as *Resolved* when there are no new changes in future upgrades.
- Query Analyzer, Realtime, and Dashboard statistics extracted files, now collect correct time/date.
- General Log Export as CSV does not show undefined and NAN in the CSV file anymore.
- MySQL Versions prior to 5.6.2, now display accurate disk space.
- Show/Hide literals in Long running query notifications, are now available for all notification channels.
- When the query contains special characters, the exported CSV file now is formatted properly.
- Resolving Conflicted CSO’s now works properly.
- SQL DM for MySQL now sends pagerduty notifications correctly when the length of the message is greater than 1024.
- Editing a LDAP Password now sets the new password accordingly.
- MariaDB roles are listed correctly.
- The values plotted in the Monitor trend charts, are now converted to bytes.
- The column QPS now sorts accurately.
- Total Time column in Query Analyzer page, now shows the value properly.

1.1.8 8.9.1 Fixed Issues

There are no Known Issues reported for this version.

1.1.9 8.9 Fixed Issues

- The Threads page now correctly displays queries that contains these “<” or “>” characters.
- The Event Details window now correctly displays the event values that contains special characters.
- The SSL settings now correctly save the changes after editing the servers.
- The client user filter option in the Sniffer settings now works correctly.
- The Locked and Locking Queries in Real-time are now available for MySQL 8.0+ servers.
- The Connection Type is no longer changing from SSL to Direct when editing a server using API.
- SQL DM for MySQL is no longer crashing when passing some arguments in the start command.
- SQL DM for MySQL now correctly exports CSV reports when applying filters in the Query Analyzer, Dashboard, and Real-time.
- All chart series are visible for MySQL 8.0+ version in the Dashboard and screens.

1.1.10 8.9.3 Miscellaneous

- MariaDB Connector is upgraded to 3.1.11.

1.1.11 8.9.2 Miscellaneous

- JQuery library is upgraded to 3.5.0.

1.2 Previous Features and Fixed Issues

This page lists all the new features and fixed issues from previous releases for SQL Diagnostic Manager for MySQL:

1.2.1 8.8.0 SQL DM for MySQL (March 2019)

1.2.1.1 New Features

- SQL DM for MySQL now monitors Google Cloud SQL server Operating System metrics.
- This release includes Google Cloud SQL server file based log monitoring for General query, Slow query, and Error log files.

⚠ This release includes new registration keys. The new keys are available for registered users from our Customer Portal(see page 30).

1.2.1.2 Fixed Issues

- Regular Expression can now be used for the "Queries Starting with" filter for Processlist based Sniffer.
- The "Flush Status" query was renamed to "Flush Local Status" in the Monitors tab.
- SQL DM for MySQL is no longer displaying the warning message "User does not seem to have PROCESS privileges on this host. Only user's own processes will be displayed".
- The message "No data to display" in the history trend analysis graph for Monitor InnoDB cache>Free Memory is no longer prompting.
- SQL DM for MySQL is no longer displaying the "Undefined" message when hovering over the trend graph with a Monitor value of zero.
- The SQL Query for the CSO Tables_Without_Constraints was modified to excludes the views.
- SQL DM for MySQL now correctly raises SNMP traps/Syslog notifications when a monitor state changes from "Warning" to "Critical".
- The password field value is now correctly shown as a masked value.
- The RDS Log monitoring "Test reading the file" now correctly checks for "DownloadDBLogFilePortion" permissions.

1.2.1.3 Miscellaneous

- The double quotes can now be used in the email subject format.
- The MOM variable MemAvailable was added. The Javascript function for the RAM Usage monitor is modified to display appropriate value depending on the kernel.
- The MariaDB C/C connector was upgraded.
- SQL DM for MySQL has included an option to show truncated query/full query in the Threads page.
- All the SQL DM for MySQL passwords are stored encrypted in the SQLite database.

1.2.2 8.7.0 SQL DM for MySQL (August 2018)

1.2.2.1 New Features

No New Features for this version.

1.2.2.2 Fixed Issues

- In some rare-cases SQL DM displayed connection timeout error in Query Analyzer page.
- SQL DM logged error “too many SQL variables” if the Custom SQL Object query returned more than 500 rows
- Auto-Refresh interval was not getting set in Threads page.
- In Query Analyzer, Sniffer Settings gave console error on saving.
- OS metrics monitoring did not work with Aurora instances when the instance was registered with IP address.
- On killing one thread in the Threads page, 2 extra rows got removed.
- Default notification subject is changed to SQL DM for MySQL | [$ALERT_TYPE] | [$SERVER_NAME] | [$MSG_DETAILS].
- Now separate SNMP traps will be sent for each alertable monitor in one collection. Earlier, a single trap contained all alertable monitors.
- When unix_socket plugin is enabled for a user then all the users were listed under the monitor “Accounts without passwords”.
- Added more information in the Query Details popup in Query Analyzer.
- In pages with multi-server selector, on selecting and deselecting the servers in the selector pop-up, the servers are updated/selected in the server selector field in the background.
- In Query Analyzer and Real-Time, the metric value in the UI and in the downloaded CSV report did not match.
- In Query Analyzer and Real-Time, column names were different in CSV report as compared to UI.
- Diagnostic report showed wrong value for “Read file from” parameter, if log monitoring is disabled.

1.2.3 8.6.0 SQL DM for MySQL (April 2018)

1.2.3.1 New Features:

- A Galera cluster can now be automatically registered specifying any single node in the cluster.
- Optimized counter migration logic. This reduces the migration conflicts on upgrading SQL DM for MySQL tremendously. Post SQL DM for MySQL-8.6.0, a conflict is raised on upgrade only if any counter definition is modified in the latest release and the user has also customized the same counter.
- Added more columns for Slow Query log (for MariaDB and Percona Server) like full_scan, full_join etc. These values are written to Slow query log when ‘log_slow_verbosity’ is set to “Query_plan”.
- Connection timeout (MySQL, SSH and SSH Tunnel) can now be set for servers individually. The default value is 30 second

1.2.3.2 Fixed Issues:

- SQL DM for MySQL logged an SQLite error “Udo.def: not an error” on fresh installation.
- “Create User..” and “Create role..” queries were shown as a schema change in audit log.
- Queries containing ‘<’ or ‘>’ were displayed with their HTML-entities ‘&lt’, ‘&gt’ in “Threads” page.

- SQL DM for MySQL failed to read the MySQL log files when 'log_output' variable was set as both File and Table.
- SQL DM for MySQL failed to connect and displayed "Operations error" when using Active Directory External Authentication.
- In rare cases SQL DM failed to raise alerts for the monitor "Recent entries of type [Error]".

1.2.3.3 Miscellaneous:

- If the server's log query path variable returns a relative path, then SQL DM for MySQL generates the absolute path – required by MONyog – automatically.
- Added the monitor "Most Advanced Node" in the Galera group.

1.2.4 8.5.0 SQL DM for MySQL (March 2018)

1.2.4.1 New Features:

- SQL DM for MySQL can now analyze MariaDB and MySQL enterprise Audit log.
- Added support for LDAP with StarTLS and SSL.
- The default path for MONyog.log can be changed using the parameter "MONyogLogPath" from the MONyog.ini file.

1.2.4.2 Fixed Issues:

- SQL DM for MySQL logged bogus SQLite errors on fresh installation.
- SQL DM for MySQL displayed console error if LDAP group name contained inverted comma.
- Changed alert condition for "Seconds behind master" monitor to consider "NULL" as an alertable condition. It considered the value "NULL" stable condition earlier.
- On upgrading, SQL DM for MySQL filled the mail alert field for sniffer with the bogus email id "admin@mydomain.com".
- On selecting a tag in server selector GUI and then returning back to the main list, the default list of tags disappeared.

1.2.4.3 Miscellaneous:

- "MySQL server restarted" alert are shown in table format.
- Added option to view locked and locking queries when hovering over the queries in the Real Time
- Upgraded SQLite library to v3.21.0.

1.2.5 8.4.1 SQL DM for MySQL (January 2018)

1.2.5.1 Fixed Issues:

- The Linux command 'ps' failed to identify SQL DM for MySQL process status. This was due to a packaging issue introduced in 8.4.0. Versions before that were not affected.
- The option "Notify when server config change detected" compared in a case-sensitive manner and thus raised false alerts when the value for the MySQL variables changed from lower to upper case and vice versa. This could happen when a MySQL server was upgraded.

1.2.6 8.4.0 SQL DM for MySQL (December 2017)

1.2.6.1 New Features:

- Added more notification channels (Slack and Pagerduty) for SQL DM for MySQL alerts.
- Option to write SQL DM for MySQL alerts in the Syslog (of the machine where SQL DM for MySQL is installed). This option is only available for Linux.
- Option to edit the subject line for SQL DM for MySQL alerts.
- Added MONyog API to disable notification for a monitor based on a server/tag.
- Complete re-design of the Settings page.

1.2.6.2 Fixed Issues:

- Export as CSV was downloading sorted based on “Average latency” column, even if sorted based on another column in Query Analyzer.
- Fixed a case where SQL DM for MySQL sent bogus alerts.
- Fixed a bug in reading RDS log files.
- The Query filter “Custom” did not work in “Threads” page.
- In rare cases, SQL DM for MySQL Disk space Monitoring alert gave false alerts.
- The “Table” column returned empty values under “Tables” filter in Real-time.
- SNMPv2 traps were having the same OID for two different objects what caused SNMP alerts failure with SNMPv2 services.

1.2.6.3 Miscellaneous:

- Increased the length of tag and server names to 2048 characters.

1.2.7 8.3.2 SQL DM for MySQL (November 2017)

1.2.7.1 Fixed Issues:

- Resolved an issue regarding access to log files on RDS. The issue occurred because the XML response from the RDS REST API has recently changed in some cases. Our XML parsing was improved to handle this.

1.2.8 8.3.1 SQL DM for MySQL (November 2017)

1.2.8.1 Fixed Issues:

- When monitoring MariaDB servers from version 10.1, bogus alerts were raised for the ‘wsrep_ready’ global status variable, if the server was not a part of a Galera cluster. The wsrep API for Galera is loaded with those servers (with ‘wsrep_ready’ returning “OFF”) even though it is not used at all.
- Edit server MONyog API did not support data directory number of the registered servers for ‘_server’ parameter.

1.2.9 8.3.0 SQL DM for MySQL (October 2017)

1.2.9.1 New Features:

- Option to set a distinct email distribution list for warning and critical alerts: Now you can define a proper demarcation between the users who receives a particular alert based on the alert status. For instance, if you want only your on-call DBAs to receive critical alerts 24/7, this feature handles that easily.
- Graph Analysis: You can group one single metric (which you find most important) from different servers into one unified chart. This allows You can group one single metric (which you find most important) from different servers into one unified chart. This allows you to visually analyse a metric across servers at various points in time.
- Disk monitoring of the system where SQL DM for MySQL is installed: In case the free space on the system where SQL DM for MySQL is installed goes below the threshold value, SQL DM for MySQL raises an alert.
- Filter to exclude unwanted long running queries in Sniffer:
- You can tell SQL DM for MySQL to ignore a particular type of query by specifying a regular expression. SQL DM for MySQL neither would kill nor send an alert for such long-running queries.
- Option to filter queries based on poor indexes, missing indexes, errors and warnings. This feature is available for PERFORMANCE SCHEMA based sniffer in Query Analyzer and "Show details" page in Dashboard. When you choose a filter, SQL DM for MySQL shows queries based on the criterias set.
- Redesigned Server selector
- Redesigned Query Analyzer: a more intuitive view of the top 5 queries based on total time.
- Added proper Y-axis unit for charts, along with the option to define the units and unit-factors.

1.2.9.2 Fixed Issues:

- SQL DM for MySQL was logging “UNIQUE constraint failed..” errors in MONyog.log
- System charts failed to show data for many data collections, when the system metrics was enabled from the Dashboard page.
- In some rare cases, real-time page hanged on switching between real-time sessions.
- Option to plot values based on data collection in History/Trend analysis (For uptime counters). Only Group by option was available for those counters.
- SQL DM for MySQL failed to recognize Aurora instances for OS monitoring. The four required fields for OS monitoring were not displayed.
- Email addresses with + in monitor level notification settings was not accepted.
- Added ‘DB’ as a column under ‘Manage Columns’ in Dashboard’s ‘show details’ page in ‘Processlist’ mode
- SQL DM for MySQL showed the total time and average latency as 0 when the execution time of the query was less than a second for table based slow query log.
- Fixed an issue with Custom SQL Object (CSO) - SQL DM for MySQL logged entries like “not an error” in MONyog.log.

1.2.10 8.2.0 SQL DM for MySQL (August 2017)

1.2.10.1 New Features:

- Added option to have same Y-axis scaling for Dashboard charts across all servers.

1.2.10.2 Fixed Issues:

- Query count could return negative value in Overview page due to integer overflow.
- The History/Trend analysis chart for the counter “Free disk space” was not rendering properly.
- The Monyog API for enabling System metrics for RDS instances was not working as expected.
- After zooming into the real-time chart, the Query column was not getting sorted.
- History/Trend could show wrong value for some counters – mainly string based counters.
- In case of RDS servers, rebuild of a server’s database, restarting SQL DM for MySQL or using “editserver” API could stop the collection for OS metrics.
- When using cname for RDS servers, SQL DM for MySQL failed to enable system metrics.

1.2.10.3 Miscellaneous:

- Redesigned Time-selector for all the features.
- Upgraded OpenSSL to current v1.0.2l. The version used before had a few non-critical security issues. Also quite a lot of libraries used internally were updated.

1.2.11 8.1.1 SQL DM for MySQL (July 2017)

1.2.11.1 Fixed Issues:

- Regression bug introduced in SQL DM for MySQL-8.1.0: SQL DM for MySQL failed to install on Linux machines with no “kernel headers” installed

1.2.12 8.1.0 SQL DM for MySQL (July 2017)

1.2.12.1 New Features:

- It is now possible to get OS metrics from Amazon RDS/Aurora (but not Azure, where interface for same is disabled).
- Added an option to generate a token in SQL DM for MySQL to be used with the Monyog API as an alternative to SQL DM for MySQL user and password.
- Added an option to define a “seconds_behind_master” setting in Replication page determining if the slave should be considered in sync or not. On some environments, slave is rarely fully in sync and in such cases, the alerting was not really useful before.
- MONyog-bin.pid location on Linux can now be changed from Monyog.ini file.
- Reintroduced the tabular view for Replication overview page. Now, the user can choose between tabular and graphical view.
- Added an option to select the way the server selector should work in SQL DM for MySQL, i.e: same servers across all pages or different across pages.

1.2.12.2 Fixed Issues:

- Seconds behind master could show wrong value in the chart.
- Export as CSV didn’t work for “Locked and Locking Queries” in Real-Time.
- On HDMI and higher resolution monitors, overview page could display distorted.

- Extra columns added from manage columns in Real-Time were getting reset to default set of columns on moving away from the page.
- In Dashboard the value of min/max/avg displayed as 0 if number of points was larger than 10,000, Also, if the chart was exploded, the chart would display empty.
- Some counters did not show values formatted properly.
- In mail alerts for counters in multi-replication monitor group a newline character in subject could cause the mail to display unformatted in some mail clients – including gmail web interface.
- In “long running query” alert there was a bare line feed character (\n) which could cause reading failures in some environments.
- History trend reports will show date as well in the timestamp if report spans more days.
- Collection and Purging interval was not setting properly while editing a UDO.

1.2.12.3 Miscellaneous:

- Redesigned the interface for adding new servers.
- It is now possible to display the “Replication Overview” page and set auto-refresh interval similar to SQL DM for MySQL before version 7.0. Different users have different preferences here.
- When installing SQL DM for MySQL on a Linux machine with kernel version less than 2.6.32, an error message will now inform that SQL DM for MySQL needs kernel version 2.6.32 or higher.

1.2.13 8.0.4 SQL DM for MySQL (Jun 2017)

1.2.13.1 Fixed Issues:

- In rare cases, Replication Overview page failed to load and a console error appeared.
- SQLite files used by Real-time were not always deleted when they should, resulting in unnecessary disk usage.
- Enabling the error log, slow query log and general query logs through the API for RDS server failed.
- Fixed a crash when connected to Azure MySQL-5.7 servers.
- Some columns were not getting sorted properly in Sniffer.

1.2.13.2 Miscellaneous:

- When stopping SQL DM for MySQL service, all running Real-time sessions will now be saved automatically.
- Upgraded LibSSH and LibCurl to the latest versions (0.7.5 and 7.54.0 respectively).
- SQL DM for MySQL will now only support Linux’es with kernel version equal to or higher than 2.6.32 . This is a consequence of an upgrade of our build environment necessary in order to support recent kernel versions and distributions.
- Upgraded the MariaDB Connector/C to 2.3.3. Due to this the MariaDB auth_gssapi (Kerberos) plugin is now supported.

1.2.14 8.0.3 SQL DM for MySQL (May 2017)

1.2.14.1 New Features:

- Option to export entire Dashboard charts.
- Option to search tags in Server selectors.
- Completely re-designed Servers page.

1.2.14.2 Fixed Issues:

- In monitors page, history chart did not plot for some counters.
- In Firefox browser, Servers page hanged with a large number of servers.
- SQL DM for MySQL failed to connect over SSH tunnel for Ubuntu-16.04.
- If 'show_compatibility_56' is off for MySQL-5.7 servers "Slave_running" counter gave a false alert. It will display "(n/a)" now.

1.2.14.3 Miscellaneous:

- API added to change the mode of Real-time Monitoring.
- Enhanced security measures to avoid vulnerabilities like clickjacking, phishing etc.

1.2.15 8.0.2 SQL DM for MySQL (April 2017)

1.2.15.1 New Features:

- Added Dashboard charts and monitors for the Aria storage engine.
- Performance schema can now be used with RDS Aurora (version 5.6.10).

1.2.15.2 Fixed Issues:

- SQL DM for MySQL failed to recognize replication topology in cases when "SHOW SLAVE STATUS" returned master host as an alias name while the master server was registered with an IP address in SQL DM for MySQL.
- Also SQL DM for MySQL could fail to recognize replication topology if SSH-tunnel was used to master.
- The replication monitors stopped updating values, when slaves stopped running as slaves.
- When logging in with LDAP group users, the onboarding tool-tips continued to display when no longer needed.
- Threads page stopped updating data once the custom query filter returned empty list.
- SQL DM for MySQL did not show queries starting with "SHOW.." in slow query log in Query Analyzer.
- "Lock time" column was not getting sorted properly for PS based real-time.
- SQL DM for MySQL returned console error for server name containing single quote(s).
- Deleting server registrations in SQL DM for MySQL could result in high CPU-load and IO. This is now reduced.
- SQL DM for MySQL RPM installer script failed to start SQL DM for MySQL in SUSE Linux Enterprise servers.

1.2.16 8.0.1 SQL DM for MySQL (April 2017)

1.2.16.1 Fixed Issues:

- Servers below MySQL-5.1.15 were shown disconnected even when the servers were connected, due to the absence of information_schema.plugins table.
- In IE 11, servers page gave console error and became unresponsive.

1.2.17 8.0.0 SQL DM for MySQL (March 2017)

1.2.17.1 New Features:

- Added support for Multi-Source Replication.
- SNMP v2 is now supported.
- Added new Monitors for a number of plugins (Failed to load plugin list, Non mature plugin, Audit log enabled, Audit log load_option FORCE_PLUS_PERMANENT, Password verification enabled).

1.2.17.2 Fixed Issues:

- The Galera counter “Total number of bytes replicated to other nodes” now uses “wsrep_replicated_bytes” and not “wsrep_replicated” variable as before.
- The Logrotate script shipped with SQL DM for MySQL failed to rotate the SQL DM for MySQL log in some cases.
- In some cases Top-5 tables in Disk Info page were not showing tables sorted based on total size.

1.2.17.3 Miscellaneous:

- Overview will now show top 10 queries only for the selected servers and not for all the registered servers in SQL DM for MySQL.
- Overview, Dashboard, Monitors, Threads and Server Config (Compare Config) pages will have their own separate server selector.
- Now the maximum no of servers that can be selected in each page is limited. Default is 10, but the number can be configured in Monyog.ini. On most systems we don’t advise raising the value as it may increase the time for loading pages.

7.x SQL DM for MySQL

1.2.18 7.0.5 SQL DM for MySQL (March 2017)

1.2.18.1 Fixed Issues:

- Fixed a crash occurring when upgrading SQL DM for MySQL from Ultimate/Trial to Enterprise/Professional.
- In rare cases dashboard failed to load charts.
- In some cases SQL DM for MySQL could fail to analyse the sniffer and returned error “Integer overflow” instead.
- Additional InnoDB deadlock information from SHOW ENGINE INNODB STATUS available in MySQL 5.6+ was not displayed in Monitors page.
- Fixed an issue where log analysis wasn’t working with Aurora instances.
- RDS servers were not being detected as RDS instances because of change in ‘basedir’ variable for RDS.
- In Monitors page, it could happen sometimes that some charts were not displayed in ‘history’ timeframe.
- The ‘Minimum time taken’ -setting for Processlist-based sniffer also affected Performance_Schema-based sniffer. This is wrong and has been corrected.
- In some cases the Overview page loaded slowly if the request for checking any unresolved counters, functions or UDOs responded late.
- Login to Monyog failed if the password had any of the characters “%”, “&”, “+” and “?”.

- A few UI/display fixes.

1.2.19 7.0.4 SQL DM for MySQL (February 2017)

1.2.19.1 Fixed Issues:

- In rare cases SQL DM for MySQL would hang while editing a server.
- Enabling “Monitor locked queries” switched back to off state after editing settings for the actual MySQL server.
- Regression fix: Monyog-v7.x releases had a higher load average than 6.x predecessors.

1.2.19.2 Miscellaneous:

- When upgrading, migration will be much faster.
- Improved Dashboard charts loading speed.

1.2.20 7.0.3 SQL DM for MySQL (February 2017)

1.2.20.1 New Features:

- Dashboard charts now load faster.

1.2.20.2 Fixed Issues:

- When editing an existing LDAP group/user, changes were not getting saved in some cases.
- When using proxy-based Query Sniffer, it could happen that some queries were not displayed in the Query Analyzer.

1.2.21 7.0.2 SQL DM for MySQL (January 2017)

1.2.21.1 New Features:

- Added two new monitors for monitoring Binary Log volume in Linux monitor group. Please note that this will work only for MySQL 5.6.2 and above.

1.2.21.2 Fixed Issues:

- For RDS instances, the Error Log files were not getting fetched. This was introduced in 7.0.
- When upgrading from a pre-7 release, LDAP settings could be overwritten. Also this was introduced in 7.0.*
- Queries containing ‘<’ or ‘>’ were not displayed properly in Overview, Real-time and Query Analyzer pages. Instead the HTML-codes ‘&lt’, ‘&gt’ displayed. Also a 7.0 bug.
- When running 7.01 as a non-registered TRIAL some pages could ‘hang’ in rare cases. 7.0 was not affected.

1.2.22 7.0.1 SQL DM for MySQL (January 2017)

1.2.22.1 Fixed Issues:

- In some cases overview page loaded slowly and would even sometimes seem to remain in “Fetching details” state forever.
- Security: Fixed an issue where a user could escalate his privileges by manipulating cookies. Though fixed already in 7.0, this is listed here for completeness.

1.2.23 7.0 SQL DM for MySQL (January 2017)

1.2.23.1 New Features:

- Overview page giving the top-level picture of all the servers running, down, critical alerts, warnings and the top 10 queries across all the servers registered with SQL DM for MySQL.
- Option to create customized dashboard with specific charts with the ability to explode a chart
- Search option for servers registered with SQL DM for MySQL.
- Query Analyzer will now show the top 5 queries based on execution time in a separate window.
- Replication overview page showing the replication hierarchy in a clean graphical way.
- Servers page providing an overview of all the servers registered and their status in one place.

6.x SQL DM for MySQL

1.2.24 6.63 SQL DM for MySQL (October 2016)

1.2.24.1 New Features:

- Monyog API now supports LDAP users and groups.

1.2.24.2 Fixed Issues:

- In rare cases SQL DM for MySQL failed to connect to SSH and returned the error “failed to connect to SFTP: SshConnect: Error while connecting to ssh server –kex error”.
- SQL DM for MySQL failed to authenticate as required when using the authentication mode “Comparison” for LDAP users and groups.
- Some galera charts were not loading due to the syntax error in the chart definition.
- SQL DM for MySQL was allowing users to add/edit the server using the API even if the user did not have privilege to add/edit the server.

1.2.25 6.62 SQL DM for MySQL (September 2016)

1.2.25.1 New Features:

- Charts are now loading faster with large time range.
- Added more pre-configured Galera charts and Performance_Schema Custom SQL Objects.

- Improved the performance of SQL-query in CSO 'Tables_Without_Constraints'.
- Added a new API "_disabledmonitorgroups" for enabling and disabling Monitor Groups at server-level.

1.2.25.2 Fixed Issues:

- Support for the 'clear-text plugin' (used by Oracle/MySQL LDAP authentication) and the 'dialog plugin' (used for MariaDB PAM authentication) was broken.
- With RDS servers SQL DM for MySQL could fail to detect log file content and size and would display the error "No queries found" and "Unable to fetch file size". It happened because of recent changes in the XML-response of Amazon's REST API.
- Slow query log/General query log paths were not always setting properly when adding/editing a server using the Monyog API.
- MySQL InnoDB-related charts displayed empty with MySQL 5.6+. The reason was removal of the 'have_innodb' server variable in MySQL 5.6. Now these charts don't depend on this variable.

1.2.26 6.61 SQL DM for MySQL (September 2016)

1.2.26.1 Fixed Issues:

- When user logged in to SQL DM for MySQL with an account other than the built-in admin account, tabs were hidden.

1.2.27 6.6 SQL DM for MySQL (September 2016)

1.2.27.1 New Features:

- SQL DM for MySQL user management was enhanced with a 'roles' feature.
- LDAP role based access: users can now map External LDAP roles to SQL DM for MySQL roles.
- Added option to restrict access to specific tabs for specific users.
- SNMP traps will now have "Tag" and the "MySQL hostname" in the message.
- Now SQL DM for MySQL will show the number of servers selected in the tag along with the total servers in that tag.
- Added a message in the charts page if system charts are enabled and SSH settings are not enabled.

1.2.27.2 Fixed Issues:

- When adding a server through Monyog API, sniffer was not enabled by default.
- The setting for "Override send notification when alert-able?" was ignored if the value specified was greater than the setting for "Send notification when alertable".
- In rare cases Real-Time failed to start a new session and showed a random number.

1.2.28 6.55 SQL DM for MySQL (August 2016)

1.2.28.1 Fixed Issues:

- The “Explain Query” dialog box in Real Time and Query Analyzer returned the message “Extended is deprecated” for MySQL 5.7.3+ servers. Now (plain) “EXPLAIN” is used to fetch the value.
- In rare cases some monitor groups were not getting Enabled/Disabled at “Manage Monitor groups” page when using Internet Explorer.
- The fields “SeriesCaption” and “SeriesValues” in “Edit chart” page did not give any error on saving when having different number of arguments, but a console error showed up while loading the charts.
- SQL DM for MySQL was not able to connect to RDS instances using SSL encryption. Attempting to do so returned the error “Failed to connect to MySQL: SSL connection error: certificate verify failed”.

1.2.28.2 Miscellaneous:

- The option “Notify when server configuration changes” will be enabled by default when notification is enabled.
- Now counters may be customized from “Events” tab as well.

1.2.29 6.54 SQL DM for MySQL (July 2016)

1.2.29.1 Fixed Issues:

- Monyog.log file logged bogus SQLite errors like “not an error” if UDOs were enabled in SQL DM for MySQL in some cases.
- The definition for the counter “Free disk space” in the “Linux” group was misleading as we were giving the free disk space of the volume where MySQL binary resided given by “basedir”, instead of giving the free disk space on the volume where data directory resides given by “datadir”.
- The column “Rows examined” in Performance_Schema based sniffer printed negative values, if the number of “Rows examined” exceeded the value 2G.
- In Real-Time, charts for Tables tab were not displayed in some rare cases.
- We are now discarding GTID-related system variables that are changed by MySQL server itself from the “Track Config” tab. These system variables (gtid_executed, gtid_owned, gtid_purged etc.) change if a new transaction occurs in Master and considering them would raise bogus alerts that configuration was changed.

1.2.29.2 Miscellaneous:

- SQL DM for MySQL will now have sniffer enabled by default for newly registered MySQL servers. If Performance_Schema is enabled for the MySQL server then Performance_Schema mode will be used, else Processlist mode will.
- Improved the load time of Monitors page. It is noticeable in particular when “History” timeframe is used with a large range (several hours or more).

1.2.30 6.53 SQL DM for MySQL (April 2016)

1.2.30.1 Fixed Issues:

- “Count” column did show empty value in Processlist based Real-time.
- Performance_Schema based Real Time failed if global sql_mode included “ONLY_FULL_GROUP_BY”.
- Clicking on locked queries were not showing its corresponding locking query and vice versa for Performance_Schema based Real Time.
- The counter “General Log Enabled?” was showing incorrect value for MySQL 5.6.1+ versions.
- In rare cases, SQL DM for MySQL would show only time with no date in x-axis if the chart was plotted for multiple days.
- Editing LDAP group was not always working as expected.
- With RDS servers SQL DM MySQL could fail to detect log files and would display the error “log file not present”. It happened because of changes in the XML-response of Amazon’s REST API introduced since we started supporting logs with RDS.

1.2.30.2 Miscellaneous:

- The default filter for “show queries only” in processlist now includes ‘Command=Execute’. Also ‘Command=Execute’ is now included in processlist-based Real Time and Query Analyzer interfaces.

1.2.31 6.52 SQL DM for MySQL (March 2016)

1.2.31.1 New Features:

- Added an option to choose data collection mode for Real-Time, Users can now choose between ‘Performance schema’ or ‘Processlist’. Old versions used ‘Processlist’ what we replaced with ‘Performance schema’ recently for servers running with needed P_S instruments enabled. However due to the aggregation and truncation of queries recorded in P_S we had several requests that both options should be there.
- Information whether a query is performing a full table scan available in Query Analyzer, Wayback Machine and Real-time.This information will help to identify queries which are not using Indexes. Note: this is available only for Performance Schema mode.
- Added a new monitor group ‘sys schema’. It contains various CSOs (Custom SQL Objects) utilizing the “sys schema” database introduced in MySQL 5.7.7.
- Added an option to either have a fixed length height or dynamic (full screen) height for Processlist in SQL DM for MySQL GUI. We have been using each in earlier versions, but it seems impossible to make everyone happy with a single solution. Now user can choose what display/design that fits him best.

1.2.31.2 Fixed Issues:

- The Monyog API failed to set the error log path.
- With MySQL 5.7+ Query Analyzer was displaying seemingly random values for the column “first seen” and “last seen” when using server logs (slow query log, general query log) as data source. This happened because of change of Timestamp format in the logs in MySQL 5.7 .
- The filter “doesn’t contain” was not working properly in Real-time.
- Fixed an issue where UDOs were displaying incorrect values after running for some time.
- While editing the server using Monyog API, SSH settings were automatically set to disabled (even if it was enabled earlier).

- For RDS/Aurora servers, “Apply the above settings to all the servers with same tags” (Edit server -> Advanced settings) was not working.
- SQL DM for MySQL was giving the error “Log file not present” if either one of the slow query log or general query log was disabled for RDS servers.
- Fixed JavaScript errors and optimized the charts for Real-time page for IE8 browser.

1.2.31.3 Miscellaneous:

- The Y-axis in Charts had just “values” written, now it shows relevant units like percentage, seconds etc. and option is provided to customize the Y-axis value.

1.2.32 6.51 SQL DM for MySQL (December 2015)

1.2.32.1 New Features:

- RDS log monitoring feature can now be configured using the Monyog API.

1.2.32.2 Fixed Issues:

- The query SHOW ENGINE INNODB STATUS fails for MySQL version less than 4.1.2. and resulted in a SQL DM for MySQL crash. Even though we don’t recommend MySQL 4.1 (and not at all such early beta of it), we should not crash.
- Fixed JavaScript errors for Query Analyzer and Real Time tab in Internet Explorer 8 (WinXP, Win2003).
- With MariaDB 10.1.x SQL DM for MySQL erroneously listed all user accounts as having no password.
- The curl library used could crash, and thus crash SQL DM for MySQL, when resolving too many hostnames at the same time.
- SQL DM for MySQL failed to send 'stable alert' when both "notify till stable" and "notify when stable" options were enabled and if different values were set for the "send notification when alertable" and "remind me after every ..".
- Fixed the height of the Processlist display for each server. Each server’s processlist window is now displayed as scrollable window of fixed height. This is a regression fix and a revert of the page design to that of older versions. With recent versions it was necessary to scroll the browser window.

1.2.32.3 Miscellaneous:

- Upgraded SQLite library to v3.9.1. This results in ~25% performance improvement on most environments.
- Renamed the “Row Access Statistics” counter to “Queries Executed” as it actually uses the ‘queries’ status variable. The old term confused some users.

1.2.33 6.5 SQL DM for MySQL (November 2015)

1.2.33.1 New Features:

- On Amazon RDS and Aurora, MySQL log files (Slow Query Log, General Query Log and Error Log) can now be read by SQL DM for MySQL. So Error Log monitoring is now possible here, and the Query Analyzer does not require logging to tables anymore. For this the Amazon “REST” API (and not SFTP) is used. Client functionalities for this API have been integrated into SQL DM for MySQL.

- Added monitors for Galera: “Presence of non-InnoDB tables” and “Tables without a primary/unique key”.

1.2.33.2 Fixed Issues:

- Monitors and Real-time page failed to load on Internet Explorer 8 (highest IE option on WinXP) due to JavaScript errors.
- Fixed a rare crash when connected to MySQL 5.7.6+.
- Accessing Realtime interface could cause a crash if connection to the server was not available.

1.2.34 6.4 SQL DM for MySQL (October 2015)

1.2.34.1 New Features:

- Performance Schema is now available in Realtime. SQL DM for MySQL can now capture all queries and display it in realtime which was not possible earlier with SHOW FULL PROCESSLIST output. To view all queries in realtime, make sure Performance_schema and ‘statements_digest’ table is enabled. If not, SQL DM for MySQL will display the output of SHOW FULL PROCESSLIST in realtime. Note: Performance_schema is available with MySQL 5.6.14 and above.

1.2.34.2 Fixed Issues:

- Users with MySQL v5.6.3+ will now see explain plan for Insert, Update and Delete queries in Wayback Machine and Query analyzer.

1.2.35 6.34 SQL DM for MySQL (September 2015)

1.2.35.1 New Features:

- Added more monitors and charts for monitoring a galera cluster. By default, these monitors are hidden. To display these monitors in Monitors page you can enable “Galera” monitor group from Customize -> Manage Monitor Groups. Charts for these new monitors can be enabled from Customize – > Manage charts.
- Optimized system charts for faster loading.

1.2.35.2 Fixed Issues:

- EXPLAIN option was not available in Wayback Machine.
- In some rare cases, SQL DM for MySQL logged SQLite errors if deadlock details contained double quotes.
- In Wayback Machine interface, SQL DM for MySQL showed wrong time in queries tab when a point was selected in the chart.

1.2.36 6.33 SQL DM for MySQL (July 2015)

1.2.36.1 New Features:

- Added monitoring of the MySQL –tmpdir volume with the “Free Disk Space” Monitor for Linux (introduced in 6.32).

- Added an option to view X-Axis labels for large time based charts.
- The Performance_Schema -based “query sniffer” has been optimized.

1.2.36.2 Fixed Issues:

- Fixed a rare case, where “Charts” page was not loading properly.

1.2.37 6.32 SQL DM for MySQL (July 2015)

1.2.37.1 New Features:

- Added a “Free Disk Space” Monitor for Linux. This new feature will collect information directly from the file system (unlike the existing ‘Disk Info’ that uses MySQL “SHOW TABLE STATUS”). Information is collected from volumes storing both MySQL data directory and InnoDB data directory (if different), and user will be alerted if either crosses a specified threshold.
- In Charts page added an option to select a time interval for display by zooming a chart using mouse and similar input devices (touchpad etc.). Note that zooming one chart will change the timeframe of all the charts displayed.
- Added an option to view queries for a selected time range in time range based charts in Charts page.

1.2.37.2 Fixed Issues:

- In rare cases, Monitors page could show wrong data for Linux counters if the HISTORY timeframe was selected for the page. Other available timeframes were not affected.
- When monitoring a MariaDB server using a galera-based replication setup we will now execute SHOW ALL SLAVES STATUS rather than just SHOW SLAVE STATUS. Before this Monyog replication page showed an empty result and alerts were not sent. However due to limitations with current SQL DM for MySQL architecture we are only able to handle the first row of SHOW ALL SLAVES STATUS with this release, if more are returned, and only information from one slave node will be available.

1.2.38 6.31 SQL DM for MySQL (June 2015)

1.2.38.1 New Features:

- What is plotted in “Charts” (formerly “Dashboard”) page may now be specified as a time interval and not only as a number of latest sample points. This increases usability of the page.
- Long running query log now includes action (kill/notify/notify and kill) and the time of action.
- Locked and locking queries tab in Real-Time interface is optimized and loads faster.
- Added Wildcard filtering using the “*” character as wildcard for user and host in Query Analyzer for Slow Query Log.

1.2.38.2 Fixed Issues:

- SQL DM for MySQL failed to send notification for an InnoDB deadlock if deadlock details (as recorded by SHOW ENGINE INNODB STATUS) contained double quotes.

1.2.39 6.3 SQL DM for MySQL (May 2015)

1.2.39.1 New Features:

- A log of long running queries can now be generated and downloaded as CSV in Sniffer interface (applies only to Processlist mode of sniffer).
- Real-Time is optimized and loading a session is much faster now. Further it shows only data from top 200 open tables and databases.
- Wayback Machine now includes user and host information as well.
- Queries in Wayback Machine can now be exported as CSV.
- SQL DM for MySQL can now send long running query notification to specific selected users only.
- Wildcard filtering using the "*" character as wildcard for user and host is introduced in Sniffer (applies only to Processlist mode of sniffer).
- Same wildcard filtering implementation is also introduced for Ignore user and host in Long running query options in Sniffer (applies only to Processlist mode of sniffer).
- SQL DM for MySQL now logs what user killed a thread in Processlist tab.
- SQLite library bundled in SQL DM for MySQL is now upgraded to the latest version. This results in performance improvement for sniffer in particular where it can be up to ~50%.

1.2.39.2 Fixed Issues:

- In some cases, SQL DM for MySQL logged SQLite errors when CSOs (Custom SQL Objects) were enabled even when the CSO worked as expected
- Small UI fixes.

1.2.40 6.27 SQL DM for MySQL (April 2015)

1.2.40.1 Fixed Issues:

- SQL DM for MySQL no longer prompts the user to re-type the admin password while upgrading on Windows.
- In Edit Server page it was possible to retrieve the MySQL password by inspecting JSON with browser's JavaScript debugging module/plugin.
- SQL DM for MySQL was not able to authenticate a user from LDAP group consisting of linked users (i.e. users with membership of more than one group).
- Monitors page failed to load in case an empty group (with no counters) existed.
- Replication tab failed to load after restart of a server when monitoring a master-master replication setup.

1.2.41 6.26 SQL DM for MySQL (March 2015)

1.2.41.1 Fixed Issues:

- When monitoring MySQL 5.6.3 and higher, SQL DM for MySQL reported false notification alert for 'innodb_thread_sleep_delay' when notifications for change in server configuration was enabled.
- Replication page failed to show data if SHOW SLAVE STATUS on one server was hanging.
- Server Config page was not always showing proper data when "show only changed values" is selected.
- In some rare cases Monitors page was not displaying data after editing SSH tunnel details.

- SQL DM for MySQL 32 bit build was not accepting port beyond 16-bit limit (> 2^16 – 1). This limit has been increased to handle full 32-bit range (up to 2^32 – 1).

1.2.41.2 Miscellaneous:

- Email Alerts in SQL DM for MySQL now contain Tag(s) and MySQL host of the corresponding server.

1.2.42 6.25 SQL DM for MySQL (February 2015)

1.2.42.1 New Features:

- Optimized Monitors page to populate faster. In particular in history/trends timeframe this is a dramatic performance improvement, as it will now populate 50 or more times faster than before with large data series.

1.2.42.2 Fixed Issues:

- An incorrect (JavaScript) monitor definition could cause a yellow bar-shaped area to appear instead of a proper error in Monitors page for the affected counter.
- Setting 'Notify Till Stable' using the Monyog API was not working as expected.
- Fixed a rare crash with a large number servers monitored. It would require more than 300 servers registered for this crash to occur on an average system.
- Minor UI fixes.

1.2.43 6.24 SQL DM for MySQL (February 2015)

1.2.43.1 Fixed Issues:

- SQL DM for MySQL was resetting "Monitor long running queries" to NO after editing a server.
- SQL DM for MySQL failed to show servers in the UI after upgrading from a pre-5.57 version.
- "Critical", "Warning" and "Stable" icons were not showing in Events tab when SQL DM for MySQL was running behind a proxy.
- SQL DM for MySQL sometimes logged an SQLite locking protocol error while opening internal database(s). But as this is handled it should not be logged as an error.
- SQL DM for MySQL allowed for saving of server registration details when "Long running query time" was specified smaller than "Minimum time taken" in sniffer settings. This makes no sense and will not save now.
- SQL DM for MySQL was not considering buffers and cached values from meminfo while calculating RAM usage. The formula for the same is updated now.

1.2.43.2 Miscellaneous:

- Charts are now enabled by default for all monitors which include MySQL status variables. For example: "Seconds behind master" in replication group.

1.2.44 6.23 SQL DM for MySQL (January 2015)

1.2.44.1 New Features:

- Added support for the ‘clear-text plugin’ (used by Oracle/MySQL LDAP authentication) and the ‘dialog plugin’ (used for MariaDB PAM authentication). There is no user setting for either. SQL DM for MySQL will transparently for user detect how to communicate with the server during authentication.

1.2.44.2 Fixed Issues:

- When data collection was stopped for a server then, in some cases, user was not able to delete that server.
- In ‘Wayback Machine’ interface, the plotted values were sometimes not visible due to how axes were scaled in the chart displayed.
- If data collection for a server was stopped, then after restarting SQL DM for MySQL.and enabling data collection for that server, the values displayed in the Monitors tab were not getting updated in some cases.

1.2.44.3 Miscellaneous:

- UI changes for ‘long running queries’ and counters with CSO disabled to make it more intuitive how to fully enable those features.
- SMTP and SNMP alerts will now as default be sent first time an alertable counter has exceeded the ‘critical’ setting.
- Formula of a monitor will now be displayed in event detail dialog in Events tab.

1.2.45 6.22 SQL DM for MySQL (December 2014)

1.2.45.1 New Features:

- SQL DM for MySQL can now resend alert notifications till a counter becomes stable based on a reminder interval setting.

1.2.45.2 Fixed Issues:

- 'Upgrade check' link in 6.21 informed that an "undefined" upgrade was available even though it was not the case.
- Performance-schema based sniffer failed if global sql_mode included "ONLY_FULL_GROUP_BY".

1.2.46 6.21 SQL DM for MySQL (November 2014)

1.2.46.1 Fixed Issues:

- In Monitors page, the small trend charts in current and delta timeframes were not displayed for some monitors.
- Charts were not displayed in Monitors page- History/Trend if data series contained "N/A" values.

- SQL DM for MySQL was not able to connect to MySQL server if password contained a leading or a trailing empty space.
- In rare cases, some server(s) could stop showing in the UI after SQL DM for MySQL restarts.
- Fixed a rare irresponsiveness ("hang") occurring when starting data collection after editing server details.
- The option to stop the current running saved session was not available in Real-Time in some cases.
- A number of minor GUI fixes.

1.2.47 6.2 SQL DM for MySQL (October 2014)

1.2.47.1 New Features:

- The Real-Time interface in SQL DM for MySQL has been completely rewritten. In addition to a more consistent/intuitive GUI it is now able to collect data from multiple MySQL servers simultaneously (one collection of Real-Time data per server). Once data collection for a server has been started it will continue until stopped by user, also if browser connection is closed in between. The sole exception is that Real-Time data collection for a server will stop after 3 days if no activity has happened inside the particular Real-Time session.
- Added an option to set line terminator for csv reports.

1.2.47.2 Fixed Issues:

- Renaming a registered server to a new name already in use would not raise any error. Instead data for the affected server would not display anymore.
- In rare cases clicking 'test SSH settings'-button would not display any response.
- When registering a 'Postmaster Enterprise SMTP server' in SQL DM for MySQL, mail notifications failed to be sent.
- 'Diagnostics report' failed to download when SQL DM for MySQL was running behind a HTTP ('reverse') proxy server.
- Clicking on editing SMTP settings from server registration page did not redirect to tools page and thus editing was not possible from here. The problem was alone a problem with the particular hyperlink. From the tools page everything worked as expected.
- The following gtid-related variables in MariaDB 10 could trigger false notifications: gtid_binlog_pos, gtid_binlog_state, gtid_current_pos, gtid_slave_pos and last_gtid.
- SQL DM for MySQL failed to establish SSH connection to Linux servers using the "OpenSSH-HPN" SSH implementation..
- When an identifier contained digits (such as 'abc12', 'ab12ab' etc.), the Query Analyzer, the WayBack Machine and Real-TIme interface would consider these digits literals, and they would be replaced with '?'.
- If SQL DM for MySQL did not have sufficient privileges to the MONyog/Data folder, adding a new server could crash SQL DM for MySQL.

1.2.47.3 Miscellaneous:

- Dashboard page was renamed to 'Charts'.
- The tab-order of SQL DM for MySQL interface has been rearranged.
- Also note that 'WayBack machine' interface is still available (unlike what was the case with the 6.2 beta, where we experimentally removed it.

1.2.48 6.1 SQL DM for MySQL (June 2014)

1.2.48.1 New Features:

- Added more counters and advisors for MySQL 5.6 configuration.
- Added an extensive set of counters for monitoring the TokuDB storage engine. The monitor groups for this are not visible as default, but can be enabled from the 'Customize' interface

1.2.48.2 Fixed Issues:

- Fixed a rare SQLite database corruption under very heavy load and with Custom SQL Objects enabled.
- Changing refresh interval setting for Processlist page was not possible for non-admin users.

1.2.48.3 Miscellaneous:

- SQL DM for MySQL installed on the Amazon Cloud from the Amazon Machine Image ("AMI") we provide as an option, will now start first time with the Amazon instance ID as the admin password and not an empty password as before. This change could take some days to propagate across the Amazon Cloud.

1.2.49 6.02 SQL DM for MySQL (June 2014)

1.2.49.1 Fixed Issues:

- CSV exports from Query Analyzer did not include the 'Db' (database) column when the Query Analyzer was used with Processlist-based sniffer or with slow log when stored in a table.

1.2.50 6.01 SQL DM for MySQL (June 2014)

1.2.50.1 Miscellaneous:

- Cleaned up the User Interface in Real-Time page. This includes removing the option to restore saved sessions for 'all selected servers'. Saved sessions for individual servers can still be restored.
- OpenSSL is updated to version 1.0.1h in all packages distributed.

1.2.51 6.0 SQL DM for MySQL (June 2014)

1.2.51.1 New Features:

- The Monyog API is enhanced with more options to manage servers registered in SQL DM for MySQL. Options now include calls to add/edit/remove servers. Refer documentation for full details.
- SQL DM for MySQL now supports Query analysis using Performance Schema tables (events_statements_summary_by_digest and events_statements_history_long tables). This feature is supported only for MySQL 5.6.14 and MariaDB 10.0 and above.
- Explain plan is available in Query analyzer for Slow_log table based logging and Processlist sniffer.

- An example query is now shown in detailed query view in Query analyzer if queries replaced with literals is selected.
- SQL DM for MySQL now has a configurable option to define the max. query length displayed. Beyond this SQL DM for MySQL will truncate. The setting has effect for Query Analyzer, Real-time and Way Back Machine interfaces. Default is now 10000 characters and maximum setting is 64000 characters). However please note that this does not apply when Performance_Schema is used with the Query Analyzer as P_S itself truncates.

1.2.51.2 Fixed Issues:

- Explain option was not available in locked and locking queries for Real-Time sessions.
- In some rare case SQL DM for MySQL logged SQLite errors while running Real-Time sessions.
- SQL DM for MySQL could crash when connecting using a 'Named Pipe' on Windows.
- When monitoring MySQL v5.6 with replication using GTIDs (global transaction IDs), SQL DM for MySQL reported false notification alert for the 'gtid_purged' variable, when notifications for change in server configuration was enabled. This particular variable was overlooked in 5.72 where there was a similar fix.
- Query literals with hex values were not replaced with '?' in Query analyzer.
- When using SSH-tunnel, connections created by the CSO (Custom SQL Object) feature were not always closed properly, and the MySQL error log as well as 'aborted_clients' status variable would record aborted connections.

1.2.51.3 Miscellaneous:

- When GTID's (Global Transaction ID's) were used with replication, SQL DM for MySQL replication page would display a lot of information about changing GTIDs. This is now hidden in the primary interface, but can still be displayed in detailed view.
- SQL DM for MySQL now displays the binary version on commandline when the command switch -version (or -v) is used.

5.x SQL DM for MySQL

1.2.52 5.72 SQL DM for MySQL (April 2014)

1.2.52.1 Fixed Issues:

- In rare cases, SQL DM for MySQL could not close alerts from Monitors page.
- When monitoring MySQL v5.6 with replication using GTIDs (global transaction IDs), SQL DM for MySQL reported false notification alert for GTID counters when notifications for change in server configuration was enabled.

1.2.52.2 Miscellaneous:

- Upgraded libraries that theoretically could be vulnerable to the 'Heartbleed' OpenSSL bug.

1.2.53 5.71 SQL DM for MySQL (March 2014)

1.2.53.1 New Features:

- With only one MySQL server selected, Dashboard page will now display charts in such a way that whitespace and the need of scrolling are reduced. With more than 1 servers selected, the Dashboard charts are drawn using a vertical layout as before.
- Real-Time interface can now show the actual query in addition to replacing literals with the "?" wildcard.

1.2.53.2 Fixed Issues:

- Using TAGS for registered servers in order to manage access for non-admin users did not always work as expected when multiple TAGS were used.
- Yet another fix for a LDAP group authentication issue.
- In Real-Time interface EXPLAIN option was not available if query length was greater than 5000.
- In Real-Time interface database and user@host were not available for aggregating queries. They are now.

1.2.53.3 Miscellaneous:

- MySQL availability and System availability charts in Dashboard are now disabled by default. Users who want them can enable from Dashboard Customize page. We removed them primarily as they consume space in the interface and have little information that is not obvious from other charts.

1.2.54 5.7 SQL DM for MySQL (March 2014)

1.2.54.1 Fixed Issues:

- 5.69 introduced an issue causing LDAP authentication to fail in some cases.

1.2.54.2 Miscellaneous:

- Code cleanups and internal optimizations. Users having a large number of MySQL servers registered will observe a lower system load with this version due to this.

1.2.55 5.69 SQL DM for MySQL (February 2014)

1.2.55.1 Fixed Issues:

- LDAP search for user in group authentication could fail with Active Directory.
- Real-time sessions could fail under rare circumstances when two or more sessions were started at the same time.
- Fixed a crash introduced in version 5.68.

1.2.56 5.68 SQL DM for MySQL (February 2014)

1.2.56.1 New Features:

- Sniffer now shows database name in long running query notification and also in Query Analyzer interface when Processlist based sniffer is selected. Please note that Processlist only exposes this information if a USE statement is associated with the query. If a 'fully qualified tablename' is used in the query this information will be empty.

1.2.56.2 Fixed Issues:

- Monitor's trend charts and history report charts were not displayed. This was introduced in version 5.67 of SQL DM for MySQL.
- Significantly improved the speed of replacing literals for queries with wildcards in Query Analyzer.
- Connection to MySQL via SSL encryption could fail with valid parameters. This bug was introduced in version 5.61 of SQL DM for MySQL.

1.2.57 5.67 SQL DM for MySQL (February 2014)

1.2.57.1 Fixed Issues:

- Fixed a rare crash occurring in CSO's (Custom SQL Objects).
- Fixed a UI issue in Realtime's lower pane. Only Safari browsers were affected.
- SQL DM for MySQL did not retain LDAP group authentication details after restart.
- The SQL DM for MySQL interface for specifying LDAP groups did not allow for more than 64 characters.
- Query Analyzer did not show queries when time range was selected with slow log and general log tables (log files were not a problem) as the source for Query Analyzer.
- CSV-reports exported from Monitors page could show wrong values for a few monitors.

1.2.57.2 Miscellaneous:

- SQL DM for MySQL now loads list of servers interface faster.

1.2.58 5.66 SQL DM for MySQL (November 2013)

1.2.58.1 New Features:

- SQL DM for MySQL can now authenticate users using LDAP user groups without the need to add them in SQL DM for MySQL.
- Explain plan for Insert, Update and Delete queries is now available in SQL DM for MySQL Processlist page for MySQL 5.6+ servers.

1.2.58.2 Fixed Issues:

- The fix for failing alerts for InnoDB deadlocks with MySQL 5.6 servers in 5.65 introduced a new issue where the time for the occurrence of the deadlock reported was wrong.
- When monitoring a Galera cluster and when cluster nodes failed to connect to each others, the error sent by cluster nodes ("unknown command") could cause SQL DM for MySQL to crash.
- When a server was deleted from SQL DM for MySQL in rare cases the corresponding data directory was not removed.

1.2.58.3 Miscellaneous:

- Linux versions of SQL DM for MySQL now include a script that can be used for log rotation of the Monyog.log.

1.2.59 5.65 SQL DM for MySQL (November 2013)

1.2.59.1 Fixed Issues:

- Parsing the output of SHOW ENGINE INNODB STATUS for InnoDB deadlocks if server was MySQL 5.6.x. For same reason alerting also failed.
- Some counters failed to report alert conditions. This bug was introduced in version 5.62 of SQL DM for MySQL.

1.2.60 5.64 SQL DM for MySQL (October 2013)

1.2.60.1 Fixed issues:

- This release includes fixes for a rare crash when deleting servers and when trying to access a SQLite database that could not be accessed for some reason.

1.2.61 5.63 SQL DM for MySQL (September 2013)

1.2.61.1 Fixed Issue:

- If SQL DM for MySQL was not able to connect to a MySQL server, replication page would 'hang' and even crash in rare cases. This was introduced in version 5.62.

1.2.62 5.62 SQL DM MySQL (September 2013)

1.2.62.1 New Features:

- Removed a hard coded limit (of 512) of the number of MySQL servers that could be registered with an 'unlimited server'-license. Now the restriction is solely with the implementation of the license in the license key.
- Added and updated a few counters in order to provide more specific support for MySQL 5.6 and MariaDB 10.

1.2.62.2 Fixed Issues:

- Cleaned up JavaScript execution running in the background that was not needed. With a large number of servers registered (from around 150 on a typical system) this could result in some slugginess in interactive parts of SQL DM for MySQL due to CPU contention.
- Explain option was not present in processlist page for queries starting with comment.
- On Windows 'Real_time' monitoring did not work since 5.61.
- Rendering of replication page failed if GTID (Global Transaction ID) mode was enabled for MySQL Slaves.
- Resolving the IP from a host name happened multiple times during display of replication page what could slow down rendering of replication page noticeably on some environments.
- Some Dashboard charts could show incorrect data.
- Dashboard failed to render charts when based on a counter saved without caption, series caption and series values.
- When restarting SQL DM for MySQL it would perform a SQLite Schema upgrade even if it was not required. Version 5.61 was affected.

1.2.63 5.61 SQL DM for MySQL (August 2013)

1.2.63.1 Miscellaneous:

- MySQL 3.23 and 4.0 servers are no longer supported and an attempt to register such will be denied.

1.2.64 5.6 SQL DM for MySQL (August 2013)

1.2.64.1 New Features:

- SQL DM for MySQL supports LDAP authentication for login into SQL DM for MySQL interface. This is available only in Ultimate version of SQL DM for MySQL.
- Added a feature in Query Analyzer to include or exclude queries based on the search string / regular expression.

1.2.64.2 Fixed Issues:

- Monitor trend charts were not updating in realtime.
- Tooltips were truncated in DiskInfo's table information.
- In Sniffer, "Queries starting with" option was not working properly if query started with comment.
- Queries starting with a comment did not show EXPLAIN option in the Real Time interface.
- Fixed a crash when viewing replication page if slave details were not properly specified (any detail NULL).

1.2.65 5.58 SQL DM for MySQL (April 2013)

1.2.65.1 New Features:

- Added a feature in realtime to include or exclude queries based on the search string / regular expression.

1.2.65.2 Fixed Issues:

- Purging mechanism was not working properly (this bug was introduced in version 5.57 of SQL DM for MySQL).
- Tag/Group names are in alphabetical order now.
- SQL DM for MySQL can now "Kill query" on Amazon RDS instances.
- The rendering of Dashboard page is further improved.

1.2.66 5.57 SQL DM for MySQL (March 2013)

1.2.66.1 Fixed Issues:

- The rendering of Monitors and Dashboard page was slow with large number of servers both registered and selected and when multiple users were connected simultaneously displaying the page. It could take one minute or more in some cases. It is now a matter of seconds.
- With a large number of servers registered, editing details for one was slow. Also this could take around 1 minute, and the improvement is in the same range as above.
- Event manager's database (Events.data) is now moved from global level to server level in order to prevent locking problems with a large number of servers. SQL DM for MySQL will migrate data first time it starts after the upgrade.
- Fixed a multi-threading issue that in rare cases could crash SQL DM for MySQL while registering more than one server at the same time.
- Fixed a rare crash in SQL DM for MySQL while viewing replication tab.
- Fixed a rare crash in SQL DM for MySQL while resolving CSOs ("Custom SQL Objects").
- SQL DM for MySQL could fail to send mail alerts if a single mail contained alerts for more than one monitor.
- SQL DM for MySQL was showing errors in Query Analyzer, Realtime and Processlist pages when a query contained characters used for specifying HTML tags.
- If a registered server was not a slave but marked as such in Advanced Settings .. Replication, SQL DM for MySQL was leaking connections every time it was collecting data and finally connections (from SQL DM for MySQL as well as other clients) could fail due to 'max_connections' setting being exceeded.
- For MySQL InnoDB-related monitors were showing "(n/a)". The reason was removal of the 'have_innodb' server variable in MySQL 5.6.

1.2.67 5.56 SQL DM for MySQL (December 2012)

1.2.67.1 Fixed Issues:

- Fixed a memory leak when reading MySQL server logs through SFTP.
- Fixed a multi-threading issue that in rare cases could crash SQL DM for MySQL running on Linux while reading the MySQL error log.
- "Apply settings to all the servers with same tags" was not working when data collection was disabled globally.
- The fix in 5.55 for indented display of a slave compared to its master was not complete. Proper indentation could still fail.

1.2.68 5.55 SQL DM for MySQL (December 2012)

1.2.68.1 Fixed Issues:

- In the replication page a slave was not displaying indented compared to its master as it should if slave and master were running on same host.
- Fixed a multi-threading issue that in rare cases could crash SQL DM for MySQL if the Monitors page was viewed simultaneously in multiple browsers.
- Stability fixes for issues found during internal stress-testing and code review. No crash, slowdown ('hang') or similar was reported due to any of those, but it could theoretically happen under specific and rare circumstances.

1.2.69 5.54 SQL DM for MySQL (November 2012)

1.2.69.1 Fixed Issues:

- Sometimes SQL DM for MySQL could hang while editing server's tag.
- Processlist did not show "view query" for prepared statements.

1.2.70 5.53 SQL DM for MySQL (November 2012)

1.2.70.1 New Features:

- Added monitors for monitoring a galera-based cluster. These monitors do not display as default but there is a 'Galera' monitor group that can be unhidden from Tools .. Manage Monitor Groups.

1.2.70.2 Fixed Issues:

- Authentication to SMTP servers could fail with very long passwords. Also authentication with Amazon SES could fail.
- When saving a connection Monyog could apparently 'hang' for a while.
- With a large number of sample points displayed in Dashboard, loading of the page was slow. It could take up to around 1 minute in some cases. Now it is a few seconds.
- SSH keys could be 'forgotten' after a Monyog restart.
- Fixed a rare crash when editing server details.
- In the 'Real Time' interface the option to EXPLAIN queries did not display in Internet Explorer 8.
- For non-admin users, the replication tab was displaying all slave servers instead of showing only those servers which the user has access to.
- Internal optimizations, library upgrades etc.

1.2.71 5.52 SQL DM for MySQL (October 2012)

1.2.71.1 Fixed Issues:

- When upgrading to SQL DM for MySQL 5.51 admin rights for users with such failed to be migrated. This release fixes this for both users that have already upgraded to 5.51 and those that have not.

1.2.72 5.51 SQL DM for MySQL (October 2012)

1.2.72.1 New Features:

- Added a number of monitors and advisors - mostly based on metrics exposed by MySQL 5.5+.
- In the 'Real Time' interface user may now define the threshold for when a query should be considered slow. Lowest possible setting is 1 second.
- In the 'Real Time' interface queries are now truncated after 5000 characters. Before it was 1000.
- 'queries' and 'slow queries' can now be exported to CSV in the 'Real Time' interface.
- Original query formatting is now retained in 'Real-Time'.
- User Manager has added a privilege to allow (or not) to 'view literals' in Query Analyzer page for non-admin users. This can be used for blocking access to read sensitive data for such users and still giving them access to use the Query Analyzer.
- Multiple saved Real-Time sessions can now be viewed at a time.
- Added a 'thread count' showing number of currently running threads on 'Processlist' page.

1.2.72.2 Fixed Issues:

- In the 'Real Time' interface displaying large BULK INSERTS was slow.
- Logging out from SQL DM for MySQL while 'Real Time' was populating could crash SQL DM for MySQL. Linux - and not Windows - was affected.
- In the replication overview page SQL DM for MySQL will now show 'binlog' file name and it's position wherever it applies (where binlogging is enabled). Before it was showing only for master servers.
- In the Monitors page the percentage of SELECTS did not display. This was unlike INSERTS, DELETES and UPDATES .
- A 'Long running query' email alert has added a note clarifying that when the alert was sent the query may still be running. Before it could be understood like the alert was sent after query had completed.

1.2.73 5.5 SQL DM for MySQL (September 2012)

1.2.73.1 New Features:

- Added a new major feature called ‘Real-Time’ to monitor MySQL. This feature details the top 200 queries, slow queries, locked queries, locking queries, tables, databases, users, hosts and query states in real time. Information is retrieved from MySQL every one second. Queries taking more than 10 seconds to execute are considered as slow queries. Note that SQL DM for MySQL depends on the ‘InnoDB Plugin’ (optional with recent MySQL 5.1 versions, standard with higher MySQL versions) to get information on locked and locking queries. Refer MySQL docs(see page 30) for more information on InnoDB plugin. Also note that constants listed here are currently 'hard-coded' but we plan to implement user settings. 'Real-Time' is an Enterprise and Ultimate feature.

- Optimized Monitors page to retrieve and display data approximately 10 times faster than before
- Added an option to choose verbosity of email notifications. 'lesser verbosity' is in particular relevant if you receive mails on a mobile phone or another small handheld device.
- Added 'Monitor level' email-address alert setting. If an email address is specified for a specific monitor, notification will be sent to that particular email-address when the monitor is alertable in addition to the email address specified globally for the server.
- In the MySQL query log settings page, the 'file path' field for general query as well as slow query log was disabled for MySQL versions lesser than 5.1.6.

1.2.74 5.3 SQL DM for MySQL (July 2012)

1.2.74.1 New Features:

- Introduced support for 'native' MySQL SSL-encryption with direct MySQL connections.
- SQL DM for MySQL now fully supports IPv6 (Internet Protocol Version 6).
- History-trends have been optimized to retrieve data faster than it was before. With large data sets this results in a speed improvement in the range of a factor ~25.
- Passwords for SQL DM for MySQL users are no longer stored in clear text. A hash is stored (for admin user in MONyog.ini - for other users in the embedded database). Also MySQL passwords as well as SSH/SSL keys, certificates and passphrases are now stored obfuscated. First time SQL DM for MySQL starts after upgrading from a version using clear text storage it will rewrite passwords etc. to the hashed/obfuscated representation.
- Added a Linux counter "Load Average" as exposed by the Linux kernel in /proc/loadavg.

1.2.74.2 Fixed Issues:

- Fixed a memory leak with SSH key based authentication.
- If HTML tags were a part of long running query, the notification email format was getting distorted.
- MySQL connection through SSH tunnel never re-established connection in Sniffer module when the connection failed for the first time
- Filter settings in the sniffer defined on a (MySQL) user with a SPACE character in its name did not work as expected.
- UI fixes.

1.2.75 5.21 SQL DM for MySQL (June 2012)

1.2.75.1 New Features:

- The Monyog API was rewritten and enhanced with more options. Options now include calls to enable/disable data collection, sniffer, notifications etc. for servers by either specifying a server name or a server tag. The old API calls have been blocked.

1.2.75.2 Fixed Issues:

- Notification emails were not being sent when the number of file descriptors used by SQL DM for MySQL exceeded a certain number.

- Fixed a bug with the Custom SQL monitor where no data was being displayed in the Monitors page when all the columns retrieved by the Custom SQL were specified as Key Columns.
- Increased readability by improving SNMP trap format.
- UI fixes and enhancements.

1.2.75.3 Miscellaneous:

- Testing slave connections has been moved to replication monitoring settings. Prior to this release it was in 'MySQL settings' tab making it necessary for users to switch between tabs for testing slave connections.

1.2.76 5.2 SQL DM for MySQL (May 2012)

IMPORTANT note:

- This release requires a new registration code. Neither the 5.1x nor the pre-5.1x key will work with this. Registered customers will get the new code from our Customer Portal. Please have the new code available before installing. Until SQL DM for MySQL is registered with new keys, it will not be collecting data from your servers.

1.2.76.1 New Features:

- Added filter for including or excluding specific hosts and(or) users in Slow log analysis.
- Added and modified a few Monitors related to Replication, InnoDB and Security. Most of this utilizes metrics exposed by MySQL from version 5.5 .
- Added a preconfigured Custom SQL Object (CSO) for Percona Servers exposing the most written and most read tables.
- There is now session-wide persistence while sorting columns in Query Analyzer.
- Improved error messages in Query Analyzer and Wayback machine.
- Usability enhancements, GUI fixes and internal optimizations.

1.2.76.2 Fixed Issues:

- SQL DM for MySQL was not working while behind Apache configured as a reverse proxy. This bug was introduced in version 5.0 of SQL DM for MySQL.

1.2.77 5.12 SQL DM for MySQL (April 2012)

1.2.77.1 Fixed Issues:

- Sometimes the default landing page failed to load after a log in. A refresh was necessary. This was introduced in version 5.0 of SQL DM for MySQL.
- Closed alerts could reappear in the Monitors page after an unsuccessful data collection and MySQL restart.

1.2.78 5.11 SQL DM for MySQL (April 2012)

IMPORTANT note:

- This release requires a new registration code. Registered customers will get the new code from our Customer Portal. Please have the new code available before installing. Until SQL DM for MySQL is registered with new keys, it will not be collecting data from your servers.

1.2.78.1 New Features:

- Introduced tag level customization in SQL DM for MySQL Monitors. SQL DM for MySQL tags for servers are now available as Monyog Object Model variable (MONyog.Connections.TagName returns an array of tag names defined for a server). Documentation has examples on how to use it in scripting. Also see the note on 'Native JavaScript properties' support in next point below.
- Native JavaScript properties (like length, indexOf, splice etc) are now fully supported in Monitors and Dashboard. Note that 'indexOf' is required to check for individual tags from the array exposed by Monyog.Connections.TagName (see point above) in case there is more than one tag.
- Optimized Dashboard page to retrieve information faster.
- Auto-registered slaves will now have server names as their host names as default.
- Auto registering slaves will now assume the SSH host name of the slaves to be same as that of MySQL host. Before this the slaves' SSH hosts were assumed to be the same as that of the master.

1.2.78.2 Fixed Issues:

- On SQL DM for MySQL restart, server edit, SQLite vacuum and MySQL restart there were 2 consecutive data collections with a short time interval. This could skew charts and MySQL metrics. Now SQL DM for MySQL collects data 2 consecutive times only when required - i.e during the first collection and after an unsuccessful data collection.
- Exporting charts as PNG exported as JPEG instead in Monitors page. This bug was introduced in version 5.0 of SQL DM for MySQL.
- UI fixes.

1.2.78.3 Miscellaneous:

- Increased SQL DM for MySQL session timeout from 30 minutes to 8 hours (the MySQL default).

1.2.79 5.1 SQL DM for MySQL (March 2012)

IMPORTANT note:

- This release requires a new registration code. Registered customers will get the new code from our Customer Portal. Please have the new code available before installing. Until SQL DM for MySQL is registered with new keys, it will not be collecting data from your servers.

1.2.79.1 New Features:

- In the 'Disk Info' page, clicking on a donnut chart in the Database level of a server will now drill down to table level.
- Query Analyzer' now has information on Average rows sent & examined while analyzing Slow Query Log. Those options has been hidden in version 5.0 and seen only in "Query details" pop up, now in v5.01 users can view those columns in QA output along with "Query details" pop up.
- Comparison of MySQL server configuration of multiple servers side by side is now faster in the 'Server Config' page.

1.2.79.2 Fixed Issues:

- The 'login' page sometimes did not work in some 'minor' IE 8 builds (if not updated with latest patches from Microsoft). Same could happen with Google Chrome that were not up to date. This happened due to 'Google Gears' (that is no long part of Chrome). It now will work with all IE8 builds and Chrome from version 9.x (current is 17.x) - also with 'Gears'.
- MySQL server configuration could show incorrect values if MariaDB Servers were compared in the 'Server Config' page.
- A MySQL server that was a master and a slave at the same time was not showing the slave status in the 'Replication' tab even if it was marked as a slave in 'Register Server'.
- In the 'Replication' tab of SQL DM for MySQL the display of binlog_do_db and binlog_ignore_db settings were interchanged.
- On a fresh installation the default SQL DM for MySQL port was 8888. This was introduced in 5.0. Its now reverted back to 5555.
- Sometimes queries could fail to display in 'Wayback Machine' interface.
- In 'Monitors' page - History/Trend view charts did not plot if data series contained '(n/a)' value(s).

1.2.79.3 Miscellaneous:

- Internet Explorer 6.x and 7.x are no longer supported. And actually this should have been announced in 5.0 GA release notes, but it was missed (it was announced in 5.0 beta1 release notes however).

1.2.80 5.0 SQL DM for MySQL (February 2012)

1.2.80.1 UI Enhancements:

- Introduced a left panel that contains the list of servers registered in SQL DM for MySQL, 'Tools' & 'Customize'. The 'List of servers' page has been removed. Users can now select/unselect servers from this panel.
- Previously available icons for Edit server/Duplicate server/Diagnostic reports etc now appear as a drop down menu next to each server.
- Previously the 'Customization' page appeared under 'Tools', now it appears as 'Customize' on the left panel.
- 'Manage changed Monitor/Advisors & Dashboard charts' are now available as 'Manage changes' under 'Customize'.
- 'Manage Custom SQL Objects' are now available under 'Manage changes'.
- The Global Notifications come on the top right hand corner before the help button.
- Pagination in Query Analyzer page. This removes the limitation of 200 queries in the result set.
- 'Monitors' is now the default landing page.

1.2.80.2 New Features:

- Added an option to add users to admin group.
- Columns in the replication tab can now be sorted.
- Register/Edit server page now has tabbed interface which eases navigation.

Enhancement:

- Improved browser level caching.

1.2.80.3 Miscellaneous:

- The Monyog API responses are now in the following JSON format: {"STATUS": "SUCCESS/FAILURE", "RESPONSE" : ""}
- Query Analyzer will now display query text up to 5000 characters. (before it was 2000)

4.x SQL DM for MySQL

1.2.81 4.81 SQL DM for MySQL (January 2012)

1.2.81.1 Fixed Issues:

- Custom SQL Objects (CSO) with a large result set could fail to display content in Monitors/Advisors page.
- Specifying the "Key Column" field has now been made mandatory while adding a new CSO. If not specified the CSO will not save.
- Small UI fixes.

1.2.82 4.8 SQL DM for MySQL (December 2011)

1.2.82.1 New Features:

- This release adds a new major feature to SQL DM for MySQL: Custom SQL Counters and Custom SQL Objects. A Custom SQL Object is an object populated from a user-defined SQL-query returning a result set. It is exposed as a JavaScript array for defining counters (Custom SQL Counters) in SQL DM for MySQL.

1.2.82.2 Fixed Issues:

- If a query contained the literal substring 'connect' SQL DM for MySQL could hang during general log analysis.
- SQL DM for MySQL returned a garbage string to the Linux 'ps' command.
- Drastically improved performance in log analysis if the option to 'replace literals' was selected.
- Some of the Monitors/Advisors were not updating in All Time/Current and Delta timeframe.

1.2.83 4.72 SQL DM for MySQL (October 2011)

1.2.83.1 Fixed Issues:

- When connected to Linux distributions with a 3.x kernel the information about Disk I/O could fail to display properly in Dashboard and Monitors/Advisors pages of SQL DM for MySQL.

1.2.84 4.71 SQL DM for MySQL (October 2011)

1.2.84.1 Fixed Issues:

- Accessing the replication tab could crash SQL DM for MySQL in rare cases with specific replication setups and where SHOW MASTER STATUS returned an empty result.
- Fixed an issue where user defined generic (JavaScript) functions added for use by customizations did not work as expected.
- Small UI fixes.

1.2.85 4.7 SQL DM for MySQL (October 2011)

1.2.85.1 New Features:

- Added a MySQL replication overview page. Here SQL DM for MySQL shows the replication topology of all registered MySQL servers, as well as SLAVE STATUS and MASTER STATUS where it applies. The display gets updated at user-specified interval. (Note: This is available only in Ultimate version of SQL DM for MySQL)
- Added a 'Wayback Machine'. In this interface a graph displaying - as per user's choice - either of the status variables 1) Threads connected or 2) # of slow queries. The graph displays aggregated values on years/ months/days/hours/ minutes depending on the data. If sniffer was running during this interval, aggregated sniffer information will be displayed. Also you can see first and last value of (optionally) all or changed variables aggregated values. The graph is zoomable by selecting a sub-interval with the mouse along with aggregated sniffer and changed variables you can also get point-in-time information by clicking on a point on the graph. When user clicks on a row in the query list of the Wayback machine a pop-up opens with information about thread-id, user and host along with full query.The list of queries in Wayback Machine will not be rendered if there are more than 2000 queries in the time period. There is a user control that can be activated in such case. The reason is that every 1000 queries take around 1 seconds to render on an average desktop system. This also means that we can now display the list of queries before zooming (provided still that there not more than 2000 queries to display). When we do a point in time select of one of the points on the graph, we get a bar chart that displays number of queries.
- With this release the dependency on Flash has been removed. Charts are now rendered using a JavaScript library. Accordingly SQL DM for MySQL can now be handled from a browser on a device not supporting Flash (such as Ipad). These charts can be exported like was the case with Flash-based charts of previous versions. Conversion to various formats uses a web service.

1.2.85.2 Fixed Issues:

- Fixed an issue where SQL DM for MySQL did not logout user successfully.
- If the time is set to 12:XX:XX in Custom filter it resets to 00:XX:XX.

1.2.85.3 Miscellaneous:

- Instructions to setup MySQL Proxy was added to SQL DM for MySQL interface.

1.2.86 4.62 SQL DM for MySQL (July 2011)

1.2.86.1 Fixed Issues:

- The counter for 'thread cache hit rate' could display incorrect values.
- Authentication with the admin password could fail in some cases on Windows. This bug was introduced in 4.61.

1.2.87 4.61 SQL DM for MySQL (July 2011)

1.2.87.1 Fixed Issues:

- In Dashboard & Monitors/Advisors page, the Delta & All-time/Current charts were showing incorrect values when values were plotted per second. This was introduced in version 4.51 of SQL DM for MySQL.
- While connecting using SSH with key based authentication in Windows, the temporary file which contains the SSH private key was not being deleted. Before this release, the temporary files could run out of names which made it impossible to create new temporary files and hence the SSH tunnel was not being created.
- SQL DM for MySQL now writes default data into MONyog.ini file on all platforms when SQL DM for MySQL starts for the first time. Before this release SQL DM for MySQL for Linux did not write default data into MONyog.ini file.
- Fixed a crash occurring when there was no data directory path mentioned in the MONyog.ini file.

1.2.88 4.6 SQL DM for MySQL (June 2011)

1.2.88.1 New Features:

- SQL DM for MySQL is now bundled with a bunch of useful dashboard charts which can be enabled/disabled from the 'manage dashboard charts' in the Dashboard page. These charts can even be re-ordered.
- Added an option to 'Alert on change in server configuration'. These alerts in the form of email's and/or SNMP traps are sent by SQL DM for MySQL whenever SQL DM for MySQL detects a change in the server variables using SET GLOBAL statements or in the MySQL configuration file.
- Added an option to copy/duplicate advisors. This is available when clicked on an advisor in the 'Monitors/Advisors' page.

1.2.88.2 Fixed Issues:

- Export as CSV in the 'Monitors/Advisors' page was showing '0' for some of the advisors when the group to which they belong was disabled.
- GUI fixes, including an issue where clicking a 'next'-link in the 'register server' page did nothing.

1.2.89 4.51 SQL DM for MySQL (May 2011)

1.2.89.1 New Features:

- This release focuses on improving performance of the SQL DM for MySQL built-in HTTP daemon. You will find up to 10 times performance improvements for most SQL DM for MySQL pages. This has been achieved by internal code optimizations, use of compression (if browser supports) and minification/optimization of HTML, JavaScript, CSS and graphics.

1.2.89.2 Fixed Issues:

- Prepared statements were not handled properly by the Query Analyzer.
- Events (as introduced in 4.5) stored in the SQL DM for MySQL embedded database did not purge as specified in the 'retention time frame' setting.
- Charts could display decimal numbers with a large number of decimals that made no sense and made charts less readable. We now round to 3 decimals.

1.2.90 4.5 SQL DM for MySQL (May 2011)

1.2.90.1 New Features:

- Added an EVENTS overview. An EVENT happens when any counter is changing its status to (yellow) WARNING alert level or to (red) CRITICAL alert level. An 'alert condition' (WARNING or CRITICAL) can be temporarily disabled (and re-enabled) for a specific server from the new EVENTS overview page as well as the Monitors/Advisors page. This can be disabled for non-admin users.
- Replication slaves can now be automatically registered. SQL DM for MySQL will detect the host/IP of active slaves. MySQL (port,username & password) and the SSH details of the slaves are assumed to be the same as that of its master. Connection details of the slaves can be edited if different from master. A slave will indent with its master in a hierarchy while displaying in the list of servers page. Auto-registering of slaves is also extended to multiple levels. 'Test Connection' in the 'Register/Edit server page' will test the slave connection details of that server in recursion, when 'Auto-register all slaves' is checked.
- In the process-list based sniffer there is now an option to alert (using mail and/or SNMP) for long-lasting LOCKs.
- Monitors/advisors groups can now be enabled/disabled on a per server basis. Before this release it was only possible to have same setting for all servers registered in SQL DM for MySQL in this respect.
- Added monitors/advisors for innodb_force_recovery, log_warnings, log_output, expire_logs_days,
- max_binlog_size and flush_time variables.
- Query Analyzer can now filter slow logs based on host and user information.
- The standard Linux daemon PID file is now created for SQL DM for MySQL. (If tools like 'Monit' are being used, then the PID file of SQL DM for MySQL would be required.)

1.2.90.2 Fixed Issues:

- Padding while exporting slow log as CSV was incorrect.
- SQL DM for MySQL could hang while parsing a log file with binary data.
- With SQL DM for MySQL running behind a Load Balancer and similar network management system, user login to SQL DM for MySQL could fail.

- SQL DM for MySQL could hang while downloading slow or general logs from a server if the server wrote new entries to the log at download-time.
- The emails sent by SQL DM for MySQL were missing the date header.
- Monitors/advisors page was displaying error for MySQL 5.5.x when Innodb engine was disabled.
- Fixed an UI issue with the Group level display of alerts (red/yellow) in monitors/advisors page.
- Fixed a bug with the system graphs in the 'dashboard' feature where the graphs were not plotting the current values.
- Lots of GUI fixes and small GUI enhancements.

1.2.91 4.2 SQL DM for MySQL (January 2011)

1.2.91.1 Fixed Issue:

- SQL DM for MySQL could crash while monitoring hundreds of servers through SSH tunneling with a small data collection interval. (1 or a few seconds)
- Charts in the I/O section under Monitor's/Advisors or Dashboard displayed wrong history. (last 14 seconds had the same values as the value in the 15th second)
- Sometimes, while analyzing the slow query log, some timestamps contained the default unix time stamp.
- If the name of a counter or caption of dashboard chart contained single quotes, the chart displayed "Invalid XML Data".
- Filtering while displaying the Sniffer output was not working when there was a change in the filter conditions.
- The Linux command 'Service MONyogd status'did not display the correct result.
- The 'Processlist page' in SQL DM for MySQL threw 'constraint failed' error when the process ID was a large number.
- At times, the 'Query Execution Time' of Query Analyzer displayed incorrect values for the milliseconds field.
- SQL DM for MySQL was not closing the file descriptors after reading from MySQL error log (introduced in 4.12) and sending SNMP traps.

1.2.91.2 Miscellaneous:

- Every time SQL DM for MySQL (re)starts it logs "MONyog has started" in MONyog.log.

1.2.92 4.12 SQL DM for MySQL (December 2010)

1.2.92.1 Fixed Issues:

- SQL DM for MySQL was leaking memory while connecting through SFTP to analyze Slow/General logs.
- SQL DM for MySQL also leaked memory when 'Processlist' page was open.
- Stopping of Data collection could crash SQL DM for MySQL.
- Sometimes SQL DM for MySQL service could not be stopped gracefully.
- SQL DM for MySQL was not able to read the Error log via SFTP for data collection interval less than 25 seconds.
- Max open .idb files counter under Innodb-Others group sent a false 'warning alert'. (Yellow light would turn on even though the value of innodb_open_files did not exceed the threshold)

1.2.93 4.11 SQL DM for MySQL (November 2010)

1.2.93.1 Fixed Issues:

- Email alerts or SNMP traps related to counters were not sent. This was introduced in 4.1.
- Minor usability fixes.

1.2.94 4.1 SQL DM for MySQL (November 2010)

1.2.94.1 New Features:

- Implemented an easy-form based interface for customizing helper functions. Any customization of helper functions or user defined functions being used with earlier versions have to be migrated manually to this version. From this version and onwards a GUI-'conflict resolver' will guide the users very similar to how it works for customized counters.
- SQL DM for MySQL now has an option to apply a group of settings to all the servers with a specific tag. (Please note: New servers registered later with the same tag are not considered)
- Monyog logs from now on contain the server name along with error, making it easy to backtrack.
- We have automated deleting dump files of size zero.
- Earlier multiple alerts for same issue would be sent if the counter was alertable for more than the specified number of data collections. Now we only send once. Settings in register server page has been updated accordingly.
- Added an option to change the refresh interval from the processlist page.
- Processlist-based sniffer will no longer show the query 'SHOW FULL PROCESSLIST' executed by SQL DM for MySQL (Please note: This only applies to the processlist-based sniffer and not to the processlist page where user has configurable filtering options).

1.2.94.2 Fixed Issues:

- Fixed a (SQLite) database lock resulting in failure to update Dashboards and Monitors/Advisors pages.
- When SFTP option was chosen to analyze 'MySQL general query log' and SSH was disabled, SQL DM for MySQL used SFTP to analyze the logs with out throwing any error.
- Small GUI enhancements.
- Resolved crashes found internally.

1.2.95 4.02 SQL DM for MySQL (September 2010)

1.2.95.1 Fixed Issues:

- Fixed a performance regression issue (introduced in 4.0) in HISTORY/TREND analysis.
- Fixed an issue where Dashboard charts were not always populated properly with 'old' data when the Dashboard page was opened. The chart itself and the data-series for the chart are independent data streams which were not always 'combined' correctly.
- The process-level SQL DM for MySQL stack size on Linux is now always expandable up to 10 MB (Linux kernel default) even if the user has defined another value. We had a report where SQL DM for MySQL would crash with a very low user setting (and this is basically an OS issue and not a SQL DM for MySQL issue).

- If a SQL DM for MySQL build with a 'lower' license than ULTIMATE was installed 'on top' of the TRIAL some ULTIMATE functionalities would still run in the background - but not controllable by user. MONyog continued to collect data internally for counters not applicable to ENT or PRO edition and the Monyog.log would log errors irrelevant for the license model

1.2.96 4.01 SQL DM for MySQL (August 2010)

1.2.96.1 Fixed Issues:

- Implemented code that makes OpenSSL (which is used by our SSH library) explicitly thread-safe, thus fixing an issue where specific cryptographic function calls under very heavy load could cause a crash.
- Fixed a SQLite corruption issue on Windows due to which SQL DM for MySQL service failed to stop.
- If both values for 'user' and 'host' were filtered in processlist-based sniffer the interface would display NULL for both.

1.2.97 4.0 SQL DM for MySQL (August 2010)

1.2.97.1 New Features:

- Compare Configuration: This feature enables the user to compare changes in configuration between two instances of MySQL. Through this feature, the user will gain a greater insight on why one server is performing better than the other as far as server configuration is concerned.
- Track Changes: This feature enables users to track changes to my.ini/my.cnf of a MySQL server over a period of time and compare changes between revisions.
- Added a new interface for customizing Monitors/Advisors and Dashboard charts requiring less knowledge of JavaScript than what was the case before.
- When upgrading an installation with customized Monitors/Advisors and Dashboard charts a 'conflict resolver' will now guide user to either migrate his old changes or discard them.
- All cache hit rate related monitors/advisors have been changed to display cache miss rate information as it more relevant when tuning databases. Also the advisor text for these monitors/advisors has been improved.

1.2.97.2 Fixed Issues:

- Fixed some issues with specific counters including issues where 'undefined' or 'NaN' were returned incorrectly.
- Graphs in 'all time' and 'delta' timeframes for the counter 'Cache Misses' for as well InnoDB Cache, MyISAM key cache and Query cache groups were not displaying but threw the error "Invalid XML data" (these counters were added in beta 1).
- Previously accepted values for the "ChartValue" property in Dashboard charts were "Current" and "Latest". This has caused confusion since both terms mean the same thing in this context, although technically "Current" refers to the current value of the metric and "Latest" to the "Delta" value. Moreover, the documentation mentions "Actual" and "Delta" as accepted values. To eliminate this confusion, the "ChartValue" property now accepts the values "Current" and "Delta" - "Current" referring to the current value of the metric, and "Delta" to the difference in the metric between the previous two collections.
- Monitors/Advisors Customization: The Group Name of a Monitor/Advisor can now be changed only when a new, custom, counter is created; once set it cannot be changed. Changing Group Names of Monitors/ Advisors shipped with SQL DM for MySQL is not allowed.
- All cache hit rate related monitors/advisors have been changed to display cache miss rate information as it more relevant when tuning databases. Also the advisor text for these monitors/advisors has been improved.

1.2.98 Miscellaneous:

- Windows 2000 is no longer supported.

3.x SQL DM for MySQL

1.2.99 3.8 SQL DM for MySQL (August 2010)

1.2.99.1 Fixed Issues:

- Monitors/Advisors on InnoDB Deadlocks were truncating the output message and hence in some cases, users were unable to view both queries involved in the deadlock. This has been changed such that the first 2048 characters of both queries are displayed.
- On Windows 7 systems changes to Monyog's .INI file from external programs had no effect upon restart (as Windows reverted the changes). Now throughout SQL DM for MySQL lifetime the .INI file will be locked so that a user may only edit the file manually when SQL DM for MySQL is not running (but users can use the SQL DM for MySQL interface to change any setting while SQL DM for MySQL is running). Any changes made to the file when SQL DM for MySQL is not running will be reflected by it once it is restarted.
- If MySQL Error Log monitoring was enabled for a particular server, the first two data collections for that server would cause large network traffic to occur if the MySQL error log was very large. This was because the entire log fie is read in the first collection. Now only the last 1MB of the file is read.

1.2.99.2 Miscellaneous:

- All libraries have been upgraded to their latest versions. Although, from a user-perspective, the changes are minimal, this has alleviated several memory leaks and other performance issues.
- This release has been checked in depth with the best leak-detection tools available for both Linux and Windows. All discovered leaks have been fixed.
- The cache parameters for the SQLite library have been modified with this release to allow for registering more servers on the same system as before. Before SQL DM for MySQL memory usage used to grow continually with time as each open SQLite handle has an associated cache which could grow to any size, irrespective of system capacity. The cache size per handle has been reduced and the total SQLite cache size will never exceed 1 GB with this build.

1.2.100 3.77 SQL DM for MySQL (July 2010)

1.2.100.1 Fixed Issues:

- Problem with SQLite where journal files were not being handled correctly leading to corruption of SQLite databases.

1.2.101 3.76 SQL DM for MySQL (May 2010)

1.2.101.1 Miscellaneous:

- Added a ten server license.

1.2.102 3.75 SQL DM for MySQL (April 2010)

1.2.102.1 Fixed Issues:

- Failure to open the SQLite resource DB located in SQL DM for MySQL installation directory could result in a crash. This crash has been fixed, but note however that SQL DM for MySQL is unable to function without this resource DB and will log an error message and perform a graceful exit.
- The Monitor/Advisor "Binary Log" --> "Transactions that got saved in temporary file" displayed "NaN" (JavaScript for Not-a-Number) for non-zero values of the associated MySQL counter.

1.2.102.2 Miscellaneous:

- Internet Explorer 6.2800 on Windows 2000 may hang when trying to display SQL DM for MySQL interface. The behavior is not seen in the next iteration of IE6.

1.2.103 3.74 SQL DM for MySQL (April 2010)

1.2.103.1 Fixed Issues:

- Log analysis could use high CPU if the option to replace literals with a placeholder was not selected and the literal string ' "' (space + doublequote) occurred in the log file.
- CSV-export from Query Analyzer only exported 200 rows (the same number as displayed in the browser view). Now CSV-export will export all rows. Also the UI makes it now clear that the browser displays the 200 uppermost rows according to current sort criteria.
- An error message regarding an integer overflow could occur in Monitor/Advisors page for some values. The SQLite datatype used could be too short for some specific data.
- Login to SQL DM for MySQL could fail if SQL DM for MySQL was running behind a proxy.

1.2.104 3.73 SQL DM for MySQL (March 2010)

1.2.104.1 Fixed Issues:

- If SHOW ENGINE INNODB STATUS returned an error that was not privilege-related, SQL DM for MySQL reported MySQL as non-available. That could happen for instance if MySQL was started with --skip-innodb option. This bug was introduced in 3.71 with the support for InnoDB deadlock detection.
- A bug in the SQL DM for MySQL startup script could on Linux have the result that SQL DM for MySQL was still reported as running if it had been killed or had crashed.

1.2.105 3.72 SQL DM for MySQL (February 2010)

1.2.105.1 New Features:

- The number of builds for Linux has been increased to the double number of what it was before. In addition to the builds based on glibc version 2.3 we now distribute builds based on glibc 2.5. We had a few reports of random crashes (typically occurring up to a few times per week) occurring on recent 64 bit CentOS servers and in one case also a RHEL5. The glibc 2.5-based build fixes this. Although we only had such reports on 64 bit Linux of 'Red Hat Family' we also included 64 bit tar.gz for all Linux platforms and 32 bit RPM builds. The general advice on which build (glibc 2.3-based versus 2.5-based) should be used in every case would be that if glibc 2.5 or higher is available on the system you should use the 2.5-based build. Or simply use the one based on 2.5 if it installs and starts on your system. Upgrading users that are in doubt can continue with 2.3 -based builds if they have no stability issues.

1.2.105.2 Fixed Issues:

- When SQL DM for MySQL encountered an SSH error while trying to read a file using SFTP it could crash. This has been fixed.
- Redundant "Server Restarted" alerts concerning monitored Linux systems could be sent by SQL DM for MySQL. Now SQL DM for MySQL will notify users only once when a system is restarted, and alert will be sent as soon as SQL DM for MySQL detects it.
- When restarting SQL DM for MySQL an error message could be written to the log indicating that an ALTER TABLE SQLite query failed to execute. Technically, it was an issue with a Schema version table in the SQL DM for MySQL database.

1.2.106 3.71 SQL DM for MySQL (February 2010)

1.2.106.1 New Features:

- Added a monitor for InnoDB deadlocks (as exposed by SHOW ENGINE INNODB STATUS - statement).
- In case of a program crash on Linux, SQL DM for MySQL will save a core dump like the Windows version already does. The dump is saved in .../MONyog/bin folder.

1.2.106.2 Fixed Issues:

- EXPLAIN from Processlist page could fail with syntax error due to a missing SPACE character in the statement.

1.2.107 3.7 SQL DM for MySQL (February 2010)

1.2.107.1 New Features:

- The SQL DM for MySQL 'admin' user can now create other users having access to a subset of available servers only. For those non-admin users access to edit server settings, to KILL queries and to execute FLUSH STATUS is optional. Also note that only 'admin' is allowed to create/delete server and user registrations.
- Monitors/Advisors page can now be exported as CSV.
- Some counters in Monitors/Advisors page were rearranged. Details about this:
    a. "Security" and "Excessive Privileges" groups have been moved to the top.
    b. "Table Cache" and "Table Locks" have been merged to a single group "Table Cache & Locks".
    c. "Thread Cache" and "Slow Launch Threads" have been combined to a single group named "Threads".
    d. "Commands" and "Schema Changes" have been merged to a single group "Commands & Schema Changes".
    e. Counters in "Network Traffic" have been moved to "Connection History".
    f. Counters in "Sort Buffers" have been moved to "Misc.".

1.2.107.2 Fixed Issues:

- FLUSH STATUS was not working for servers having space(s) in their names.
- Warning also displayed when global notifications are disabled.
- Duplicate 'Server restarted' messages were sometimes sent when SQL DM for MySQL is restarted.

1.2.107.3 Miscellaneous:

- SQL DM for MySQL is now available in three different editions: Professional, Enterprise and Ultimate (full-featured).

1.2.108 3.65 SQL DM for MySQL (December 2009)

1.2.108.1 New Features:

- You can now specify a comma-separated list of users to be ignored/excluded by notify and/or kill actions for long-running queries in 'query sniffer' interface.
- Added a 'diagnostic info' icon to servers in 'list of servers' page. Clicking this will generate a report in plain text with the most basic information about the server.
- Added an option to stop/start data collection for a server in 'list of servers' page (without opening the detailed pages for this server).
- Enabling/disabling data collection from and/or alerting about all servers can be done from tools .. preferences .. maintenance.
- Enabling/disabling data collection from and/or alerting about specific servers can be done by calling the SQL DM for MySQL URL with parameters. This can be scripted and scheduled using any standard method available with the OS. Please refer to documentation for details.
- Mail alerts will now have a direct link to open SQL DM for MySQL login page.
- Added an option to customize colors used in Dashboard.
- If global wait_timeout setting for a server is lower than the sample interval we will now increase session wait_timeout setting in order to avoid the MONyog.log to grow with a "MySQL server has gone away" message for each data retrieval.

1.2.108.2 Fixed Issues:

- Fixed some issues with validation of user input. For specific data invalid and unusable input could be saved by SQL DM for MySQL. This includes 1) invalid separator between email addresses 2) Entering a decimal where an integer is required.
- SMTP return codes were neither logged nor were specific messages displayed to the user based on them.
- In history/trends an empty graph would display if server details were wrong. Now proper JavaScript errors are displayed.
- In connection details page for a server the previous/next links would hide with specific menu items selected.
- In History/Trends, warning message ("No data found in this range, try changing history range!") didn't show up for the time-frame in which server data collection was disabled.
- If Adobe Flash was not installed, the message to "install the latest Adobe Flash Player" was displayed more than once.
- Small GUI improvements and fixes.

1.2.109 3.62 SQL DM for MySQL (December 2009)

1.2.109.1 Fixed Issues:

- When using the option (introduced in 3.6) to alert for long-running queries from the Query Sniffer interface, such long-running query could raise multiple alerts.

1.2.109.2 Miscellaneous:

- Added 'User and Host' information for long running query alert.

1.2.110 3.61 SQL DM for MySQL (December 2009)

1.2.110.1 Fixed Issues:

- If notifications were enabled, program threads were not always closed properly what ultimately could lead to a program 'hang'. This bug was introduced in 3.6.

1.2.111 3.6 SQL DM for MySQL (November 2009)

1.2.111.1 New Features:

- Delayed alert notifications. It can now be defined that a problem must have existed for a number of sample intervals continuously ('in a row') for an alert to be sent. A global setting for each server is available from GUI. For individual counters the global setting can be overridden by defining the RetryOverride (like 'RetryOverride:3') property of the (JavaScript) counter definition in Tools... Customize interface.
- Added counters for monitoring I/O parameters on Linux systems (similar to 'iostat').
- Host and user information is added (optionally) to the Query Analyzer output. Note: This option is currently not available for proxy based sniffer.
- Duplicate/Clone Server functionality added to GUI.
- Added direct link to customize counters from the Monitors/Advisors page.
- Grouping of servers: one or more 'tags' can now be specified for a server. In 'register servers' page the list of servers are grouped as per tags. As more tags may be specified for a server, it may appear in more groups. A group established this way can be selected in one operation.
- Added an option to send alert if server was restarted between two data retrievals. With managed hosting the MySQL server may be restarted automatically as part of routine maintenance or after a server crash. Knowing when this has occurred can be useful.
- Process-list based sniffer now has an option to notify about and kill long running queries.
- New and updated counters/advisors.
- Added automatic and manual update check.

1.2.111.2 Fixed Issues:

- Notifications did not take 'base time' setting and/or 'uptime_since_flush-status' status variable into account.

- If SSH setting for retrieving Linux system information were once selected and later deselected again, graphs without any information would appear in Dashboard and in MONyog.log errors were recorded.
- Now "busy" icon is displayed while saving/deleting a connection. When SQL DM for MySQL was controlled from a browser running on a remote machine the absence of this could cause confusion about if change had effect.
- A garbage string like "0NaN/0NaN/NaN 0NaN:0NaN:0NaN" could be displayed when editing log analyzer filter settings.
- With specific settings Dashboard graphs could hide other displayed dialogs.
- Notification mails now have information about the value that triggered the alert.
- In log analysis a divisor '1024' was use for calculation of 'kilo rows'. Now the divisor '1000' is used.
- Exporting charts as graphics did not use a consistent naming convention. Now files will be named 'MONyog_chart_ConnectionName_Chart_Title.file_extension'.
- For slave servers process-list options to "Kill", "Copy", "View Query" and "Explain" were not always accessible.
- GUI improvements, improved error messages etc.

1.2.111.3 Miscellaneous:

- Added a warning in Disk Info page that the calculation may be expensive. In particular with databases with thousands of objects the calculations may cause unacceptable load.
- Default data collection interval is 5 minutes.

1.2.112 3.5 SQL DM for MySQL (October 2009)

1.2.112.1 New Features:

- Alerts can now be sent as SNMP traps.
- Support for MySQL error log. An alert can be sent when the error log has an entry of type [ERROR] and the error log entry can be viewed in SQL DM for MySQL web interface as well.
- Query Analyzer data can now be exported as CSV.
- Updated and added advisors.
- EXPLAIN/View query from Processlist page will now open in a new browser tab.
- Alerts (mails or traps) will now list the IP of the machine running the SQL DM for MySQL instance (request from users who had more than one SQL DM for MySQL instance).
- Graphs/charts will now have new colors that make the graphs more readable.
- Added next/previous links in register/edit servers page.

1.2.112.2 Fixed Issues:

- Lots of fixes for browser-specific issues.
- If identical queries were logged from more clients with same timestamp, Query Analyzer/slow query log would only count that query once.
- Long BULK INSERTS would not display properly in Query Analyzer if the option to 'replace literals' was selected.
- Deleting a server registered as number '0001' did not delete the sniffer.data file for that server.
- Windows Vista and higher could warn that SQL DM was not properly installed due to lack of an Application Manifest. It was false alarm but now such Manifest is included with the installer.

1.2.113 3.16 SQL DM for MySQL (September 2009)

1.2.113.1 Fixed Issues:

- Some queries were not parsed correctly from server logs. Affected queries were queries enclosed in (brackets) and some queries with comments. Known issues still are: 1) queries starting with a conditional comment. 2) queries containing a 'quoted string' with a " ; " (semicolon) followed by a line-break.
- Query time could fail to be read correctly from logs.
- Fixed a memory issue when parsing logs.

1.2.114 3.15 SQL DM for MySQL (August 2009)

1.2.114.1 Fixed Issues:

- More CPU optimizations with processlist-based sniffer.
- When installing Windows version a js32.dll - error could stop installation (after waiting a few seconds and clicking 'retry' it installed successfully).
- Some counters of YES|NO type had an icon for displaying graph what made no sense.

1.2.115 3.14 SQL DM for MySQL (August 2009)

1.2.115.1 Fixed Issues:

- Significant CPU optimization with processlist-based sniffer.
- Saving mail settings with Linux version could cause a program 'hang'.

1.2.116 3.13 SQL DM for MySQL (August 2009)

1.2.116.1 Fixed Issues:

- Fixed a major memory leak issue in processlist based sniffer. Also some other small leaks were fixed.
- Using the option to 'show only changed values' in HISTORY/TRENDS would show an empty result for counters having non-integer values.
- On systems running Kaspersky security software 'test connection' could show an empty message box. Kapersky filtered. We have changed the text of the message so that it does not happen now.

1.2.117 3.12 SQL DM for MySQL (July 2009)

1.2.117.1 New Features:

- All charts displayed by SQL DM for MySQL can now be exported as PDF/JPG/PNG (from chart context menu).
- Query Analyzer will now sort in descending order as default first time (before it was ascending).

1.2.117.2 Fixed Issues:

- Validation of connection details was not proper when saving.
- In Query Analyzer options that do not apply for sniffer and log files are now disabled where they do not apply.
- In Query Analyzer 'show details' did not preserve the original formatting of a query.
- Max column in sniffer output in Query Analyzer could show erroneous values in some cases.
- Mail alerts sent by SQL DM displayed TRIAL at bottom even after registration of SQL DM for MySQL.
- Various errors like "ErrCode:-1 ErrMsg:PathMgr::GetExeName readlink failed" could occur in MONyog.log. In some cases the error would cause failure to register.
- Sniffer data are purged automatically as per the server 'retention timeframe' setting, but it could also happen that Sniffer data that were not old enough to qualify for purging were deleted automatically. This would result in some queries missing after analyzing.
- Wrong error was logged by sniffer if MONyog.lua version 3.0 was used with SQL DM for MySQL version 3.1. However in connection details section, on "Test Proxy", proper error was thrown.

1.2.117.3 Miscellaneous:

- Due to a bug in specific builds of the MySQL proxy version 0.72 (it returns incorrect version) Monyog Query Analyzer will not work with those builds affected. Affected builds include versions for Windows and OS-X (but not Linux where this Monyog release works fine with proxy 0.72). We hope that MySQL will fix this soon, but if not we will consider a patch in SQL DM for MySQL itself.
- With Internet Explorer versions 8.0.6001.18783 and higher (latest available for Vista and Win7) exporting graphics will not work. The 'Save' button (a FLASH object) does nothing in those browsers. We will release again as soon as a fix is possible.

1.2.118 3.11 SQL DM for MySQL (June 2009)

1.2.118.1 Fixed Issues:

- Fixed a bug when converting timestamp format (human readable timestamp, unix_timestamp) when reading server logs. The bug could result in time being displayed 1 hour wrong in Query Analyzer if server was in a timezone using 'daylight saving time'.
- A SQLite database LOCK could prevent access to the 'system' database. Linux Operating System counters would then not be available.
- Fixed a GUI issue with IE browser in licensing form page (command button behavior while a command was in process).
- SQL DM for MySQL could crash when user saved specific details (port and password) from 'preferences' .

1.2.119 3.1 SQL DM for MySQL (June 2009)

1.2.119.1 Packaging:

- The same binary/installer is now used for any license model (whether TRIAL or whatever number of servers). A license key defines and controls how many servers can be registered. You may retrieve your license key from our Customer Portal. Before installation please ensure that you have registration details available and also soon after installation has completed please connect to SQL DM for MySQL from your browser and enter registration details. This change of packaging concept simplifies the release cycle for us, as there are now much fewer installers that will need to be built and tested.

- Also now all distributed builds will run as a TRIAL will run for a month (not till a fixed date) and in the TRIAL period an unlimited number of servers can be registered.
- Unfortunately and as a consequence of the changed packaging concept it is not possible to install a 3.1 RPM build on top of a previous version as the file name format (ie. version substring) will need to change to be functional by RPM installers ('rpm' console command and various distribution-specific commands and GUI tools). Previous RPM versions installed must be uninstalled this time before installing 3.1 (but note that previously collected data will still be preserved and available after this). For Windows .exe and Linux .tar.gz builds there is no such issue.

1.2.119.2 New Features:

- Added option to purge all stored sniffer data. Also sniffer default settings were changed. Quite a lot of evaluating users experienced an empty report due to previous default settings.
- Added support for MySQL Proxy version 0.7.1. A new MONyog.lua script is shipped with this build. The old script must be replaced with this one if you use the proxy-based sniffer option. This new script also works with MySQL Proxy 0.61.
- Filtering options were re-arranged in Query Analyzer as per user requests.
- Lots of GUI fixes, clarifications, usability improvements etc.

1.2.119.3 Fixed Issues:

- On RedHat type Linux 'service MONyogd start' command would sometimes not start SQL DM for MySQL in first attempt if SQL DM for MySQL process had been killed or had crashed.
- The mail header of SQL DM for MySQL mail alerts did not handle unicode. Now utf8 encoding is used everywhere.

1.2.119.4 Miscellaneous:

- Do not forget to replace the MONyog.lua script if you use the Query Analyzer with proxy-based sniffing!
- RPM users must uninstall previous versions before installing this one!

1.2.120 3.09 SQL DM for MySQL (June 2009)

1.2.120.1 Fixed Issues:

- If SQL DM for MySQL user for monitoring a replicating slave did not have 'replication client' privilege', MySQL availability would be shown as 'No' in Monitors/Advisors page (even though 'test connection' was successful).
- If mail alerts were enabled, a syntax error in customized JavaScript could cause a crash.
- Charts for a server in Dashboard page would not resume updating after a disconnect and reconnect to that server. A browser refresh was required.
- The chart 'popups' from Monitors/Advisors page would not populate with historical data from the SQL DM for MySQL database but only with fresh data sampled.
- After sorting in Query Analyzer the expand/collapse state was lost.

1.2.121 3.08 SQL DM for MySQL (May 2009)

1.2.121.1 Fixed Issues:

- Query Analyzer did not show correct output when displaying a sorted result. A wrong LIMIT clause was generated when querying the temporary database containing queries.
- Fixed a bug in checksum calculation on 64 bit builds. The checksum is used by Query Analyzer for indexing. On 64 bit wrong sort order could result from this.
- If SSH-tunneling to a MySQL server had been used and later the setting for 'Use SSH' was changed to "No", SQL DM for MySQL would continue to use SSH tunnel as long as valid tunneling details were available.

1.2.122 3.07 SQL DM for MySQL (May 2009)

1.2.122.1 Fixed Issues:

- With SMTP notifications turned "ON" various issues (connection failure, program crash) could occur if SMTP authentication was not used. This bug was introduced with SMTP authentication support in 3.05.
- Fixed a crash with SSH tunneling. Only 64 bit build for Linux was affected.
- Counters "cache hit rate", "pruned as percentage of inserts" and "queries not cached" reported wrong in 'History/trends' timeframe.

1.2.123 3.06 SQL DM for MySQL (April 2009)

1.2.123.1 Fixed Issues:

- Installation on Windows could fail with a js32.dll-related error. Waiting a few seconds and 'retry' would solve this but now the error does not come.
- Fixed a 'false alert' related to settings for temporary tables (the error occurred because of comparing an integer with a non-integer).
- With slow query logs generated from servers greater than 5.1 Query Analyzer "Max" column would show a value without decimal point if the query count was greater than 1.
- Chart pop ups from Monitors/Advisors page would sometimes only populate from next time data was sampled (it was a concurrency issue in code).
- 32 bit builds for Linux failed to populate the Monitors/Advisors page. This bug was introduced in 3.05.
- 3.05 TRIAL build for Windows would expire at a fixed date and not after one month as it should.

1.2.124 3.05 SQL DM for MySQL (March 2009)

1.2.124.1 New Features:

- SSL/TLS mail encryption is now supported for mail alerts.
- Added an option to rebuild SQL DM for MySQL databases. Using this option periodically may result in better performance - including shorter startup time when OS is rebooted. Basically using this option will defragment the database including indexes. It is not possible to provide an 'absolute' advice on how this option should be used. But the larger the databases and the shorter the sample interval the faster there is a chance of fragmentation occurring after a huge amount of INSERT´s and DELETE´s to the database (note

that only with databases files of around 1 GB and larger we have seen the need for this. A 'general' advice could be to execute monthly with large database files).

- Improved error messages for Query Analyzer.

1.2.124.2 Fixed Issues:

- When "Test Connection" or "Test Path" was clicked SQL DM for MySQL seemed inactive. Now the mouse cursor will change to an 'active state cursor' indicating that SQL DM for MySQL is performing the requested operation.
- With query sniffer enabled the 64 bit Linux build could crash. Reports about this include RedHat and Ubuntu Linux distributions with very recent Linux Kernels.

1.2.125 3.04 SQL DM for MySQL (March 2009)

1.2.125.1 New Features:

- In case of invalid JavaScript in a counter definition, SQL DM for MySQL will now tell details about this error.
- Optimizations in the handling of the SQL DM for MySQL embedded database. In particular users with a very large retention timeframe setting and rather short sample interval (resulting in large databases) will benefit from that.

1.2.125.2 Fixed Issues:

- The above will also fix an issue where SQL DM for MySQL seemed to 'hang' when saving connection details for a connection having large database files.

1.2.126 3.02 SQL DM for MySQL (February 2009)

1.2.126.1 Fixed Issues:

- The alerts and advisors for slave 'read-only' mode were in conflict. We will now advise for a slave (in production) that 'read-only' mode should be ON (only the replication thread can write) and alert accordingly.
- Add/Edit connection failed with connection names using SPACE character.
- Proxy connection was not established when proxy was started for a server present in a remote system.

1.2.127 3.01 SQL DM for MySQL (February 2009)

1.2.127.1 Fixed Issues:

- For 5.1.x server SQL DM for MySQL was not fetching the log details like slow query log and general query log file path in Connection details section.

1.2.128 3.0 SQL DM for MySQL (February 2009)

1.2.128.1 New Features:

- The 'query sniffer' can now connect to and retrieve data from a running instance of the MySQL-proxy program used by one or more clients to connect to MySQL. A LUA script for controlling the proxy for use with is provided with this build. With this sniffer option the SHOW FULL PROCESSLIST statement will not be sent and thus use of 'query sniffer' will not put any load on MySQL.
- SSH connections will now attempt more authentication methods in case the first method attempted failed. This will solve problems reported with tunneling to MySQL on FreeBSD that as per default does not support the full range of SSH authentication methods (and possible similar problems).
- Updated advisors. This is a complete and major revamp of advisors in . It is still in progress (around 80% is finished).
- Log Analyzer will expose call statements.
- In Log Analyzer ''Rows Sent' and 'Rows Examined' column will be displayed as KBs/MBs.
- Connection Details (Connection name, MySQL user, SSH user, SSH tunneling user which are saved in connection details are exposed for customization in Monitors/Advisors. (For instance 'connection name' can be accessed using MONyog.Connections.ConnectionName, 'MySQL user' as MONyog.connections.MySQLUser etc. Please refer documentation for more detailed list of all counters exposed.). Using this option counters can be made specific for specific SQL DM for MySQL connection details. For instance you can have two connections defined to the same server that will display each their set of counters.
- SQL DM for MySQL now has an ability to fully utilize the caching abilities of modern browsers. Now when refreshing a page where there is no change there will be almost no network traffic generated and the page will refresh almost instantaneously.
- SQL DM for MySQL is able to read microseconds from proxy and if available in slow query log and will (for readability) show in milliseconds.
- Sniffer output will now display MAX, first seen time, last seen time, query occurrence %. Also query occurrence % and Lock Time for a query was added to slow query log and query occurrence % for a query was added to general query log displays.
- Added an option to execute FLUSH STATUS from inside SQL DM for MySQL.

1.2.128.2 Fixed Issues:

- Once a connection in connection page was defined ssh = 'yes' for a Linux server and changed on 'no' then the GUI would not update with proper information.

1.2.128.3 Miscellaneous:

- Grouping of queries is now case insensitive for all forms of query analysis.
- The browser cache support added in beta2 has been temporary removed. There is a problem with Firefox3 with cache and AJAX (update requests were not sent from a cached page - only current cache content was reloaded). Note that FF3 users who had beta2 installed will need to clear the browser cache after installing this beta3.
- Note that the qd.lua script for PROXY-based sniffer has been updated and renamed to 'MONyog.lua'. For use with this SQL DM for MySQL version the old script must be replaced.
- Also note that Log Analyzer has been renamed to 'Query Analyzer'.

2.x SQL DM for MySQL

1.2.129 2.9 SQL DM for MySQL (December 2008)

1.2.129.1 New Features:

- This release adds a 'base time' setting in SQL DM for MySQL. This setting (if defined by user) will be used for calculation of uptime-based counters. The reason for this implementation is that if FLUSH STATUS is executed with a MySQL server, specific server status variables will be reset to the same value as would be after a server restart. However the 'uptime' status variable itself is not affected by FLUSH STATUS. And as uptime-based counters will relate the value of cumulative status variables with some initial time, using the 'uptime' variable as the initial time will result in calculation of misleading values if FLUSH STATUS was executed. So to get true uptime-based counters in SQL DM for MySQL with servers that do not support the 'uptime_since_flush_status' variable (and currently only 5.0 COMMUNITY servers from 5.0.37 do - not ENTERPRISE servers and not any other major branch than 5.0 and also not 5.0 before 5.0.37) you will need to define a 'base time' in SQL DM for MySQL greater than or equal to the time where FLUSH STATUS was executed last time. Also you can now discard data older than a specific time by using this setting. Refer to documentation for full details.
- Improved the purging logic with the Monyog embedded database. Now also system CPU load is considered and purging operations will be skipped if CPU is high. This is a further improvement to the change in purging logic introduced in version 2.5.
- SQL DM for MySQL can now be running behind an Apache server configured as 'reverse proxy'. Before this release SQL DM for MySQL JavaScript was not accessed due to absolute paths, as a result SQL DM for MySQL pages were not displaying properly.

1.2.129.2 Fixed Issues:

- Fixed high CPU usage in the 'query sniffer' if sniffer "Minimum time taken" value was set to "0" (or left blank).

1.2.130 2.83 SQL DM for MySQL (November 2008)

1.2.130.1 Fixed Issues:

- SFTP log downloads used excessive memory on Linux.
- Version 2.82 introduced an unresolved dependency with a libcrypt library on some Linux distros (including some CentOS and OpenSuSE distros).
- Installation on Windows 2000 could fail due to a dependency on a .dll not standard on this Windows version. Also this was introduced in 2.82.
- The status indicator for log downloads could display wrong for large files (> 2 GB).

1.2.131 2.82 SQL DM for MySQL (October 2008)

1.2.131.1 Fixed Issues:

- SFTP log downloads failed with the Bitvise 'WinSSHD' SSH implementation for Windows.

1.2.132 2.81 SQL DM for MySQL (October 2008)

1.2.132.1 Fixed Issues:

- Fixed an issue where data collection would continue to fail with 'database locked' error message in the log. We now roll back the query generating the lock and collection can continue (this was reported happening by a single user (only) after SQL DM for MySQL upgrade).

1.2.133 2.8 SQL DM for MySQL (September 2008)

1.2.133.1 New Features:

- Added new monitoring module displaying disk usage information for databases and tables.
- Display of slow log data in Log Analyzer had added more information (about when query occurred, information about number of rows examined and sent).

1.2.133.2 Fixed Issues:

- When a multibyte character was used in a connection name the display would garble and also the connection could not be deleted from GUI.
- When connection was not available the caption for counters should display "n/a". Since 2.71 it would instead display a string exposing an internal value (example: "NumCounterWithSeconds").

1.2.134 2.72 SQL DM for MySQL (September 2008)

1.2.134.1 Fixed Issues:

- On 64 bit Linux systems running SQL DM for MySQL 64 bit builds queries could be considered identical by Log Analyzer even if they were not.
- Authentication failed if the "#" character was used in password.

1.2.135 2.71 SQL DM for MySQL (September 2008)

1.2.135.1 Fixed Issues:

- Fixed some issues with JSON strings that could prevent objects in the SQL DM for MySQL browser interface to populate with data. Most important 'Monitors/Advisors' page would fail to display if a username contained a doublequote character.

1.2.136 2.7 SQL DM for MySQL (September 2008)

1.2.136.1 New Features:

- CPU optimizations in both SQL DM for MySQL (server) process (in particular when 'Monitors/Advisors' page was being viewed) and browser processes (viewing Dashboard could cause high CPU usage in Firefox browser in particular).

1.2.136.2 Fixed:

- Fixed a dependency with the SSH library that could cause failure to install in Win2K. This was an issue with the updated SSH library introduced with version 2.5.

1.2.136.3 Miscellaneous:

- The RPM installer scripts for Linux was updated in various ways. This solves issues on specific distributions including: 1) Uninstall with the SuSE/YaST 'software management' GUI was not possible. 2) It was not always possible to install on top of a running SQL DM for MySQL service - sometimes it was necessary to stop service manually before upgrading. Also note: when updating from trial to single/multi server or from single server to multi server it is still necessary to uninstall the old version first. And when installing this version any older version will have to be removed with the "rpm -e ..." command in advance (distribution-specific GUI tools may not work with versions before this one, so please use the RPM console command). We apologize for this but there is no other way.

1.2.137 2.6 SQL DM for MySQL (August 2008)

1.2.137.1 New Features:

- Log Analyzer code has been completely refactored resulting in better performance and stability and more maintainable code.
- The progress indicator for downloads in Log Analyzer page is now a true quantitative progress bar.
- Error messages in Log Analyzer were rewritten. Every single possible error is now differentiated by a specific message providing help for the situation.

1.2.137.2 Fixed Issues:

- With SFTP connection to the log file selecting "ALL" for log size to be downloaded in Log Analyzer page could have the result that the log file download would last forever.
- Analyzing log chunks containing BULK INSERT statements could in special situations cause a program crash.
- "Group by" option in history/trends page could deliver incorrect results if MySQL and SQL DM for MySQL were using different time-zones.
- Fixed a bug with resolvement of time in log analyzer that could result that no queries or less queries than what should be were returned. The most recent queries would sometimes not be shown. Issue was only if log was retrieved with SFTP.
- When retrieving logs stored in tables the substitution of literals would fail for binary data .

- Installing a RPM multi/single server builds on top of a trial build (and vice versa) resulted in that they were installed in parallel. Also on SUSE SQL DM for MySQL was not registered with the correct run level. The RPM has been updated to fix this.
- When downloading a log SQL DM for MySQL would continue to download in the background until end of file even after STOP button was clicked.
- In the display of queries from the Processlist page special characters could garble.
- Also in Log Analyzer page special characters would not always be identified correctly. This bug also affected records with binary data. Both display and grouping was affected (depending on where those occurred - in plain strings or in identifiers).
- Repeatedly switching between sorting and show details for a slow query log in Log Analyzer could in certain situations crash SQL DM for MySQL.

1.2.137.3 Miscellaneous:

- Display of servers is now sorted in the order of the server/connection name. Earlier the internal ID was used for sorting. Also note that now no SQL DM for MySQL restart is required for changing sort order. A connection rename alone will do.

1.2.138 2.51 SQL DM for MySQL (July 2008)

1.2.138.1 Fixed Issues:

- Depending on the chosen installation environment and the Linux distro, the RPM build might not install 'on top' of a previous installation. Before installing a new version it was necessary to uninstall the previous version. This is a limitation with all RPM builds released till now. From this build and onwards it will always be possible to 'upgrade on top' of a previous version. Note: this was an issue with the RPM installer - not the program binary. There is no change in the binary as such and also no change in the Windows installer and the .gz build for Linux.

1.2.139 2.5 SQL DM for MySQL (July 2008)

1.2.139.1 New Features:

- MySQL Server Log Analysis: SQL DM for MySQL is now able to retrieve (completely or partially) the General Query Log and the Slow Query Log (whether stored as files or tables on the server) from the MySQL servers it connects to and analyze them.
- Query Sniffer: A 'query sniffer' is implemented that will record a 'pseudo log' on the client (SQL DM for MySQL) side. The sniffer is useful when 'real logs' are not available, but also the sniffer has various filtering options what will sometimes make it more useful than the 'real server logs' also when they are available.

The log analysis component will (whether operating on a 'real log' or a 'sniffer pseudo log') only display identical queries once but print a 'count' for every query. Important information (like execution time, number of instances for this query etc.) will be displayed in a sortable table view.

In the log analyzer 'Filter settings' (not to be confused the filter settings of the 'query sniffer') there further is an option to 'replace literals'. The purpose of this option is to eliminate small differences between 'almost identical' queries. Currently 'quoted strings' and numbers are replaced with the dummy string 'XXX' only. We may extend this feature to support more (like special keywords and functions (like NULL, now()), operators etc.) in the next releases. It will depend on the demand for this as compared to the demand for other features. Also note that the current implementation has the 'imperfection' that also numbers that are not values (like the '1' in identifiers like `column1`) will be replaced.

- More Linux system counters added: Counters related to Linux memory (including swap) usage added.
- RPM build now supports SUSE and Mandrake/Mandriva Linux distributions.

1.2.139.2 Fixed Issues:

- Under specific circumstances the SQL DM for MySQL service would stop very slow.
- If network connection was temporarily unavailable the first CPU value after network came back told 100% CPU load.
- The temporary files for storing public/private keys are now being created with very restrictive file permission in Linux. So only the owner will be able to read/write these temporary files. Although these files would exist only for a fraction of second, in earlier versions of SQL DM for MySQL other users in the same Linux system could get access to them within that small timeslice.
- Fixed an issue with trending where graphs would not always display properly for uptime-based counters.
- MONyog.ini file and SQL DM for MySQL database files on Linux are now only readable for user who starts/installs (.gz and RPM build respectively) SQL DM for MySQL (and of course for any user too that was given access by a root user). Before the files were world-readable. This applies to fresh installations only - there is no change with existing installations.
- Solved an issue where slow startup could occur if lots of servers were registered (Technical: If SQL DM for MySQL had not been running for a while, large amounts of data could require deletion from the Monyog database. The queries used to identify those data would generate significant I/O at startup. The retention process is now 'smoothed over time' and indexes have been added for more efficient deletions).

1.2.140 2.06 SQL DM for MySQL (June 2008)

1.2.140.1 Fixed Issues:

- Fixed: The new version 3 release of Mozilla Firefox (now in RC) has a flickering issue with SQL DM for MySQL and the login page did not load correctly in that browser version.

1.2.141 2.05 SQL DM for MySQL (May 2008)

1.2.141.1 Fixed Issues:

- Fixed: OS monitoring issue in Fedora Core 6 Zod.

1.2.142 2.04 SQL DM for MySQL (May 2008)

Performance improvement:

- Earlier SQL DM for MySQL was semaphore intensive. That could result in reaching the 'limit of semaphores that can be created by the kernel' in some systems. SQL DM for MySQL 2.04 will use less semaphores. Another symptom was that a large number of small and zero-size files could be left behind (in system TEMP folder on Windows and in MONyog/bin folder on Linux). This is also solved with this.

1.2.143 2.03 SQL DM for MySQL (May 2008)

1.2.143.1 Fixed Issues:

- In some situations where SSH connections failed (including if SQL DM for MySQL was not able to connect to MySQL through tunnel, wrong SSH authentication details for tunneling etc.), sockets were not being closed and it could result in that many sockets were kept in CLOSE_WAIT state. (Note: Connections to MySQL not using SSH were not affected!)

1.2.144 2.02 SQL DM for MySQL (April 2008)

1.2.144.1 Fixed Issues:

- In the 'delta' timeframe CPU load displayed was incorrect.

1.2.145 2.01 SQL DM for MySQL (April 2008)

1.2.145.1 Fixed Issues:

- Whenever Dashboard was drawn for the first time or redrawn with new configuration settings all nodes displaying CPU load were erroneously drawn with the latest available value. New values appearing after that were drawn correctly. Only CPU load counter was affected.

1.2.146 2.0 SQL DM for MySQL (March 2008)

1.2.146.1 New Features:

- SSH tunneling to MySQL servers.
- Analysis of historical data (trends).
- Dashboard graph size is now configurable.
- You can choose to display x-axis values or not in Dashboard (in 'large' display mode). Also remember that no matter if axis-values are displaying you can retrieve values by pointing the mouse to a node in the charts.
- The no. of samples displayed in dashboard is configurable.
- Above dashboard settings can be changed from the dashboard page itself.
- 64 bit binaries for Linux included.
- Option to display counters as 'value/sec'.
- Reorganized the connection page for better clarity.
- Performance improvements.
- JavaScript code was 'broken down' to smaller units for easier customization. Every counter group now has its own 'custom' script.
- The menu was reorganized and other GUI improvements for easier navigation between the different functionalities.
- The 'looks and feel' (of graphics in particular) has been vastly improved.
- Documentation/help was updated and reorganized with better structure, more hyperlinks - and also now much more better looking.

1.2.146.2 Fixed Issues:

- SQL DM for MySQL would not install on some older Linux distro's (including RHEL3).
- In Linux it was possible to start more identically named Monyog instances. This is now prevented.
- When stopping SQL DM for MySQL service it could happen that the service was reported to be stopped before the task was complete. Starting service in this situation would return an error. Now it will report 'running' as long as it is not fully stopped.
- Fixed an issue with the shell script for non-RPM Linux where it could be reported that service had started when it had not.
- Fixed some calculation bugs (affected 'hit rate' and 'percentage of used blocks' counters).

1.2.146.3 Miscellaneous:

- Help/documentation page will open in a separate browser window or tab (depending on browser setting).
- 'Show All' page was renamed to 'Monitors/Advisors'.
- System counters have been renamed to 'Linux counters'.

More details on implementation of SSH-tunneling and History/Trends analysis:

- SSH-tunneling: The implementation does (unlike current SQLyog implementation) not start any external process and does not occupy a local port. The performance for this 'home grown' SSH-tunnel wrapper is very high (you may find it faster than direct connection to MySQL on a remote host)!
- History/Trends analysis: This is implemented in the 'Monitors/Advisors' page. You can select whatever time interval you want for the analysis. Counters that are UPTIME based (or if you like: CUMULATIVE counters as opposed to SNAPSHOT counters) can be aggregated using SUM function and common time intervals (HOUR, DAY, MONTH etc.). This is semantically similar to the query "SELECT SUM () ... GROUP BY 📅 " (but the real query behind this interface is much more complex of course!).

1.x SQL DM for MySQL

1.2.147 1.52 SQL DM for MySQL (February 2008)

1.2.147.1 Fixed Issues:

- Installing recent versions on top of 1.1 or earlier could disable connection to the OS (not to MySQL) of servers already registered (this bug was introduced when we started supporting key authentication with 1.5).
- Alert icons (red and yellow circles) were disappearing from the group in 'Show All' interface if the user clicked the next group.

1.2.148 1.51 SQL DM for MySQL (January 2008)

1.2.148.1 Fixed Issues:

- Adding a server with notification enabled was storing wrong information in the 'connection.data' configuration file. After a restart of the SQL DM for MySQL service the wrong information that was read from that file would cause notifications to stop working. Note that if a connection was configured in 'two steps' where notification settings were added to an existing connection there was no error.

- Fixed a memory leak occurring with public/private key authentication when connection was unsuccessful (server not available, invalid key pair etc.).
- The shortcuts installed by SQL DM for MySQL installers now point directly to '127.0.0.1' (and not 'localhost'). This solves an issue with Firefox on Windows Vista where resolution of the URL 'localhost' was very slow.

1.2.149 1.51 SQL DM for MySQL (January 2008)

1.2.149.1 Fixed Issues:

- Adding a server with notification enabled was storing wrong information in the 'connection.data' configuration file. After a restart of the SQL DM for MySQL service the wrong information that was read from that file would cause notifications to stop working. Note that if a connection was configured in 'two steps' where notification settings were added to an existing connection there was no error.
- Fixed a memory leak occurring with public/private key authentication when connection was unsuccessful (server not available, invalid key pair etc.).
- The shortcuts installed by SQL DM for MySQL installers now point directly to '127.0.0.1' (and not 'localhost'). This solves an issue with Firefox on Windows Vista where resolution of the URL 'localhost' was very slow.

1.2.150 1.5 SQL DM for MySQL (December 2007)

1.2.150.1 New Features:

- Major speed improvements (in both data collection and as regards browser responsiveness).
- Private/Public key authentication is now supported with SSH connection to remote hosts. Key format must be OpenSSH standard key format - special formats (like the one used by the 'Putty' program) are not supported.
- Added security-related counters and advisors.
- Added counters and advisors for schema and table operations.
- PROCESSLIST display will not by default 'filter out'. A button has been added in the 'Change Filter' dialog to switch between a non-filtered result and the (old) filtered result for the PROCESSLIST page.
- PROCESSLIST can be paused/resumed.
- Default SSH timeout (3 minutes) was replaced with 30 seconds.

1.2.150.2 Fixed Issues:

- Fixed some small memory leaks.
- Deleting a connection did not remove all the files belonging to that connection.
- Fixed a rare issue where browser could report that the page cannot be displayed while saving/editing server details or connection (IE issue only).
- Fixed an issue with system counters where 100% CPU usage erroneously could be reported when SQL DM for MySQL started.

1.2.151 1.11 SQL DM for MySQL (October 2007)

1.2.151.1 Fixed Issues:

- HTML formatting of queries in the PROCESSLIST tab was not correct. As a result display of queries could be truncated.

1.2.152 1.1 SQL DM for MySQL (October 2007)

1.2.152.1 New Features:

- Support for SHOW FULL PROCESSLIST - SQL DM for MySQL monitor processes on the servers in real time. Various filtering options for PROCESSLIST are available. Further displays, copy and EXPLAIN (EXTENDED) a running query from inside SQL DM for MySQL.
- Documentation was updated - including information about the PROCESSLIST feature, up-to-date screenshots etc.
- Counters and Advisors - Small adjustments!
- Debug DUMP facility - SQL DM for MySQL (Windows version) has an ability to store crash-dumps fit for debugging similar to SQLyog as explained here: Debug Facility[9].
- Menu rearrangement etc. - or clarity we have decreased the number of menus displaying. Those that 'went away' are available as links or submenus.

1.2.153 1.04 SQL DM for MySQL (August 2007)

1.2.153.1 New Features:

- Improved counters and advisors.
- Customize page now will show the .js-components that user changed with a red * (star). Note that existing .js-components will be overwritten when you upgrade. The red star will help you to see what you need to back up before upgrading.
- Default port is now 5555. With certain names used for the connections there was a conflict with a standard 'rule' of Norton Internet Security when port 9999 was used.
- The option to display graphs from the 'monitors/advisors... custom timeframe' interface was temporarily removed.

1.2.153.2 Fixed Issues:

- Fixed a crashing bug in mailing section if a .js-component was corrupted.
- Adding the option to send mail after registering a server required SQL DM for MySQL service to be restarted.

[9] http://wiki.idera.com/x/6QAnBg

1.2.154 1.03 SQL DM for MySQL (August 2007)

1.2.154.1 New Features:

- A BELL icon displays in the 'Monitors/Advisors' page for alertable counters.
- Various small GUI improvements.
- Documentation updated with details about system Objects + various small corrections.

1.2.154.2 Fixed Issues:

- %age of queries doing full table scan metric was incorrect.
- With very small mail interval setting it could occur that selected servers became deselected.
- Fixed a bug in the calculation of full table scans.

1.2.155 1.02 SQL DM for MySQL (August 2007)

1.2.155.1 Fixed Issues:

- Fixes an issue in previous 1.x releases that may result in too early expiry of the TRIAL.

1.2.156 1.01 SQL DM for MySQL (August 2007)

1.2.156.1 Fixed Issues:

- Replication counters did not display in LINUX builds (both .rpm and .tar.gz).

Additional note on file storage positions and upgrading:

Version 1.0 RC changed the storage position for various files used by SQL DM for MySQL. This release is in this respect identical to 1.0 RC and 1.0.

By default the embedded database is now stored:

- Windows 2K/XP/2003: {System_drive}:\Documents and Settings\All Users\ApplicationData\Webyog\MONyog\Data
- Windows Vista:{System_drive}:\ProgramData\Webyog\MONyog\Data

... but user can specify another location if for instance she does not want to store on the system disk/partition at all.

Also the monyog.ini file and the log file is now no longer stored in 'program files' file tree (or whatever localized name it has). The storage position is:

- Windows 2K/XP/2003: {System_drive}:\Documents and Settings\All Users\ApplicationData\Webyog\MONyog
- Windows Vista:{System_drive}:\ProgramData\Webyog\MONyog

Users upgrading from 0.9 who want to use existing data must after installing this version:

1. Stop SQL DM for MySQL service.
2. Copy the folders containing databases named with a number like "0001" or "0002".
3. Start SQL DM for MySQL service.

No such intervention is required when upgrading from version 1.0 RC or 1.0 (unless storage position is changed again, of course!).

Upgrading from versions before 0.9 is not possible. These versions must be uninstalled and installation folder deleted or this version must be installed to another folder.

1.2.157 1.0 SQL DM for MySQL (August 2007)

1.2.157.1 New Features:

- Installs and runs on Windows 2000.
- Every counter is available in every timeframe.
- User can specify where the embedded database is stored.

1.2.157.2 Fixed Issues:

- Retrieval of memory information from Linux servers was broken in 1.0 RC.
- Windows version: Solved an issue with a dependency on a .dll file that was reported missing on some systems.

Additional note on file storage positions:

Version 1.0 RC changed the storage position for various files used by SQL DM for MySQL. This release is in this respect identical to 1.0 RC.

By default the embedded database is now stored:

- Windows 2K/XP/2003: {System_drive}:\Documents and Settings\All Users\ApplicationData\Webyog\MONyog\Data
- Windows Vista:{System_drive}:\ProgramData\Webyog\MONyog\Data

... but user can specify another location if for instance she does not want to store on the system disk/partition at all.

Also the monyog.ini file and the log file is now no longer stored in 'program files' file tree (or whatever localized name it has). The storage position is:

- Windows 2K/XP/2003: {System_drive}:\Documents and Settings\All Users\ApplicationData\Webyog\MONyog
- Windows Vista:{System_drive}:\ProgramData\Webyog\MONyog

Users upgrading from 0.9 who want to use existing data must after installing this version.

1. Stop SQL DM for MySQL service.
2. Copy the folders containing databases named with a number like "0001" or "0002".
3. Start SQL DM for MySQL service.

No such intervention is required when upgrading from version 1.0 RC (unless storage position is changed again, of course!).

Upgrading from versions before 0.9 is not possible. These versions must be uninstalled and installation folder deleted or this version must be installed to another folder.

0.x SQL DM for MySQL

1.2.158 0.9 SQL DM for MySQL Beta1 (July 2007)

Note that this release is a TRIAL build. It has the restrictions that,

1. Maximum 2 servers can be registered.
2. It will expire on August 15th 2007.

Before expiry of this beta we plan to release version 1.0 FINAL and publish purchasing and licensing terms.

Also notice that the schema of the embedded database has changed compared to previous versions. With this beta we will not provide a database upgrade script. If you have a previous version installed you must either,

- install to another folder.
- or uninstall the old version.

1.2.158.1 New Features:

- Authentication is in place. SQL DM for MySQL pages can be watched from another machine where SQL DM for MySQL service is running. SQL DM for MySQL pages can be watched from another machine where SQL DM for MySQL service is running.
- Two-level alerts (WARNING/CRITICAL).
- Mail alerts.
- Various timeframes available for display of data in 'Monitors/Advisors'.
- Replication counters.
- Documentation (1st version) included.
- And lots of minor improvements.

Known issues:

- Installation on Windows 2000 is (still) not possible (due to a dependency in the SSH library that we use). We hope to fix this soon.
- We are aware of some (rare) problems that can occur on Windows Vista including:
    a. Graphs will not populate with old (stored) data.
    b. Pages will not update properly.

1.2.159 0.20 SQL DM for MySQL (May 2007)

1.2.159.1 Fixed Issues:

- Fixed a crashing issue while installing the SQL DM for MySQL over existing SQL DM for MySQL installation.
- Introduced locks to make thread safe.
- Fixed a leak of GDI's (Graphics Device Interfaces). Number of open GDI's would increase and when SQL DM for MySQL was running for very long time, SQL DM for MySQL service could ultimately crash.
- Fixed few JavaScript issues.

1.2.160 0.16 SQL DM for MySQL (March 2007)

1.2.160.1 Fixed Issues:

- AJAX is now fully implemented in the Monyog main screen.
- Improved advisors - in particular regarding the Query Cache.
- Fixed all reported (and some non-reported) GUI issues.
- Lots of code changes for better structure, performance and maintainability.

1.3 Known Issues

IDERA strives to ensure our products provide quality solutions for your SQL Server needs. If you need further assistance with any issue, please contact Customer Support Portal(see page 95).

In this section you can find information of all the Known Issues existing for this version.

1.3.1 8.9.4 Known Issues

There are no Known Issues reported for this version.

1.3.2 8.9.3 Known Issues

There are no Known Issues reported for this version.

1.3.3 8.9.2 Known Issues

There are no Known Issues reported for this version.

1.3.4 8.9.1 Known Issues

There are no Known Issues reported for this version.

1.3.5 8.9 Known Issues

There are no Known Issues reported for this version.

2 Get started

SQL Diagnostic Manager for MySQL is a monitor and advisor application for MySQL database administrations. It provides you with tools to manage more database servers, to tune your current database infrastructure, and to aid you in finding and fixing problems with your database applications before they develop into more serious issues or costly outages.

With SQL Diagnostic Manager for MySQL, you can proactively monitor enterprise database environments and obtain expert advice so that even administrators new to MySQL can tighten security, optimize performance and reduce downtime of their MySQL powered systems.

2.1 Get Started with SQL Diagnostic Manager for MySQL

- Introduction[10]
- Security and authentication[11]
- Architecture[12]
- Installation[13]
- Configuration[14]
- Server Registration[15]

2.2 Introduction

SQL Diagnostic Manager for MySQL provides monitors and advisors for MySQL database administrators, helping you to manage more database servers, tune your database infrastructure, and identify any issues with your database applications. It monitors enterprise environments and provides expert advise so that even someone new to MySQL can tighten security, optimize performance, and reduce downtime of MySQL powered systems.

2.2.1 Features

In technical terms, SQL DM for MySQL is essentially a web server with a very low resource footprint. As such, it has much in common with popular web servers such as Apache, Nginx, IIS, Lighttpd and so on, but there are some important differences.

In order to access SQL DM for MySQL, you need a web browser that is capable of handling AJAX. This has been tested on Google Chrome, Internet Explorer (versions 11.0.96 and later), Mozilla Firefox and related browsers, and all recent releases of the Opera browser.

- Monitoring Optimization(see page 97)
- Unlimited Servers Side-by-Side(see page 97)
- Scripting with JavaScript(see page 97)
- Agent-less Monitoring(see page 97)
- Built-in High Performance Database(see page 98)

[10] http://wiki.idera.com/display/SQLDMYSQL/Introduction
[11] http://wiki.idera.com/display/SQLDMYSQL/Security+and+Authentication
[12] http://wiki.idera.com/display/SQLDMYSQL/Architecture
[13] http://wiki.idera.com/display/SQLDMYSQL/Installation
[14] http://wiki.idera.com/display/SQLDMYSQL/Configuration
[15] http://wiki.idera.com/display/SQLDMYSQL/Server+Registration

2.2.1.1 Monitoring Optimization

Where web servers are general purpose applications, SQL DM for MySQL is a more specialized server application. It is built for collecting and storing information on MySQL database servers and the systems that host them. You can not use it to host your own webpages as it is locked to only display those pages that the SQL DM for MySQL application was designed to serve.

SQL DM for MySQL uses the Asynchronous JavaScript and XML (AJAX) language to communicate with the server and generate pages divided into small, independent objects that it can then individually update as required. This reduces bandwidth consumption, given that you do not need to reload the entire page for updates and allows you to display data on as many MySQL servers as you want.

Each of the HTML objects in SQL DM for MySQL AJAX pages refresh automatically when new data becomes available. This refresh is independent for each MySQL server being displayed on the page. Refreshes happen at the server-level.

2.2.1.2 Unlimited Servers Side-by-Side

When using SQL DM for MySQL Enterprise, Charts show real-time graphs of all import metrics that provide a consolidated view into the availability and performance of all the MySQL servers across the enterprise. From these real-time charts, the MySQL database administrator can instantly see:

- The availability status of all MySQL servers,
- Import operating system metrics that may affect MySQL,
- Side-by-side comparison of similar servers,
- Which MySQL servers need attention, and
- Where and how they need to spend limited time.

2.2.1.3 Scripting with JavaScript

SQL DM for MySQL includes an embedded JavaScript engine, similar to most modern browsers. The entire SQL DM for MySQL application logic is handled through JavaScript objects, such as defining threshold values, calculating performance metrics and sending alerts. The complete source code for these JavaScript objects is available for customization, making them fully editable, extensible, and configurable as your use-cases require.

2.2.1.4 Agent-less Monitoring

Typically, monitoring and advisory tools use external PHP or Java agents to allow traditional web servers to connect to MySQL. Unlike these tools, SQL DM for MySQL is compiled with the MySQL client code through the C API, allowing to connect directly to MySQL database servers without needing agents.

This greatly simplifies SQL DM for MySQL deployments, given that you don not need to install and maintain agents on each MySQl host. Instead, SQL DM for MySQL uses the MySQL client to connect to and retrieve database information. To retrieve operating system-level metrics it uses SSH when connecting to Linux hosts.

You install SQL DM for MySQL on one host and that's it. It can retrieve whatever information it needs using the same sources and methods as a database administrator.

2.2.1.5 Built-in High Performance Database

Internally, SQL DM for MySQL manages the data it collects using a high-performance embedded database. Whatever server parameters it retrieves from MySQL servers are also stored in this database. Various methods in displaying counters or metrics are based on the data it provides.

2.3 Security and Authentication

The fact that SQL DM for MySQL is a dedicated and specialized web server also makes it very secure.

- **SQL DM for MySQL - by design is very secure:** It is practically immune to any kind of malicious code. Whether they be unwanted popups, advertisements, trojans, attempts at phishing, hijacking or the like. Any HTML pages with such content are not sent by SQL DM.
- **Password protection:** During install you (or your SysAdmin) provided a password. Your browser prompts you for this password whenever you connect to the SQL DM for MySQL service. You cannot access the stream of data from the SQL DM for MySQL service if you are not able to provide the correct password.
- **Additional security considerations:** Note that on the server the SQL DM for MySQL password is stored obfuscated in the MONyog.ini file . Also, the SQL DM for MySQL embedded database has stored credentials (user IDs and passwords for MySQL servers, SSH, and SMTP servers).

The SQL DM for MySQL authentication system to work, must have cookies enabled in the browser. A 'medium high' security setting are appropriate for most users. The exact setting (and how it is named) varies from browser to browser.

2.4 Architecture

In technical terms, SQL DM for MySQL is basically a very low footprint web server. As such it has much in common with popular web-servers like Apache, IIS, Lighttpd etc., but there are also important and significant differences.

SQL DM for MySQL is a dedicated and specialized server program for collecting and storing information concerning MySQL servers and the systems on which they are installed. You can not use it to display your own webpage. It is 'locked' to display the pages that SQL DM for MySQL program is designed for.

✓ SQL DM for MySQL is basically a low footprint web server that is optimized for Monitoring MySQL servers.

2.4.1 SQL DM for MySQL uses state-of-the-art web technology

These pages use AJAX (Asynchronous JavaScript and XML) to communicate with SQL DM for MySQL server. With AJAX, it is possible to generate pages that are divided into small independent objects that are updated individually when required. This reduces the bandwidth consumption as compared to traditional method of reloading the whole page. And well-designed AJAX pages provide a much more pleasant user experience than traditional HTML pages. You can display and compare as many MySQL servers simultaneously as you want. It is handled by the browser itself. Also, you will need not reload the page to display the latest data. Each of the small HTML objects in SQL DMfor MySQL AJAX pages refresh automatically when new data are available. And this refresh is done independently for every MySQL server that is displayed. Actually, this refresh happens at the server level which means for "Server A" interval is 15 seconds to collect server data and for "Server B" it is 10 minutes, then the respective display of data for every server refreshes after that interval.

2.4.2 SQL DM for MySQL is configurable using the de-facto web scripting language: JavaScript

SQL DM for MySQL also includes an embedded JavaScript (JS) engine - not unlike most modern browsers. In fact, the entire SQL DM for MySQL "application logic" like defining threshold values, calculating performance metrics, and sending alerts are defined as JS objects. The complete source code of these JS objects is available to the user for customization. The JavaScript objects are fully editable, extensible, and configurable.

2.4.3 SQL DM connects to MySQL directly: no JAVA, no PHP, no 'agents' install required

Also, unlike other web-servers SQL DM for MySQL has the MariaDB Connector/C 'compiled in'. It can connect to MySQL directly without depending on any sort of external code (PHP, JAVA or whatever) as other web-servers do. Also, it does not make use of any 'agents' on the computers where the MySQL servers are running. You install SQL DM for MySQL on one computer and that is it!

2.4.4 SQL DM for MySQL has a built-in high-performance database

Further! SQL DM for MySQL has a built-in high-performance embedded database. Whatever server parameters are retrieved from the MySQL servers are also stored in that database. Various ways to display 'counters' or 'metrics' based on these data are provided.

2.4.5 SQL DM for MySQL is compatible with all modern browsers

All you need, to view the results that SQL DM for MySQL throws up, is a web browser that is capable of handling AJAX. This has been tested on Google Chrome, Internet Explorer (versions 11.09 and later), Mozilla Firefox and related browsers, and all recent releases of the Opera browser.

2.5 SQL DM for MySQL Requirements

Consider the following requirements when installing SQL Diagnostic Manager for MySQL.

Type	Requirement
Operating System	Windows (32-bit): • Windows Server 2008 • Windows Server 2008 R2 • Windows Server 2012 • Windows Server 2012 R2 • Windows Server 2016 • Windows Vista • Windows 7 • Windows 8 • Windows 8.1 Linux (64-bit): • All distributions • Kernel 2.6.32
Database Platforms	Oracle MySQL Enterprise 5.1, 5.5, 5.6, 5.7, 5.8 MySQL Community Server 5.1, 5.5, 5.6, 5.7, 8.0 MariaDB (5.1, 5.2, 5.3) 5.5, 10, 10.1, 10.2, 10.3 Percona Server for MySQL 5.1, 5.5, 5.6, 5.7 Galera Cluster for MySQL 3 MariaDB Galera Cluster 5.5, 10.0 Amazon RDS for MySQL, Amazon RDS for MariaDB, Amazon for Amazon Aurora Azure MySQL Google Cloud SQL For MySQL Oracle MySQL Cloud Service

2.5.1 SQL Diagnostic Manager for MySQL hardware sizing guidelines

The following guidelines provide an estimation of the hardware resources required to deploy SQL Diagnostic Manager for MySQL depending on the number of servers you want to monitor.

2.5.1.1 Small and medium size deployments for less than 500 MySQL servers

Type	Requirement
RAM	4GB
CPU	2 Cores

Type	Requirement
Storage Space	5GB per monitored MySQL Server

2.5.1.2 Large size deployments for more than 500 MySQL Servers

Type	Requirement
RAM	32GB
CPU	16 Cores
Storage Space	1.5 TB

2.6 Installation

SQL DM for MySQL is supported on Microsoft Windows and Linux operating systems. Once installed, you can access it from any AJAX-supported web browser, including those available on mobile devices.

Once you have SQL DM for MySQL installed, it can connect to and collect data from MySQL database servers running on any platform. When connecting to a MySQL server running on Windows, SQL DM for MySQL can use the Windows Management Instrumentation to retrieve additional information. When connecting to a server running on Linux, it can do the same using SSH.

⚠ SQL DM for MySQL can only retrieve system information from Linux servers.

2.6.1 Downloading SQL Diagnostic Manager for MySQL

Currently, SQL DM for MySQL is supported on Microsoft Windows 2008 and later, and on Linux operating systems. It does not support specific distributions of Linux, but provides a package for those based on RPM standards.

2.6.1.1 Getting SQL Diagnostic Manager for MySQL

To obtain the download links for SQL DM for MySQL, you need to purchase a license or request a free trial from IDERA.

Once you have purchased the license or requested a free trial, you receive links to the installation media by email. For more information on the different levels of support see Pricing(see page 101) or Trial Center(see page 101).

2.6.1.2 Choosing the Correct Download

When you receive the download links, there are several available for you to use. The specific link you need depends on the system you want to run SQL DM for MySQL on, both in terms of hardware architecture and the operating system.

- Windows

If you want to use SQL DM for MySQL on a Windows Server installation, there is only one link available to you. Select the **Executable (.exe)** link and follow the instructions on Windows Installation to install SQL DM for MySQL. For Linux installations, it is a little more complicated.

- Linux

There are four download links available for Linux installations, divided by distribution and hardware architecture. Select the download link that matches your installation. Currently, IDERA only provides RPM-based packages. In the event that you use Fedora, CentOS, Red Hat Enterprise Linux, openSUSE, SUSE Linux Enterprise Server, or a similar RPM-based Linux distribution, select the **RPM** download link for your hardware. For all other distributions, select the **TAR** download link.

> ⓘ SQL DM for MySQL runs on both 32-bit and 64-bit operating systems. In the event that you want to install SQL DM on a server and do not know which architecture it uses, you can find out with the `uname` command:
>
> ```
> $ uname -a Linux actual 3.16.0-4-amd64 #1 SMP Debian 3.16.43-2(2017-04-30) x86_64
> GNU/Linux
> ```
>
> Reference to `amd64` or `x86_64` indicate a 64-bit operating system. `i386` or `i686` indicate 32-bit.

2.6.2 Installing SQL Diagnostic Manager for MySQL

Once you have downloaded the package for your system, you can install it on your system. The specific installation directions depends on the operating system you want to use to host SQL DM for MySQL:

- Linux[16]
- Windows[17]

2.6.3 Linux

SQL DM for MySQL runs as a web application on a server, from which it connects to and monitors MySQL database servers. The MySQL servers can run on any operating system. SQL DM for MySQL can run on Linux <install-windows> or any distribution of the Linux operating system.

2.6.3.1 Installing SQL DM for MySQL

Webyog provides two types of installation media for Linux operating systems: RPM and TAR. Each is available for 32-bit and 64-bit architectures. For RPM-based Linux distributions, you can install SQL DM for MySQL through the package manager or the `rpm` command. For all other distributions, you can manually install SQL DM for MySQL from a tarball.

[16] http://wiki.idera.com/display/SQLDMYSQL/Linux
[17] http://wiki.idera.com/display/SQLDMYSQL/Windows

Unlike the Windows installation, neither of these methods prompts the user to supply a default password or port number. By default, SQL DM for MySQL runs on port 5555, with the user admin and a blank password. You can change them through the web UI later.

Installation

The link downloads an `.rpm` package. The filename pattern is `IderaSQLdmforMySQL-<version>.<architecture>.rpm`. You can install SQL DM for MySQL by calling the package manager on this file.

- For Fedora and newer Red Hat-based distros, you can install SQL DM for MySQL with DNF:

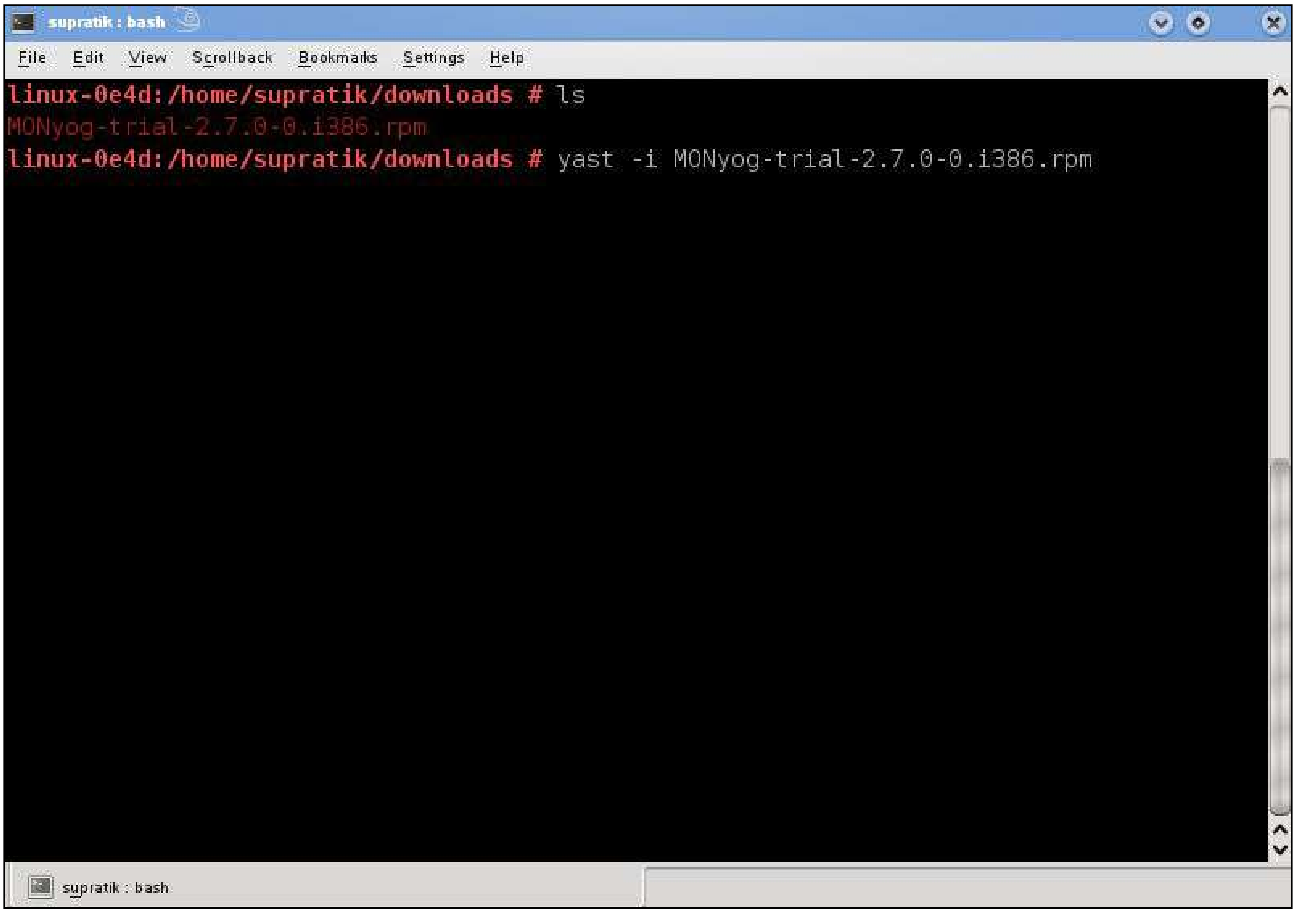
```
# dnf install /path/to/IderaSQLdmforMySQL-*.rpm
```

- For older Red Hat-based distros, you can use YUM:

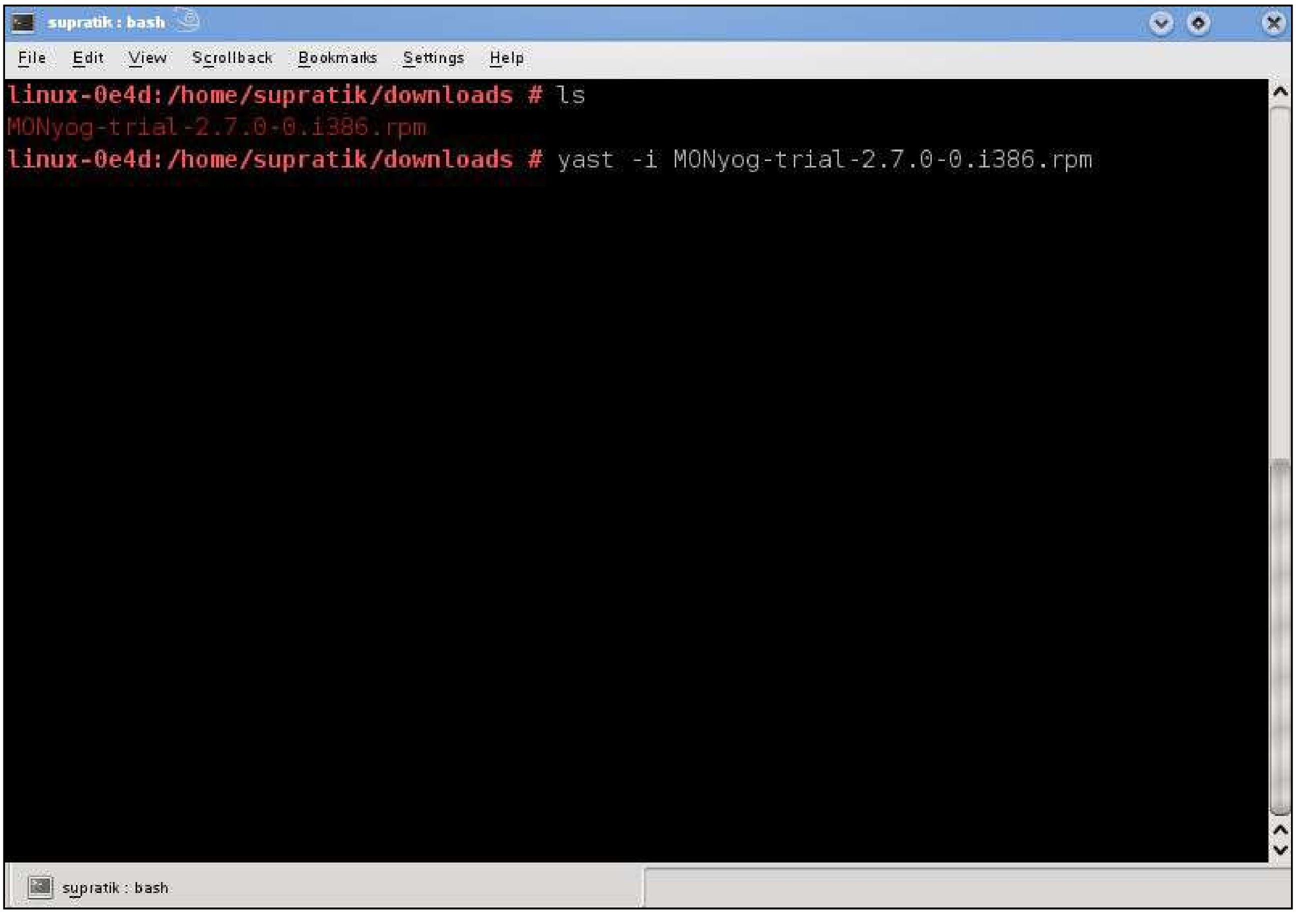
```
# yum install /path/to/IderaSQLdmforMySQL-*.rpm
```

- For openSUSE and SUSE-based distros, use YaST:

```
# yast -i /path/to/IderaSQLdmforMySQL-*.rpm
```



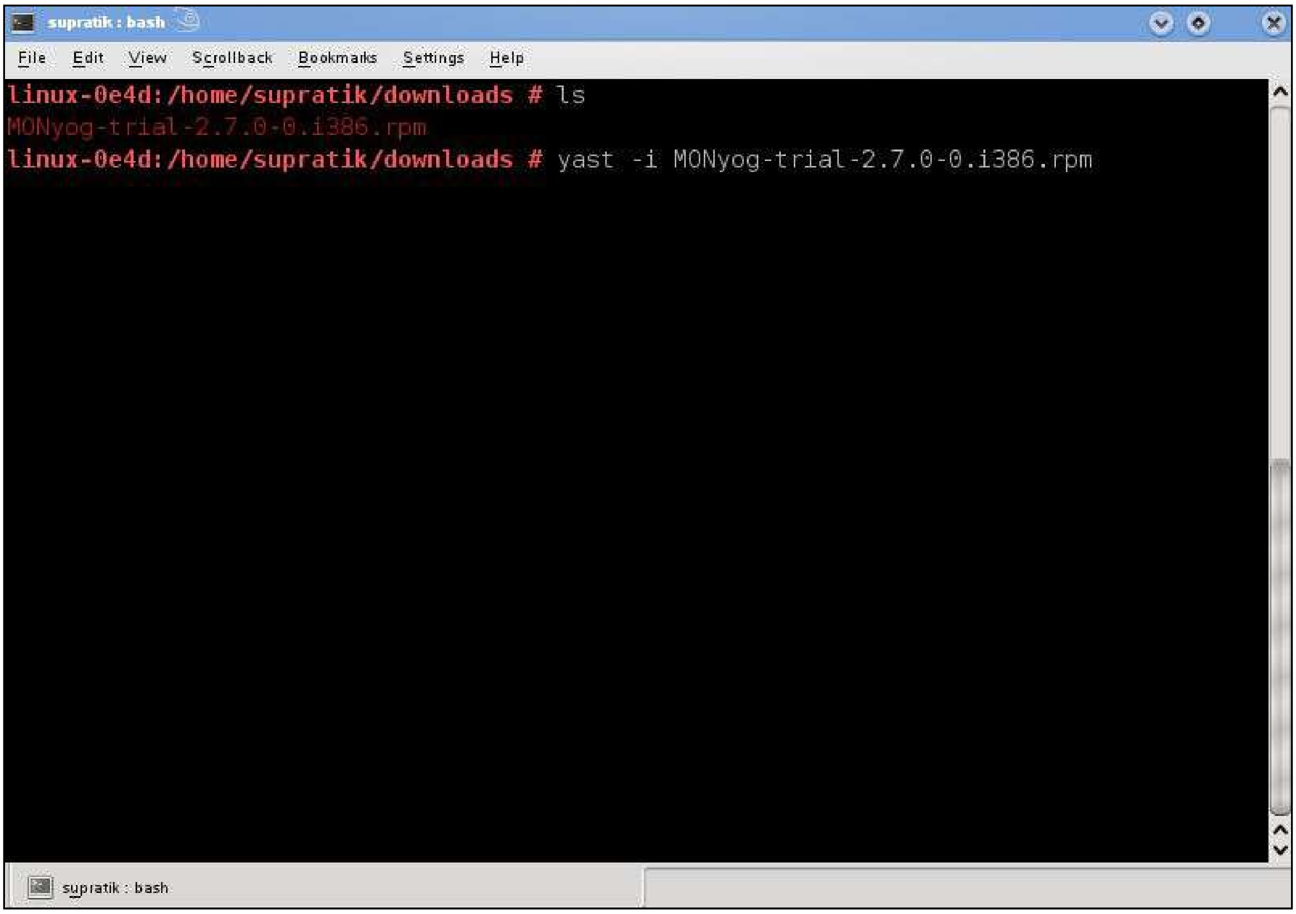
```
linux-0e4d:/home/supratik/downloads # ls
MONyog-trial-2.7.0-0.i386.rpm
linux-0e4d:/home/supratik/downloads # yast -i MONyog-trial-2.7.0-0.i386.rpm
```



```
linux-0e4d:/home/supratik/downloads # ls
MONyog-trial-2.7.0-0.i386.rpm
linux-0e4d:/home/supratik/downloads # yast -i MONyog-trial-2.7.0-0.i386.rpm
```

- For any RPM-based distro, you can also use RPM itself:

```
# rpm -ivh /path/to/IderaSQLdmforMySQL-*.rpm
```



```
linux-0e4d:/home/supratik/downloads # ls
MONyog-trial-2.7.0-0.i386.rpm
linux-0e4d:/home/supratik/downloads # yast -i MONyog-trial-2.7.0-0.i386.rpm
```

Whichever package manager you use, it installs SQL DM for MySQL in the `/usr/local/MONyog/` directory. You can now begin configuring and using SQL DM for MySQL.

Start/Stop

- In Redhat and Fedora systems if SQL DM for MySQL is installed from the RPM package, the daemon script 'MONyogd' can be used to start/stop the server. This script is in"/etc/init.d/" directory.

  You can use one of the following command to start SQL DM for MySQL:

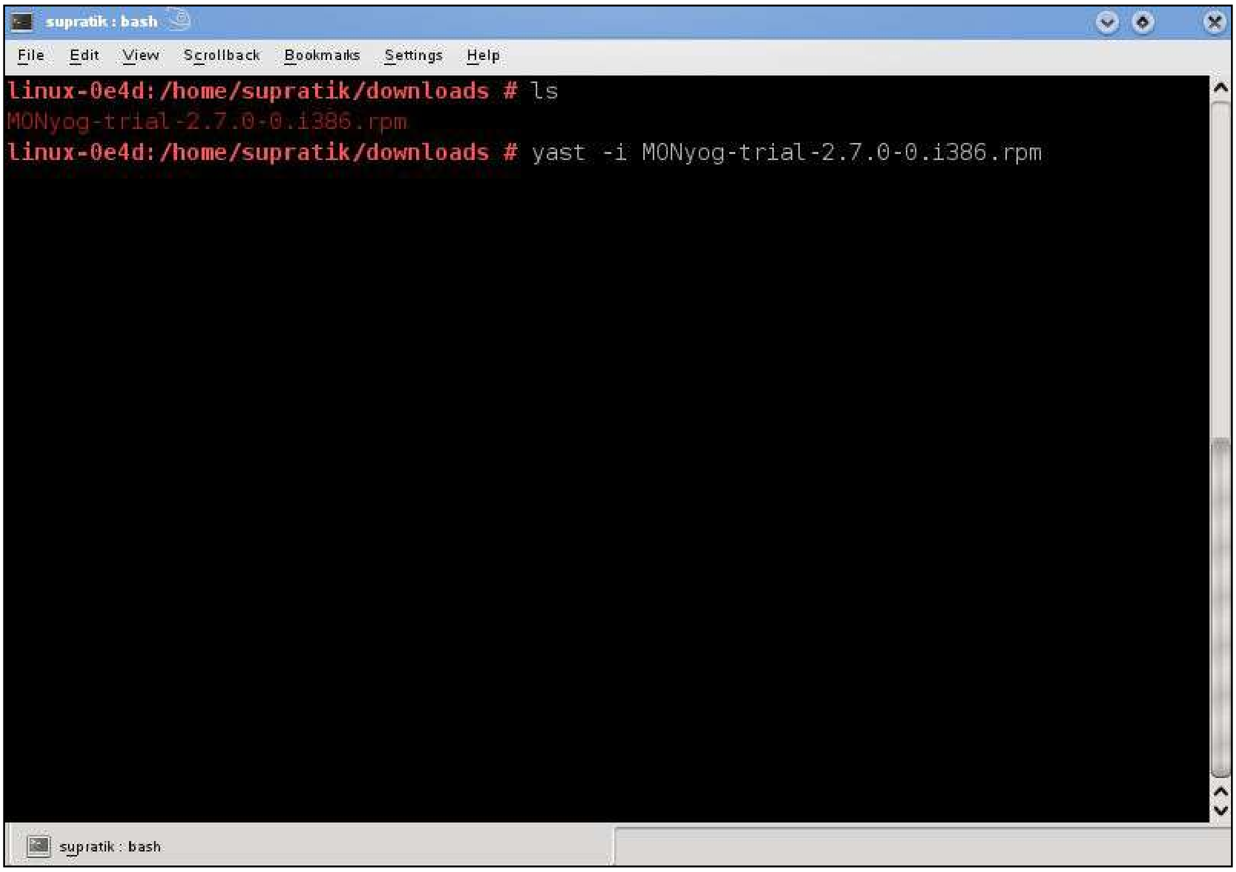
```
# service MONyogd start
```

or,

```
# /etc/init.d/MONyogd start
```

- To start the MONyog(SQL DM for MySQL) service on a system that uses systemd, run this command instead:

```
# systemctl start MONyog
```



```
linux-0e4d:/home/supratik/downloads # ls
MONyog-trial-2.7.0-0.i386.rpm
linux-0e4d:/home/supratik/downloads # yast -i MONyog-trial-2.7.0-0.i386.rpm
```

similarly for stopping:

```
# service MONyogd stop
```

or,

```
# /etc/init.d/MONyogd stop
```

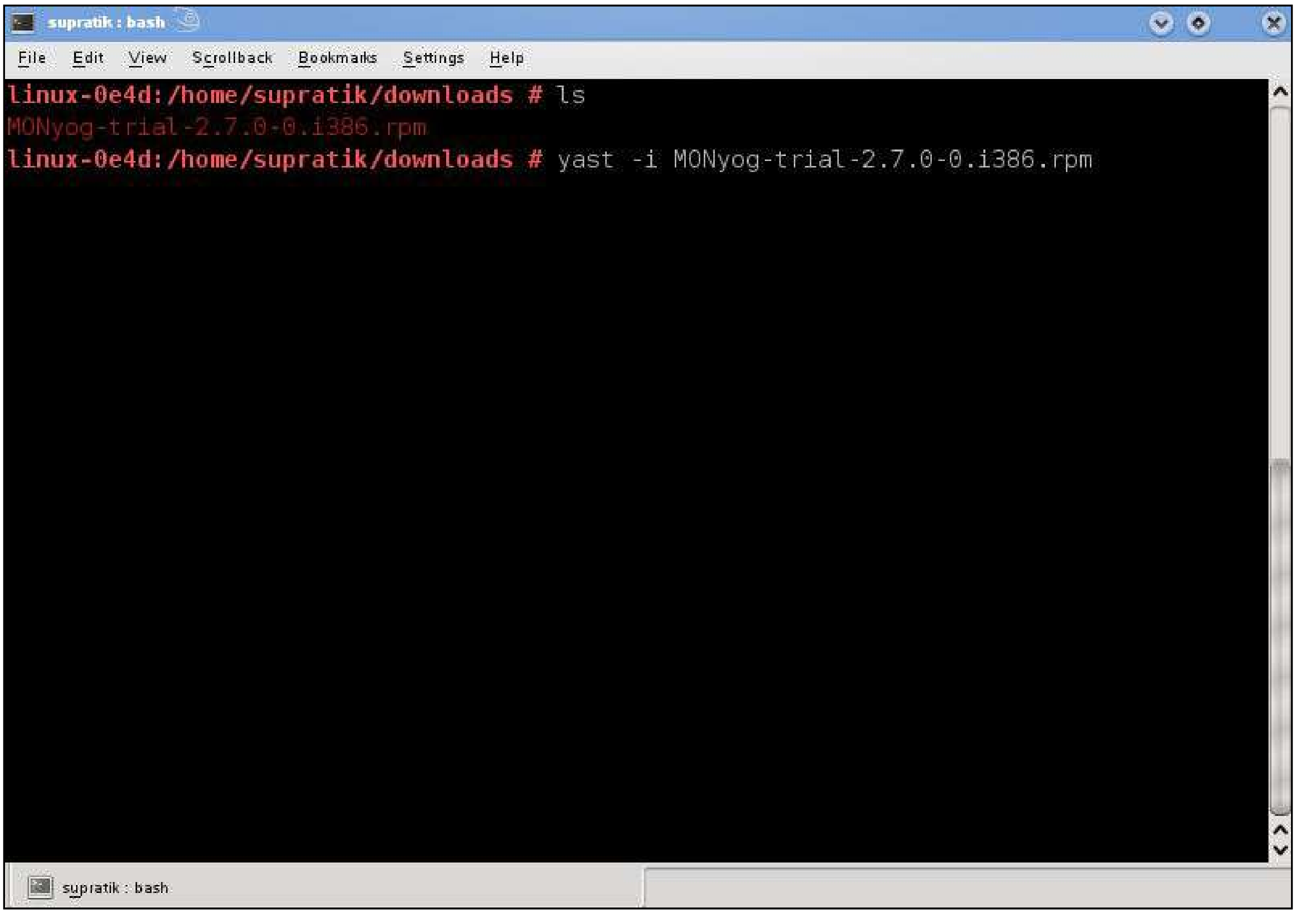
ⓘ By default in these systems SQL DM for MySQL will be started automatically even after a restart.

- In SUSE systems if SQL DM for MySQL is installed from a RPM package, the YaST 'run-level tool' can be used to start/stop the server. You may enable/disable the MONyog(SQL DM for MySQL) service from **YaST (Administrative settings) > System > System Services**.

Upgrading

To upgrade SQL DM for MySQL in Redhat/Fedora from a previous installation the following command can be used:

```
# rpm -Uvh <IderaSQLdmforMySQL_package>.rpm
```



```
linux-0e4d:/home/supratik/downloads # ls
MONyog-trial-2.7.0-0.i386.rpm
linux-0e4d:/home/supratik/downloads # yast -i MONyog-trial-2.7.0-0.i386.rpm
```

2.6.3.2 Uninstalling SQL DM for MySQL

To uninstall SQL DM for MySQL in Redhat/Fedora the following command can be used:

```
# rpm -e IderaSQLdmforMySQL
```

⚠ SQL DM for MySQL upgrade is supported from MONyog 2.7.0-0 onwards, currently SQL DM for MySQL supports upgrade from lower version of SQL DM for MySQL to upper version of SQL DM for MySQL.

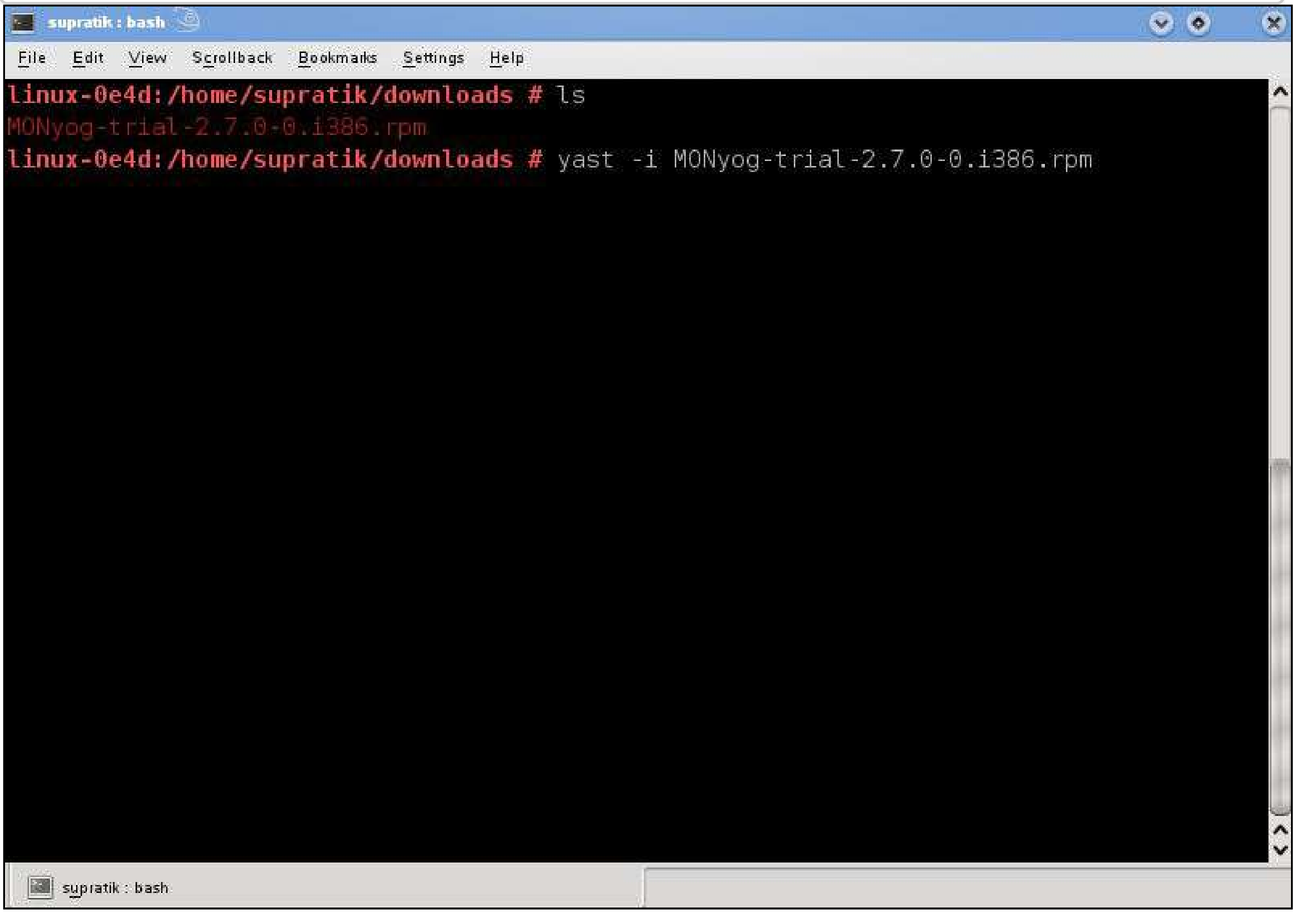
2.6.3.3 Tar Installation

In the event that your server runs a Linux distribution that isn't RPM-based, such as Debian or Ubuntu, you can still manually install SQL DM for MySQL from a tarball. Note that SQL DM for MySQL does not require any sort of compiling. You just need to unpack it. Select the download link that corresponds to your hardware: **32 Bit Tar** or **64 Bit TAR**.

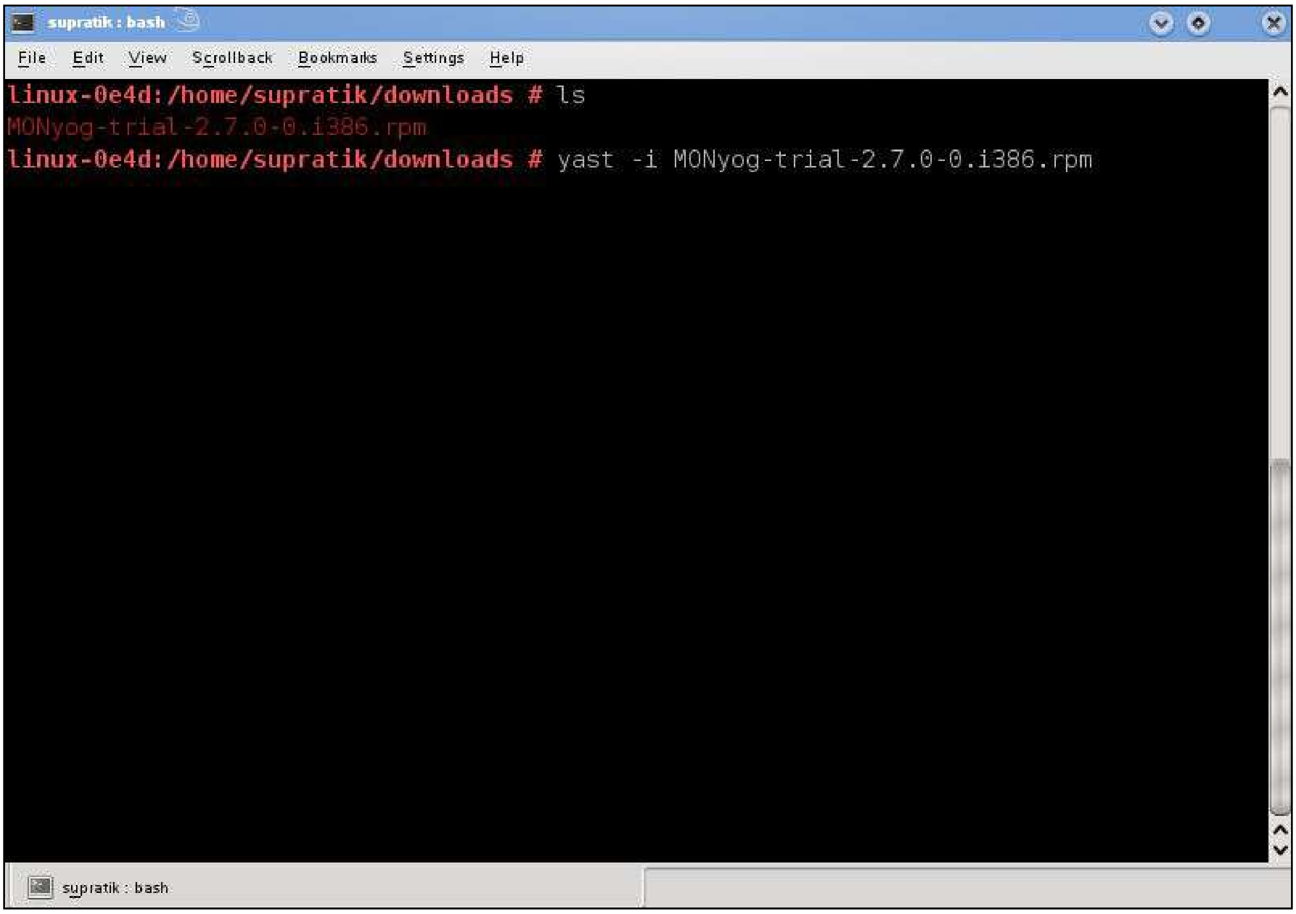
Installation

The download link retrieves a `.tar.gz` file from IDERA. The pattern for the filename is `IderaSQLdmforMySQL-<version>.<architecture>.tar.gz`. To install it run Tar setting the output to `/usr/local` or any prefix appropriate to your system:

```
# tar -xvf /path/to/IderaSQLdmforMySQL-*.tar.gz -C /usr/local
```



```
linux-0e4d:/home/supratik/downloads # ls
MONyog-trial-2.7.0-0.i386.rpm
linux-0e4d:/home/supratik/downloads # yast -i MONyog-trial-2.7.0-0.i386.rpm
```



```
linux-0e4d:/home/supratik/downloads # ls
MONyog-trial-2.7.0-0.i386.rpm
linux-0e4d:/home/supratik/downloads # yast -i MONyog-trial-2.7.0-0.i386.rpm
```

It installs SQL DM for MySQL in the same path as the RPM package. In order to use the MONyog(SQL DM for MySQL) service, you also need to copy the `MONyog` service script to `/etc/init.d/` directory.

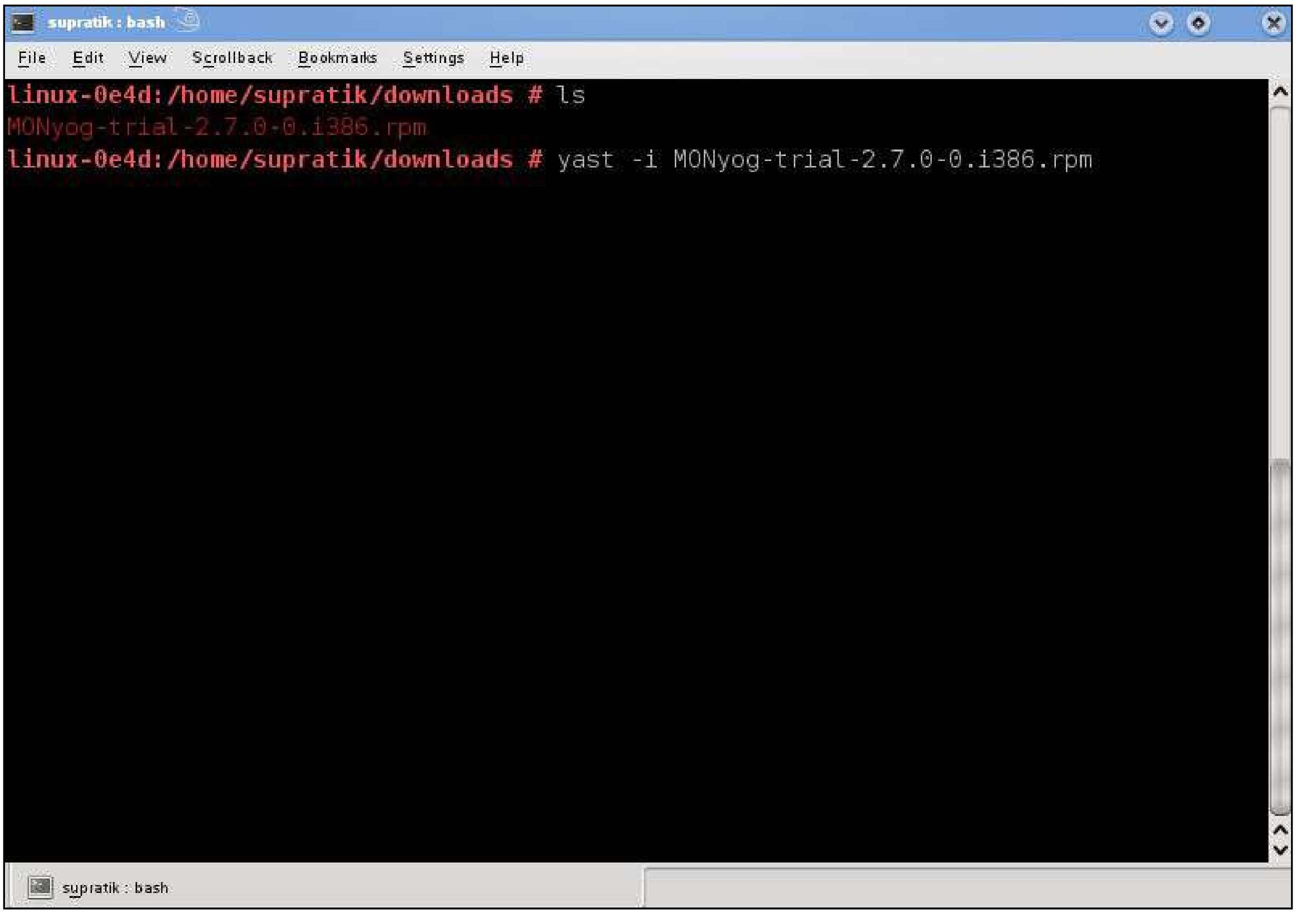
```
# cp /path/to/MONyog/bin/MONyog /etc/init.d/
```

Start/Stop

This description applies if you have extracted SQL DM for MySQL(Monyog) from zipped tar(.tar.gz) package.

There is one shell script named "MONyog" within "MONyog/bin" directory. For example if SQL DM for MySQL(Monyog) has been extracted to ~/MONyog/ directory, you can start SQL DM for MySQL by typing:

```
$ ~/MONyog/bin/MONyog start
```



```
linux-0e4d:/home/supratik/downloads # ls
MONyog-trial-2.7.0-0.i386.rpm
linux-0e4d:/home/supratik/downloads # yast -i MONyog-trial-2.7.0-0.i386.rpm
```

Similarly, to stop:

```
$ ~/MONyog/bin/MONyog stop
```

Upgrading

Untar the SQL DM for MySQL package in the same directory where you had untarred the previous SQL DM for MySQL, use the command: 64 bit tar.

```
# tar -xvf /path/to/IderaSQLdmforMySQL-*.tar.gz -C /usr/local
```

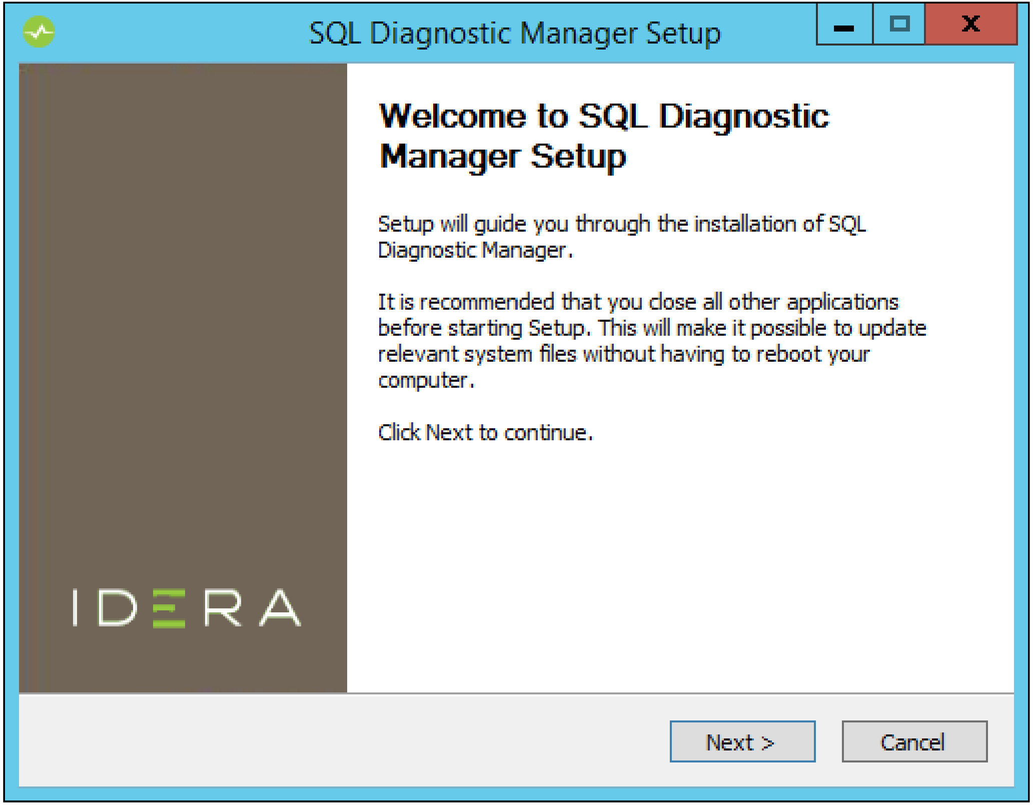
2.6.4 Windows

SQL DM for MySQL runs as a web application on a server, from which it connects to and monitors MySQL database servers. The MySQL servers can run on any operating system. SQL DM for MySQL can run on any distribution of the Linux <install-linux> operating system or it can run on Microsoft Windows Server.

2.6.4.1 Installing SQL DM for MySQL

Webyog provides an executable file to install SQL DM for MySQL on Windows. When you receive the download link, select **Executable (.exe)** to get this installer. Once you have the installer, transfer it to the Windows Server instance you want to host SQL DM for MySQL.

1. Double-click the installer, then click **Next**.



SQL Diagnostic Manager Setup

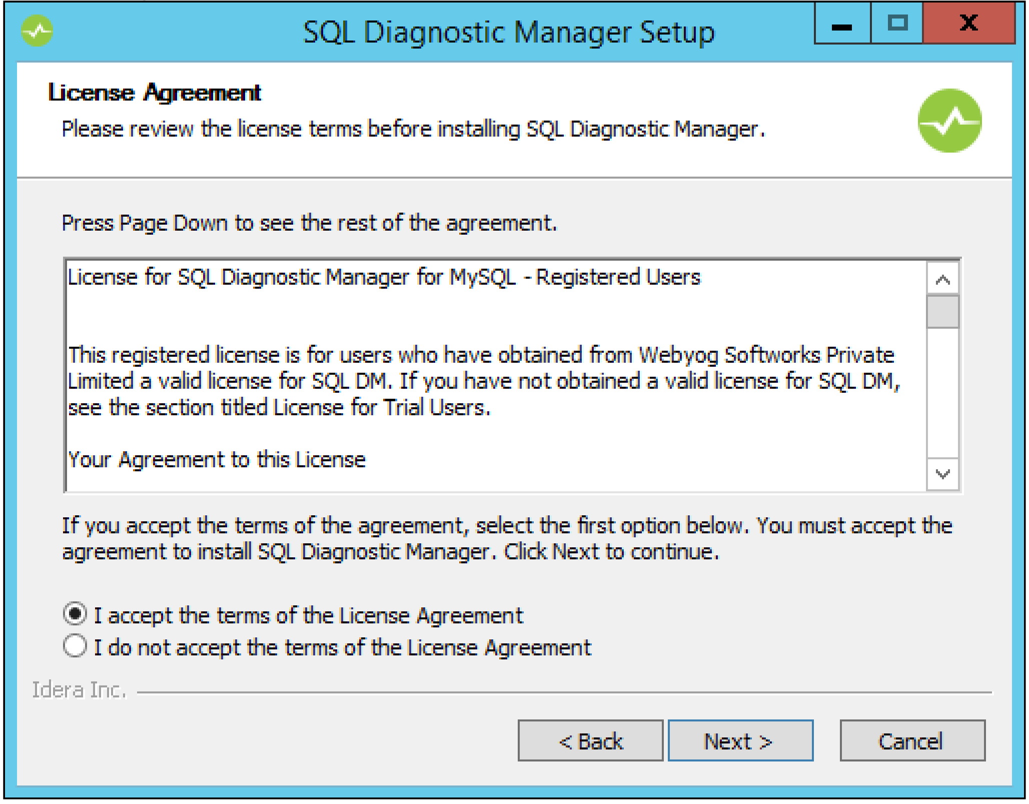
Welcome to SQL Diagnostic Manager Setup

Setup will guide you through the installation of SQL Diagnostic Manager.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

2. Read and accept the License Agreement, then click **Next**. Using SQL DM for MySQL requires that you accept the license agreement.



License Agreement

Please review the license terms before installing SQL Diagnostic Manager.

Press Page Down to see the rest of the agreement.

License for SQL Diagnostic Manager for MySQL - Registered Users

This registered license is for users who have obtained from Webyog Softworks Private Limited a valid license for SQL DM. If you have not obtained a valid license for SQL DM, see the section titled License for Trial Users.

Your Agreement to this License

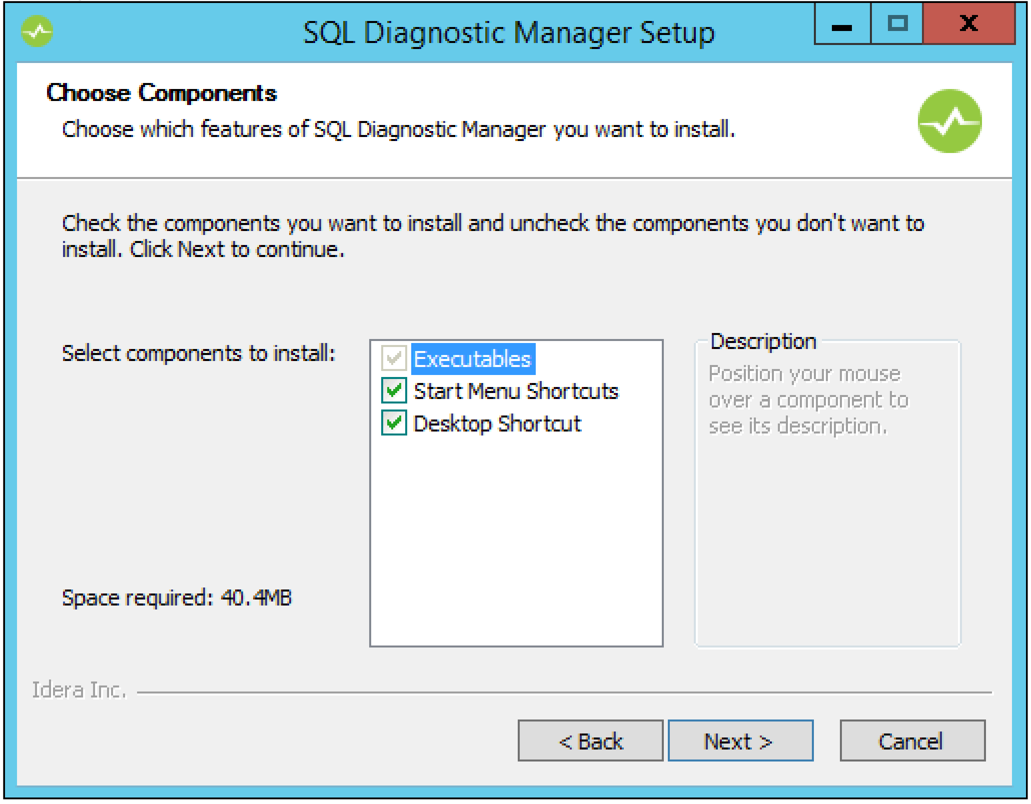
If you accept the terms of the agreement, select the first option below. You must accept the agreement to install SQL Diagnostic Manager. Click Next to continue.

● I accept the terms of the License Agreement

○ I do not accept the terms of the License Agreement

Idera Inc.

3. Select the components you want to install for SQL DM for MySQL, including executables, Start Menu and Desktop shortcuts, then click **Next**.



SQL Diagnostic Manager Setup

Choose Components

Choose which features of SQL Diagnostic Manager you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

☑ Executables
☑ Start Menu Shortcuts
☑ Desktop Shortcut

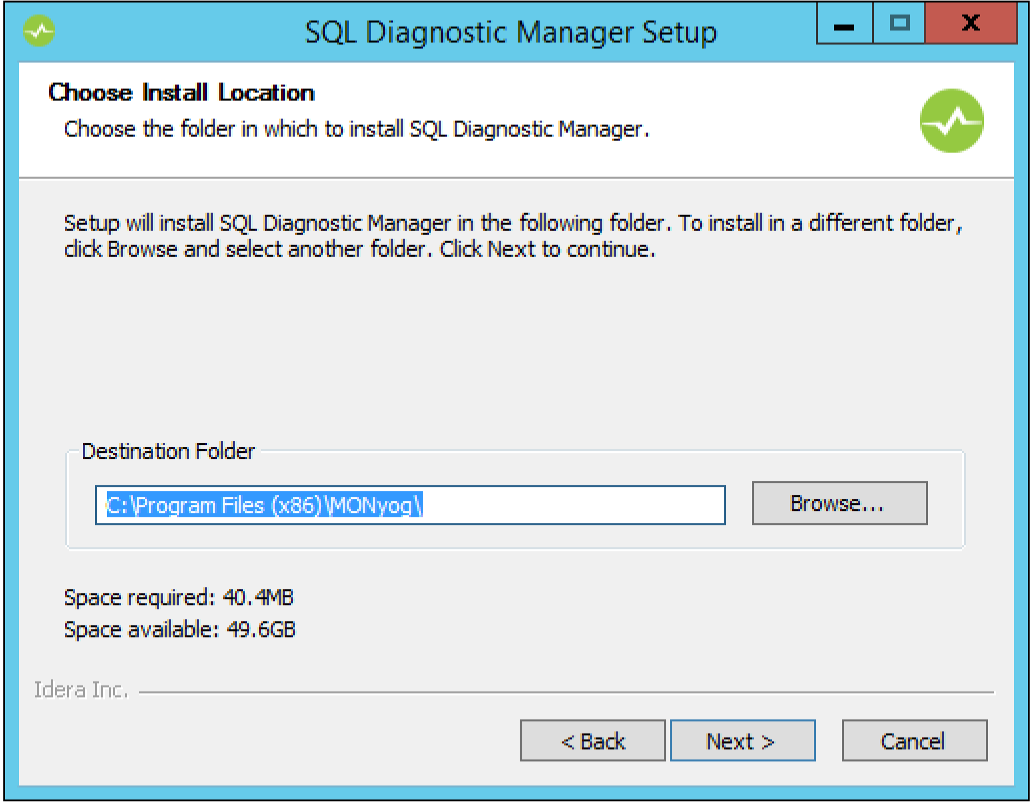
Description

Position your mouse over a component to see its description.

Space required: 40.4MB

Idera Inc.

4. Enter the appropriate install path for SQL DM for MySQL, then click **Next**.



SQL Diagnostic Manager Setup

Choose Install Location

Choose the folder in which to install SQL Diagnostic Manager.

Setup will install SQL Diagnostic Manager in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

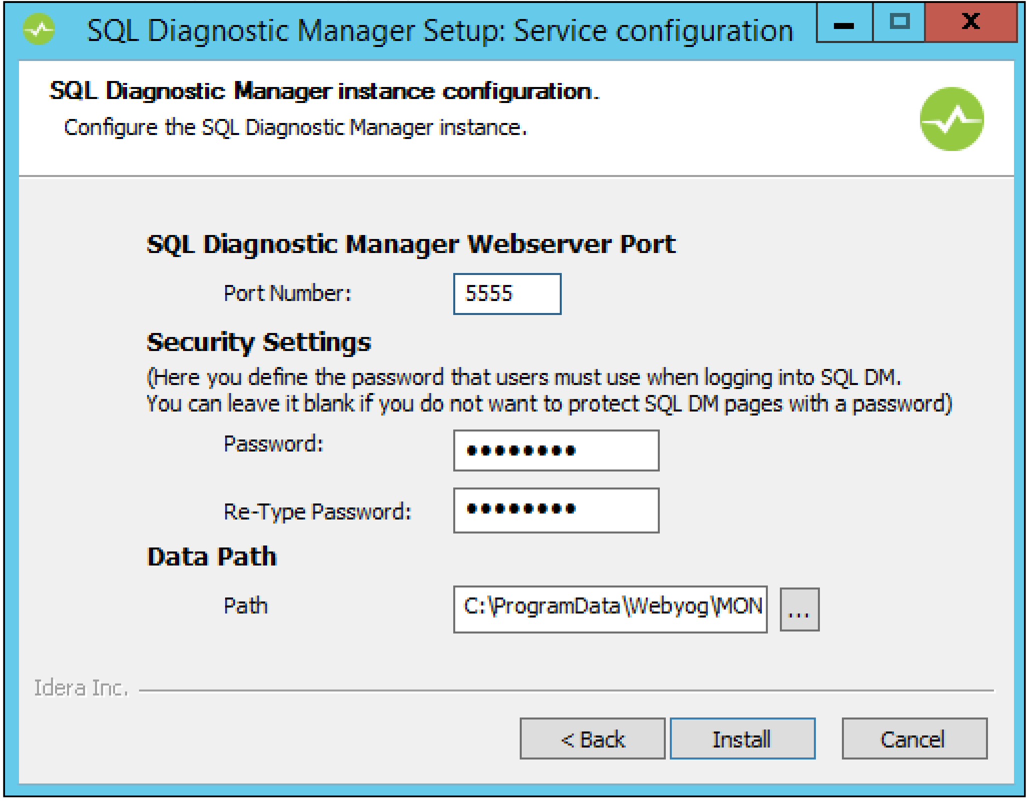
Destination Folder

C:\Program Files (x86)\MONyog\

Space required: 40.4MB
Space available: 49.6GB

Idera Inc.

5. Enter the default port number, password and data path, then click **Install**.



SQL Diagnostic Manager Setup: Service configuration

SQL Diagnostic Manager instance configuration.

Configure the SQL Diagnostic Manager instance.

SQL Diagnostic Manager Webserver Port

Port Number: 5555

Security Settings

(Here you define the password that users must use when logging into SQL DM.
You can leave it blank if you do not want to protect SQL DM pages with a password)

Password: ••••••••

Re-Type Password: ••••••••

Data Path

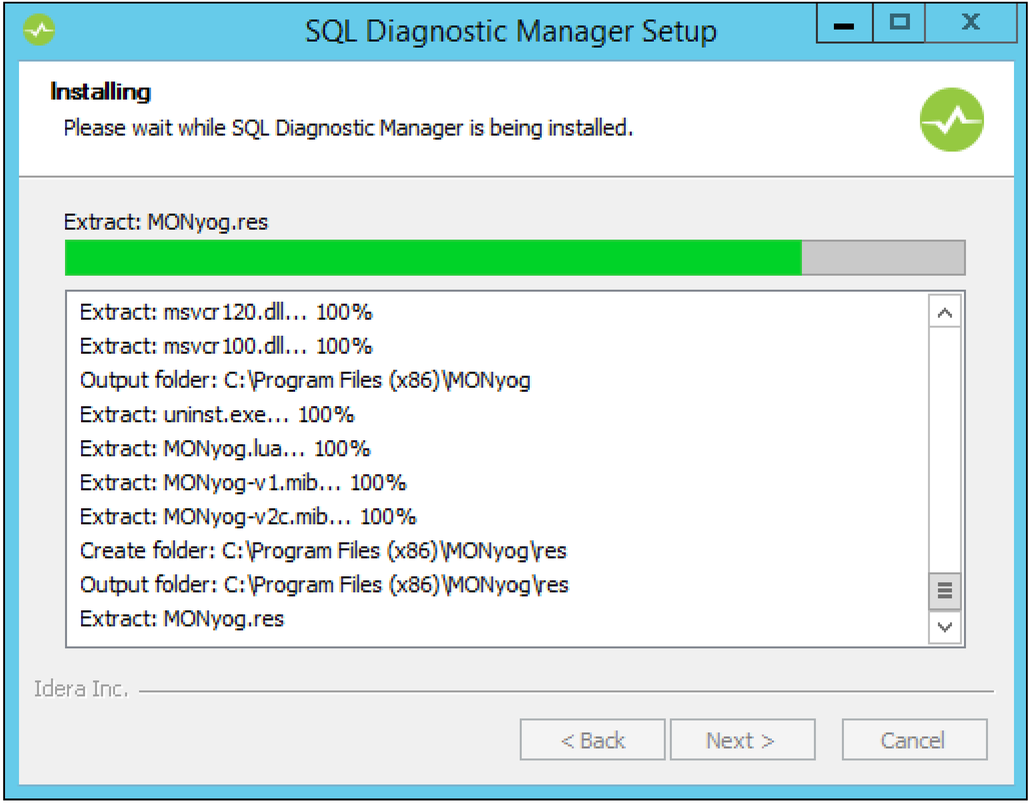
Path C:\ProgramData\Webyog\MON

Idera Inc.

In the event that you decide later that you would like to use different port numbers or login credentials, you can change them by editing the `MONyog.ini` file and restarting MONyog(SQL DM for MySQL) Service. The location of this file varies depending on what version of Windows you use:

- **Windows 2008 and later:** `C:\ProgramData\Webyog\MONyog\`
- **Windows Vista and later:** `C:\ProgramData\Webyog\MONyog`

6. To start the installation, click **Next**.



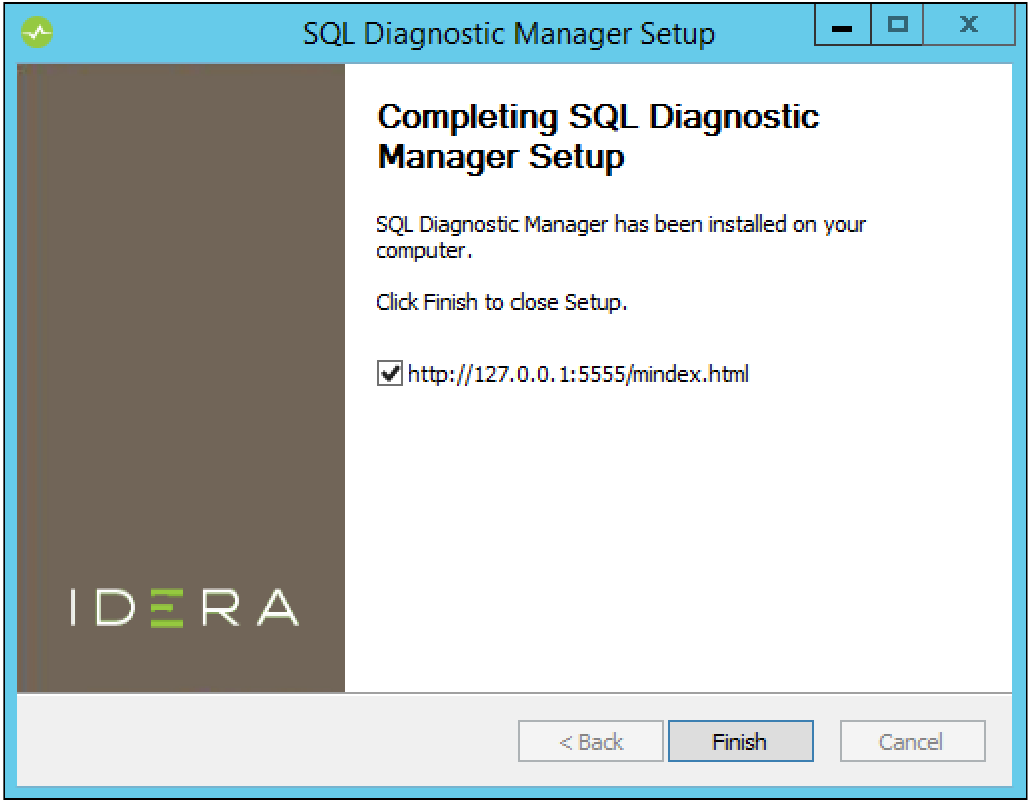
Installing

Please wait while SQL Diagnostic Manager is being installed.

Extract: MONyog.res

```
Extract: msvcr120.dll... 100%
Extract: msvcr100.dll... 100%
Output folder: C:\Program Files (x86)\MONyog
Extract: uninst.exe... 100%
Extract: MONyog.lua... 100%
Extract: MONyog-v1.mib... 100%
Extract: MONyog-v2c.mib... 100%
Create folder: C:\Program Files (x86)\MONyog\res
Output folder: C:\Program Files (x86)\MONyog\res
Extract: MONyog.res
```

7. To close the wizard, click **Finish**.



Completing SQL Diagnostic Manager Setup

SQL Diagnostic Manager has been installed on your computer.

Click Finish to close Setup.

☑ http://127.0.0.1:5555/mindex.html

2.6.4.2 Managing the MONyog(SQL DM for MySQL) Service

SQL DM for MySQL installs as a service on Windows. You can manage the SQL DM for MySQL service, named MONyog from the Windows Service Manager as you would any other service. When the installation is complete, the installer automatically starts the MONyog service.

You can access SQL DM for MySQL through a web browser at `http://127.0.0.1:5555`, or at the port you configured it to use.

2.6.5 Monyog AMI

SQL DM for MySQL AMI comes with the default installation of 64 bit SQL DM for MySQL and nginx reverse proxy on Amazon Linux. You can find SQL DM for MySQL installer in the home directory.

If SQL DM for MySQL AMI default security group is used, SQL DM for MySQL User Interface can be accessed on port 80(http) and 443(https).

The URL to access SQL DM for MySQL would be:

- `http://<Public DNS of the Amazon instance>`
- `https://<Public DNS of the Amazon instance>`

2.6.5.1 Upgrading And Installing

Upgrading and installing SQL DM for MySQL can be either done by using:

- YUM to upgrade:

  ```
  # yum update MONyog
  ```

  or to install SQL DM for MySQL:

  ```
  # yum install MONyog
  ```

- Downloading the binaries and upgrading it with RPM:

  ```
  # rpm -Uvh MONyog-*.rpm
  ```

  or, installing it with RPM:

  ```
  # rpm -ivh MONyog-*.rpm
  ```

2.6.5.2 Nginx Configuration

SQL DM for MySQL AMI is configured with nginx reverse proxy. Nginx listens on ports 80 and 443 and forwards the incoming traffic to SQL DM for MySQL default port: 5555.

SQL DM for MySQL AMI comes with an SSL certificate which is not verfied by any Certificate Authority(CA).

Note

We highly recommend you to change the SSL certificates. You can either have a self-signed SSL certificate or get one signed by any Certificate Authority(CA). Refer the following to know more about generating SSL certificates with OpenSSL: CA.pl(see page 117). Also refer FAQ 14(see page 298) points 5-7.
To change SSL certificates in Nginx follow the FAQ 13(see page 296) section of the HELP file.Reset the admin password before continuing to use SQL DM for MySQL.
Reset the admin password before continuing to use SQL DM for MySQL.

2.7 Configuration

With SQL DM for MySQL installed on your server, you can begin to configure systems and databases in your infrastructure for its use.

Use SQL DM for MySQL to monitor a server, only grant its user access to MySQL and then register the server through the Web UI.

2.7.1 MySQL Configuration

When you register a server with SQL DM for MySQL, it collects information and monitors the server's health through standard MySQL client connections. You need to configure a user account with the appropriate permissions on each MySQL server you want to monitor. Fully enabling SQL DM for MySQL requires that you grant `SELECT` , `RELOAD` , `PROCESS` , and `SUPER` to this user. You can use whatever username you find convenient, with the host set to the server hosting SQL DM for MySQL.

For instance, say you have installed SQL DM for MySQL on a server at 192.168.1.150 and that you would like use the user `monyog` as the user name. To fully enable SQL DM for MySQL in this scenario, you would issue the following command:

```
GRANT SELECT, RELOAD, PROCESS, SUPER ON *.*

TO 'monyog'@'192.168.1.150';
```

Granting these permissions makes the MySQL server fully accessible to SQL DM for MySQL. To actually start collecting data on this server you also need to register it with the application.

2.7.2 Running SQL DM for MySQL as unprivileged user

By default, SQL DM for MySQL runs under the 'root' account in Linux. This may be a security nightmare for some.

Here's how you can create and use a user account exclusively for running SQL DM for MySQL:

- First add a new user and group; replace <GID>, with the group ID, and <PASSWORD>:

```
# groupadd monyog
```

```
# useradd -g <GID> -p <passwd> monyog
```

- Copy the original files to the new location and change the file ownership.

```
# cp -r /usr/local/MONyog/. /home/monyog/
```

```
# chown -R monyog:monyog /home/monyog
```

- Next, we need to change the initialization script. Be sure to make a backup first!

```
# cp /etc/init.d/MONyogd /etc/init.d/MONyogd.orig
# vi /etc/init.d/MONyogd
```

- Change the following lines for the new path.

```
PREFIX="/home/monyog"
```

```
MONYOGBIN="$PREFIX/bin/MONyog"
```

- In the same file, you also need to update the start command to the following.

```
# Start MONyog
...
else
action "`su - monyog -c \"$MONYOGBIN -s\"`" /bin/true
fi
```

- Now you need to alter the configuration file.

```
# vi /home/monyog/MONyog.ini
```

- Change the data path to the new directory.

```
Data_path=/home/monyog/data
```

- After this you should be able to start SQL DM for MySQL running as an unprivileged user.

```
# service MONyog start
```

2.8 Server Registration

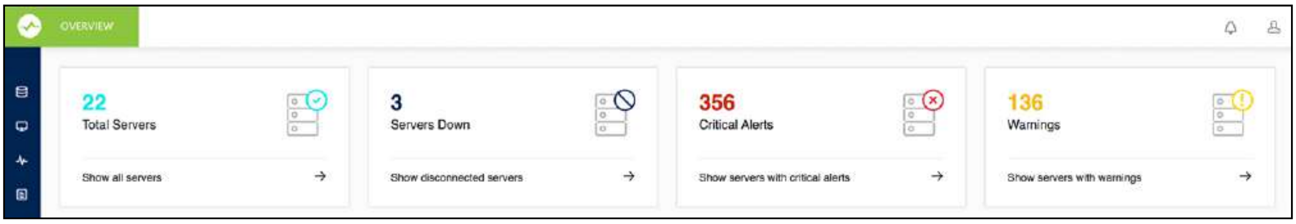
At this point, you should have SQL DM for MySQL installed and configured on a Linux or Windows server and one or more servers running MySQL that you have configured to provide monitoring data to SQL DM for MySQL. However, in order to get SQL DM for MySQL to access that data, you need to register the server with the application.

A single instance of SQL DM for MySQL can monitor hundreds of MySQL database servers. With tagging, you can categorize them into logical groups, ensuring that they're easier to keep track of and identify. During the Trial period, you can register up to 1000 MySQL servers, after which you'll need to purchase a license.

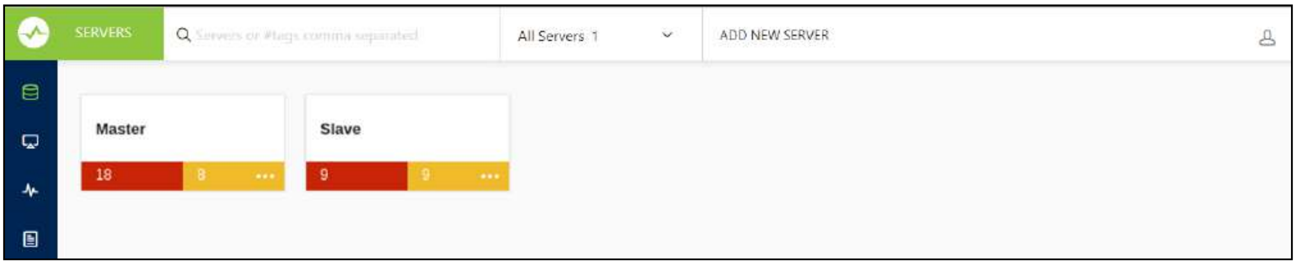
⚠ For more information on licenses and pricing, see Pricing(see page 120).

2.8.1 Adding Servers

The web interface for SQL DM for MySQL provides a convenient form for adding new MySQL database servers to monitor. The link to this form is found on the Servers tab. To get there, click the **Servers** icon on the dock.

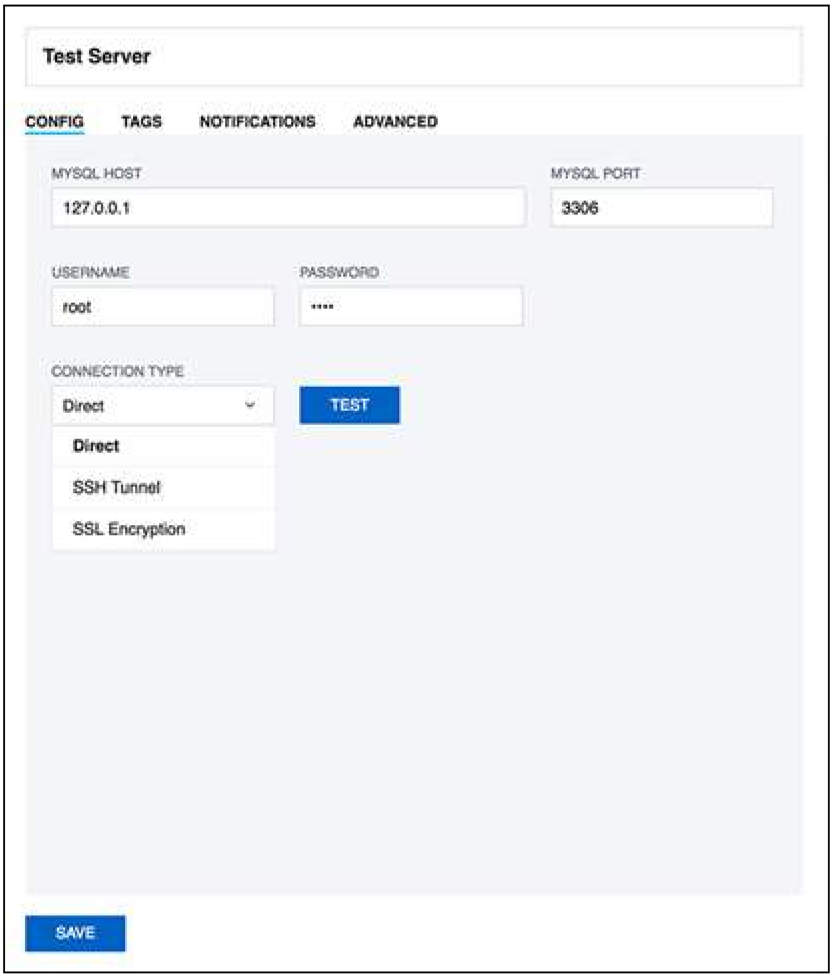

OVERVIEW
22
Total Servers
Show all servers
3
Servers Down
Show disconnected servers
356
Critical Alerts
Show servers with critical alerts
136
Warnings
Show servers with warnings

On the Servers tab, click the **Add New Server** link at the top of the page:



Clicking this link opens an overlay form which you can use to configure a new server for SQL DM for MySQL.

For general information, you need to provide a name, the IP address or domain name, port number for the MySQL host, the username, and password for the SQL DM for MySQL user on the database. Also, you have the option of setting tags to group different servers together, and setting the connection type (direct, SSH tunneling, or SSL encryption).



Test Server

MYSQL HOST: 127.0.0.1

MYSQL PORT: 3306

USERNAME: root

PASSWORD: ****

CONNECTION TYPE: Direct

When you finish configuring the new MySQL server, click **Save**.

SQL DM for MySQL now collects data on the new server. You can view its state, compare its configuration to another servers, and much more.

3 Register your servers

3.1 Connecting to SQL DM for MySQL

In this section it is assumed that SQL DM for MySQL is installed and running. Connect to it from a browser, start your browser, and enter it following the format: `http://host:port`

The host can be:

- An IP address (like "10.0.0.1")
- The host name of the system
- An alias being read from the local host-file (like `localhost`). The option Port is where you or your Sys Admin specified when installing SQL DM for MySQL. By default, SQL DM for MySQL UI can be accessed on port 5555.

3.2 Managing multiple servers

Single instance of SQL DM for MySQL is capable of monitoring hundreds of MySQL servers. With tagging, you can categorize your servers into logical groups, and avoid clutter.

You can register as many servers as your SQL DM for MySQL license allows. Also, SQL DM for MySQL installations registered with an Unlimited license, can register up to 1000 servers. If you are a customer with 'unlimited' license and want to register more licenses, please contact us by sending an email to IDERA Support[18].

✓ The SQL Diagnostic Manager TRIAL installation can register up to 1000 servers.

3.3 Registering replicas

SQL DM for MySQL has an option to auto-register slaves. Please refer to Replication Settings(see page 0) in Advanced Settings for more details.

Review the following options to continue with the Registration of your servers:

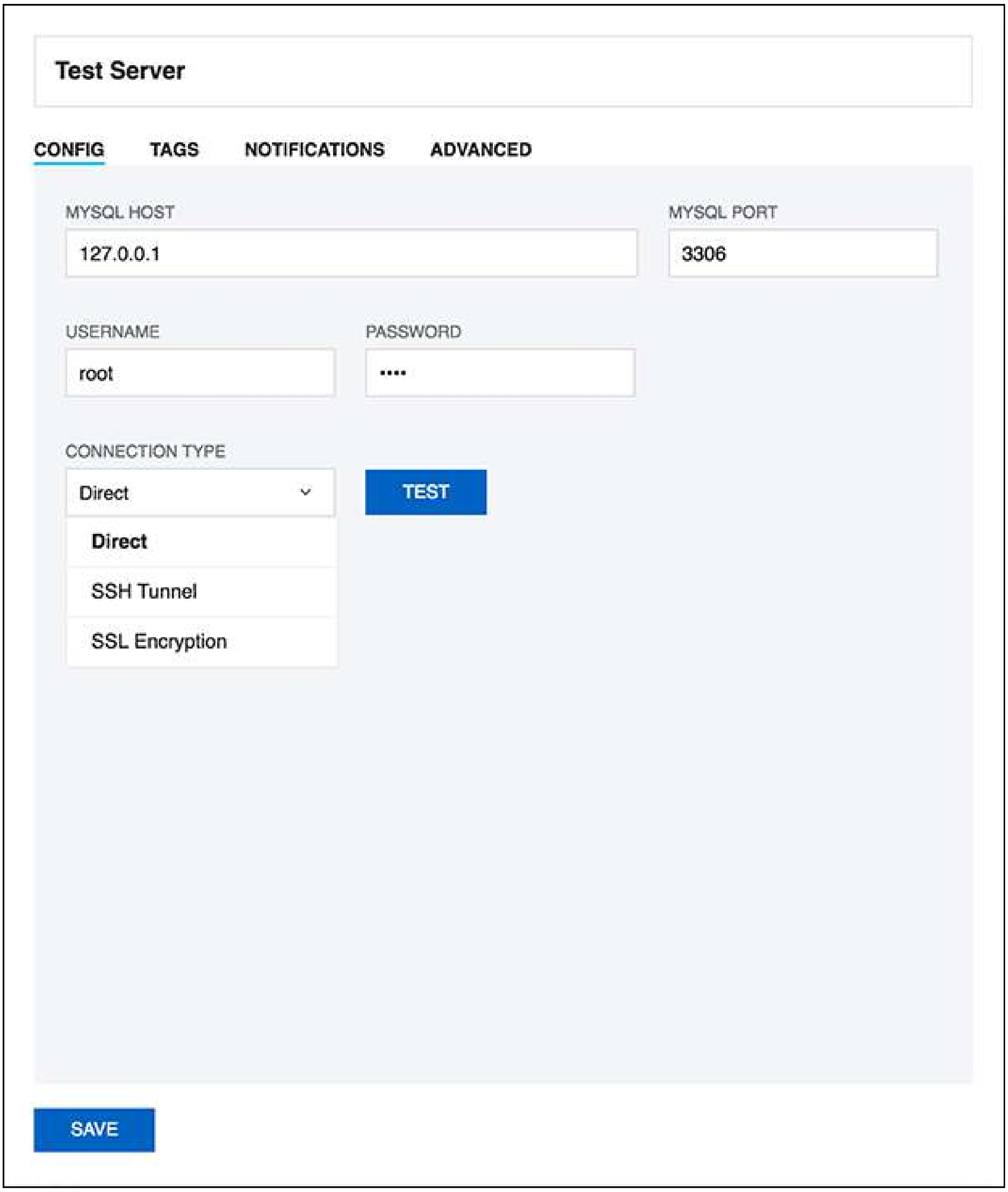
- Connecting to MySQL Server[19]
- Tags[20]
- Notification Settings[21]
- Advanced Settings[22]
- MySQL Privileges[23]
- System Privileges[24]
- RDS OS and File based Log Monitoring Privileges[25]

[18] mailto:ideramysqlsupport@idera.com
[19] http://wiki.idera.com/x/dgEvvgE
[20] http://wiki.idera.com/x/dwEvvgE
[21] http://wiki.idera.com/x/eAEvvgE
[22] http://wiki.idera.com/x/eQEvvgE
[23] http://wiki.idera.com/x/egEvvgE
[24] http://wiki.idera.com/x/ewEvvgE
[25] http://wiki.idera.com/x/fAEvvgE

3.4 Connecting to MySQL Server

3.4.1 Registering Servers

First you must register the MySQL servers to which SQL DM for MySQL connects to. For every server you specify here, an embedded database is created. SQL DM for MySQL connects to those servers to retrieve and store information from that server.



Test Server

MYSQL HOST: 127.0.0.1

MYSQL PORT: 3306

USERNAME: root

PASSWORD: ••••

CONNECTION TYPE: Direct

For a **Direct connection**, enter the connection information as you would do from any client.

To retrieve system information (CPU, memory load etc.) you need to give the information required to create a remote command shell (Linux) from the server machine on which SQL DM for MySQL is running.

3.4.2 SSH tunneling

SQL DM for MySQL can send and receive encrypted authentication information as well as monitored data from MySQL using SSH tunneling. Also, it is possible to connect to MySQL with SSH tunneling even if the MySQL port (normally 3306) is blocked by a firewall or if users are not allowed to connect from remote hosts. An operating system user is required so that SQL DM for MySQL SSH client functionalities can use this user to connect to the SSHD daemon on the server.

If you want to use SSH tunneling to your MySQL server, you have to provide details for creating a SSH connection. Please refer to Using SSH connections, below.

3.4.2.1 To connect using SSH tunneling

- TCP port forwarding must be allowed in the SSH server. For openSSH servers it should be done in the "sshd_config" configuration file. In Linux usually it is in, `/etc/ssh/sshd_config`.

  Set the "AllowTcpForwarding" option to "yes". So it should look like,

```
AllowTcpForwarding yes
```

- MySQL host should be specified 'relative' to the SSH server. Say, your SSH server is running in M1. Then you should ask yourself the following question: How should I connect to the target MySQL server from M1?

If your MySQL server is running in the same system M1, you can choose "**localhost**", "**127.0.0.1**" or the IP address of MySQL server which M1 can see.

SSH will listen on a specified port on the client machine, encrypt the data it receives, and forward it to the remote SSH host on port 22 (the SSH protocol port). The remote SSH host decrypts the data and forwards it to the MySQL server. The SSH host and the MySQL server do not have to be on separate machines, but separate SSH and MySQL servers are supported.

3.4.3 Using SSH connections

To create a SSH connection you need the following:

- **SSH Host**: Host of the machine on which SSH server is running.
- **SSH Port**: Port on which SSH server is listening. By default, it is 22.
- **SSH Username**: Username to access the SSH server (Note: not the MySQL server).

- **Authentication type**: Specify the type of authentication to use. This can be either key based or password based.
- If you have specified authentication type as Password - Provide the password.

- If you have specified authentication type as Key - You should note that SQL DM for MySQL only supports ''OpenSSH standard key format'' for key based authentication in SSH connections.

- **Private Key**: Paste the content of your private key file. Again, do not specify the path to your private key file.
- **Passphrase**: Enter the passphrase for your private key file (if any). This can be left blank, if no passphrase was given for the private key.

3.4.4 Using SSL Connection

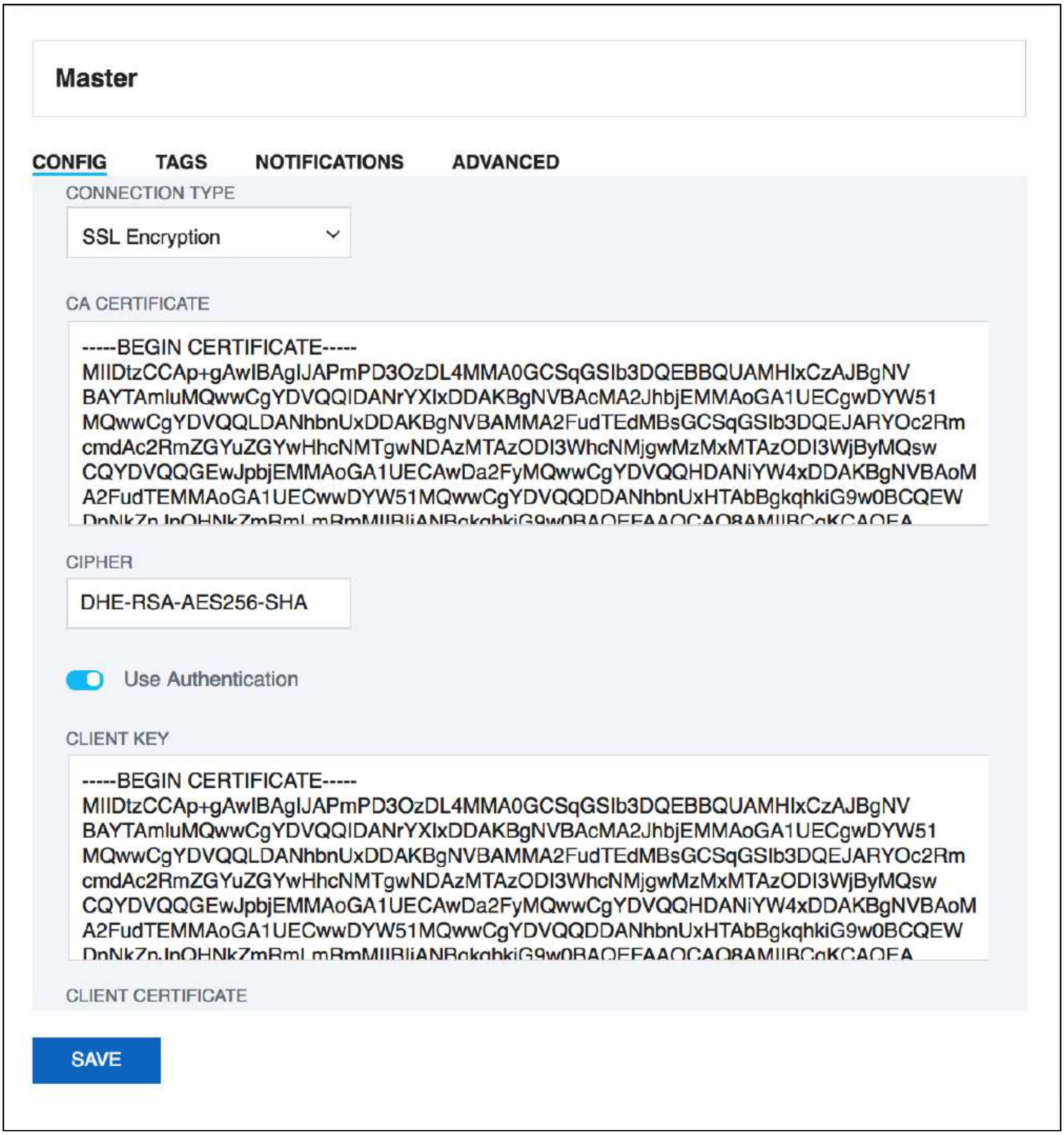
The Secure Sockets Layer (SSL) is a commonly-used protocol for managing the security of a message transmission on the Internet. SQL DM for MySQL provides native MySQL SSL encryption for direct MySQL connections.

To use SSL encryption you will be asked for:

- **CA Certificate**: The digital certificate issued by CA.
- **Cipher**: Encryption algorithm like DES, AES etc.

If you need client authentication then,

- **Client Key**: Private key of the client that is needed for encryption.
- **Client Certificate**: The client certificate.



Master

CONNECTION TYPE

SSL Encryption

CA CERTIFICATE

```
-----BEGIN CERTIFICATE-----
MIIDtzCCAp+gAwIBAgIJAPmPD3OzDL4MMA0GCSqGSIb3DQEBBQUAMHIxCzAJBgNV
BAYTAmluMQwwCgYDVQQIDANrYXIxDDAKBgNVBAcMA2JhbjEMMAoGA1UECgwDYW51
MQwwCgYDVQQLDANhbnUxDDAKBgNVBAMMA2FudTEdMBsGCSqGSIb3DQEJARYOc2Rm
cmdAc2RmZGYuZGYwHhcNMTgwNDAzMTAzODI3WhcNMjgwMzMxMTAzODI3WjByMQsw
CQYDVQQGEwJpbjEMMAoGA1UECAwDa2FyMQwwCgYDVQQHDANiYW4xDDAKBgNVBAoM
A2FudTEMMAoGA1UECwwDYW51MQwwCgYDVQQDDANhbnUxHTAbBgkqhkiG9w0BCQEW
DnNkZnJnQHNkZmRmLmRmMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
```

CIPHER

DHE-RSA-AES256-SHA

Use Authentication

CLIENT KEY

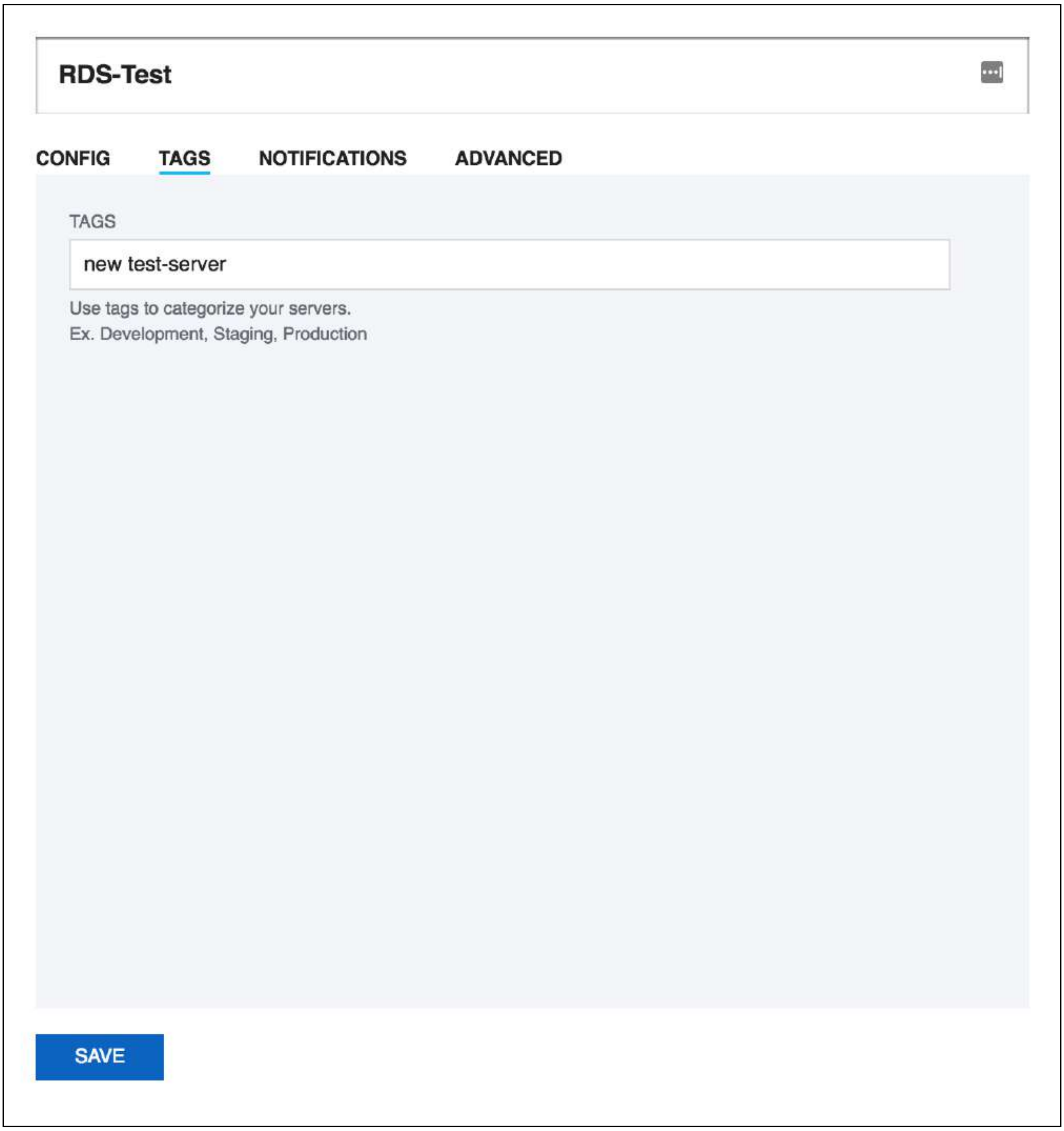
```
-----BEGIN CERTIFICATE-----
MIIDtzCCAp+gAwIBAgIJAPmPD3OzDL4MMA0GCSqGSIb3DQEBBQUAMHIxCzAJBgNV
BAYTAmluMQwwCgYDVQQIDANrYXIxDDAKBgNVBAcMA2JhbjEMMAoGA1UECgwDYW51
MQwwCgYDVQQLDANhbnUxDDAKBgNVBAMMA2FudTEdMBsGCSqGSIb3DQEJARYOc2Rm
cmdAc2RmZGYuZGYwHhcNMTgwNDAzMTAzODI3WhcNMjgwMzMxMTAzODI3WjByMQsw
CQYDVQQGEwJpbjEMMAoGA1UECAwDa2FyMQwwCgYDVQQHDANiYW4xDDAKBgNVBAoM
A2FudTEMMAoGA1UECwwDYW51MQwwCgYDVQQDDANhbnUxHTAbBgkqhkiG9w0BCQEW
DnNkZnJnQHNkZmRmLmRmMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
```

CLIENT CERTIFICATE

3.5 Tags

Tags are used for grouping servers. When you have few servers it is easy to keep track of them but, it is increasingly difficult to keep track of servers when there are too many of them. To reduce the difficulty in dealing with too many servers you can use tags.

Using tags you can categorize the servers by grouping a server under a single tag or you can also enter multiple tags separated by 'Space' for each server. Also, note that if you do not provide any tag then by default the server will be categorized under 'Not Tagged'.



RDS-Test

TAGS

new test-server

Use tags to categorize your servers.
Ex. Development, Staging, Production

Using tags, it is easy to search for a group of servers at once, simply enter the tag name in the server selector with **#** (ex: #test) and all the tags with the entered keyword are listed and you can further select the desired tag to select servers tagged under it.

3.6 Notifications Settings

To use the SQL DM for MySQL Notification alert functionality to send you alerts about a MySQL server you need to configure the different Notification channels from **Settings -> Notification & Maintenance**. Once you configure the notification channels from the Settings page, those channels are now be available for sending alerts for each of the servers. If you have enabled all the channels, then you get the following alert options: Email, SNMP, Slack, Pagerduty, and Syslog.

3.6.1 Different Notification Channels

3.6.1.1 Email

Use this option, to get SQL DM for MySQL alerts via Email. Once you enable it, enter the addresses to which SQL DM for MySQL is to send e-mails.

There is an Option to set a distinct email distribution list for warning and critical alerts. You can define a proper demarcation between the users who receives a particular alert based on the alert status. For instance, if you want only your on-call DBAs to receive critical alerts 24/7, this feature handles that easily. Please refer to Mail Settings(see page 261) for more information. Details about the SMTP server to use. SQL DM for MySQL does not have a SMTP server of its own. It needs to connect to a SMTP server on the Internet, your local network or your local computer. The settings are the same you enter when installing or configuring a mail client program like Outlook or Outlook Express.

3.6.1.2 SNMP

Enable this option to get alerts via SNMP traps. Refer to SNMP settings(see page 264) for more information

3.6.1.3 Write to Syslog

Enable this option if you want SQL DM for MySQL to write the alerts in the Syslog (of the machine where SQL DM for MySQL is installed). Refer to SYSLOG Settings(see page 267) for more information.

3.6.1.4 Slack Notifications

Use this option to send alerts to Slack. Like Pagerduty, you get a drop-down option to select a slack channel to which the alerts can be sent to. Refer to Slack Settings(see page 266) for more information.

3.6.1.5 Pagerduty Notifications

Enable this option to get alerts on Pagerduty. After enabling, you get a drop-down option to select a rule which you created at **Settings -> Notification & Maintenance -> PAGERDUTY**. All the alerts are sent to this rule in Pagerduty. Refer to Pagerduty Settings(see page 265) for more information.

3.6.2 MONITORS (CRITICAL & WARNING)

Notify When the Monitor is in Alert State For: Here, you can specify the number of times a counter should be in an alertable state, consecutively, before a notification is sent.

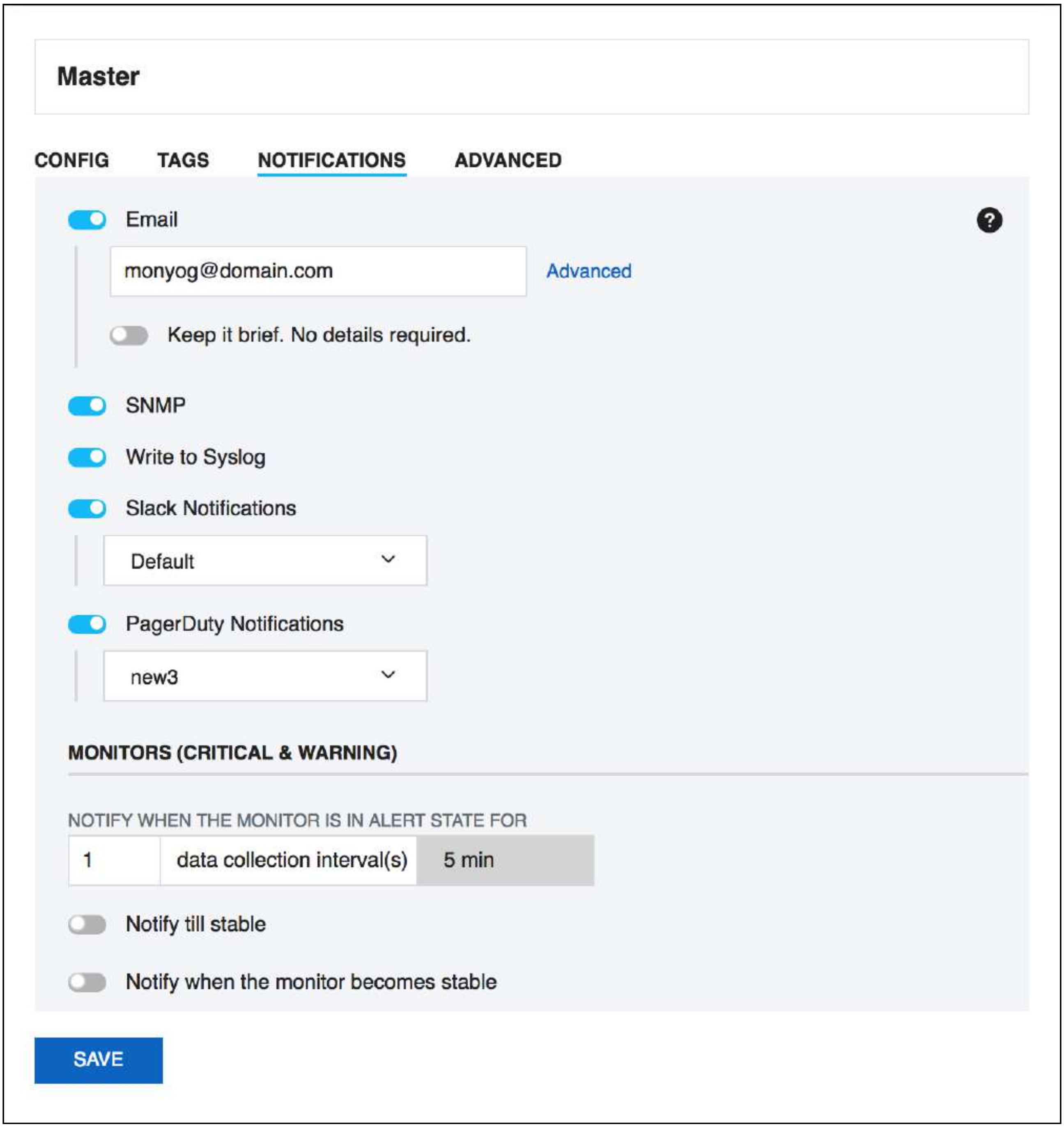
Notify till stable: SQL DM for MySQL sends mail alert notification until a given monitor becomes stable. The value in the "remind me after every" specifies how often you want to receive the mail alert notification i.e. if set to three, then for every three consecutive data collection for which the monitor is not stable, an e-mail alert is sent.

Notify when the monitor becomes stable: If a monitor goes into alertable state and then becomes stable, a stable e-mail or an SNMP trap is sent by SQL DM for MySQL.

3.6.3 OTHER

Notify when server restarts: If you enable this option, it sends an alert if the server restarts between two data retrievals. With managed hosting, the MySQL server may be restarted automatically as part of routine maintenance or after a server crash. Knowing when this has occurred can be useful.

Notify when changes to MySQL configuration is detected: If you enable this option, SQL DM for MySQL sends an alert whenever there is a change in MySQL configuration. These alerts are sent whenever SQL DM for MySQL detects a change in server variables using SET GLOBAL statements or in the MySQL configuration file.



Master

Email: on

monyog@domain.com

Keep it brief. No details required.: off

SNMP: on

Write to Syslog: on

Slack Notifications: on

Default

PagerDuty Notifications: on

new3

MONITORS (CRITICAL & WARNING)

NOTIFY WHEN THE MONITOR IS IN ALERT STATE FOR

1 data collection interval(s) 5 min

Notify till stable: off

Notify when the monitor becomes stable: off

3.7 Advanced Settings

You can configure the following Advanced settings when connecting to MySQL server:

- System Metrics[26]
- Data Collection[27]

[26] http://wiki.idera.com/display/SQLDMYSQL/System+Metrics
[27] http://wiki.idera.com/display/SQLDMYSQL/Data+Collection

- Replication[28]
- Galera[29]
- MySQL Error Log[30]
- MySQL Query Log[31]
- Audit Log[32]
- Sniffer Settings[33]
- Deadlock Settings[34]
- Monitors[35]
- Real-Time[36]
- Connection[37]

3.7.1 System Metrics

3.7.1.1 System Metrics (applicable for Linux based systems)

In the event that you would like SQL DM for MySQL to use SSH when communicating with this server, you can configure it from this tab. SQL DM for MySQL disables SSH communications by default. In order to use it, you need to click the **Enable System Metrics** switch. Doing so provides a series of configuration options needed to enable SSH on the server.

Please refer to System Privileges[38] for further details of privileges needed for this feature.

If SSH tunneling to MySQL is configured successfully for this registration you can use those same details here too, provided that SSH tunnel user has enough privileges.

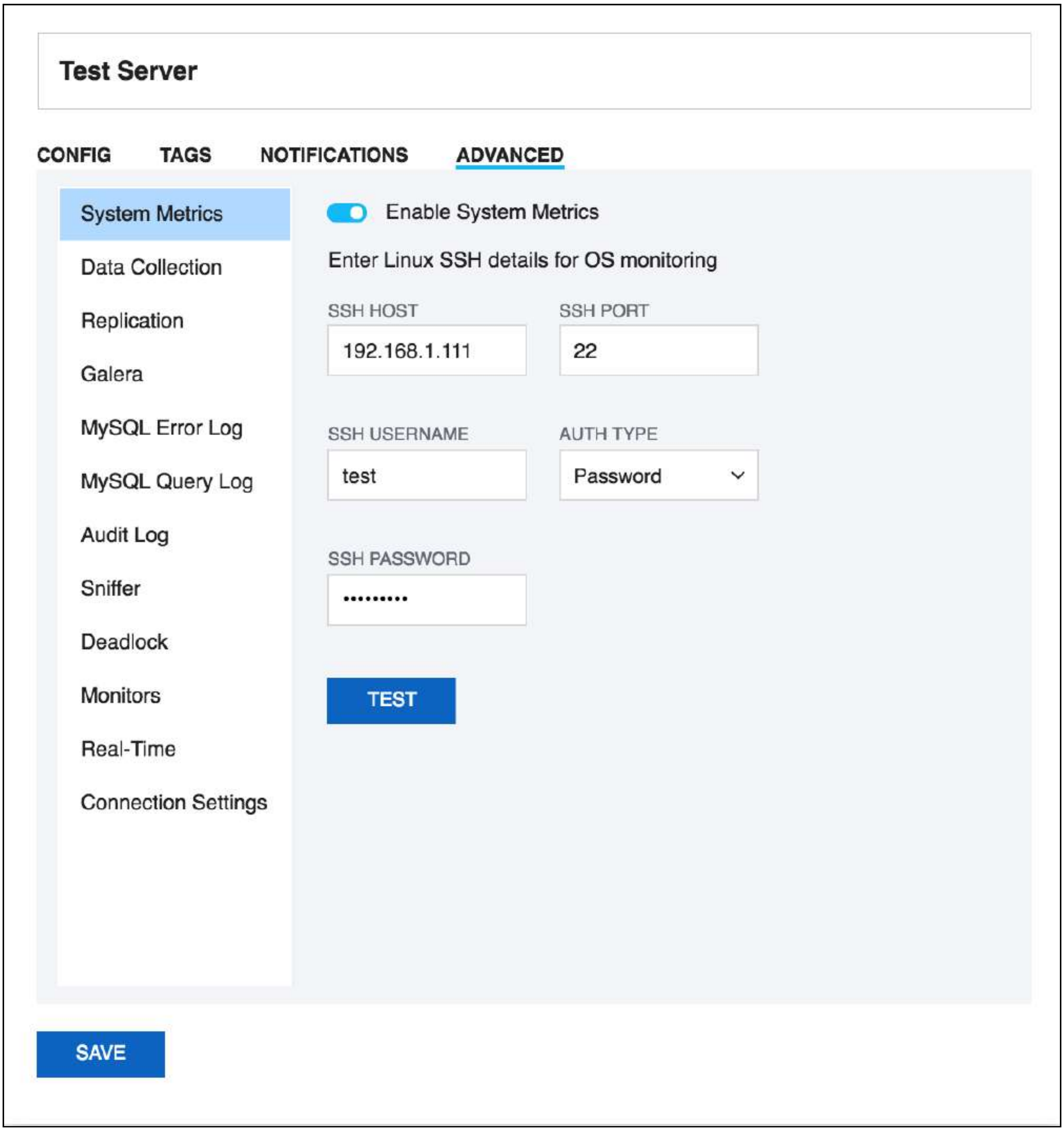
3.7.1.2 Using SSH connections

To create a SSH connection you need the following details:

- **SSH Host:** Host of the machine on which SSH server is running.
- **SSH Port:** Port on which SSH server is listening. By default, it is 22.
- **SSH Username:** Username to access the SSH server (Note: not the MySQL server).
- **Authentication type:** Specify the type of authentication to use. This can be either key based or password based.
- If you have specified Authentication type as Password - Provide the password.
- If you have specified, Authentication type as Key - You should note that SQL DM for MySQL only supports ''OpenSSH standard key format'' for key based authentication in SSH connections.

- **Private Key:** Paste the content of your private key file. Again, do not specify the path to your private key file.
- **Passphrase:** Enter the passphrase for your private key file (if any). This can be left blank, if no passphrase was given for the private key.

[28] http://wiki.idera.com/display/SQLDMYSQL/Replication+Settings
[29] http://wiki.idera.com/display/SQLDMYSQL/Galera
[30] http://wiki.idera.com/display/SQLDMYSQL/MySQL+Error+Log+Settings
[31] http://wiki.idera.com/display/SQLDMYSQL/MySQL+Query+Log
[32] http://wiki.idera.com/display/SQLDMYSQL/Audit+Log+Settings
[33] http://wiki.idera.com/display/SQLDMYSQL/Sniffer+Settings
[34] http://wiki.idera.com/display/SQLDMYSQL/Deadlock+Settings
[35] http://wiki.idera.com/display/SQLDMYSQL/Monitors+Settings
[36] http://wiki.idera.com/display/SQLDMYSQL/Real-Time+Settings
[37] http://wiki.idera.com/display/SQLDMYSQL/Connection+Settings
[38] http://wiki.idera.com/display/SQLDMYSQL/System+Privileges



Test Server

Enable System Metrics

Enter Linux SSH details for OS monitoring

SSH HOST: 192.168.1.111

SSH PORT: 22

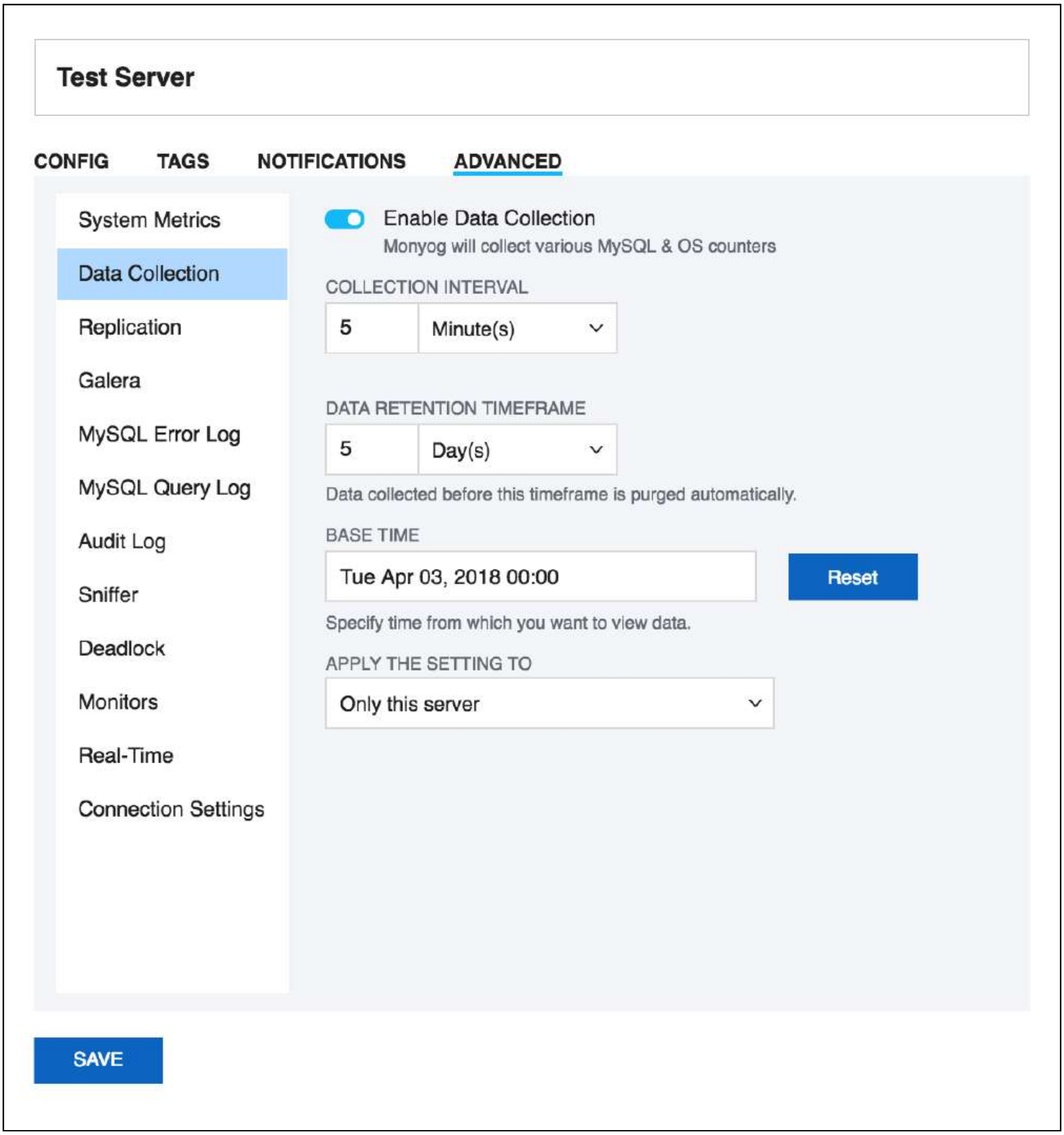
SSH USERNAME: test

AUTH TYPE: Password

SSH PASSWORD: •••••••••

3.7.2 Data Collection

If you want to collect data from the server, you need to select **Enable Data Collection**, so SQL DM for MySQL collects and stores various MySQL and OS metrics.



Test Server

Enable Data Collection
Monyog will collect various MySQL & OS counters

COLLECTION INTERVAL: 5 Minute(s)

DATA RETENTION TIMEFRAME: 5 Day(s)
Data collected before this timeframe is purged automatically.

BASE TIME: Tue Apr 03, 2018 00:00
Specify time from which you want to view data.

APPLY THE SETTING TO: Only this server

3.7.2.1 Define the collection interval for every server as you want

You can also define the time interval between two successive retrievals of data. For production systems a setting between 2 and 10 minutes is a good place to start.

3.7.2.2 Data retention time frame

SQL DM for MySQL is designed for storing large amounts of data for long periods of time. Data collected before the specified time frame is purged automatically. Time frame may be specified in seconds, minutes, hours and days for a particular server.

3.7.2.3 Base time

For calculation of uptime-based counters the current value of each status variable is compared with either of those,

- server status variable 'uptime'
- server status variable 'uptime_since_flush_status'
- SQL DM for MySQL 'base time' setting

If SQL DM for MySQL 'base time' setting is defined and server status variable uptime_since_flush_status is available then, `uptime_since_flush_status` is used, if it is not available then base time is used.

The reason for this implementation is that if `FLUSH STATUS` is executed with a MySQL server, the server status variables are reset to the same value as after a server restart. There is one important exception however and that is the 'uptime' status variable itself. This single status variable is not affected by `FLUSH STATUS`.

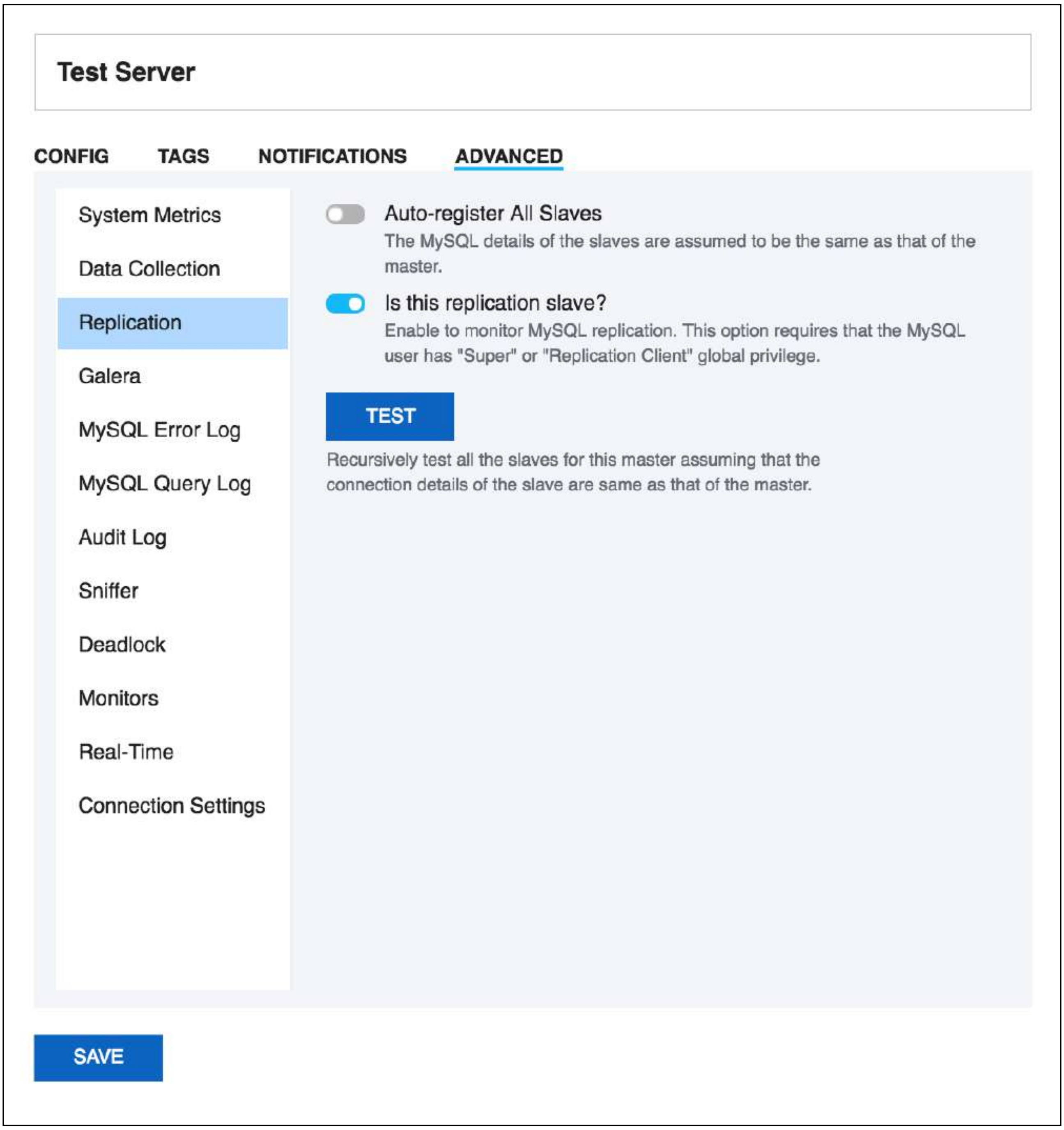
So, to get true uptime-based counters in SQL DM for MySQL with servers that do not support the `uptime_since_flush_status` variable you need to define a 'base time' in SQL DM for MySQL greater than or equal to the time where `FLUSH STATUS` was executed last time.

But also if `uptime` and/or `uptime_since_flush_status` is large ('old') you may use 'base time' setting to analyze uptime-based counters on an interval defined by you. For instance, if the server has been running for months you may choose to analyze uptime-base counters based on data collected from a specific time only as you have defined it.

Also, note that if the 'base time' is smaller than `uptime` (or `uptime_since_flush_status` if available), then 'base time' setting is ignored. Using a 'base time' larger than 'uptime' and/or `uptime_since_flush_status`, then base time is considered. If a base time is in future, then most recent collection time is considered (similar to Delta).

3.7.3 Replication Settings

SQL DM for MySQL monitors MySQL replicas by issuing a SHOW SLAVE STATUS on the slaves. SQL DM for MySQL can also auto-register slaves, given the master details.



Test Server

Auto-register All Slaves

The MySQL details of the slaves are assumed to be the same as that of the master.

Is this replication slave?

Enable to monitor MySQL replication. This option requires that the MySQL user has "Super" or "Replication Client" global privilege.

Recursively test all the slaves for this master assuming that the connection details of the slave are same as that of the master.

3.7.3.1 Replication Slave

Select the toggle switch for Is this a replication slave? To monitor MySQL replication. This option requires that the MySQL user has "Super" or "Replication Client" global privilege.

3.7.3.2 Automatic registering of slaves

This feature of SQL DM for MySQL saves you time from registering each slave individually. In order for SQL DM for MySQL to auto-register all slaves, select the toggle switch to enable **Auto-Register slaves** in the Advanced settings tab while registering a slave. If a master is already registered, click **Edit Server,** and check the Auto-Register slaves in the Advanced settings tab. The MySQL and the SSH details of the slaves are assumed to be the same as that of the master. In case the slave details are different from that of the master, you have to manually edit that server and change details.

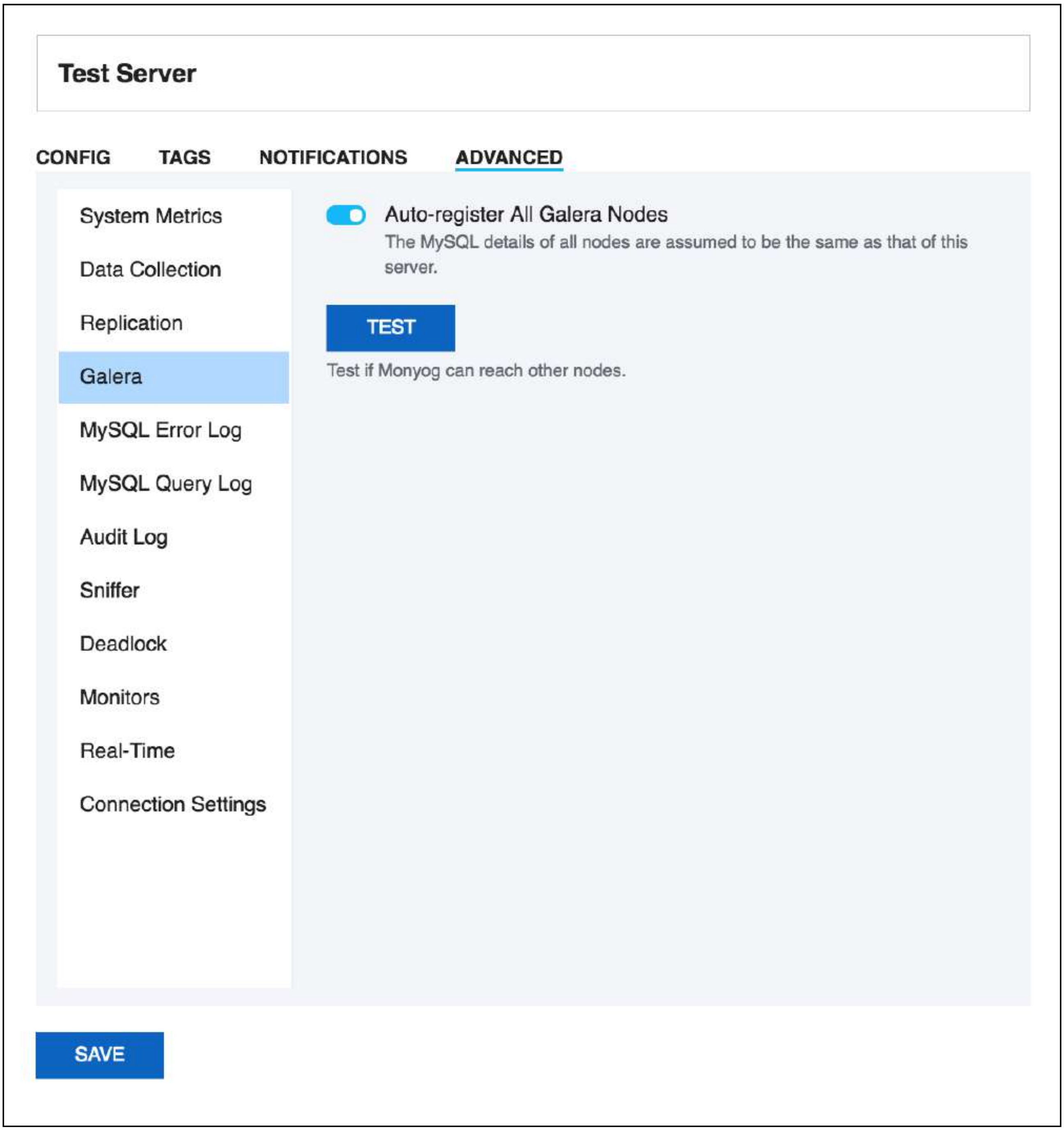
The auto-registering of slaves is extended to multiple levels of replication. For instance, lets say Server A is a Master that has Server B as the slave. And Server B has Server C as its slave. In such case, while registering Server A, if you check **Auto-Register Slaves**, then it registers A, B, and C provided the MySQL and the SSH details of the A is same as that of B.

How does SQL DM for MySQL auto-register all slaves of a given master? SQL DM for MySQL shoots a SHOW FULL PROCESSLIST on the master, and checks for all the slaves connected. (It assumes that MySQL and the SSH details of the slaves are the same as that of the master.) To view replication topology click the Replication tab[39].

3.7.4 Galera

Use this option to Auto register all the Galera nodes of your cluster with SQL DM for MySQL. The MySQL and the SSH details of the nodes are assumed to be the same as that of the node on which you are enabling this option. In case the other node details are different from that of the node on which you are enabling this option, you need to manually edit that server, and change details. You can do a **Test** to check if SQL DM for MySQL is able to connect to the other nodes. If the **Test** gives a successful message then you can go ahead and click **Save**. SQL DM for MySQL registers the detected nodes and redirects you to the Servers page where you can see all the registered nodes.

[39] http://wiki.idera.com/display/SQLDMYSQL/Replication



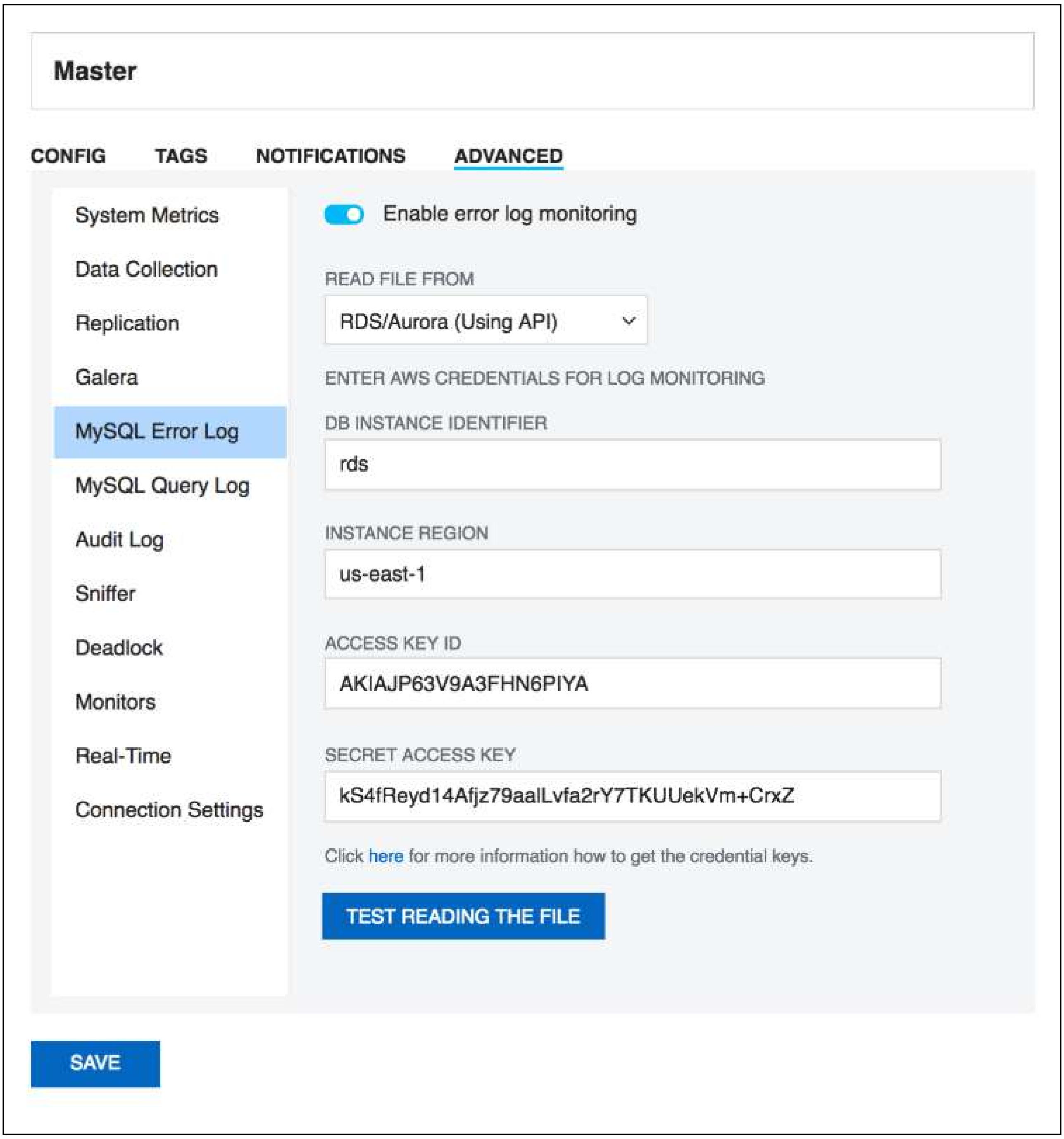
Test Server

Auto-register All Galera Nodes
The MySQL details of all nodes are assumed to be the same as that of this server.

Test if Monyog can reach other nodes.

3.7.5 MySQL Error Log Settings

The MySQL error log is quintessential in determining the health of the server. You can **Enable error log monitoring** to allow SQL DM for MySQL to keep an eye on your MySQL Error Log and notify you of important information.



Master

Enable error log monitoring

READ FILE FROM: RDS/Aurora (Using API)

ENTER AWS CREDENTIALS FOR LOG MONITORING

DB INSTANCE IDENTIFIER: rds

INSTANCE REGION: us-east-1

ACCESS KEY ID: AKIAJP63V9A3FHN6PIYA

SECRET ACCESS KEY: kS4fReyd14Afjz79aalLvfa2rY7TKUUekVm+CrxZ

Click here for more information how to get the credential keys.

3.7.5.1 Enable error log monitoring:

Select the toggle switch to **Enable error log monitoring**.

3.7.5.2 Read file from:

There are 3 ways of accessing the log files: Select ***"Local path"*** if the logs are in the machine where SQL DM for MySQL is running, or if they can be accessed by SQL DM for MySQL on a shared network drive. Choose ***"Via SFTP"*** if you have configured SQL DM for MySQL to use SSH. Select ***"RDS/Aurora (Using API)"*** if your server is a RDS/Aurora instance. In case of RDS/Aurora (Using API) for file based logging, four additional fields have to be filled, which are:

- **DB instance identifier:** A unique name to identify your RDS/Aurora instance.
- **Instance region:** The region in which your instance is hosted, for e.g: us-east-1
- **Access key ID:** It is a 20 character long key ID which can be created from the AWS Management console. It is used to make programmatic request to AWS.
- **Secret access key:** It is 40 character long and can be created from the AWS Management console. You can refer to the documentation, on how to generate credential keys here: Getting Your Access Key ID and Secret Access Key(see page 138).

Fetch error log details:

SQL DM for MySQL can automatically get the path of the error log from the MySQL server. Just click the fetch button, and SQL DM for MySQL will do the rest for you.

File path:

If you choose to enter the error log file path manually, you may do so here.

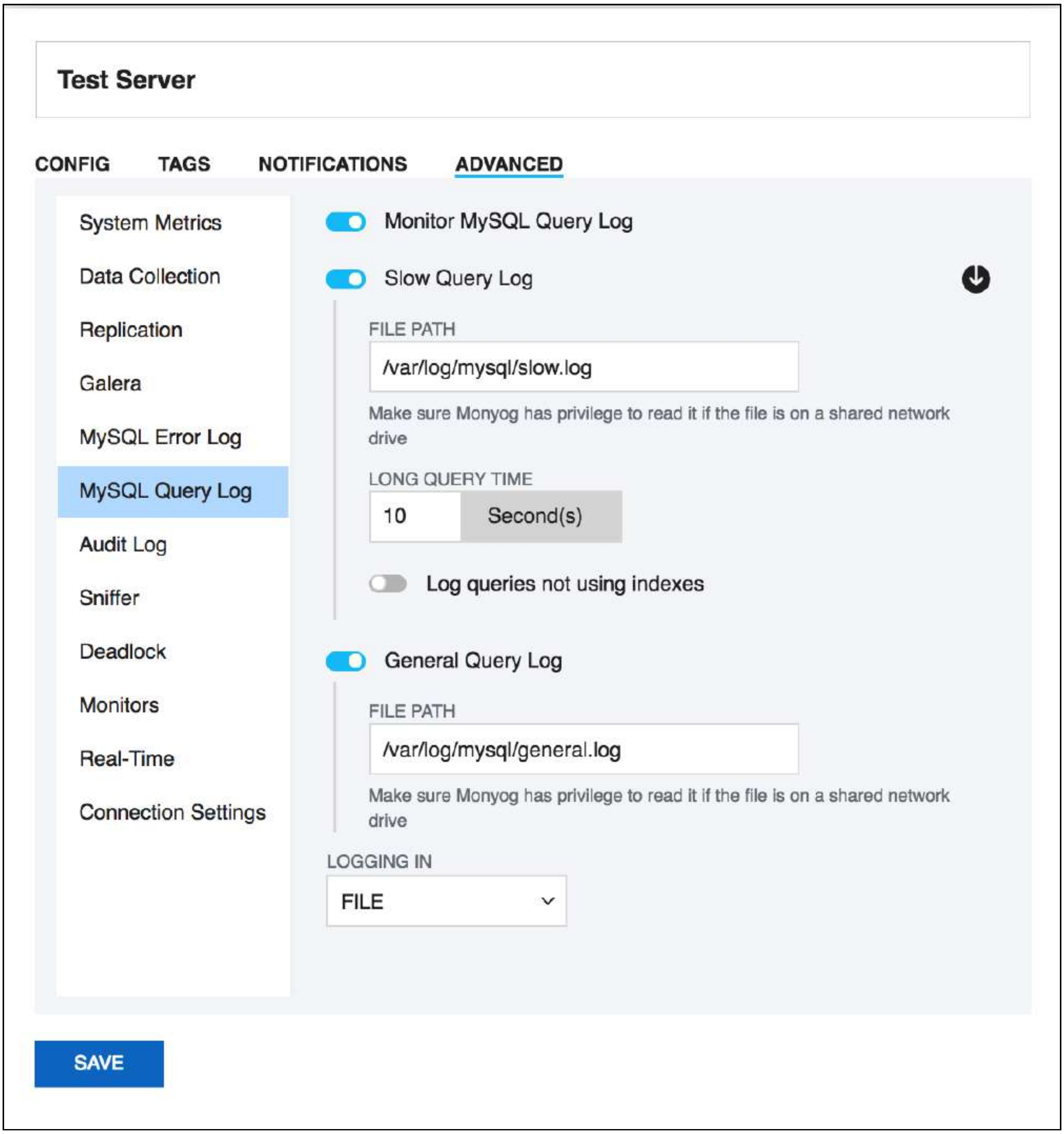
3.7.5.3 Test path:

Click this button to check if SQL DM for MySQL can access the file specified in the File path.

3.7.6 MySQL Query Log

SQL DM for MySQL retrieves (completely or partially) the General query log and the Slow query log from the MySQL servers it connects to, and analyzes them. Here, we see how to set up details for the connection, so that log analysis is available with SQL DM for MySQL. You have to set up details for the general query log and the slow query log independently. Enabling **Slow Query Log** 'log queries not using indexes' instead needs SUPER privilege. Refer to the MySQL documentation on how to enable and configure logging. MySQL server logs can be written to files on the server machine or to tables in the MySQL database itself.

The MySQL server (since version 5.0) has an option to log (in the slow log) queries that do not use an index. Such queries need not be slow if there are only a few hundred or few thousand records in the table(s) involved. But they are 'potentially slow' and should be identified if they access tables,which will continue to grow. You can enable and disable this from here too (SQL DM for MySQL will send the appropriate SET of statements to MySQL) Note: Only DML and DDL queries are recorded in the slow query log.



Test Server

Monitor MySQL Query Log

Slow Query Log

FILE PATH

/var/log/mysql/slow.log

Make sure Monyog has privilege to read it if the file is on a shared network drive

LONG QUERY TIME

10 Second(s)

Log queries not using indexes

General Query Log

FILE PATH

/var/log/mysql/general.log

Make sure Monyog has privilege to read it if the file is on a shared network drive

LOGGING IN

FILE

3.7.6.1 Logs written to files:

First, lets consider the situation where server logs are stored as files on the server machine. This is the most common situation and the only one available with MySQL servers before version 5.1. First time you configure a server with this option, click the **Fetch query log details** button. The MySQL server knows (it is stored in server variables) what logs are enabled and how logging is configured. Click **Test Path** to verify that the path SQL DM for MySQL connects and verifies the existence of the file (but not its content).

The log files can be accessed from the local file system (if SQL DM for MySQL and MySQL is running on the same computer) or by using SFTP (if SQL DM for MySQL and MySQL is running on different computers) or by using RDS/ Aurora (Using API) if you are using a RDS/Aurora instance. Note that you must use the file and path syntax of the machine where the logs are.

If the log files can be accessed from a shared drive, over a network, or from a network enabled file system (like NFS on Linux), then SQL DM for MySQL can access them as if they were local files. No additional SSH/SFTP configuration is required in this case: the operating system takes care of the file transfer transparently.

When **via SFTP** option is chosen, then SSH server details as defined in SSH server details settings are used to read the file from the remote system. Note that the SSH user must have read access to the log files.

When RDS/Aurora (Using API) option is chosen, make sure that you have the required Access credentials with you to fetch the log files. The Access credentials can be generated from the AWS Management Console. You can refer to the documentation, on how to generate credential keys here: Getting Your Access Key ID and Secret Access Key(see page 140).

By default, MONyog(SQL DM for MySQL) service runs under Local System Account. If you have Slow query or General query logs in a Mapped Network Drive, SQL DM for MySQL is not be able to reach it. You need to use UNC notation for SQL DM for MySQL to be able to access them. See FAQ 31(see page 304) for details.

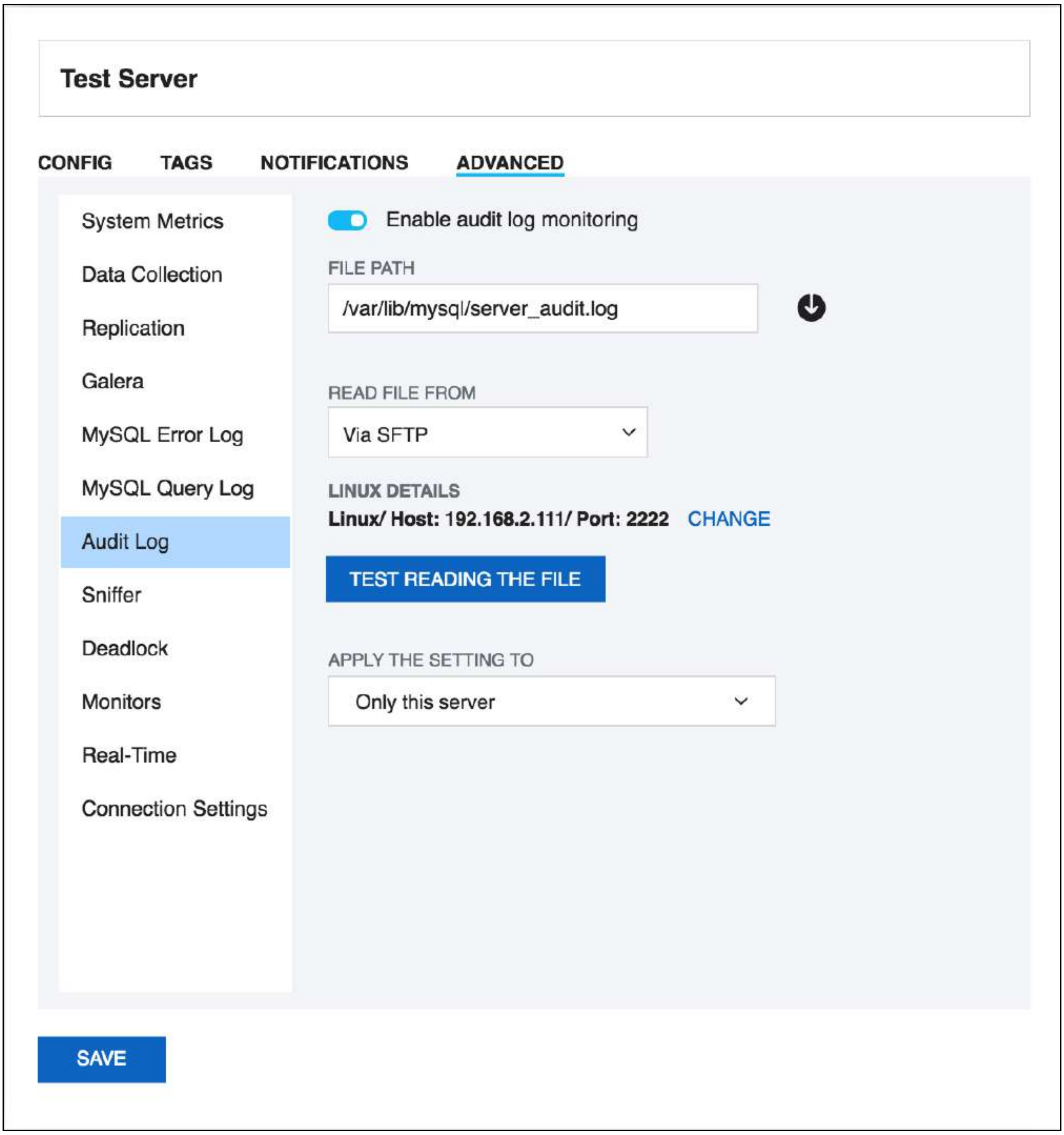
3.7.6.2 Logs written to MySQL tables:

This option is supported by MySQL from version 5.1. Also, SQL DM for MySQL supports when this option is available. Here, you click the **Fetch Log Details From MySQL** button. When this option is used there is no file path to configure and no SSH details to consider. SQL DM for MySQL can retrieve the server log by sending simple `SELECT` statements. Only the MySQL user used by SQL DM for MySQL to connect to MySQL must have `SELECT` privileges to the tables.

3.7.7 Audit Log Settings

This works exactly the same way as the MySQL Error log and MySQL Query Log. On enabling **audit log monitoring**, SQL DM for MySQL fetches the path from the server and displays it in the **File Path** box. Note that, we get the path from the variable "server_audit_file_path", and by default it just returns the audit log file name. In such cases, you have to manually enter the path for the audit log (by default, the path is same as datadir path).

Next, depending on where you have the MySQL server running, select an appropriate option for **Read File From**. If the server is on the same machine as SQL DM for MySQL, choose **Local path**. If it is on a remote machine, choose **Via SFTP** and give the corresponding SSH details. If the server is a RDS/Aurora server, then choose **RDS/Aurora (Using API)**.



Test Server

Enable audit log monitoring

FILE PATH

/var/lib/mysql/server_audit.log

READ FILE FROM

Via SFTP

LINUX DETAILS

Linux/ Host: 192.168.2.111/ Port: 2222

APPLY THE SETTING TO

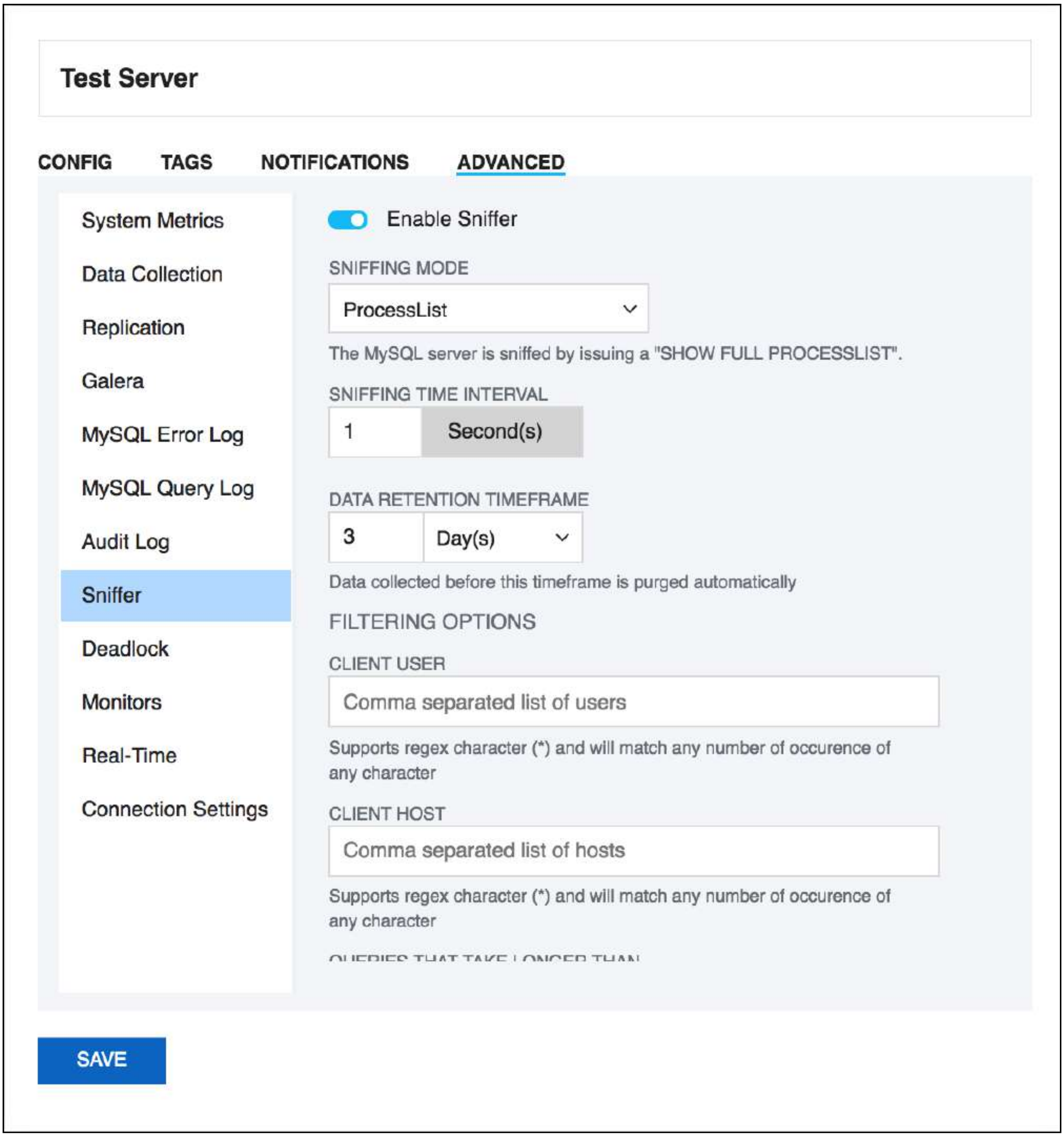
Only this server

3.7.8 Sniffer Settings

SQL DM for MySQL query sniffer is a functionality that records a pseudo server log and stores it in the SQL DM for MySQL embedded database. With **Sniffer** enabled, SQL DM for MySQL can populate the pseudo server log in three different ways at the intervals you specify:

- By utilizing Performance Schema tables (events_statements_summary_by_digest, events_statements_history_long) and collecting snapshots at regular intervals.

- By sending the query SHOW FULL PROCESSLIST to the MySQL server.
- Or by connecting to a running instance of the MySQL-Proxy program that is used by one or more clients to connect to a MySQL server.



Enable Sniffer

SNIFFING MODE: ProcessList

The MySQL server is sniffed by issuing a "SHOW FULL PROCESSLIST".

SNIFFING TIME INTERVAL: 1 Second(s)

DATA RETENTION TIMEFRAME: 3 Day(s)

Data collected before this timeframe is purged automatically

FILTERING OPTIONS

CLIENT USER: Comma separated list of users

Supports regex character (*) and will match any number of occurence of any character

CLIENT HOST: Comma separated list of hosts

Supports regex character (*) and will match any number of occurence of any character

QUERIES THAT TAKE LONGER THAN

For MySQL 5.6.14 and above you can use Performance schema(if Performance Schema is enabled), Proxy and Processlist for query analysis. If using MySQL version less than 5.6.14 then you can use Processlist mode.

Performance Schema on MySQL contains queries executed on server along with other information

- Number of rows sent and examined

- Number of temporary tables created on disk
- Number of temporary tables created on memory
- Number of joins performed and the type of join
- Whether sorting happened and the type of sort
- Whether index used
- Whether good index used

SQL DM for MySQL uses performance schema statement digest `table(events_statements_summary_by_digest)` to get the above information and is dependent on the `statements_digest` in `setup_consumers` table. By default, this is enabled. If not, it can be enabled by executing the following:

```
UPDATE performance_schema.setup_consumers
SET enabled = 'YES'
WHERE name = 'statements_digest';
```

Example query is available in `events_statements_history_long` table and has to be enabled and is dependent on the `events_statements_history_long` in `setup_consumers` table. By default, this is not enabled and should be enabled by executing the following:

```
UPDATE performance_schema.setup_consumers
SET enabled = 'YES'
WHERE name = 'events_statements_history_long';
```

The `performance_schema.events_statements_summary_by_digest` table size is dependent on `performance_schema_digests_size` global variable. By default, this size is set to 5000 rows. Once it reaches this limit you may lose the queries. SQL DM for MySQL provides an option to truncate the performance schema digest table when it reaches 80% of `performance_schema_digests_size`.

Although configuring a Proxy instance is a little more complicated, the PROXY-based sniffer has several advantages over the PROCESSLIST-based, including:

1. All queries that was handled by the Proxy will be recorded by SQL DM for MySQL sniffer when PROXY option is used. When PROCESSLIST option is used very fast queries may execute completely between two SHOW FULL PROCESSLIST queries and will then not be recorded.
2. You can choose to analyze queries from specific client(s)/application(s) only. Simply let (only) the clients that you want to focus on at the moment connect through the Proxy.
3. When using the PROXY option you can distribute part of the load generated by the sniffer on the machine that fits best in your deployment scenario (like on the one that has most free resources available) by deciding where to have the PROXY: The MySQL machine, the SQL DM for MySQL machine (if not the same) or quite another machine. The machine running MySQL will have no additional load due to the sniffer if the Proxy is not running on that machine.

Also note that, if more SQL DM for MySQL instances use the same PROXY they use the same data collected, when the Proxy Sniffing is enabled by the first SQL DM for MySQL instance. To work with SQL DM for MySQL sniffer the MySQL Proxy instance must be started with the name of a LUA script called MONyog.LUA (LUA is a scripting/programming language) as argument and is distributed with SQL DM for MySQL. You can find it in the MONyog program folder after installing (Windows and Linux RPM) or unpacking (Linux .tar.gz) the SQL DM for MySQL program package as downloaded from the IDERA website. The MySQL Proxy program however you need to download from MySQL website (we cannot include it for license reasons). SQL DM for MySQL works with Proxy versions from 0.61 to

0.81(latest currently) with the exception of 0.7x versions for windows and Mac due to a bug in those specific builds. For more information on Proxy, see MySQL Proxy(see page 143).

To start a Proxy instance for use with SQL DM for MySQL use the command:

- For v0.81(Alpha) and later, run the following common from the Proxy installation folder:

```
# mysql-proxy --proxy-backend-addresses=192.168.x.x:3305 \
--proxy-address=192.168.y.y:4045 \
--admin-username=root \
--admin-password=root \
--admin-lua-script=MONyog.lua \
--proxy-lua-script=MONyog.lua
```

- For Older versions, from the Proxy installation folder, run:

```
# mysql-proxy --proxy-backend-addresses=192.168.x.x:3305 \
--proxy-address=192.168.y.y:4045 \
--proxy-lua-script=MONyog.lua
```

(It is assumed that the 'MONyog.LUA' was copied to the folder where the PROXY binary is). Also note that, if no port is specified the PROXY listens on port 4040. Now, you can connect to the Proxy from one or more clients/applications. The Proxy sends queries to MySQL and the results back to the client. But when started with the LUA script for SQL DM for MySQL sniffer it also sends information to SQL DM for MySQL that SQL DM for MySQL uses to populate the sniffer 'pseudo log'.

Once this 'seudo log has been recorded (in either of the three ways described: Performance Schema, PROCESSLIST or PROXY-based) the SQL DM for MySQL log analysis functionalities can operate on the pseudo log as well as the real logs. The data recorded in the pseudo log is purged automatically based on the data retention timeframe option set by you.

Further some filtering options are provided. This filtering happens before storing to the SQL DM for MySQL database. This prevents the sniffer database to grow out of control. The filtering options are as follow:

- **User and host**: You can choose to store queries executed only by a specific combination of users and/or hosts.
- **Minimum time taken**: For every `PROCESSLIST` returned to SQL DM for MySQL, the queries are recorded in the embedded database only if they have been executing for a time greater than the specified minimum execution time. Furthermore, if a query with the same structure and details (like process ID) as one already recorded is encountered, the embedded database is UPDATED, and the statement is recorded only once.

> ⚠ This setting should be somewhat larger than the sample interval (and also consider the latency of the connection). If set lower it would not really make much sense.

- **Queries starting with**: Enter any string and only queries starting with that string are recorded. Examples: `SELECT *`, `UPDATE Customer_Base`.

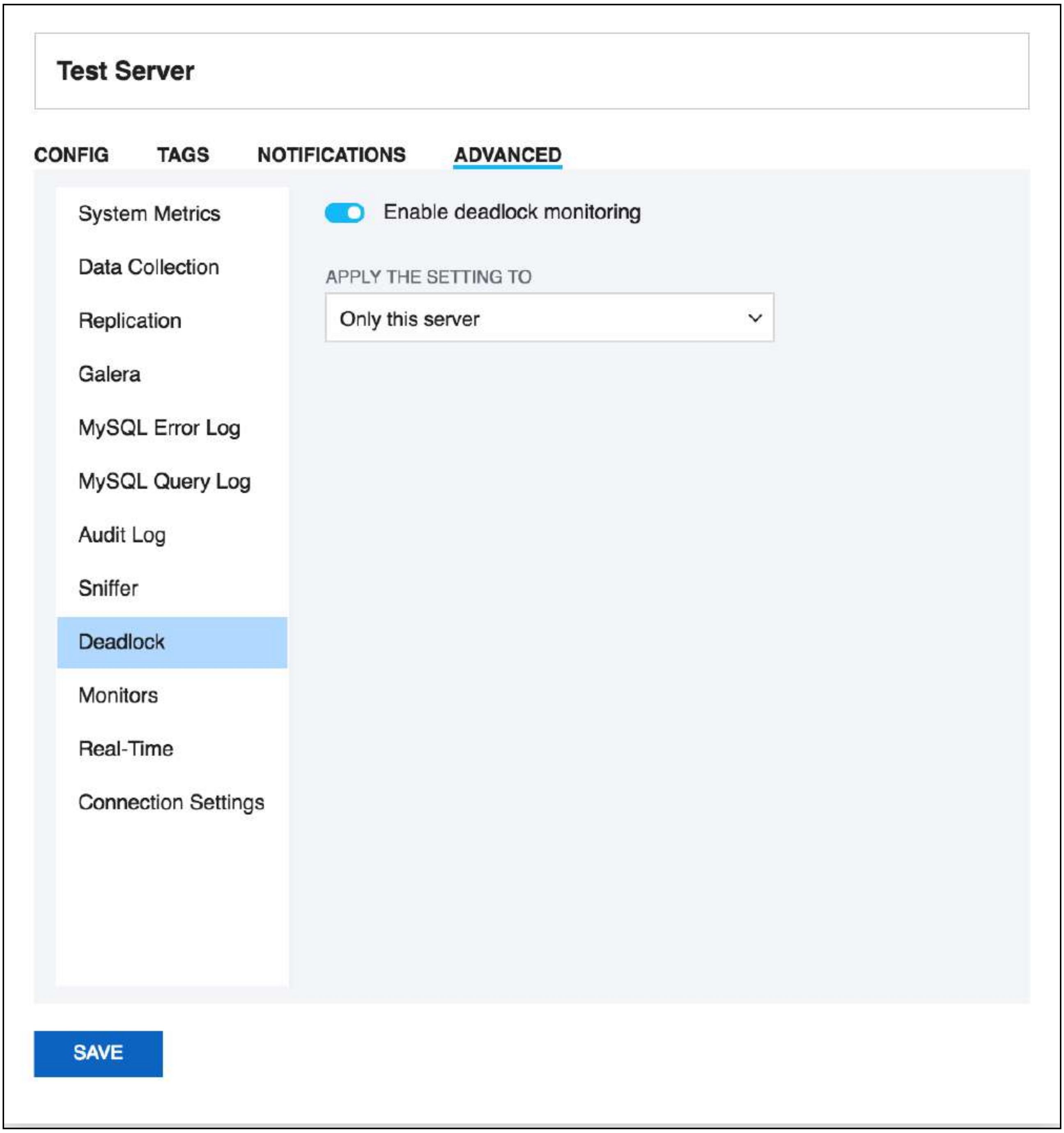
Also note that in `PROCESSLIST` Sniffer we have an option **Long Running Query Options** where you can monitor the long running queries by notifying or killing a query which takes more than a time specified by you. You can also specify users whose queries will be ignored (i.e. queries by such user are never killed by SQL DM for MySQL) and never raise an alert even if they take a longer than the time specified under 'LONG RUNNING QUERY TIME' you specified.

Clicking **Monitor only locked queries** would only monitor those long queries that are locked.

You should note that the query sniffer is not a complete 'general log'. Very fast statements may or may not be recorded as they may or may not finish executing between two PROCESSLISTs generated. The time interval between subsequent data collections for the 'pseudo log' depends on the connection to the MySQL server.

3.7.9 Deadlock Settings

In transactional databases deadlocks are a classic problem, but these deadlocks are not too dangerous unless they are so frequent that you cannot run certain transactions at all. To trace the deadlocks reported by `INNODB STATUS`, you can select the toggle switch to "**Enable deadlock monitoring**" option.



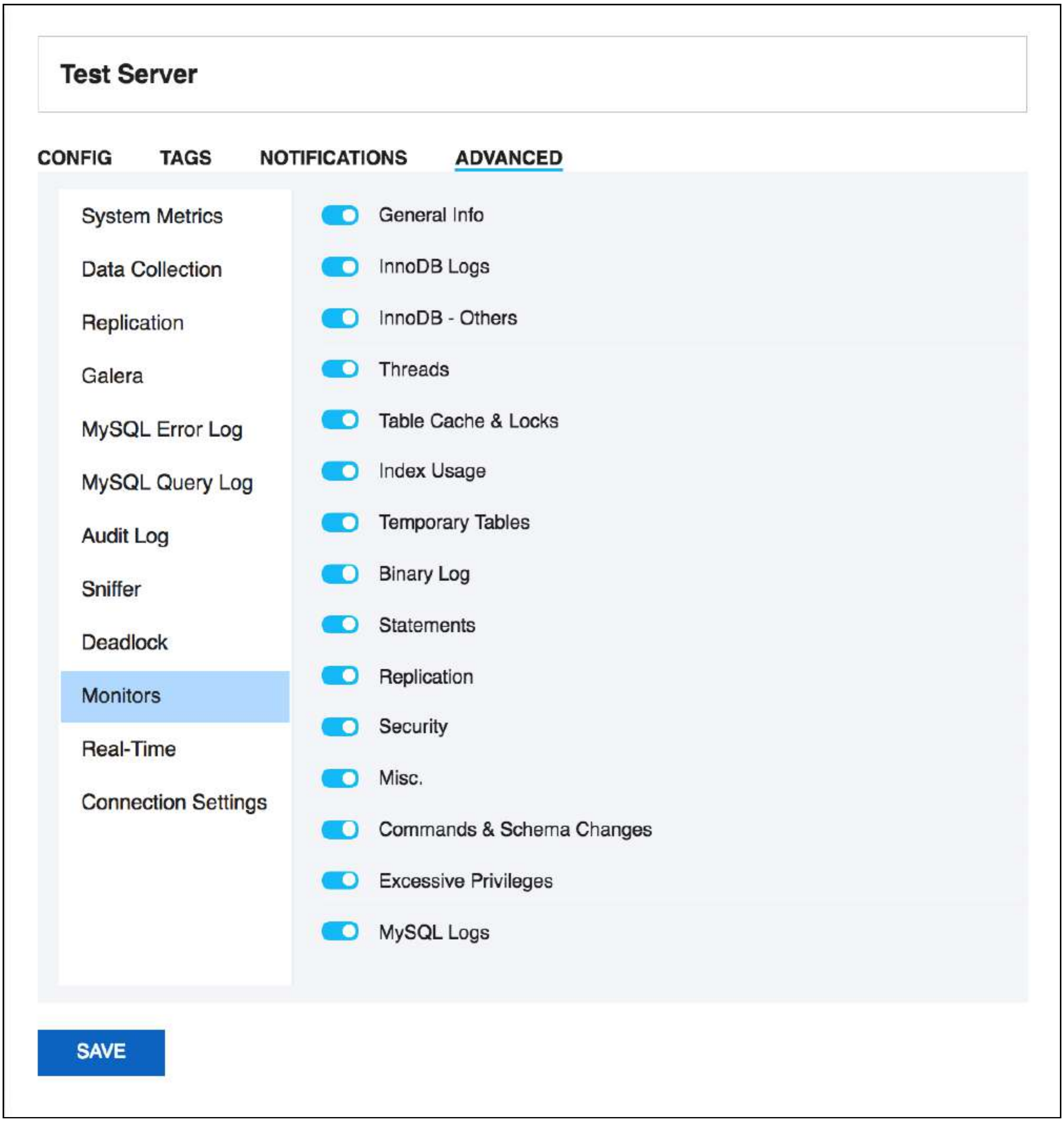
Test Server

Enable deadlock monitoring (enabled)

APPLY THE SETTING TO: Only this server

3.7.10 Monitors Settings

SQL DM for MySQL provides a way of disabling an entire group of Monitors. For instance if a MySQL server is not a replication slave, then the replication group can be disabled.



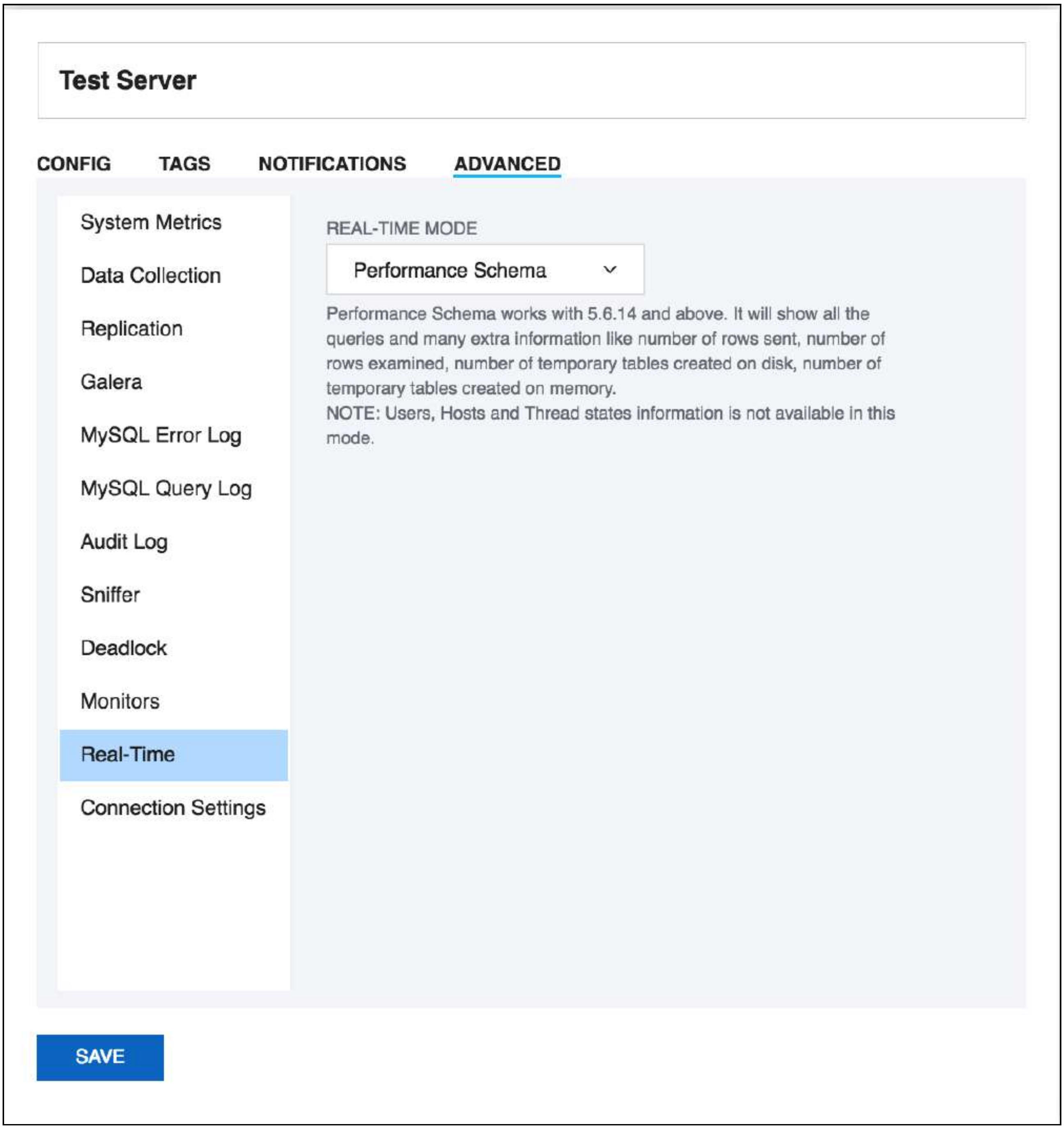
Test Server

- General Info
- InnoDB Logs
- InnoDB - Others
- Threads
- Table Cache & Locks
- Index Usage
- Temporary Tables
- Binary Log
- Statements
- Replication
- Security
- Misc.
- Commands & Schema Changes
- Excessive Privileges
- MySQL Logs

3.7.11 Real-Time Settings

SQL DM for MySQL provides you the option to choose the data collection mode for Real-time monitoring.

You can choose between **Processlist** and **Performance schema**. Select **Performance schema** mode if your MySQL version is 5.6.14 or above and if performance schema is enabled, otherwise you can go with **Processlist** mode.



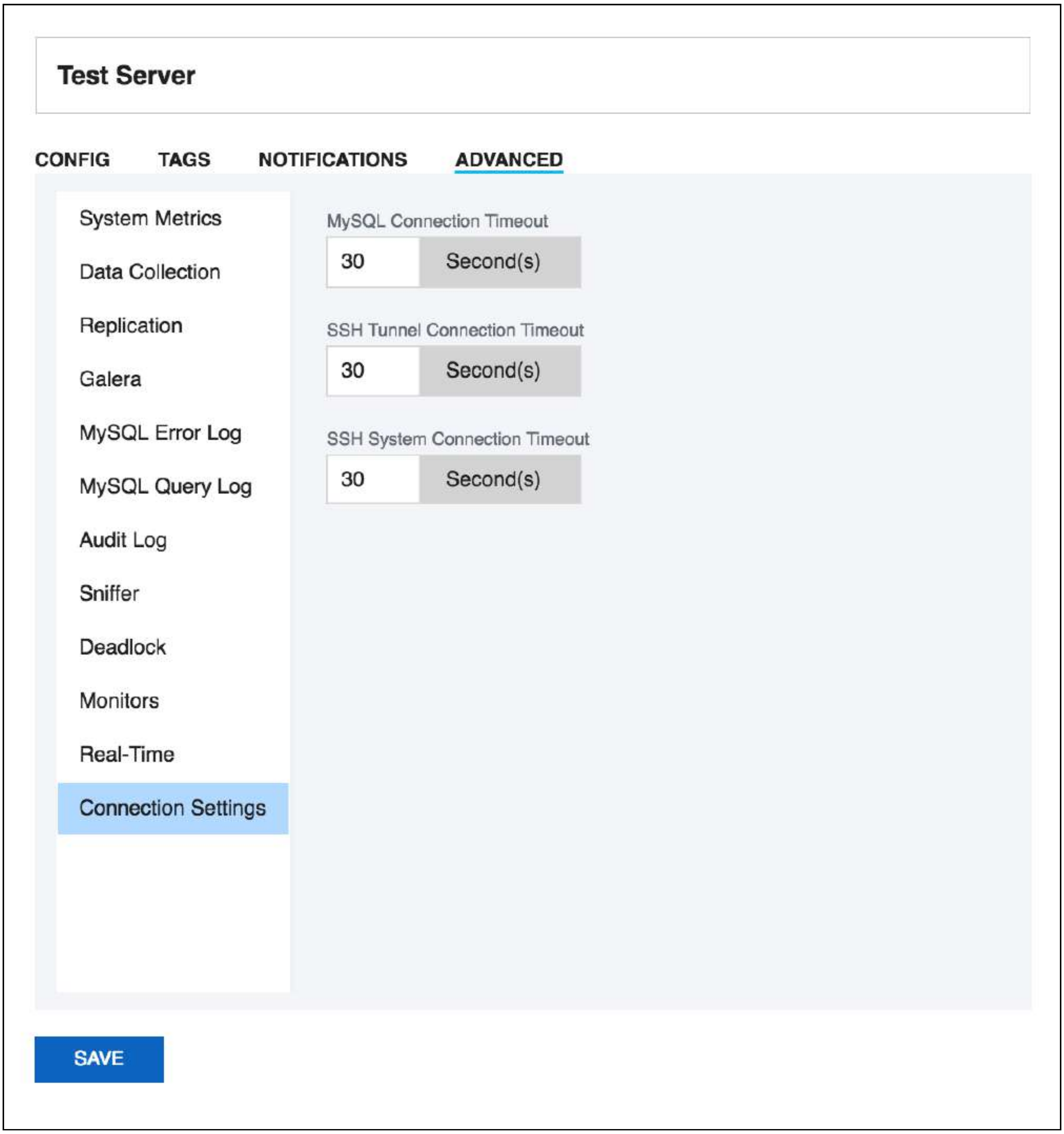
REAL-TIME MODE

Performance Schema

Performance Schema works with 5.6.14 and above. It will show all the queries and many extra information like number of rows sent, number of rows examined, number of temporary tables created on disk, number of temporary tables created on memory.
NOTE: Users, Hosts and Thread states information is not available in this mode.

3.7.12 Connection Settings

You can specify the **MySQL Connection Timeout** value for your server. This option simply means that SQL DM for MySQL waits for this long to get a response from the server before it throws an error. This comes handy to avoid false positives when connection to some specific servers is slow. You can setup a larger timeout in such cases. If you have SSH Tunneling enabled to the MySQL then you can specify a **SSH Tunnel Connection Timeout**, and a **SSH System Connection Timeout** if System Metrics is enabled. The default value for all being 30 seconds.



MySQL Connection Timeout: 30 Second(s)

SSH Tunnel Connection Timeout: 30 Second(s)

SSH System Connection Timeout: 30 Second(s)

3.8 MySQL Privileges

3.8.1 MySQL Privileges

Whether you are running a single MySQL database server or multiple servers in a cluster, SQL DM for MySQL provides a comprehensive list of performance variables that you can monitor in real-time to check the health and performance of your MySQL server or servers.

In order to access the data for these metrics, SQL DM for MySQL requires a dedicated user on each MySQL database server it monitors, with the appropriate privileges to collect the information it needs.

3.8.1.1 Privileges

Most SQL DM for MySQL functionalities do not require any privileges at all for the user it uses in establishing client connections with MySQL. You can create it without any special global or object privileges, using the following command:

```
GRANT USAGE ON *.*
TO 'user'@'host';
```

Fully enabling all SQL DM for MySQL functionality requires that you grant the user some additional privileges, ensuring that it can access all the data it needs.

⚠ For an example of granting full functionality to the SQL DM for MySQL user, see Database Configuration <db-config> below.

3.8.1.2 Security Counters

SQL DM for MySQL user requires `SELECT` privileges to the `mysql.user` table to enable SQL DM for MySQL security counters.

Additionally, if the server was started with the `skip_show_databases` configuration variable, you must also explicitly grant the `SHOW DATABASES` privilege.

3.8.1.3 Replication Counters

In order to collect replication metrics from a slave, the MySQL user on the slave requires the `REPLICATION_CLIENT` privilege. You can also enable this functionality using the `SUPER` privilege.

When registering MySQL servers with SQL DM for MySQL, there is an option **Is this a Replication Slave?** to include replication information. You must set this to **Yes**, when registering the server. It defaults to **No**, when set to No, SQL DM for MySQL ignores replication data.

When testing the connection from the server registration page, SQL DM for MySQL displays the error message:

```
Access denied. You need the SUPER/REPLICATION CLIENT
privilege for retrieving REPLICATION details!
```

You can ignore this error if the given MySQL server is not a replication slave or if you do not want to monitor replication.

3.8.1.4 InnoDB Deadlock Monitoring

Depending on the MySQL server version, the `SUPER` or `PROCESS` privileges allow you to monitor for deadlocks with the InnoDB storage engine.

> Beginning in the 5.1.24 version of MySQL, you can enable this feature using `PROCESS`. Older versions require `SUPER`.

3.8.1.5 Processlist Feature

In order to collect data on the MySQL server processes for all users, you must enable the `PROCESS` privilege. Additionally, when granted the `SUPER` privilege, you can use SQL DM for MySQL to kill running processes.

In order to use the `EXPLAIN` option on the processlist, you need to grant `SELECT` and `SHOW VIEW` to the objects accessed by the statements you want to explain, or simply, grant the global `SELECT` privilege.

> SQL DM for MySQL displays `N/A` on counters where the user lacks the required privileges and logs a record of the MySQL server error for every attempt to retrieve those counters from the server. Be aware the log may grow considerably if this is the case.

3.8.1.6 Performance Schema-based Sniffer

Collecting data on the Performance schema requires `SELECT`, `DROP` and `UPDATE` privileges.

SQL DM for MySQL uses the `SELECT` privilege to read the Performance Schema tables. The `DROP` privilege allows it to truncate these tables. The `UPDATE` privilege enables the statement digest and statement history log in the performance schema tables.

3.8.1.7 Log Retrieval

In order to retrieve log information when stored in a table, (which is supported in version 5.1 and newer of MySQL), the SQL DM for MySQL user requires the `SELECT` privilege on the log tables.

3.8.1.8 Flush Status

In order to execute `FLUSH STATUS` commands, the SQL DM for MySQL user also requires the `RELOAD` privilege.

3.9 System Privileges

To fully enable all SQL DM for MySQL functionalities you may also need to create one or more operating system users for specific purposes.

- **SSH tunneling:** SQL DM for MySQL can send and receive encrypted authentication information as well as monitoring data from MySQL using SSH tunneling. Also, it is possible to connect to MySQL with SSH tunneling even if the MySQL port (normally 3306) is blocked by a firewall or if users are not allowed to connect from remote hosts. An operating system user is required so that SQL DM for MySQL SSH client functionalities can use this user to connect to the SSHD daemon on the server.
- **System information (currently available for Linux servers):** For retrieving system information you need an Operating System user. The user must be a 'shell' user (it must be possible for this user to issue system commands). This user is specified for the SSH-based connection to the host.

  This user must have read-access to the Linux '/proc' folder. Normally, this is always the case for Linux users. Also to retrieve memory related information this user must have read access to the .pid files created by the MySQL server. An easy way to ensure this is to add the user to the `mysql` user group.

- **MySQL log information:** For retrieving MySQL log information (when stored as files - not when stored as MySQL tables) read-access to the MySQL log files are also required.

  You may (but need not) use the same operating system user for all the three above ways that SQL DM for MySQL connects to the OS.

3.10 RDS OS and File based Log Monitoring Privileges

AWS IAM Policy can be used to create permissions that specify which RDS actions a user, or a group of users in your AWS account can perform. IAM Policy is basically a JSON document that consists of one or more statements which defines the action to be taken on AWS resources. It can be used to determine who is allowed to create, delete, or modify RDS instances.

SQL DM for MySQL needs the following permissions to fetch the log files:

1. DescribeDBLogFiles: This API fetches a list of log files available for your instance.

2. DownloadDBLogFilePortion: This API downloads the specified log file.

For fetching the OS metrics, the following permission is needed:

1. GetMetricStatistics: This API fetches the average of CloudWatch metrics.

You can create a simple IAM Policy, giving the above permissions, for e.g:

```
{
"Version":"2012-10-17",
"Statement": [{
"Effect":"Allow",
"Action": [
"rds:DescribeDBLogFiles",
"rds:DownloadDBLogFilePortion"
"cloudwatch:GetMetricStatistics"
],
"Resource":"*"
}]
}
```

You can restrict the Resource for the above policy using this link(see page 154). In case, you want to perform either OS monitoring, or file-based log monitoring for your RDS/Aurora instance then you can include only those actions in the above policy.

You can use the default policy **CloudWatchReadOnlyAccess** provided by AWS for OS monitoring, in case you do not want to create a custom policy. Keep in mind that this policy grants more permissions than SQL DM for MySQL requires to fetch your RDS/Aurora metrics.

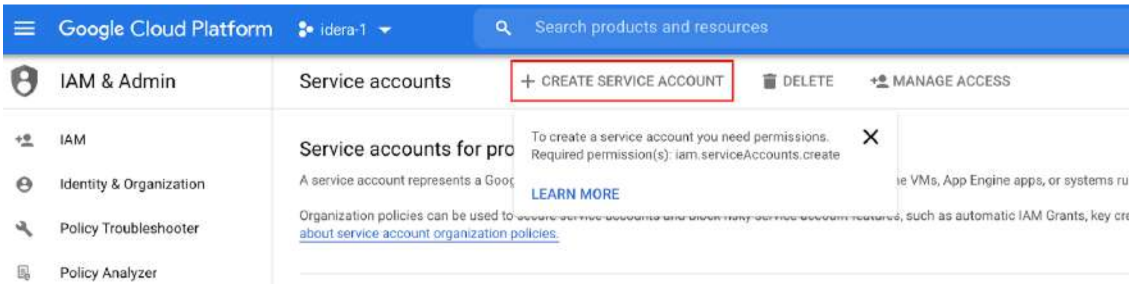
3.11 Google Cloud SQL Monitoring

For Google Cloud Monitoring you need the following prerequisites:

1. 3.11.1 to create a service account in your Google Cloud account and download the content of the JSON file

To generate service-account credentials and view the public credentials already created, follow these steps:

- Open the Service accounts page[40]. When prompted, select a project.
- Click **CREATE SERVICE ACCOUNT**.



Service accounts for pro

To create a service account you need permissions. Required permission(s): iam.serviceAccounts.create

A service account represents a Goog ie VMs, App Engine apps, or systems ru

Organization policies can be used to , such as automatic IAM Grants, key cre about service account organization policies.

- In the **CREATE SERVICE ACCOUNT** window, type a name for the service account, and select **Furnish** a new private key. Click **Create**.
- When you create a key, your new public/private key pair is generated and downloaded to your machine.

2. 3.11.2 To configure access to your Cloud SQL instance

Go to the Cloud SQL Instances page[41] in the **Google Cloud Console**.

- From the client machine, use What's my IP[42] to see the IP address of the client machine.
- Copy that IP address.
- Go to the Cloud SQL Instances page[43] in the **Google Cloud Console**.

[40] https://console.cloud.google.com/iam-admin/serviceaccounts
[41] https://console.cloud.google.com/sql/instances
[42] http://ipv4.whatismyv6.com/
[43] https://console.cloud.google.com/sql/instances

- Open the instance **Overview** page, and record the IP address.
- Select the **Connections** tab.
- Under **Authorized networks**, click **Add Network** and enter the IP address of the client machine.

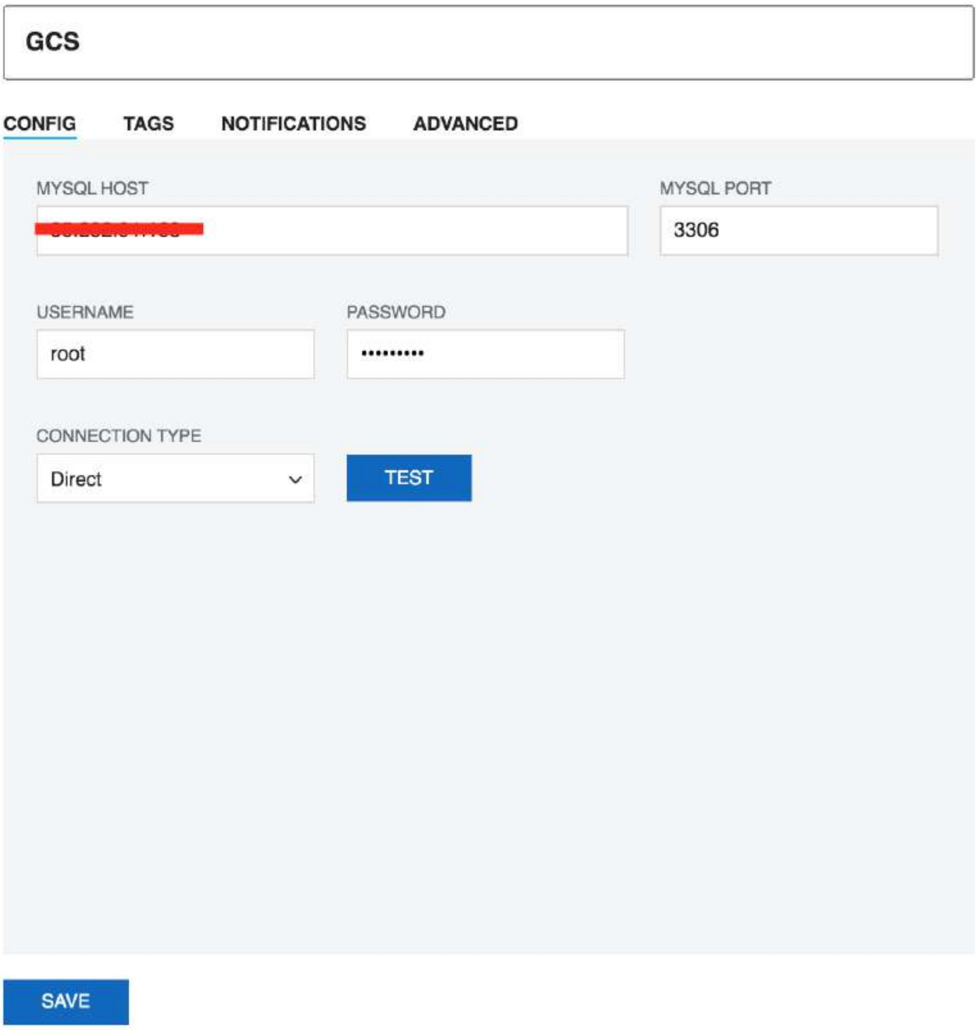
> The IP address of the instance and the MySQL client IP address you authorize must be the same IP version: either IPv4 or IPv6.

- Click **Done** and then **Save** at the bottom of the page to save your changes.

3.11.3 Adding your Google Cloud Server

To add your **Google Cloud Server** to SQL DM for MySQL, you need to follow these steps:

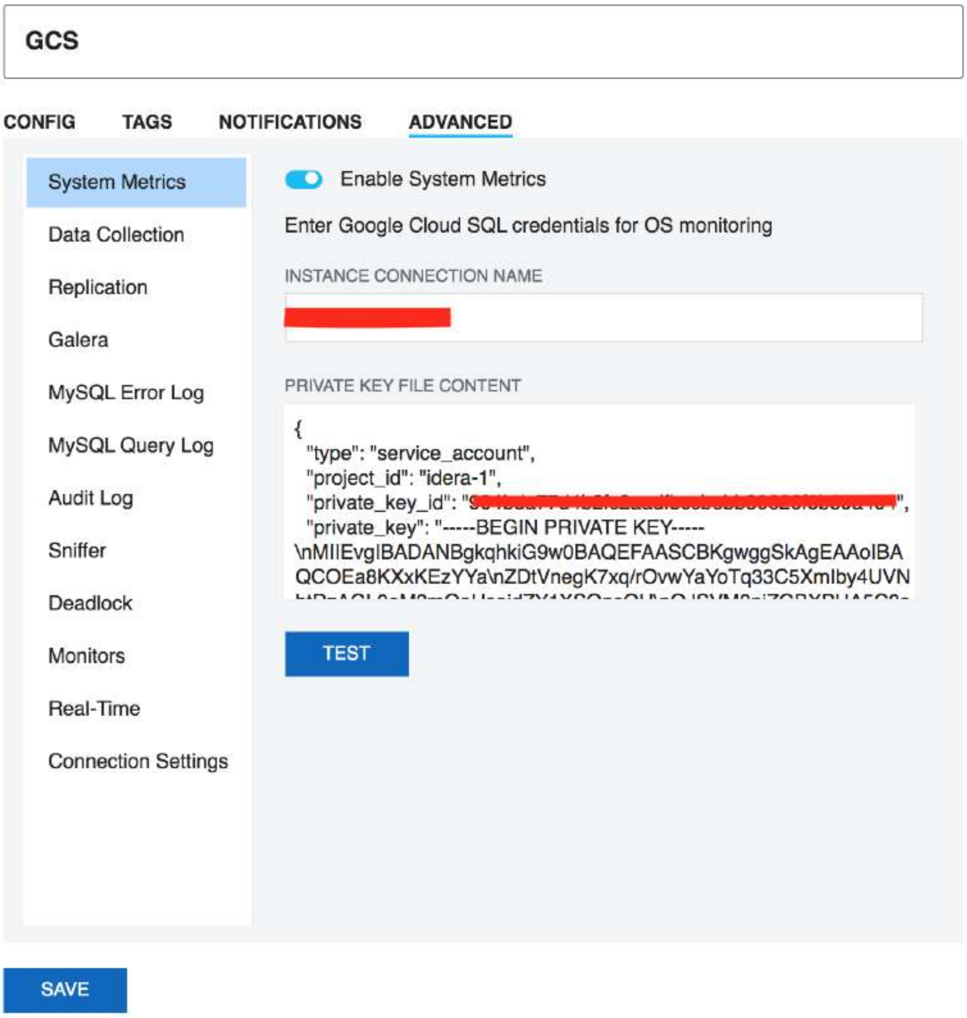
1. Provide the Host IP, MySQL Username, and Password to register your Google cloud SQL server in SQL DM for MySQL.



GCS

MYSQL HOST: [redacted]
MYSQL PORT: 3306
USERNAME: root
PASSWORD: •••••••••
CONNECTION TYPE: Direct

2. Enable **System Monitoring** and enter the instance name with the content of the private key JSON file. Go to Edit Server > Advanced > System Metrics.



GCS

● Enable System Metrics

Enter Google Cloud SQL credentials for OS monitoring

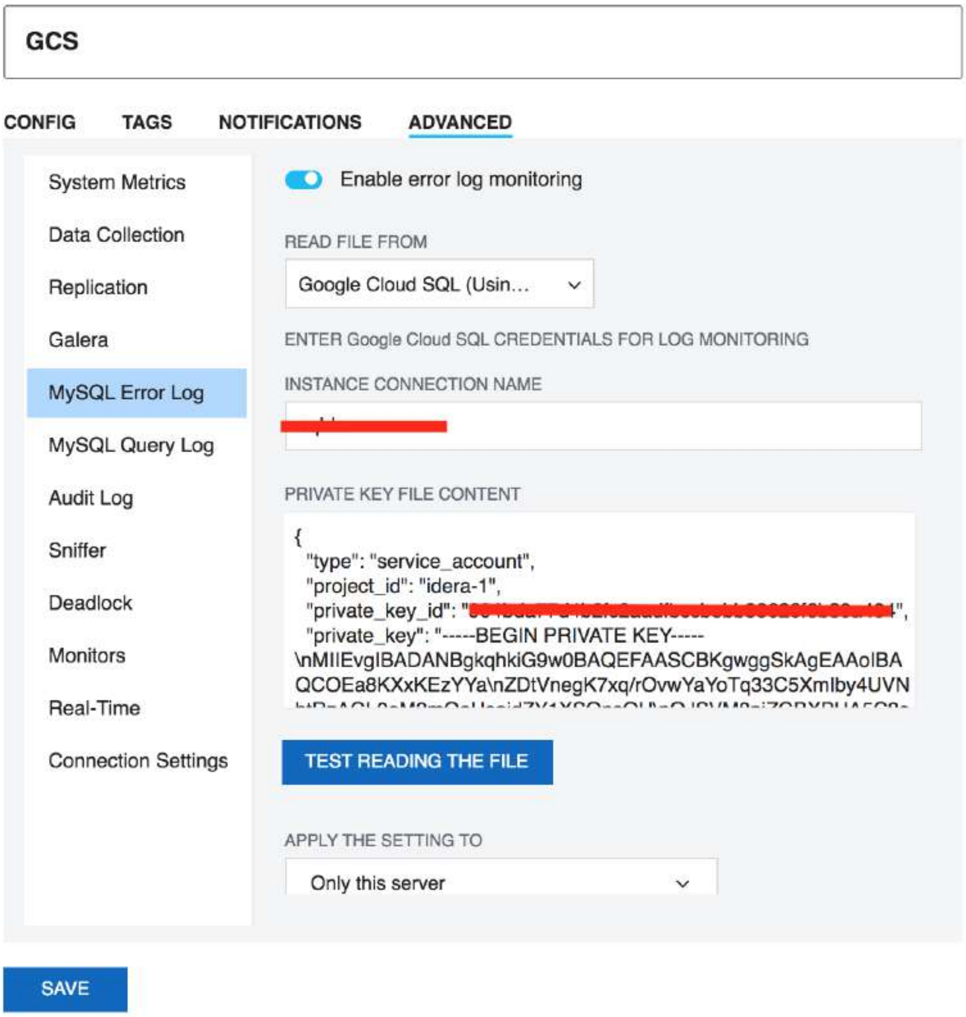
INSTANCE CONNECTION NAME

[redacted]

PRIVATE KEY FILE CONTENT

```
{
  "type": "service_account",
  "project_id": "idera-1",
  "private_key_id": "[redacted]",
  "private_key": "-----BEGIN PRIVATE KEY-----
\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBA
QCOEa8KXxKEzYYa\nZDtVnegK7xq/rOvwYaYoTq33C5XmIby4UVN
```

3. To enable **MySQL Error Log** monitoring, review the below screenshot:



Enable error log monitoring

READ FILE FROM: Google Cloud SQL (Usin...

ENTER Google Cloud SQL CREDENTIALS FOR LOG MONITORING

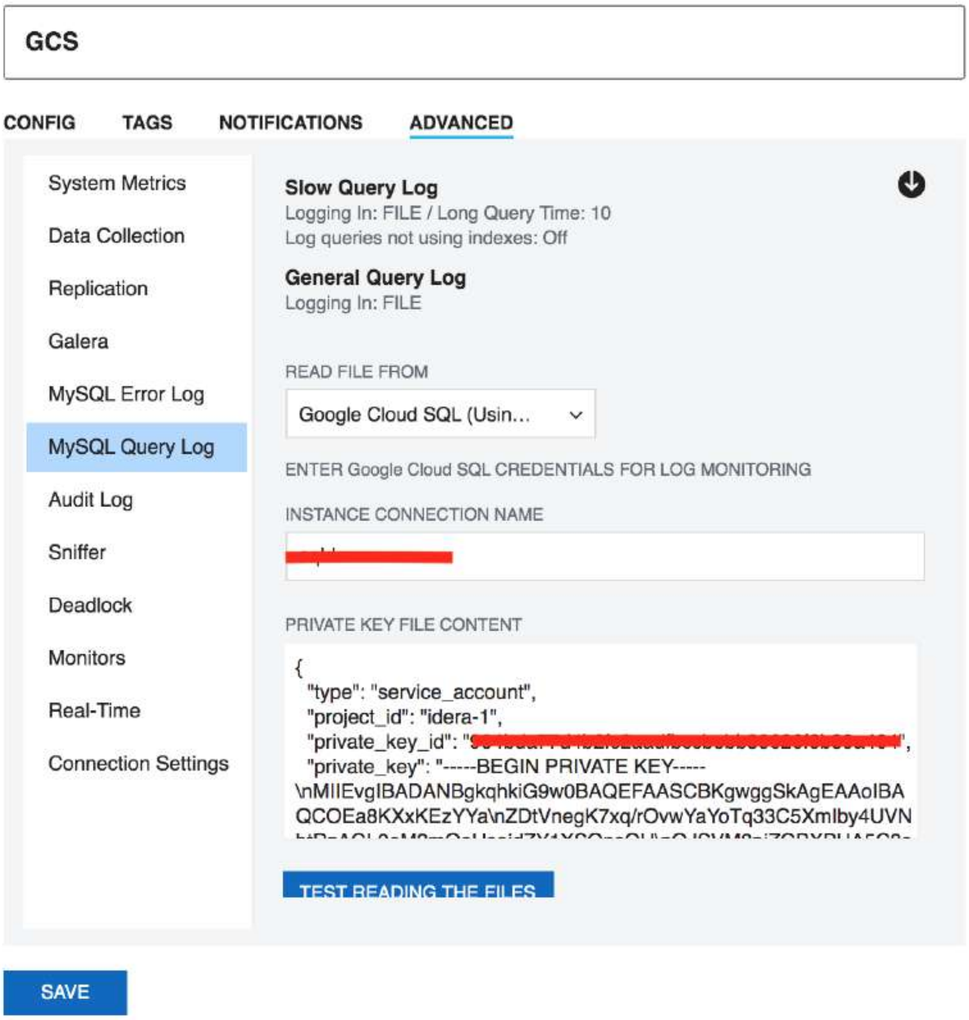
INSTANCE CONNECTION NAME: [redacted]

PRIVATE KEY FILE CONTENT

```
{
  "type": "service_account",
  "project_id": "idera-1",
  "private_key_id": "[redacted]",
  "private_key": "-----BEGIN PRIVATE KEY-----
\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBA
QCOEa8KXxKEzYYa\nZDtVnegK7xq/rOvwYaYoTq33C5XmIby4UVN
```

APPLY THE SETTING TO: Only this server

4. To enable **MySQL Query Log** monitoring, review the below screenshot:



Slow Query Log
Logging In: FILE / Long Query Time: 10
Log queries not using indexes: Off

General Query Log
Logging In: FILE

READ FILE FROM
Google Cloud SQL (Usin...

ENTER Google Cloud SQL CREDENTIALS FOR LOG MONITORING

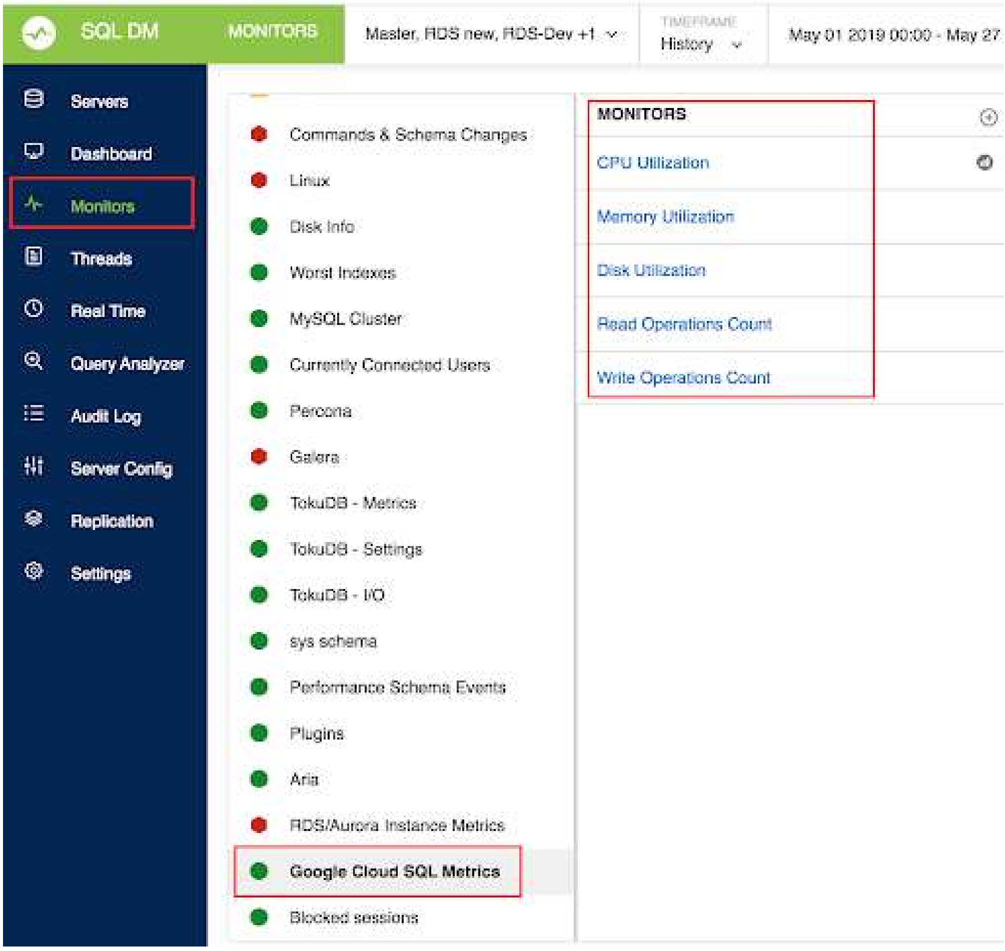
INSTANCE CONNECTION NAME
[redacted]

PRIVATE KEY FILE CONTENT

```
{
  "type": "service_account",
  "project_id": "idera-1",
  "private_key_id": "[redacted]",
  "private_key": "-----BEGIN PRIVATE KEY-----
\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBA
QCOEa8KXxKEzYYa\nZDtVnegK7xq/rOvwYaYoTq33C5XmIby4UVN
```

5. To enable **GCS Objects**, go to the **Monitors** page, click “**+**” near Monitors, and select **Manage Google Cloud SQL Objects**.

6. You can enable the below Monitors to monitor the **Google Cloud SQL Metrics**:
   - database/cpu/utilization
   - database/memory/utilization
   - database/disk/utilization
   - database/disk/read_ops_count
   - database/disk/write_ops_count



- ● Commands & Schema Changes
- ● Linux
- ● Disk Info
- ● Worst Indexes
- ● MySQL Cluster
- ● Currently Connected Users
- ● Percona
- ● Galera
- ● TokuDB - Metrics
- ● TokuDB - Settings
- ● TokuDB - I/O
- ● sys schema
- ● Performance Schema Events
- ● Plugins
- ● Aria
- ● RDS/Aurora Instance Metrics
- ● **Google Cloud SQL Metrics**
- ● Blocked sessions

MONITORS

- CPU Utilization
- Memory Utilization
- Disk Utilization
- Read Operations Count
- Write Operations Count

4 Navigate SQL DM for MySQL

SQL DM for MySQL is an agentless MySQL monitoring tool that helps DBAs reduce downtimes, tighten security, and optimize performance of your MySQL powered systems. Other characteristics can include reducing excessive CPU usage, query execution time, fast identification of queries causing performance issues, and high availability among others.

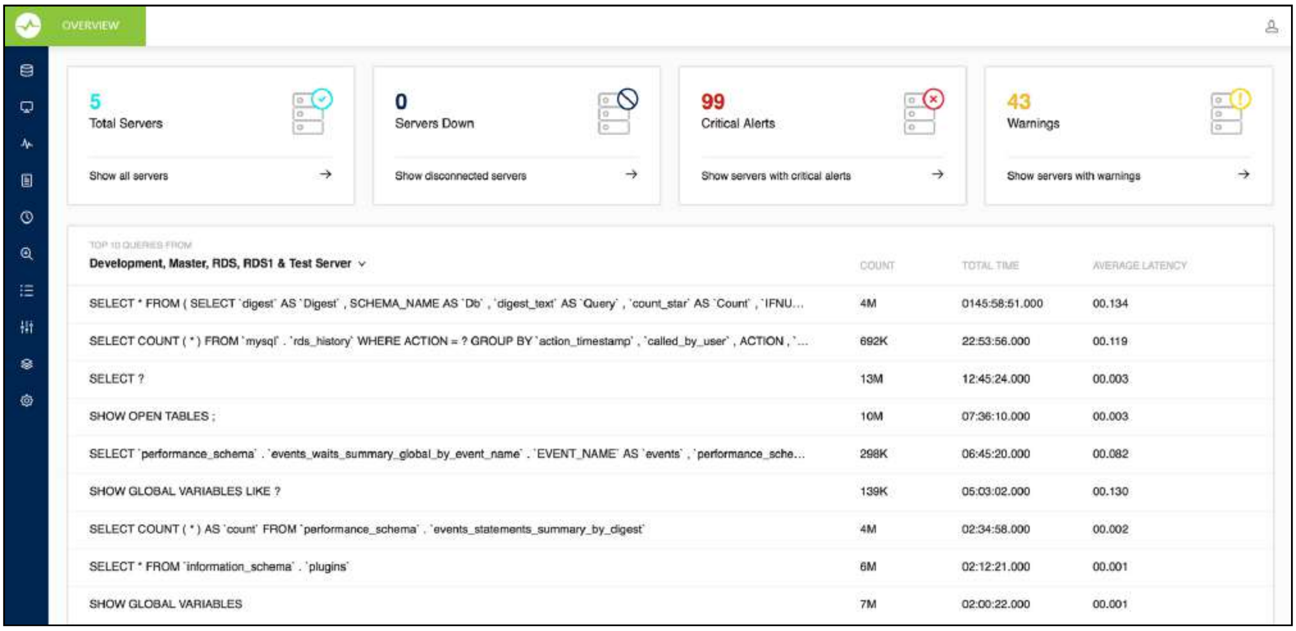
Below, review the different views and options that SQL DM for MySQL offers:

- Overview[44]
- Servers[45]
- Dashboard[46]
- Monitors[47]
- Threads[48]
- Real-Time[49]
- Query Analyzer[50]
- Audit Log[51]
- Server Configuration[52]
- Replication[53]

4.1 Overview

The Overview gives the top level picture of all the servers registered with SQL DM for MySQL. It gives the count of the total servers registered with SQL DM for MySQL, total number of disconnected servers, servers having critical alerts, and the servers having warnings. You can further drill down to Servers page with the corresponding categories from the Overview page.

[44] http://wiki.idera.com/x/fgEvvgE
[45] http://wiki.idera.com/x/fwEvvgE
[46] http://wiki.idera.com/x/ggEvvgE
[47] http://wiki.idera.com/x/iAEvvgE
[48] http://wiki.idera.com/x/mgEvvgE
[49] http://wiki.idera.com/x/oAEvvgE
[50] http://wiki.idera.com/x/owEvvgE
[51] http://wiki.idera.com/x/pQEvvgE
[52] http://wiki.idera.com/x/pwEvvgE
[53] http://wiki.idera.com/x/qgEvvgE



5 Total Servers — Show all servers →

0 Servers Down — Show disconnected servers →

99 Critical Alerts — Show servers with critical alerts →

43 Warnings — Show servers with warnings →

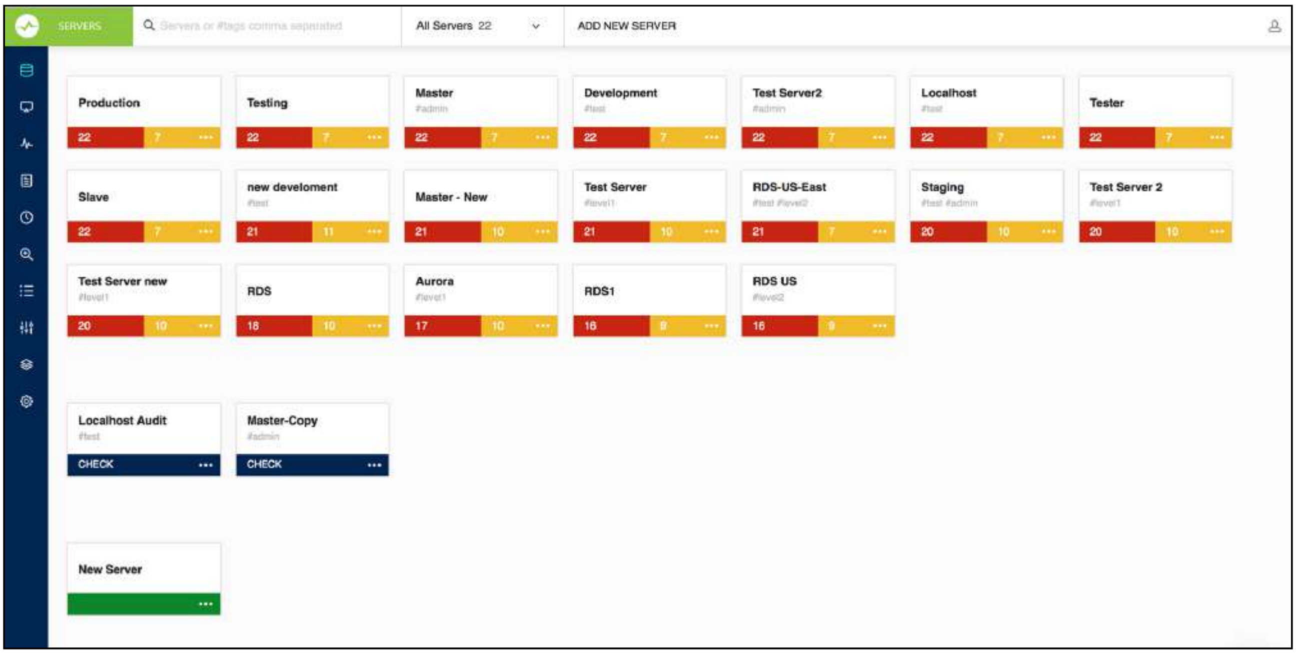
TOP 10 QUERIES FROM
Development, Master, RDS, RDS1 & Test Server

	COUNT	TOTAL TIME	AVERAGE LATENCY
SELECT * FROM (SELECT `digest` AS `Digest` , SCHEMA_NAME AS `Db` , `digest_text` AS `Query` , `count_star` AS `Count` , `IFNU...	4M	0145:58:51.000	00.134
SELECT COUNT (*) FROM `mysql` . `rds_history` WHERE ACTION = ? GROUP BY `action_timestamp` , `called_by_user` , ACTION , `...	692K	22:53:56.000	00.119
SELECT ?	13M	12:45:24.000	00.003
SHOW OPEN TABLES ;	10M	07:36:10.000	00.003
SELECT `performance_schema` . `events_waits_summary_global_by_event_name` . `EVENT_NAME` AS `events` , `performance_sche...	298K	06:45:20.000	00.082
SHOW GLOBAL VARIABLES LIKE ?	139K	05:03:02.000	00.130
SELECT COUNT (*) AS `count` FROM `performance_schema` . `events_statements_summary_by_digest`	4M	02:34:58.000	00.002
SELECT * FROM `information_schema` . `plugins`	6M	02:12:21.000	00.001
SHOW GLOBAL VARIABLES	7M	02:00:22.000	00.001

Overview also gives you the Top 10 Queries across all the servers registered with SQL DM for MySQL based on total execution time and also lists the count and Average latency for each of the queries. The queries are basically filtered from the "*sniffer*.data" file of all the registered servers. You can click the queries to get the query details which also gives you the list of server names on which the particular query was executed. Clicking the server names to open Sniffer for the selected server, with the time-range selected as the first seen and the last seen of the query.

4.2 Servers

Servers provides an unified view of all the servers registered with SQL DM for MySQL. Users can view which servers are running at the optimum level and identify servers which require critical attention. The servers page allows you to add or delete a server as required.



Production 22 7 •••

Testing 22 7 •••

Master #admin 22 7 •••

Development #test 22 7 •••

Test Server2 #admin 22 7 •••

Localhost #test 22 7 •••

Tester 22 7 •••

Slave 22 7 •••

new develoment #test 21 11 •••

Master - New 21 10 •••

Test Server #level1 21 10 •••

RDS-US-East #test #level2 21 7 •••

Staging #test #admin 20 10 •••

Test Server 2 #level1 20 10 •••

Test Server new #level1 20 10 •••

RDS 18 10 •••

Aurora #level1 17 10 •••

RDS1 16 9 •••

RDS US #level2 16 9 •••

Localhost Audit #test CHECK •••

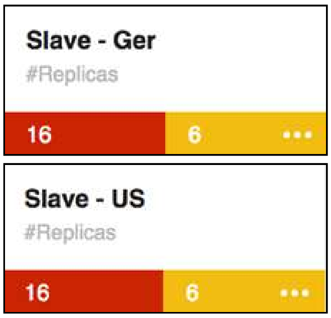
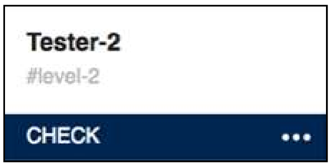
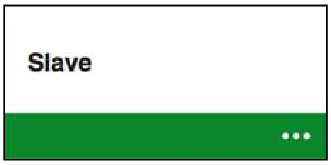
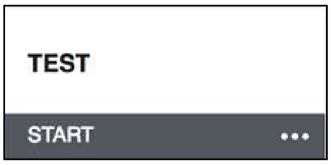
Master-Copy #admin CHECK •••

New Server •••

With the All-New SQL DM for MySQL, you are to able to easily filter MySQL servers and find the details of your preferred server.

4.2.1 Server Block

The Server Block lets you identify the state of the MySQL server:

Status	Description	Server Block
Alerting	MySQL servers that has critical alerts and warnings.	 Slave - Ger #Replicas 16 6 ••• Slave - US #Replicas 16 6 •••
Disconnected	SQL DM for MySQL is unable to connect with the MySQL server.	
Stable	MySQL servers that do not have critical alerts and warnings.	
Stopped	SQL DM for MySQL is not collecting data.	

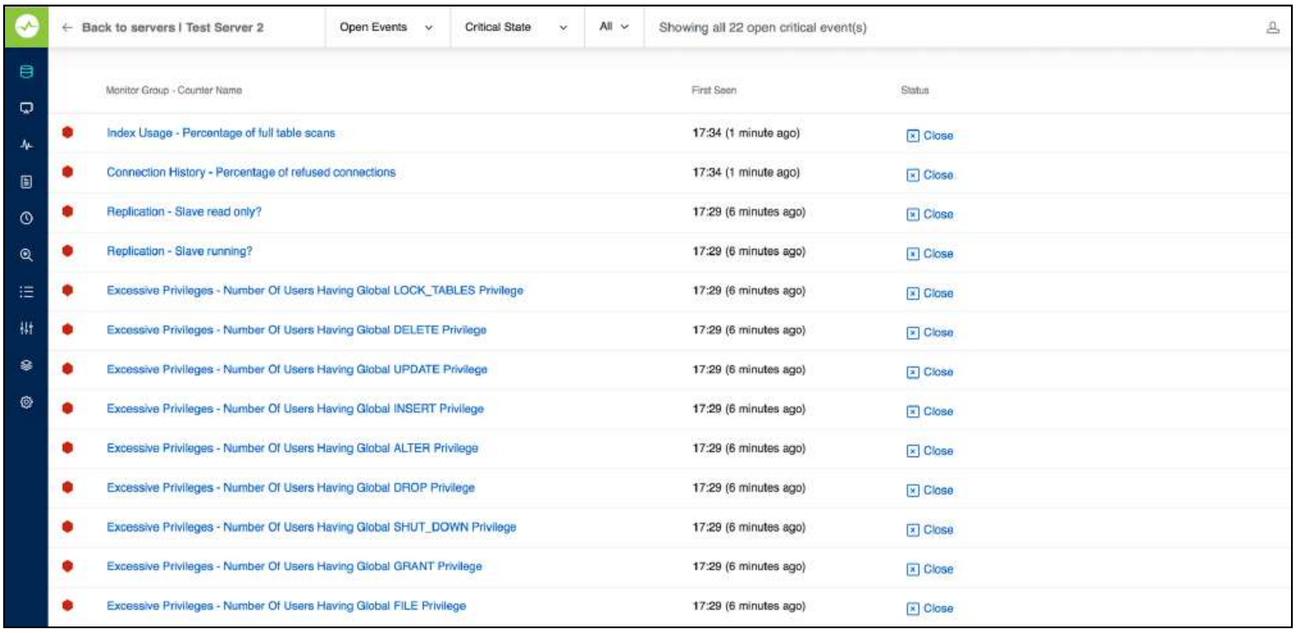
4.2.2 Server options

The server tab provides different options to the user to manage their registered MySQL servers. Users can edit, duplicate or delete MySQL server(s), command SQL DM for MySQL to stop the data collection for a particular server, rebuild database (please refer to SQL DM for MySQL's Data Maintenance[54]), check disk info, analyze all the events, and run diagnosis reports.

[54] http://wiki.idera.com/x/sQEvvgE

4.2.3 Events

An EVENT happens when any counter is changing its status to (yellow) WARNING alert level or to (red) CRITICAL alert level. And this EVENT is over when the counter becomes stable. The EVENT table gives an overall view of all such EVENTS that have happened on each server.



← Back to servers I Test Server 2 | Open Events | Critical State | All | Showing all 22 open critical event(s)

	Monitor Group - Counter Name	First Seen	Status
●	Index Usage - Percentage of full table scans	17:34 (1 minute ago)	☒ Close
●	Connection History - Percentage of refused connections	17:34 (1 minute ago)	☒ Close
●	Replication - Slave read only?	17:29 (6 minutes ago)	☒ Close
●	Replication - Slave running?	17:29 (6 minutes ago)	☒ Close
●	Excessive Privileges - Number Of Users Having Global LOCK_TABLES Privilege	17:29 (6 minutes ago)	☒ Close
●	Excessive Privileges - Number Of Users Having Global DELETE Privilege	17:29 (6 minutes ago)	☒ Close
●	Excessive Privileges - Number Of Users Having Global UPDATE Privilege	17:29 (6 minutes ago)	☒ Close
●	Excessive Privileges - Number Of Users Having Global INSERT Privilege	17:29 (6 minutes ago)	☒ Close
●	Excessive Privileges - Number Of Users Having Global ALTER Privilege	17:29 (6 minutes ago)	☒ Close
●	Excessive Privileges - Number Of Users Having Global DROP Privilege	17:29 (6 minutes ago)	☒ Close
●	Excessive Privileges - Number Of Users Having Global SHUT_DOWN Privilege	17:29 (6 minutes ago)	☒ Close
●	Excessive Privileges - Number Of Users Having Global GRANT Privilege	17:29 (6 minutes ago)	☒ Close
●	Excessive Privileges - Number Of Users Having Global FILE Privilege	17:29 (6 minutes ago)	☒ Close

An alert (WARNING or CRITICAL) can be closed (and re-opened) for a specific server from both the EVENTS overview page as well as the Monitors page.

Closing an event is temporarily disabling an EVENT meaning, disabling alerting with the yellow/red on the monitors page or sending notifications until it becomes stable, and then goes to the CRITICAL/WARNING (yellow/red) state.

EVENTS can be closed from both the Monitors page (by clicking on the red/yellow alert) and the EVENTS page. Opening the closed EVENTS can be done from the EVENTS page. Closing ALL EVENTS for a certain server from the Monitors page is possible.

⚠ Opening/Closing EVENTS requires user-level privileges.

4.2.4 Disk Info

The Disk Info is an easy to use disk space usage analyzer that allows you to see which MySQL server takes up the most space. It displays number of databases with their data and index sizes. It gives you the snapshot of the disk space occupied by MySQL database objects on your servers as well as a graphical chart display to quickly spot the largest databases.

The Disk information is available at two information levels:

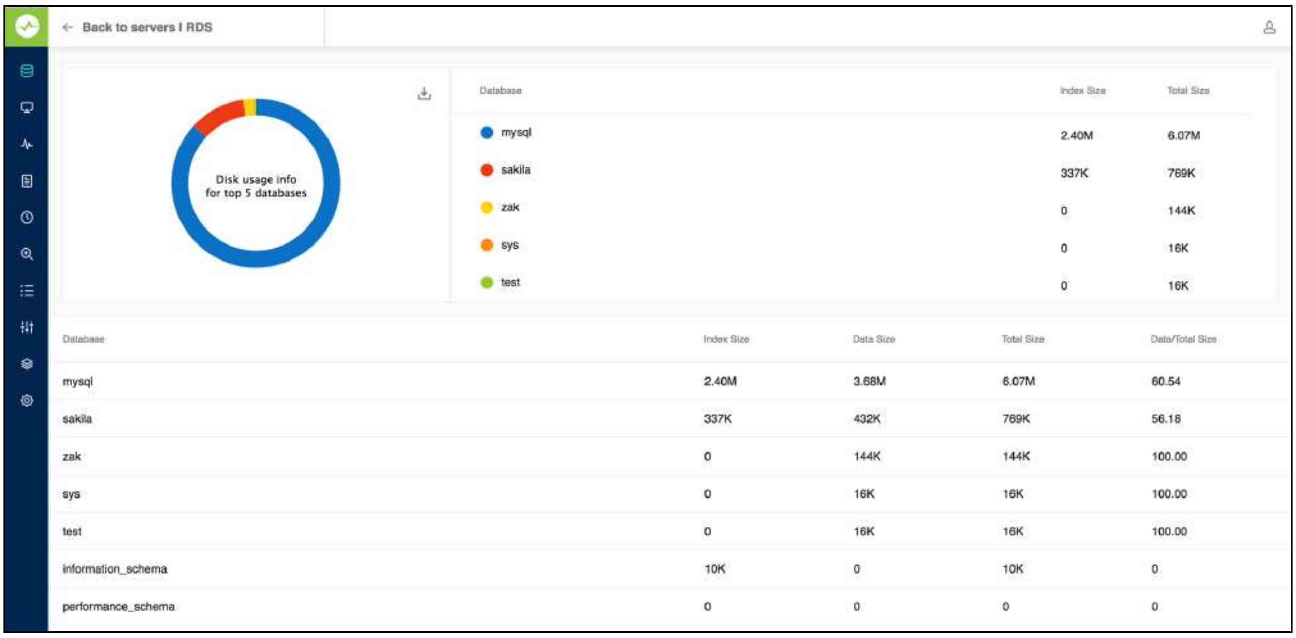
- Database level(see page 165)
- Table level(see page 165)

In Database and Table information levels the data are illustrated by a graphical chart (a 'Doughnut' chart type) for an easy overview. If there are few objects only (databases on server or tables in database) they display in the graph.

4.2.4.1 Overview

Database Level

You reach at this level by clicking the Disk Info option provided for each server in the Servers page. In this view, you get an overview of the space occupied by individual MySQL databases on the server you selected.



Database	Index Size	Total Size
● mysql	2.40M	6.07M
● sakila	337K	769K
● zak	0	144K
● sys	0	16K
● test	0	16K

Database	Index Size	Data Size	Total Size	Data/Total Size
mysql	2.40M	3.68M	6.07M	60.54
sakila	337K	432K	769K	56.18
zak	0	144K	144K	100.00
sys	0	16K	16K	100.00
test	0	16K	16K	100.00
information_schema	10K	0	10K	0
performance_schema	0	0	0	0

Table Level

You reach at this level by clicking one of the databases in the Database Level view. In this screen, you get an overview of the space occupied by the individual tables in the MySQL database you selected.

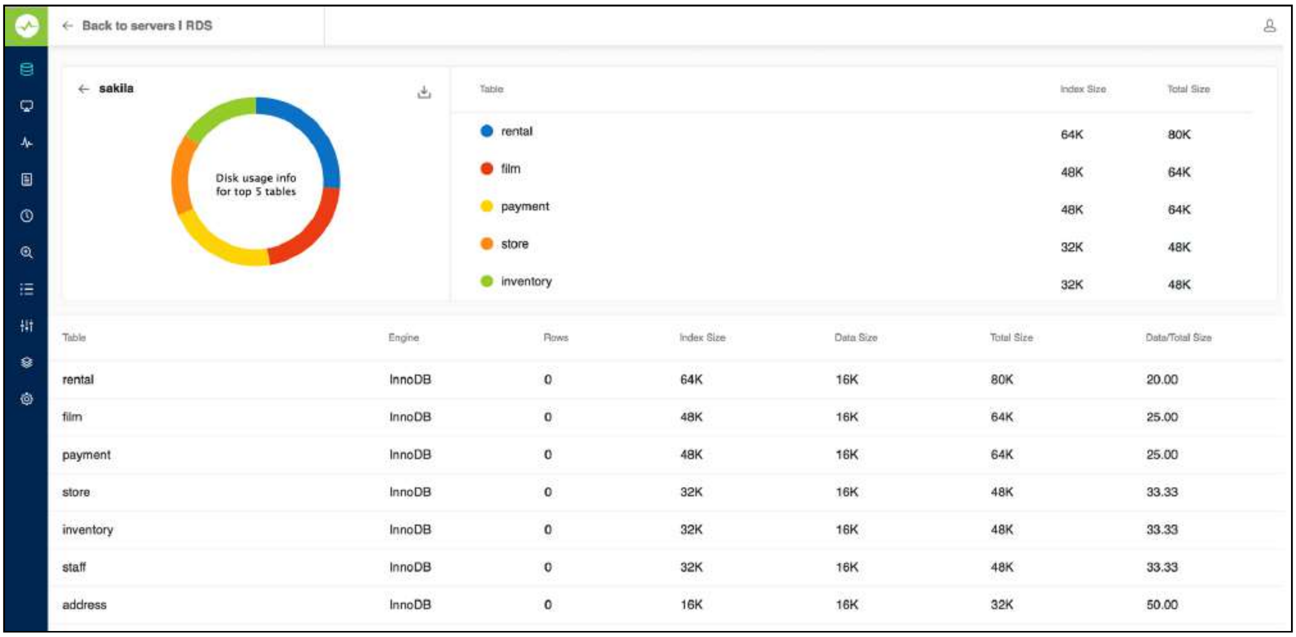


Table	Index Size	Total Size
● rental	64K	80K
● film	48K	64K
● payment	48K	64K
● store	32K	48K
● inventory	32K	48K

Table	Engine	Rows	Index Size	Data Size	Total Size	Data/Total Size
rental	InnoDB	0	64K	16K	80K	20.00
film	InnoDB	0	48K	16K	64K	25.00
payment	InnoDB	0	48K	16K	64K	25.00
store	InnoDB	0	32K	16K	48K	33.33
inventory	InnoDB	0	32K	16K	48K	33.33
staff	InnoDB	0	32K	16K	48K	33.33
address	InnoDB	0	16K	16K	32K	50.00

Also, you get a detailed view of the disk space occupied by data and indexes, along with values indicating how efficiently disk space for every object is utilized. Note that for specific table ENGINES such information is not always available and N/A displays.

For table ENGINES having a shared tablespace for all databases like INNODB (if not the file_per_table option is set in configuration) the Connection Level returns the size of the tablespace itself not the cumulative sum of the databases as shown in the Database Level. This may be rather misleading as the tablespace does not shrink when objects are dropped. Same may apply to third-party engines (like SolidDB) also having a shared tablespace for all databases. Also, with third-party ENGINES and ENGINES in beta, other issues may occur that can affect the accuracy and usability of the information. Finally, navigate back to the Database level from the Table level by clicking the database name adjacent to the chart. You can navigate back to the servers page by clicking the **Back to servers** link.

✓ Disk Info charts displayed can now be exported as JPEG/PNG/PDF formats.

4.3 Dashboard

4.3.1 Real time Charts for Monitoring All MySQL Servers

One of the biggest challenges the MySQL DBA faces is managing an ever-growing number of MySQL servers and databases. Regardless of the size of the MySQL environment, each server requires specific attention when it comes to basic administration, security, performance monitoring, and availability. To give the MySQL DBA a proactive advantage in all of these areas, SQL DM for MySQL provides the Dashboard.

The Dashboard gives the flexibilty to display only a particular set of charts and create as many dashboards as you want. Using the Dashboards, DBAs can create their own set of charts and monitor MySQL and OS specific metrics for single or groups of servers. The Charts are designed so DBAs can easily understand the complete security, availability, and performance picture of all their MySQL servers in one place, all from a slick AJAX interface.

4.3.2 Default Dashboards

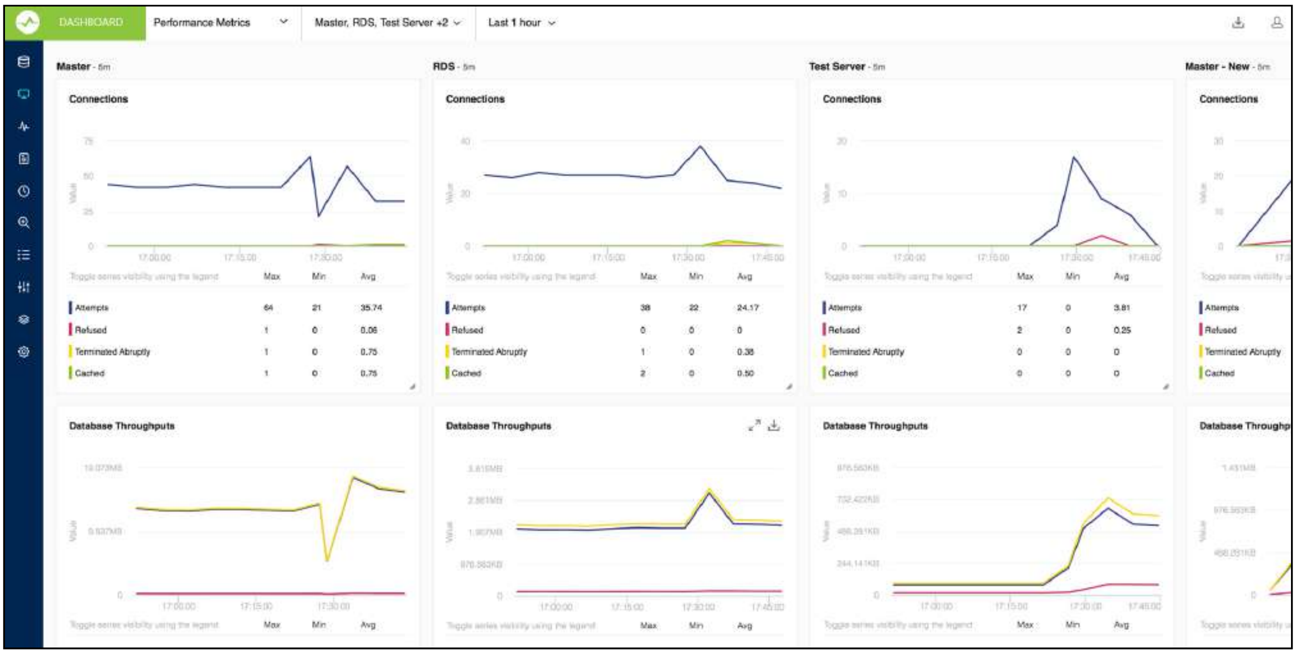
The Dashboard is a page displaying a graphical view of the server parameters and metrics that in most situations will give you a quick overview of the server load and performance. SQL DM for MySQL ships with a default dashboard called "Performance metrics" which includes the metrics:

4.3.2.1 MySQL metrics:

- Connections
- Cache misses
- Statements
- Database Throughputs

4.3.2.2 System metrics:

- Disk IO
- CPU usage



Master – Connections

	Max	Min	Avg
Attempts	64	21	35.74
Refused	1	0	0.06
Terminated Abruptly	1	0	0.75
Cached	1	0	0.75

RDS – Connections

	Max	Min	Avg
Attempts	38	22	24.17
Refused	0	0	0
Terminated Abruptly	1	0	0.38
Cached	2	0	0.50

Test Server – Connections

	Max	Min	Avg
Attempts	17	0	3.81
Refused	2	0	0.25
Terminated Abruptly	0	0	0
Cached	0	0	0

4.3.2.3 How is the refresh interval defined?

Every graph displays related metrics using different colors. In the Charts interface the X-axis timestamps are not printed, and the time interval between each reflects the server-specific setting for the sample interval. By default, the chart displays the last one-hour timeframe. If no data is available for a period because MONyog(SQL DM for MySQL) service was stopped, two empty sample points display in the grid no matter how long time has passed.

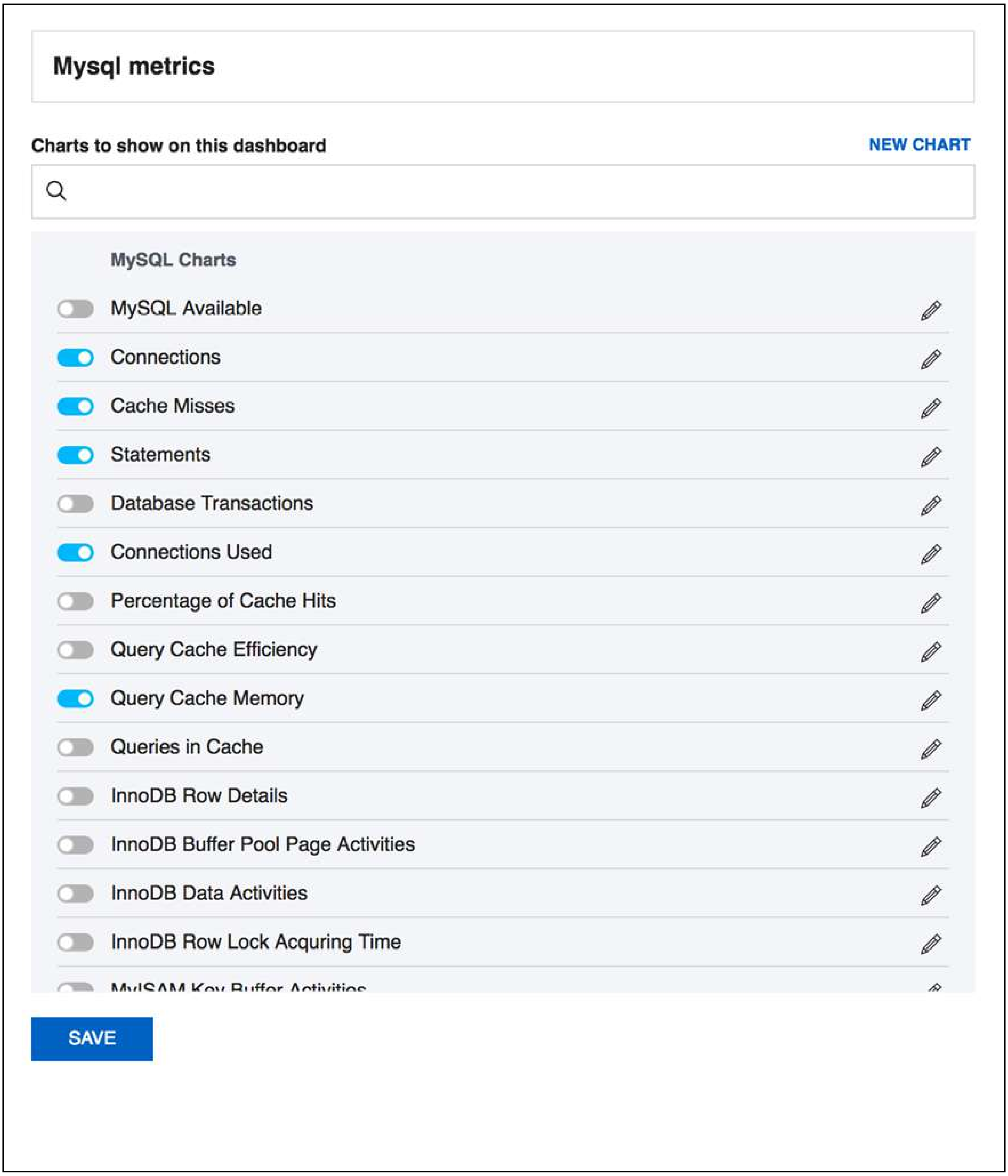
4.3.2.4 Adding Dashboards

Just click the drop down menu containing the Dashboard names, and select the **Add New Dashboard** option. This opens the page where you can give a suitable dashboard name and decide which charts to enable for your dashboard.

4.3.2.5 Dashboards settings

Dahboard settings can be changed from the dashboard page itself. These are local (browser specific) settings that are stored in a cookie.

- The size of the Chart is configurable, you can stretch/reduce the charts size.
- You can hover on the anchor points on the charts to see the actual values.
- You can customize the look of charts by changing the chart color under **Settings -> General -> CHART COLOR**
- You can rename the dashboard name, add/remove charts to the dashboard, and delete the dashboard from the dashboard name drop-down menu.



Mysql metrics

Charts to show on this dashboard

NEW CHART

MySQL Charts

- ○ MySQL Available
- ● Connections
- ● Cache Misses
- ● Statements
- ○ Database Transactions
- ● Connections Used
- ○ Percentage of Cache Hits
- ○ Query Cache Efficiency
- ● Query Cache Memory
- ○ Queries in Cache
- ○ InnoDB Row Details
- ○ InnoDB Buffer Pool Page Activities
- ○ InnoDB Data Activities
- ○ InnoDB Row Lock Acquring Time
- ○ MyISAM Key Buffer Activities

SAVE

4.3.2.6 Charts recommendation

Recommended number of collections is 50 for better visibility.

⚠ This recommendation is based on 1280 x 1024 screen resolution. So, depending upon your screen resolution the number of collections may vary to some extent.

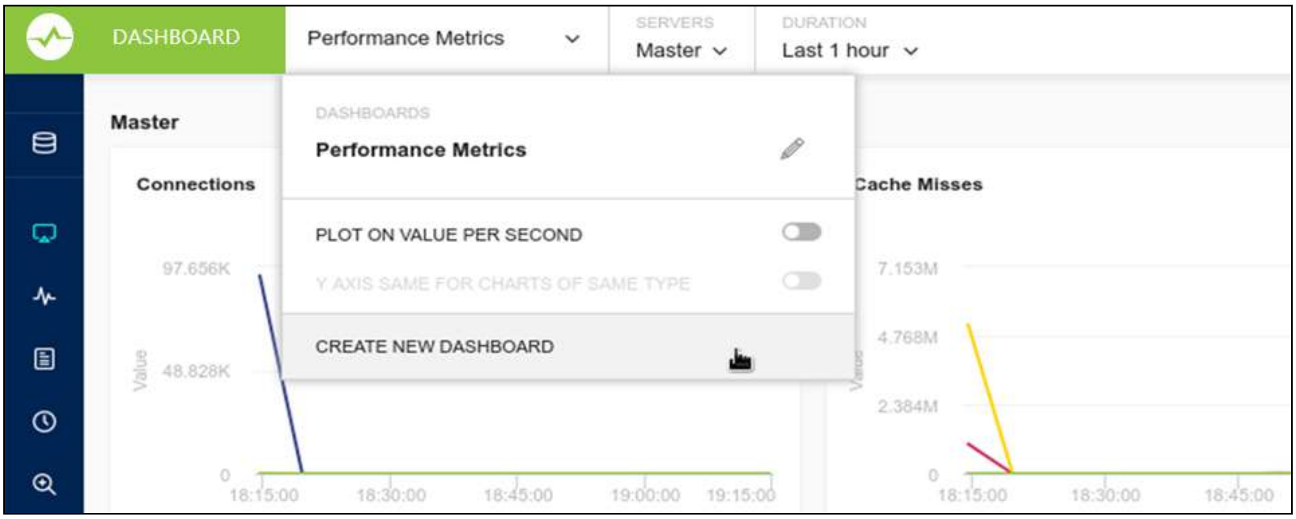
Also, charts can be exported as PDF/JPG/PNG formats, for more information review Exporting graphs[55].

4.3.3 Customizing Dashboard

SQL DM for MySQL provides a series of charts that address general monitoring needs for MySQL systems. It also ships with a general purpose dashboard for monitoring database performance. Also, you can create your own custom dashboards and charts.

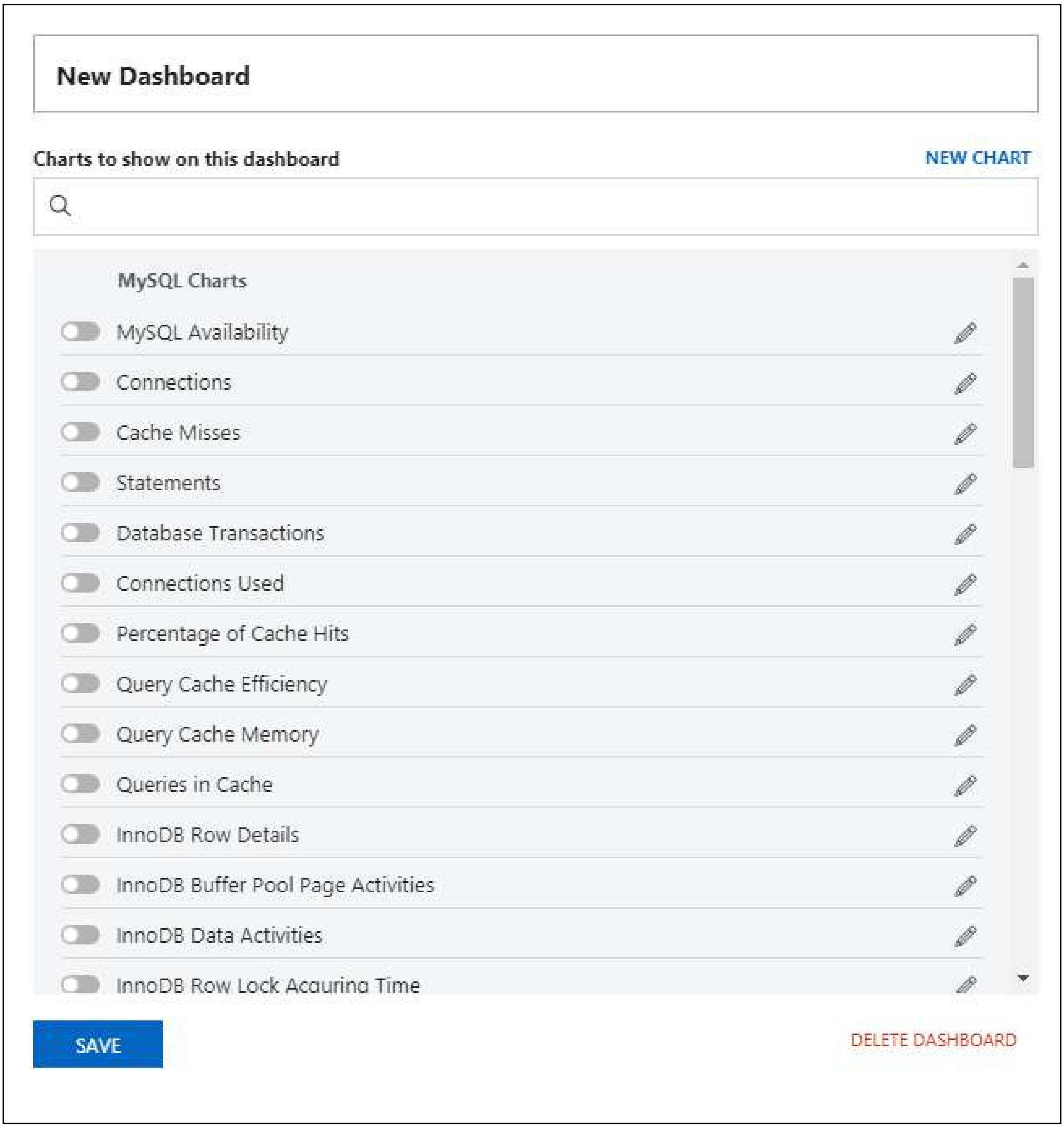
4.3.3.1 Custom Dashboards

To create a custom dashboard, open the dashboard selection context menu. Then click **Create New Dashboard**. It opens an overlay to configure the dashboard, select the charts you want to display. In the event that you would like to modify an existing dashboard, click the edit icon beside it in the same menu.



From the overlay, enter a logical name for the dashboard in the text field, then select the charts that you want to be added to the dashboard. In the event that you need a chart for something that is not available, you can create a new chart. Click **Save** to make the new dashboard available in SQL DM for MySQL.

[55] http://wiki.idera.com/x/hwEvvgE



New Dashboard

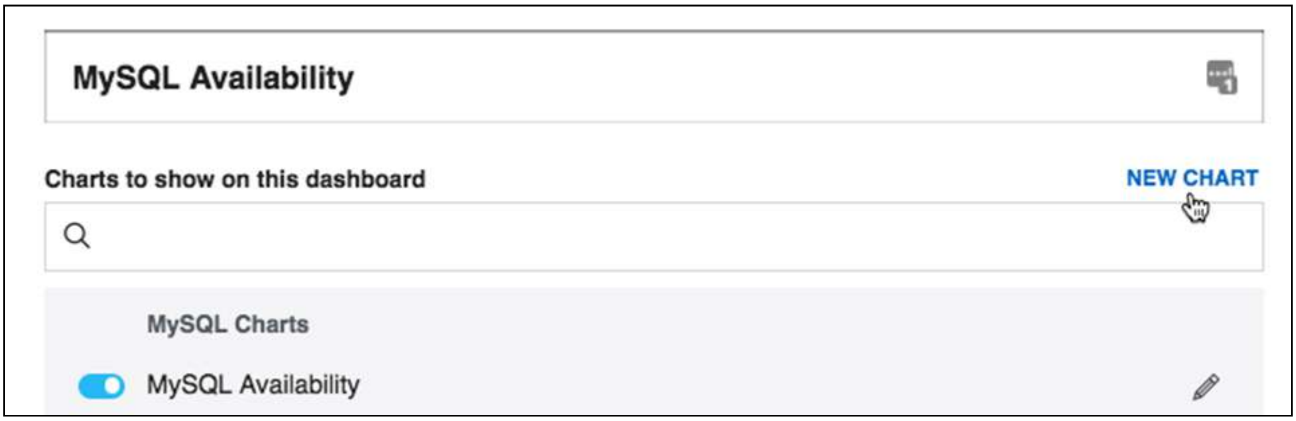
Charts to show on this dashboard

MySQL Charts

- MySQL Availability
- Connections
- Cache Misses
- Statements
- Database Transactions
- Connections Used
- Percentage of Cache Hits
- Query Cache Efficiency
- Query Cache Memory
- Queries in Cache
- InnoDB Row Details
- InnoDB Buffer Pool Page Activities
- InnoDB Data Activities
- InnoDB Row Lock Acquring Time

4.3.3.2 Custom Charts

In addition to the many charts that SQL DM for MySQL provides by default, you can create custom charts to suit your own particular needs. To create a chart, create a new dashboard or edit an existing one, click the **New Chart** link in the overlay.



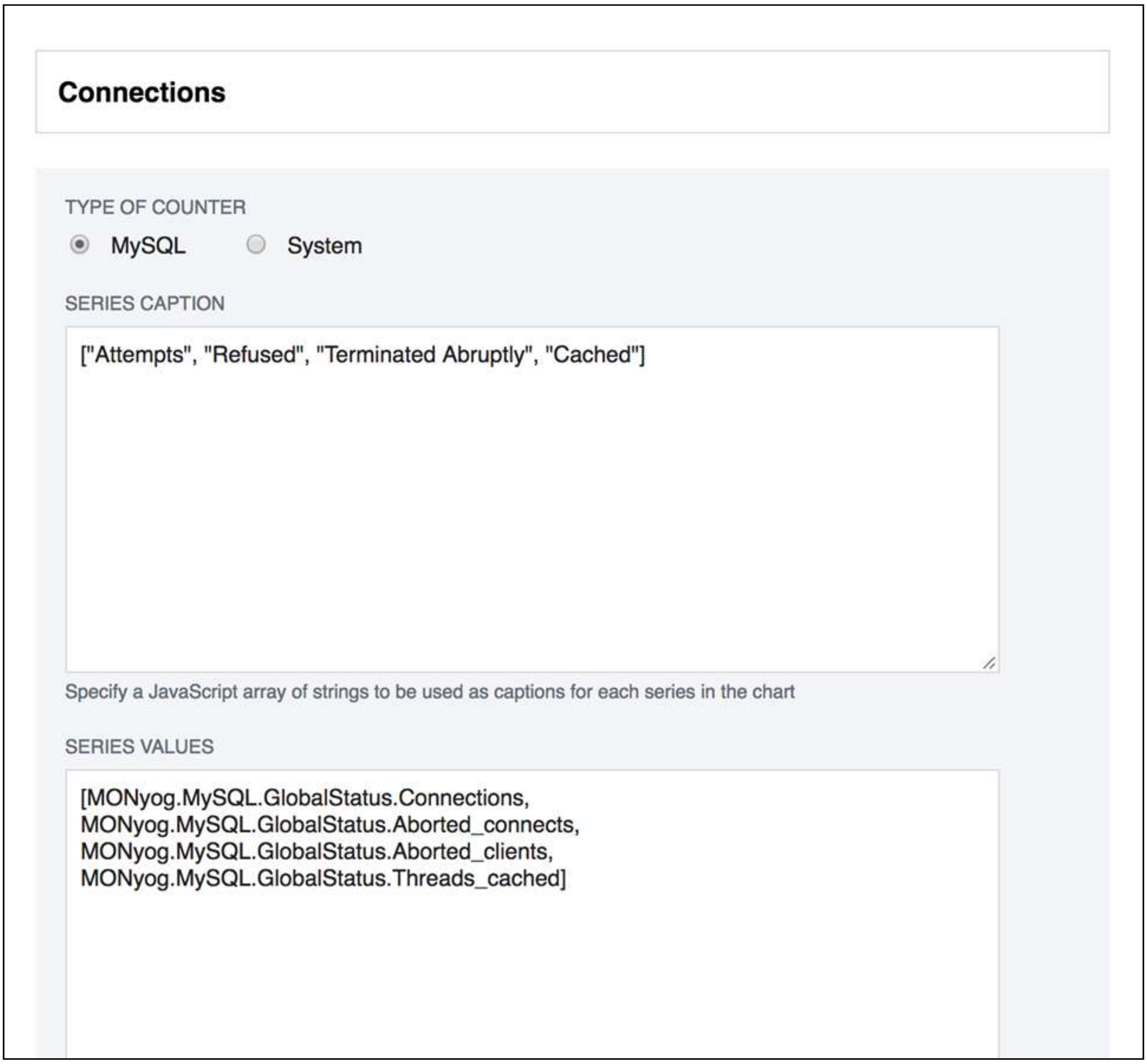
MySQL Availability

Charts to show on this dashboard

MySQL Charts

MySQL Availability

When clicking **New Chart**, the second overlay opens to configure the new chart.



Connections

TYPE OF COUNTER

● MySQL ○ System

SERIES CAPTION

```
["Attempts", "Refused", "Terminated Abruptly", "Cached"]
```

Specify a JavaScript array of strings to be used as captions for each series in the chart

SERIES VALUES

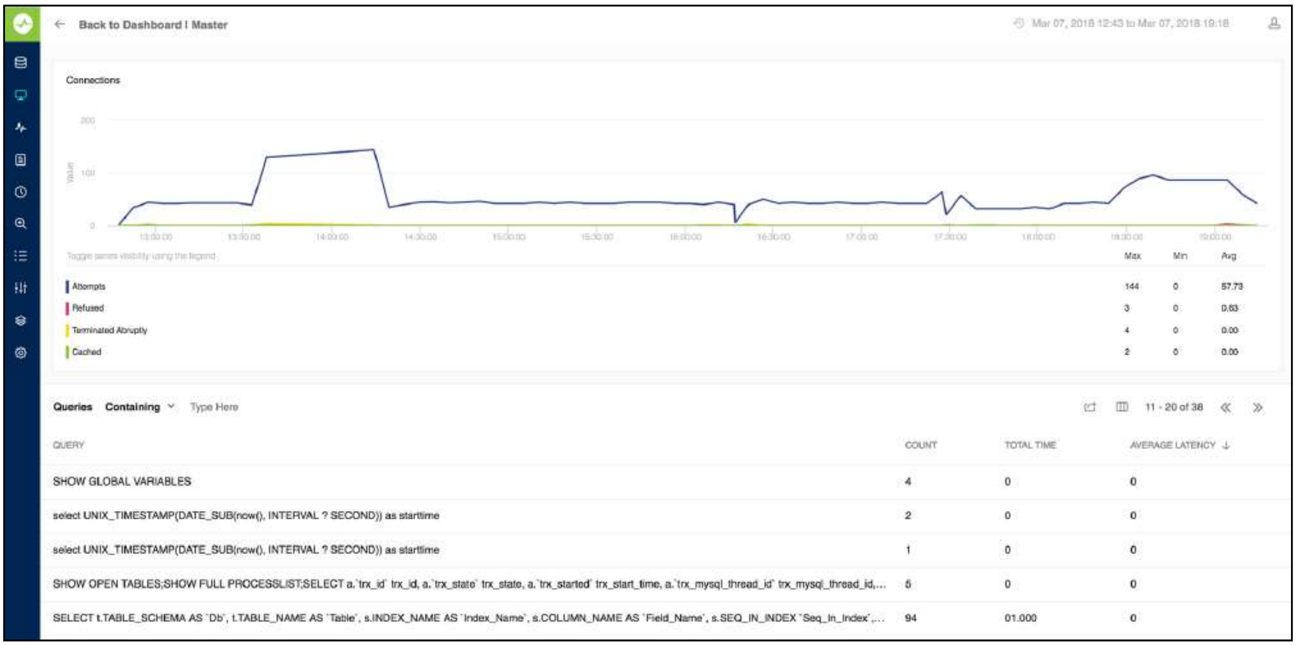
```
[MONyog.MySQL.GlobalStatus.Connections,
MONyog.MySQL.GlobalStatus.Aborted_connects,
MONyog.MySQL.GlobalStatus.Aborted_clients,
MONyog.MySQL.GlobalStatus.Threads_cached]
```

Charts are of the MySQL or System type. MySQL charts get their data from the MySQL client interface. System charts get their data from the operating system, (through SSH, for instance). For each chart you can define a Series Caption, indicating the usual one to four values shown on the chart and series values to specify how to generate those values. Both are written as JavaScript expressions.

Once your new chart is ready, click **Save**. SQL DM for MySQL adds the chart to those available on the configuration overlay, where you can enable it for any dashboard.

4.3.4 Query Details

You can zoom into spikes in the charts and get to see the queries involved in the charts, global status, and global system variables for that time range. Just hover over the chart, and click the **Show details** icon to open the page with the above mentioned information.



	Max	Min	Avg
Attempts	144	0	57.73
Refused	3	0	0.63
Terminated Abruptly	4	0	0.00
Cached	2	0	0.00

QUERY	COUNT	TOTAL TIME	AVERAGE LATENCY ↓
SHOW GLOBAL VARIABLES	4	0	0
select UNIX_TIMESTAMP(DATE_SUB(now(), INTERVAL ? SECOND)) as starttime	2	0	0
select UNIX_TIMESTAMP(DATE_SUB(now(), INTERVAL ? SECOND)) as starttime	1	0	0
SHOW OPEN TABLES;SHOW FULL PROCESSLIST;SELECT a.`trx_id` trx_id, a.`trx_state` trx_state, a.`trx_started` trx_start_time, a.`trx_mysql_thread_id` trx_mysql_thread_id,...	5	0	0
SELECT t.TABLE_SCHEMA AS `Db`, t.TABLE_NAME AS `Table`, s.INDEX_NAME AS `Index_Name`, s.COLUMN_NAME AS `Field_Name`, s.SEQ_IN_INDEX `Seq_In_Index`,...	94	01.000	0

If sniffer was running during this interval, aggregated sniffer information is displayed for the time interval. Also, you can see first and last value of (optionally) all or changed variables aggregated values. The graph is zoomable by selecting a sub-interval with the mouse.

When user clicks a row in the query list, a pop-up opens with information about thread-id, user, and host along with full query.

The list of queries will not be rendered if there are more than 2000 queries in the time period. There is a user control that can be activated in such case. The reason is that every 1000 queries take around 1 seconds to render on an average desktop system. And, it displays the list of queries before zooming (provided still that there not more than 2000 queries to display).

4.3.5 Adding New Charts

Click **New Chart** option on the Create New Dashboard page.

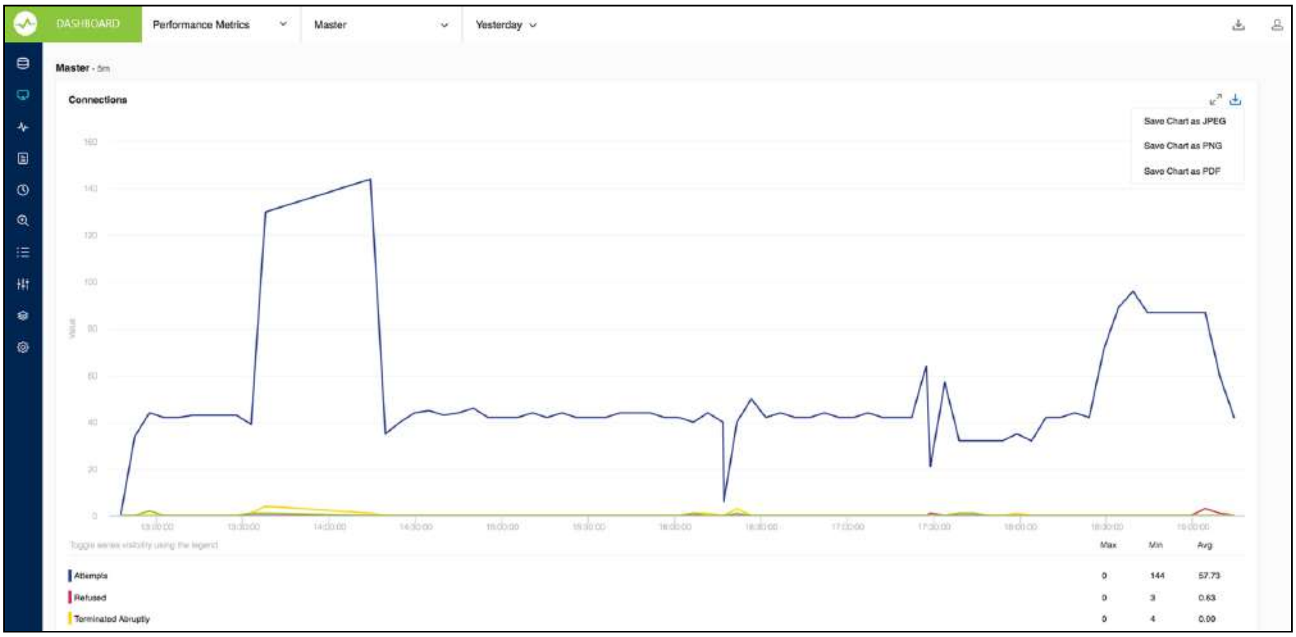
Select the desired type of counter - MySQL or System.

- **Caption:** Enter an appropriate name for your chart.

- **SeriesCaption:** Specify the names of all the graphs that are going to be plotted. Example: ["Attempts", "Refused", "Terminated abruptly"] Here 'Attempts', 'Refused' and 'Terminated abruptly' are the names of the graphs that are going to be plotted on this new chart being created.
- **SeriesValues:** Specify a JavaScript array of strings to be used as captions for each series in the chart. Please refer Monyog object model for more details.
- **This chart shows boolean values:** Enable this option if the chart displays only on and off status.
- **Unit:** Specify a JavaScript array of strings to be used as y-axis unit in the chart. Ex: ['','KB','MB','GB','TB']
- **Unit Factor:** Define the limit on reaching which the unit should be incremented. Ex: if this value is 1024 then, 1024 = 1KB, 1024KB=1MB and so on.
- **ChartValue:** Controls how values in the series are plotted. Specifying "Delta" causes the difference between values from the last two data collections to be plotted. Specifying "Current" causes the current value to be plotted as it is.
- **Uptime:** Selecting "Yes" indicates that this Charts chart will plot cumulative values which increase with time.

4.3.6 Exporting Graphs

All charts displayed by SQL DM for MySQL in Dashboard can be exported as PDF/JPG/PNG formats. To export a chart select the option from the drop-down context menu.



	Max	Min	Avg
Attempts	0	144	57.73
Refused	0	3	0.63
Terminated Abruptly	0	4	0.00

4.4 Monitors

Server monitoring is one of the base components of server management. The critical nature of servers require someone or something to constantly monitor the status of the managed servers. SQL DM for MySQL easily automates this tedious job and detects server problems as soon as they occur.

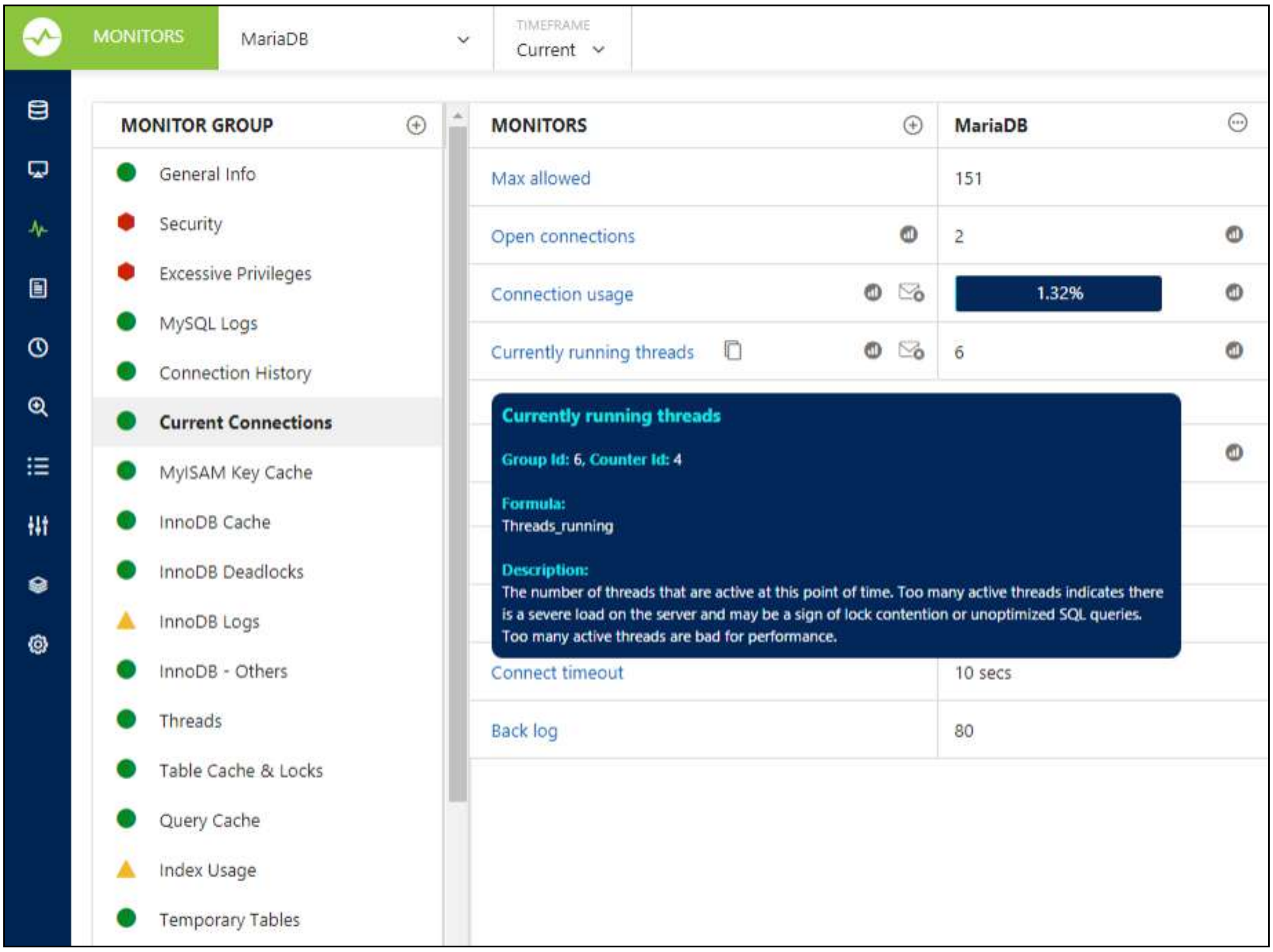
'Monitors' interface lets you access any server metric you can imagine.

The Monitors page shows a detailed display of server parameters and metrics. The left column displays the metrics that SQL DM for MySQL displays per server. For every server that you registered with SQL DM for MySQL, a column, with data pertinent to that server, is displayed. Some data is displayed as cumulative values (accumulated over a time frame that you can specify) while others are averaged over a time interval (typically per second).

4.4.1 Layout_View of Monitors

To see how every metric is calculated by SQL DM for MySQL you need to position the mouse over the name of each metric. The name of the monitor, formula used by the calculation displays with an explanation as a 'tool tip popup'. The tool-tip also explains how the result shown is evaluated, and how the value relates to an overall or some specific server performance metric.

✓ Every metric is documented in the SQL DM for MySQL web interface itself.



MONITORS	MariaDB
Max allowed	151
Open connections	2
Connection usage	1.32%
Currently running threads	6
Connect timeout	10 secs
Back log	80

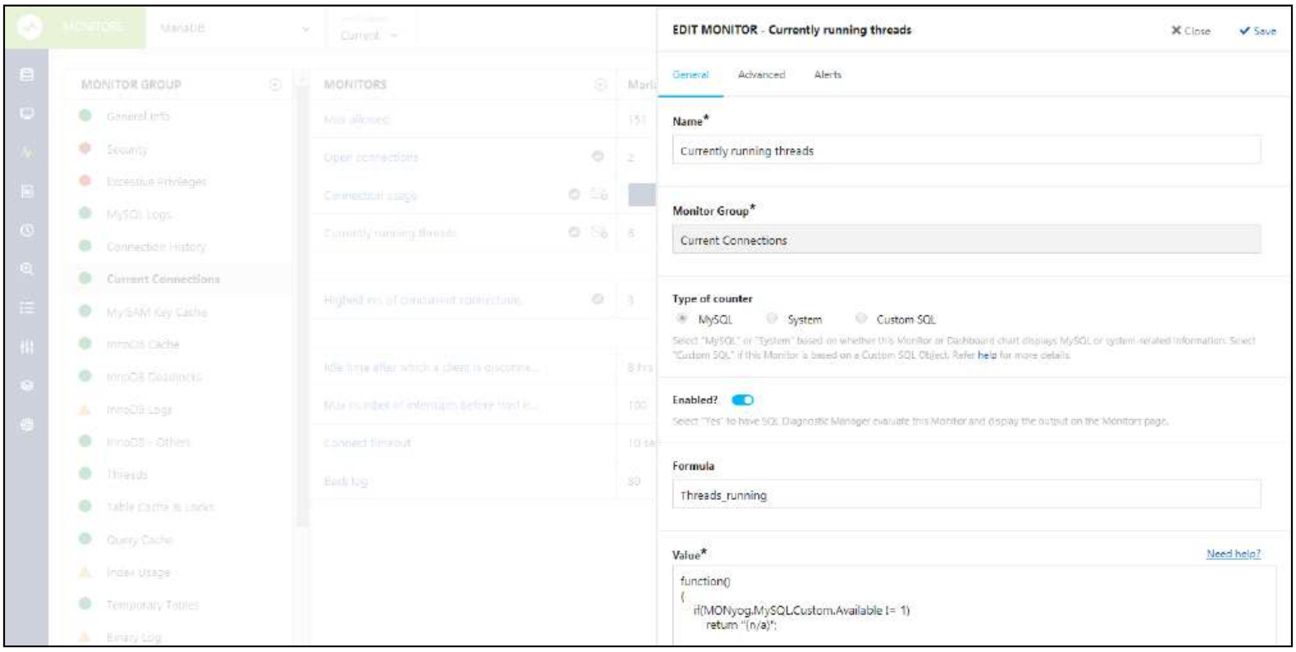
Currently running threads

Group Id: 6, **Counter Id:** 4

Formula:
Threads_running

Description:
The number of threads that are active at this point of time. Too many active threads indicates there is a severe load on the server and may be a sign of lock contention or unoptimized SQL queries. Too many active threads are bad for performance.

If you require more details on the metric, click the hyperlink to get a detailed view.



EDIT MONITOR - Currently running threads

Name*: Currently running threads

Monitor Group*: Current Connections

Type of counter: ● MySQL ○ System ○ Custom SQL

Select "MySQL" or "System" based on whether this Monitor or Dashboard chart displays MySQL or system-related information. Select "Custom SQL" if this Monitor is based on a Custom SQL Object. Refer help for more details.

Enabled? ●

Select "Yes" to have SQL Diagnostic Manager evaluate this Monitor and display the output on the Monitors page.

Formula: Threads_running

Value*:

```
function()
{
  if(MONyog.MySQL.Custom.Available != 1)
    return "(n/a)";
```

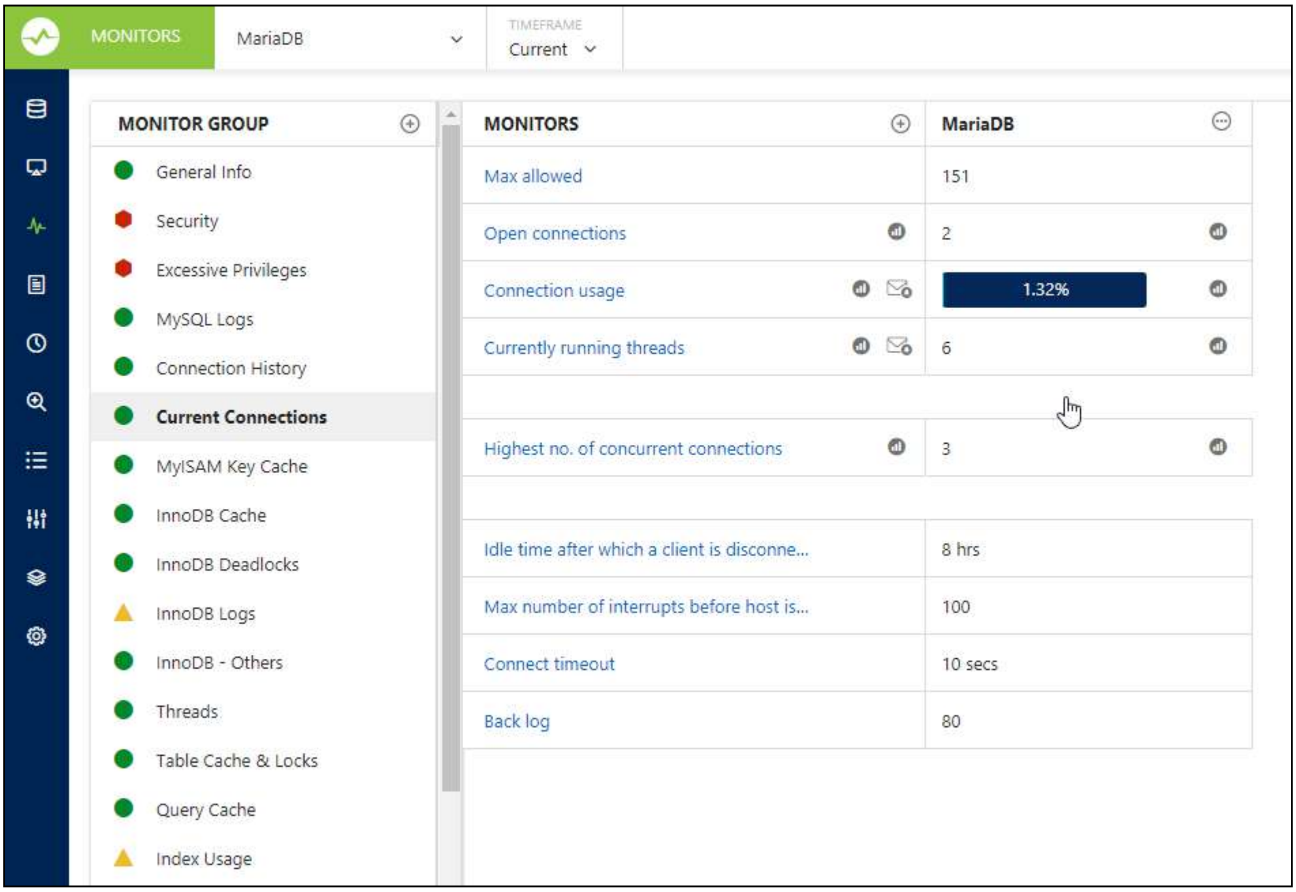
The monitors can be edited or duplicated by clicking the icons adjacent to the name of the monitor. The mail alert icon is displayed adjacent to the name of the monitor specifying an alert condition. The blue coloured mail alert icon indicates that the notification has been enabled with the alert condition specified.

4.4.2 Mail and Graph icon

4.4.2.1 Mail icon

The Mail icon ✉ indicates that the corresponding counter will send you an alert when:

- You selected the option to send alerts for the specific server.
- The value for either the 'current' or the latest time frame exceeds threshold level.



MONITOR GROUP

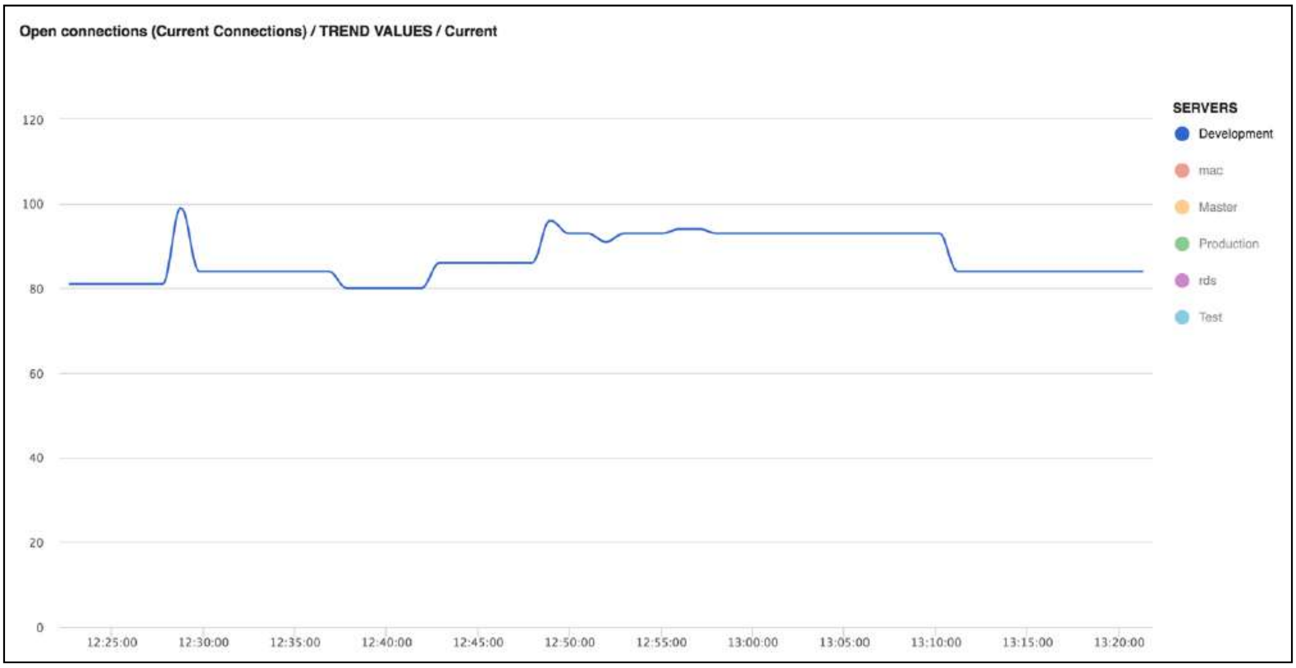
- ● General Info
- ⬢ Security
- ⬢ Excessive Privileges
- ● MySQL Logs
- ● Connection History
- ● **Current Connections**
- ● MyISAM Key Cache
- ● InnoDB Cache
- ● InnoDB Deadlocks
- ▲ InnoDB Logs
- ● InnoDB - Others
- ● Threads
- ● Table Cache & Locks
- ● Query Cache
- ▲ Index Usage

MONITORS	MariaDB
Max allowed	151
Open connections	2
Connection usage	1.32%
Currently running threads	6
Highest no. of concurrent connections	3
Idle time after which a client is disconne...	8 hrs
Max number of interrupts before host is...	100
Connect timeout	10 secs
Back log	80

4.4.2.2 Graph icon

A graph or chart is a type of information graphic or graphic organizer that represents tabular numeric data and/or functions. Graphs are often used to make it easier to understand large quantities of data and the relationship between different parts of the data. Graphs can usually be read more quickly than the raw data that they come from.

When clicking the graph icon a graph displays, depicting the current status of some metric of the server, which is updated in real-time.



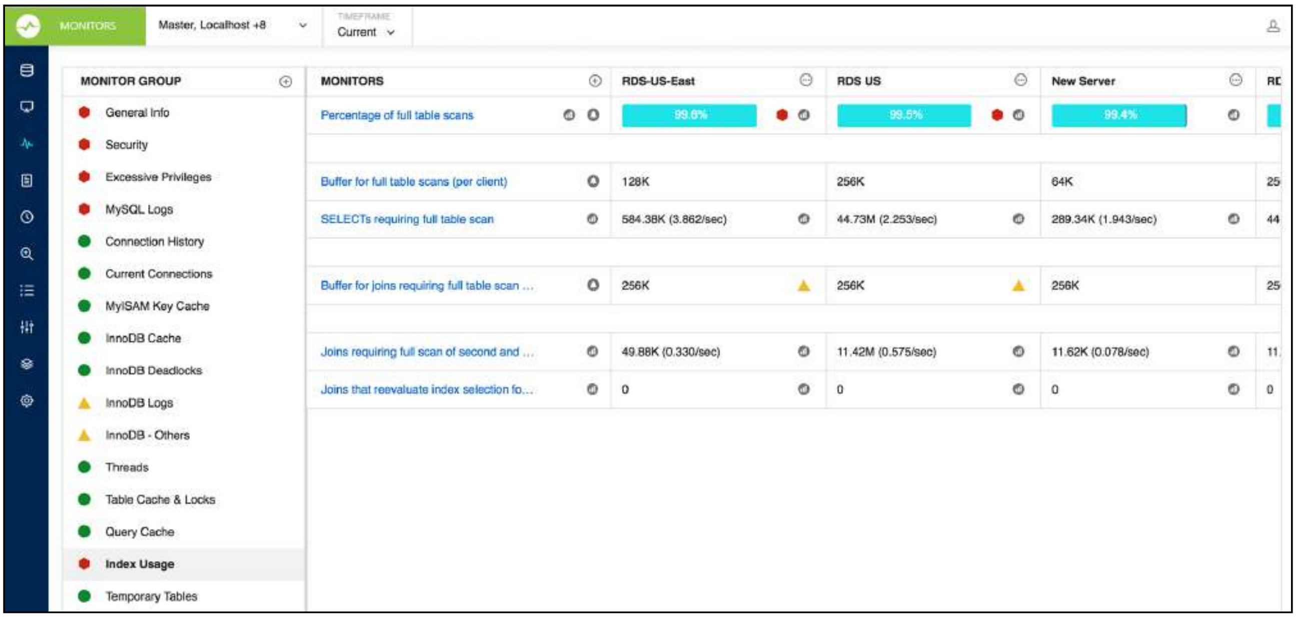
Charts and graphs displayed by the Monitors can now be exported in PDF/JPG/PNG formats. To export a chart select the option from the drop-down context menu.

4.4.3 Critical and Warning Alerts

In case a red hexagon or yellow triangle appears beside a metric, it means that the current value of the metric exceeds the set threshold value, ie., the symbols signify alert conditions. The definition of what should trigger a red hexagon or yellow triangle alert is defined in the JavaScript component defining every metric. The default values that SQL DM for MySQL ships with have been configured for average servers used for mixed purposes: a mix of websites and databases with corporate data, for example.

You may want to change the values depending on the type of databases. Threshold values for the metrics (and also actual, realizable, values) varies depending, at the very least, on the type of applications that use the database. For instance with certain web applications like Forums a very high 'cache hit rate' typically can be reached (as a large percentage of queries return the most popular post in the Forums) whereas in a production planning system it is typically not possible to achieve a very high hit rate (as identical queries and results only rarely occur here).

✓ Red and Yellow alerts will quickly tell what could be issues to investigate.



MONITOR GROUP
🟥 General Info
🟥 Security
🟥 Excessive Privileges
🟥 MySQL Logs
● Connection History
● Current Connections
● MyISAM Key Cache
● InnoDB Cache
● InnoDB Deadlocks
▲ InnoDB Logs
▲ InnoDB - Others
● Threads
● Table Cache & Locks
● Query Cache
🟥 **Index Usage**
● Temporary Tables

MONITORS	RDS-US-East	RDS US	New Server	RD
Percentage of full table scans	99.6% 🟥	99.5% 🟥	99.4%	
Buffer for full table scans (per client)	128K	256K	64K	25
SELECTs requiring full table scan	584.38K (3.862/sec)	44.73M (2.253/sec)	289.34K (1.943/sec)	44
Buffer for joins requiring full table scan ...	256K ▲	256K ▲	256K	25
Joins requiring full scan of second and ...	49.88K (0.330/sec)	11.42M (0.575/sec)	11.62K (0.078/sec)	11.
Joins that reevaluate index selection fo...	0	0	0	0

4.4.4 Difference between All time_Current, Delta

Define yourself the timeframe of data in the database that shall be used for generating the display.

The Monitors page can be defined to operate in 2 'dynamic modes' and 1 'cumulative mode'. The dynamic modes auto-update with the most recent data without user interaction. The 'cumulative mode' is History/Trend Analysis where you can define any time interval for the actual display of the data.

- **Current:** Display values based on Current data values collected by SQL DM for MySQL for every specific server. Some server metrics will only be meaningful after the server has been running for a while (ie, after the server has been 'warmed-up').
- **Delta:** Display results based on data for the period between the last data collection and the collection before it. The result gives you a better idea of the current situation, and how much this current situation differs from the average or normal situation.

4.4.5 Difference between Cumulative and Point in time counters

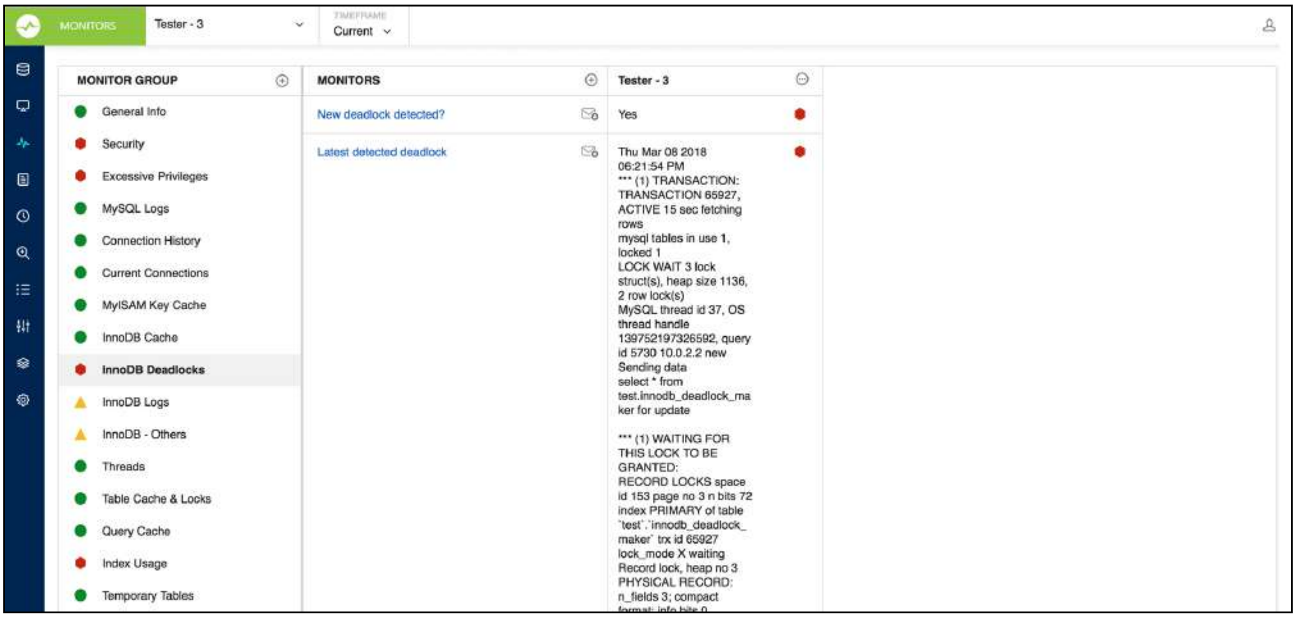
Cumulative Metrics (increase with time and can be recorded for a period (like 'Key_reads', 'Maximum no. of Connections that Ever Was')) are calculated differently instead of point-in-time metrics (like 'Server Availability', 'Currently Running Threads').

- **Cumulative metrics:** SQL DM for MySQL send alert if thresholds are crossed for last collection interval.
- **Point-in-time metrics:** SQL DM for MySQL send alert only based on 'Current' values (for instance current no. of connections is close to Max Allowed Connections - there is nothing more to consider!).

4.4.6 Innodb Deadlock Monitoring

The InnoDB deadlock monitor is a little different than the rest of the monitors and require a little additional explanation. Whereas all other monitors are based on the return of `SHOW GLOBAL STATUS`, `SHOW GLOBAL VARIABLES` and `SHOW SLAVE STATUS`, the 'InnoDB deadlock' monitor is based on output from `SHOW ENGINE INNODB STATUS`. This statement differs from the first-mentioned in the sense that it does not return as single discrete value (like a single number), but a rather long string instead with lots of information. From this long string we extract what is related to deadlocks.

With this monitor, it does not make sense to distinguish between 'Current' and 'Delta' timeframes. These timeframes display the same in SQL DM for MySQL:



New deadlock detected? Yes

Latest detected deadlock

```
Thu Mar 08 2018
06:21:54 PM
*** (1) TRANSACTION:
TRANSACTION 65927,
ACTIVE 15 sec fetching
rows
mysql tables in use 1,
locked 1
LOCK WAIT 3 lock
struct(s), heap size 1136,
2 row lock(s)
MySQL thread id 37, OS
thread handle
139752197326592, query
id 5730 10.0.2.2 new
Sending data
select * from
test.innodb_deadlock_ma
ker for update

*** (1) WAITING FOR
THIS LOCK TO BE
GRANTED:
RECORD LOCKS space
id 153 page no 3 n bits 72
index PRIMARY of table
`test`.`innodb_deadlock_
maker` trx id 65927
lock_mode X waiting
Record lock, heap no 3
PHYSICAL RECORD:
n_fields 3; compact
format; info bits 0
```

- **New Deadlock Detected** indicates if a new deadlock was reported by INNODB STATUS between the last two data retrievals.
- **Latest Deadlock Detected** shows information about the last deadlock detected (if any). To view all deadlock situation that SQL DM for MySQL has information about, use the 'History' timeframe.

There is a limitation with deadlock information returned by `SHOW ENGINE INNODB STATUS` that will also affect SQL DM for MySQL: It returns only information about the last deadlock. So if more than one deadlock has occurred between two data retrievals, SQL DM for MySQL only has information about the latest.

4.4.7 Getting automatic alert notifications

4.4.7.1 Get alerts for what you want and when you want

SQL DM for MySQL can send you alerts over mail, SNMP, Slack,PagerDuty, and Syslog. Stay on top of significant events and avoid many sleepless nights!

Some features of the alerting system are:

- It uses the concept of "Delayed alert notifications". It can now be defined that a problem must have existed for a number of sample intervals continuously (in a row) for an alert to be sent. A global setting for each server is available from GUI. For individual counters the global setting can be overridden by defining the RetryOverride (like 'RetryOverride:3') property of the (JavaScript) counter definition.
- Also, you can choose to be notified when SQL DM for MySQL detects that a problem, which existed previously, has been resolved.

⚠ Alerts for every registered server are sent independently.

4.4.7.2 Configuring alerts.

Please refer to Notification & Maintenance Settings[56] for more information regarding Notification Settings.

4.4.7.3 What alerts contain and when is it sent?

In the Alert Condition field there is a condition defined to decide when the monitor will be in critical or warning state. SQL DM for MySQL will evaluate AlertCondition and if AlertCondition is "Critical" or "Warning" then only it will trigger an Alert. Further, if WarmUpRequired is defined and it is "Yes", then SQL DM for MySQL evaluates the AlertCondition if the server is up for at least three hours (the default warmup time). "Critical" and "Warning" correspond to red or yellow dots in the GUI interface respectively.

The mails are optimized for display in a HTML enabled mail client but also readable in a client supporting text mails only.

The alert contains the following information:

- The MySQL Host.
- There is a detailed rule defining if and when a alert is being sent like All Time/Current, Delta - Last two collections.
- Name of the alerting Monitor.
- The corresponding Monitor Group.
- Type of Alert: Critical, Warning, etc.
- The defined Threshold Limit.
- Value of the Monitor when the alert was generated.
- The Advice text.

This is an example of a typical SQL DM for MySQL mail alert.

Server: LMT	
Sampling timeframe: **All Time/Current**	
Name	Available?
Group	General Info
Type	Critical
Value	No
Advice	If MONyog service is not able to connect to MySQL, it simply means that connection is not possible for one of the following (or similar) reasons: • There is no MySQL server running at the specified host. • Connection to the MySQL server is not all.

[56] http://wiki.idera.com/x/twEvvgE

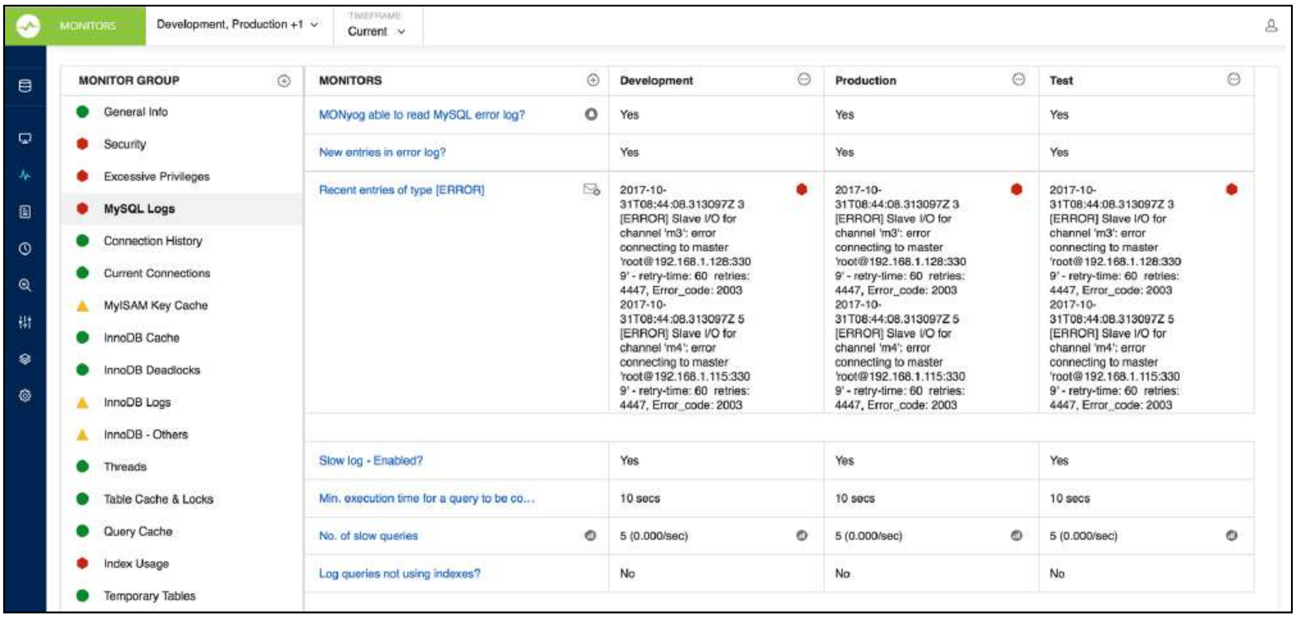
4.4.8 MySQL Error Log

MySQL error log contains information indicating when MySQL was started and stopped and also any critical errors that occur while the server is running. As such, monitoring the error log is crucial – changes to the log are indicative of disastrous outages.

SQL DM for MySQL makes the task of monitoring the error log very simple for you.

SQL DM for MySQL alerts you of changes in the error log, and if there is an entry of type `[ERROR]` in the log, SQL DM for MySQL extracts the corresponding message and send it to you.

> ⚠ If there are more entries of type `[ERROR]` , then SQL DM for MySQL will show only the last 1024 characters of type `[ERROR]` .



MONITORS	Development	Production	Test
MONyog able to read MySQL error log?	Yes	Yes	Yes
New entries in error log?	Yes	Yes	Yes
Recent entries of type [ERROR]	2017-10-31T08:44:08.313097Z 3 [ERROR] Slave I/O for channel 'm3': error connecting to master 'root@192.168.1.128:3309' - retry-time: 60 retries: 4447, Error_code: 2003 2017-10-31T08:44:08.313097Z 5 [ERROR] Slave I/O for channel 'm4': error connecting to master 'root@192.168.1.115:3309' - retry-time: 60 retries: 4447, Error_code: 2003	2017-10-31T08:44:08.313097Z 3 [ERROR] Slave I/O for channel 'm3': error connecting to master 'root@192.168.1.128:3309' - retry-time: 60 retries: 4447, Error_code: 2003 2017-10-31T08:44:08.313097Z 5 [ERROR] Slave I/O for channel 'm4': error connecting to master 'root@192.168.1.115:3309' - retry-time: 60 retries: 4447, Error_code: 2003	2017-10-31T08:44:08.313097Z 3 [ERROR] Slave I/O for channel 'm3': error connecting to master 'root@192.168.1.128:3309' - retry-time: 60 retries: 4447, Error_code: 2003 2017-10-31T08:44:08.313097Z 5 [ERROR] Slave I/O for channel 'm4': error connecting to master 'root@192.168.1.115:3309' - retry-time: 60 retries: 4447, Error_code: 2003
Slow log - Enabled?	Yes	Yes	Yes
Min. execution time for a query to be co...	10 secs	10 secs	10 secs
No. of slow queries	5 (0.000/sec)	5 (0.000/sec)	5 (0.000/sec)
Log queries not using indexes?	No	No	No

For more information, see Advanced Settings.[57]

4.4.9 History_Trend and Graph Analysis

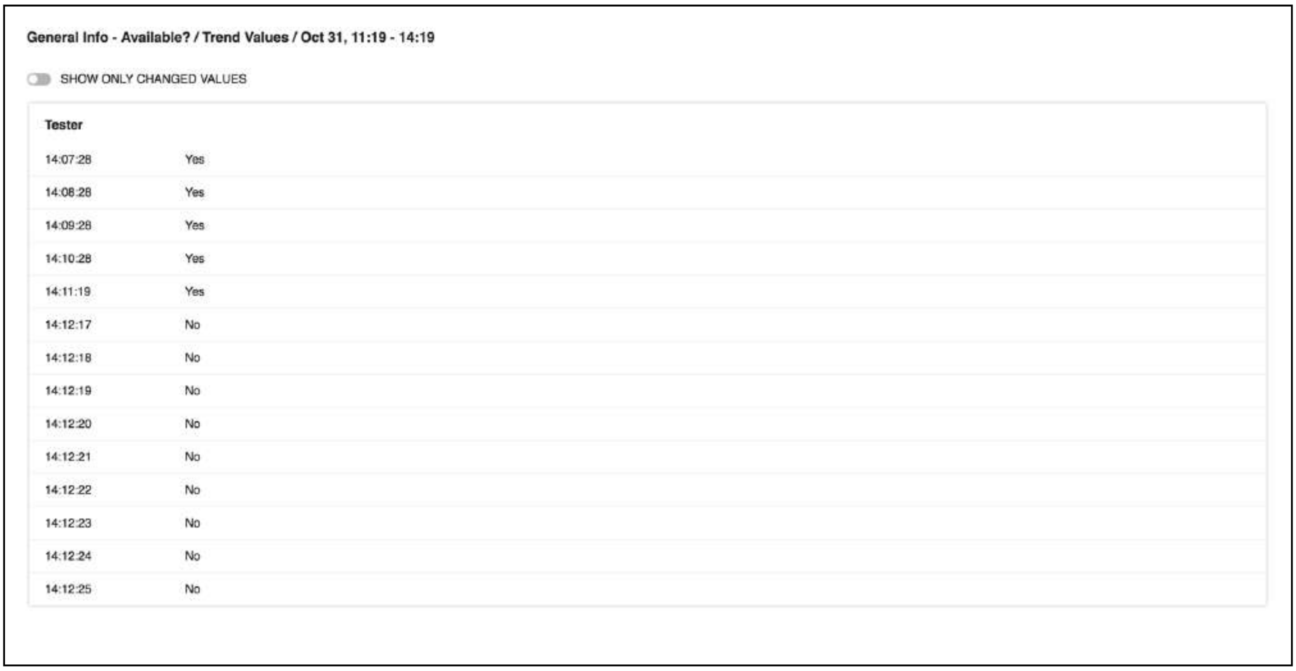
The term Trend Analysis refers to the concept of collecting information and attempting to spot a pattern, or trend, in the information. In some fields of study, the term Trend Analysis has more formally-defined meanings. Today, trend analysis often refers to the science of studying changes in social patterns, including fashion, technology, and the consumer behavior.

Analyzing history reports give you the means of tracking trends and identifying problem areas in your network. You can use this information to find recurring problems, which helps you prevent future problems. This data can also help you plan maintenance schedules and equipment replacement.

SQL DM for MySQL concerns itself with answering the following question:

[57] http://wiki.idera.com/x/egEGBg

- **Was your MySQL down in the last one month?**

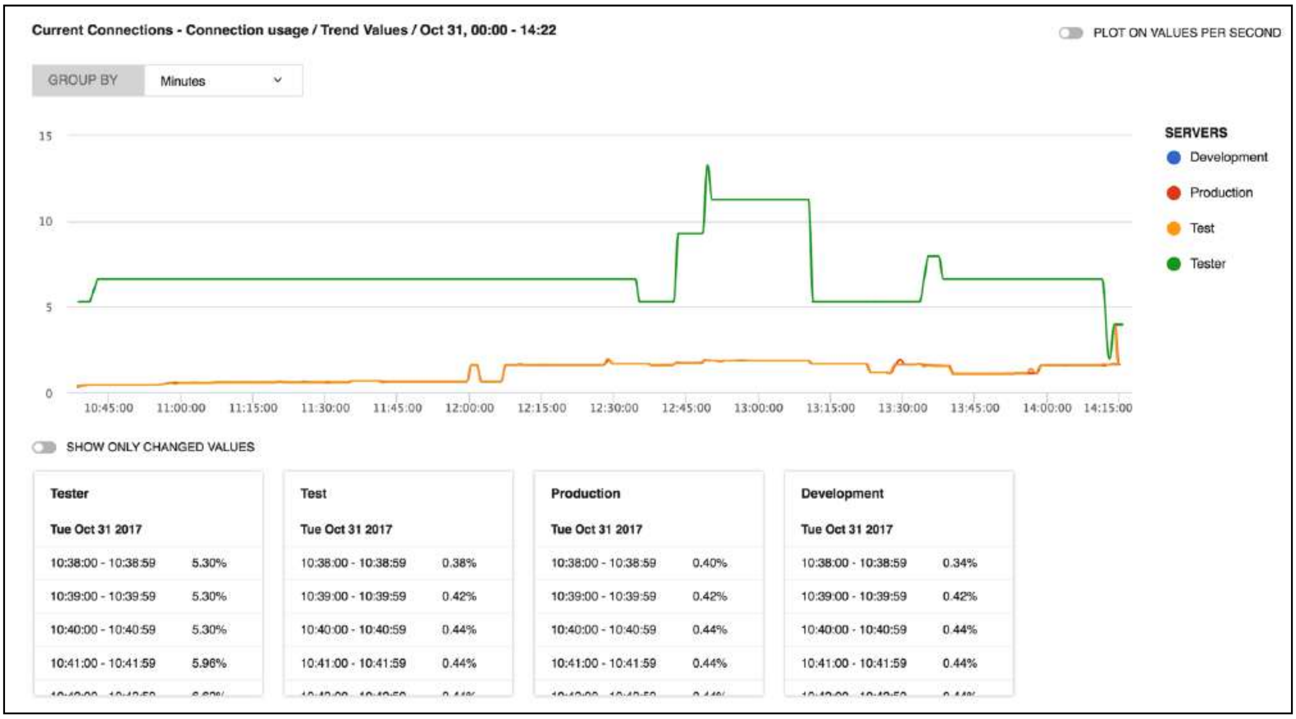


General Info - Available? / Trend Values / Oct 31, 11:19 - 14:19

SHOW ONLY CHANGED VALUES

Tester	
14:07:28	Yes
14:08:28	Yes
14:09:28	Yes
14:10:28	Yes
14:11:19	Yes
14:12:17	No
14:12:18	No
14:12:19	No
14:12:20	No
14:12:21	No
14:12:22	No
14:12:23	No
14:12:24	No
14:12:25	No

- **When did your MySQL traffic peak in the last few days?**



Current Connections - Connection usage / Trend Values / Oct 31, 00:00 - 14:22

PLOT ON VALUES PER SECOND

GROUP BY: Minutes

SHOW ONLY CHANGED VALUES

Tester	
Tue Oct 31 2017	
10:38:00 - 10:38:59	5.30%
10:39:00 - 10:39:59	5.30%
10:40:00 - 10:40:59	5.30%
10:41:00 - 10:41:59	5.96%

Test	
Tue Oct 31 2017	
10:38:00 - 10:38:59	0.38%
10:39:00 - 10:39:59	0.42%
10:40:00 - 10:40:59	0.44%
10:41:00 - 10:41:59	0.44%

Production	
Tue Oct 31 2017	
10:38:00 - 10:38:59	0.40%
10:39:00 - 10:39:59	0.42%
10:40:00 - 10:40:59	0.44%
10:41:00 - 10:41:59	0.44%

Development	
Tue Oct 31 2017	
10:38:00 - 10:38:59	0.34%
10:39:00 - 10:39:59	0.42%
10:40:00 - 10:40:59	0.44%
10:41:00 - 10:41:59	0.44%

And these are the few instances where we can find the answers for the following questions:

- When replication slave went down?
- Whether their was a hacking attempt?
- Which counters do you want to analyze?
- What time period do you want to analyze?

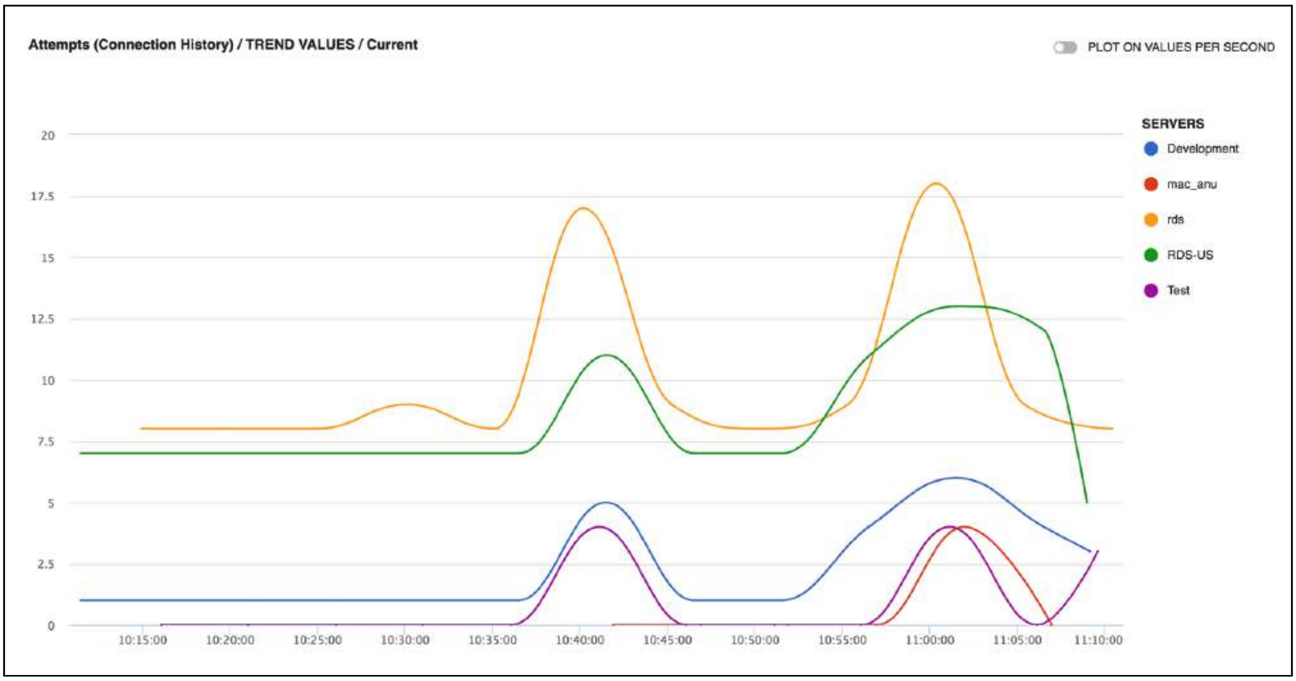
Answering the question thus becomes a matter of comparing different results and of analyzing their differences.

4.4.9.1 Graph Analysis

This feature is a further extension of the History/Trend analysis and lets you view the graphical representation for a monitor across different servers in one unified chart. You can either see the Current trend analysis (shows the graph for last 1 hour) or the Historical trend analysis for a selected time-frame.

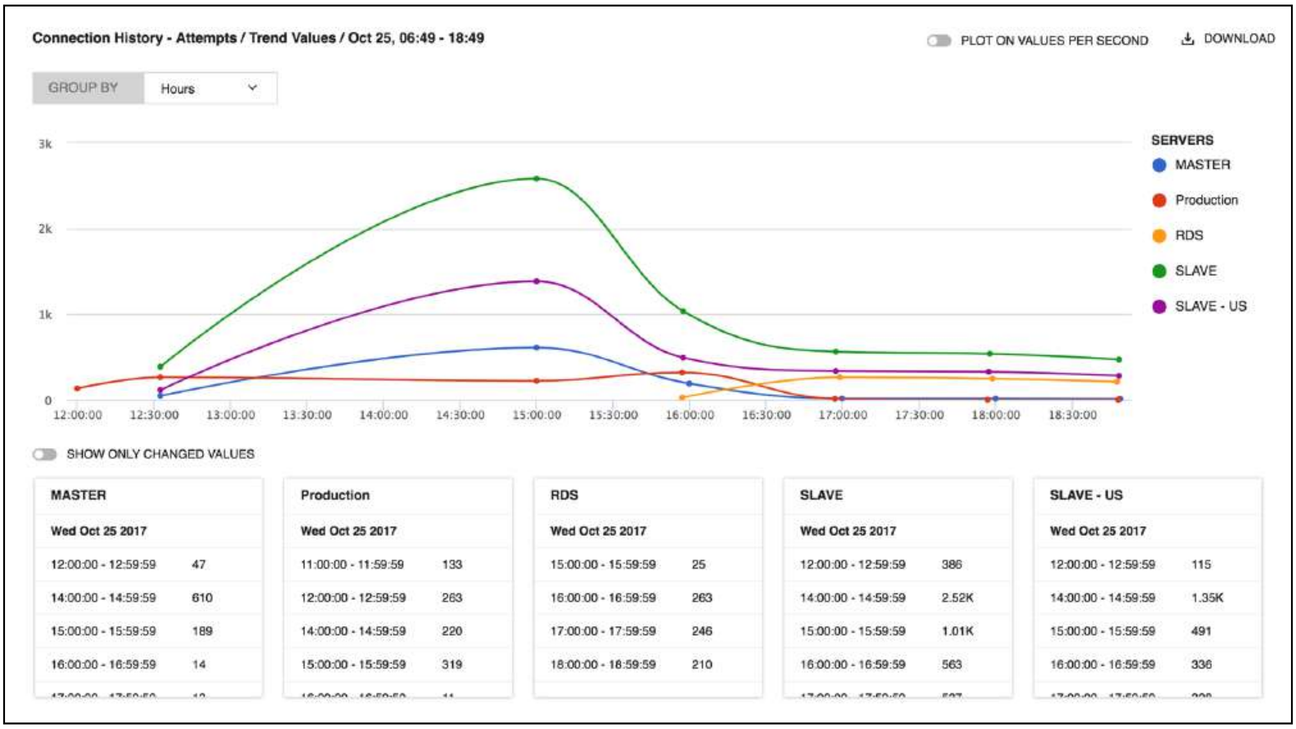
Current Trend Graph

With the Current time-frame selected, the Trend graph icon is present just beside the monitor name. When clicking, it displays the trend graph of last 1 hour duration for the selected servers in the Monitors page. The selected servers list is displayed on the right hand side, the servers can be selected or unselected by clicking the server name.



Historical Trend Graph

This is similar to the Current trend graph, the difference being that you can select any desired time-frame for which you want to see the graph for. You can choose the time-frame as History and then click the **Trend Graph** icon beside the counter name to see the graph. Along with the graph, it also shows the values of the monitor in the tabular form. Enable the option **Show Only Changed Values** to check when the monitor value was changed.



Connection History - Attempts / Trend Values / Oct 25, 06:49 - 18:49

MASTER	
Wed Oct 25 2017	
12:00:00 - 12:59:59	47
14:00:00 - 14:59:59	610
15:00:00 - 15:59:59	189
16:00:00 - 16:59:59	14

Production	
Wed Oct 25 2017	
11:00:00 - 11:59:59	133
12:00:00 - 12:59:59	263
14:00:00 - 14:59:59	220
15:00:00 - 15:59:59	319

RDS	
Wed Oct 25 2017	
15:00:00 - 15:59:59	25
16:00:00 - 16:59:59	263
17:00:00 - 17:59:59	246
18:00:00 - 18:59:59	210

SLAVE	
Wed Oct 25 2017	
12:00:00 - 12:59:59	386
14:00:00 - 14:59:59	2.52K
15:00:00 - 15:59:59	1.01K
16:00:00 - 16:59:59	563

SLAVE - US	
Wed Oct 25 2017	
12:00:00 - 12:59:59	115
14:00:00 - 14:59:59	1.35K
15:00:00 - 15:59:59	491
16:00:00 - 16:59:59	336

4.4.10 Custom SQL Objects (CSOs)

The CSO feature is to utilize information available in Information_Schema (as well as Performance_Schema of MySQL 5.5+) that are not exposed in the basic `SHOW` statements we have been using in the monitors/advisors. In addition to the `Performance_Schema` `SELECT` queries any query which returns a result set can be monitored.

CSOs not only lets you monitor server metrics but also lets you monitor server data.

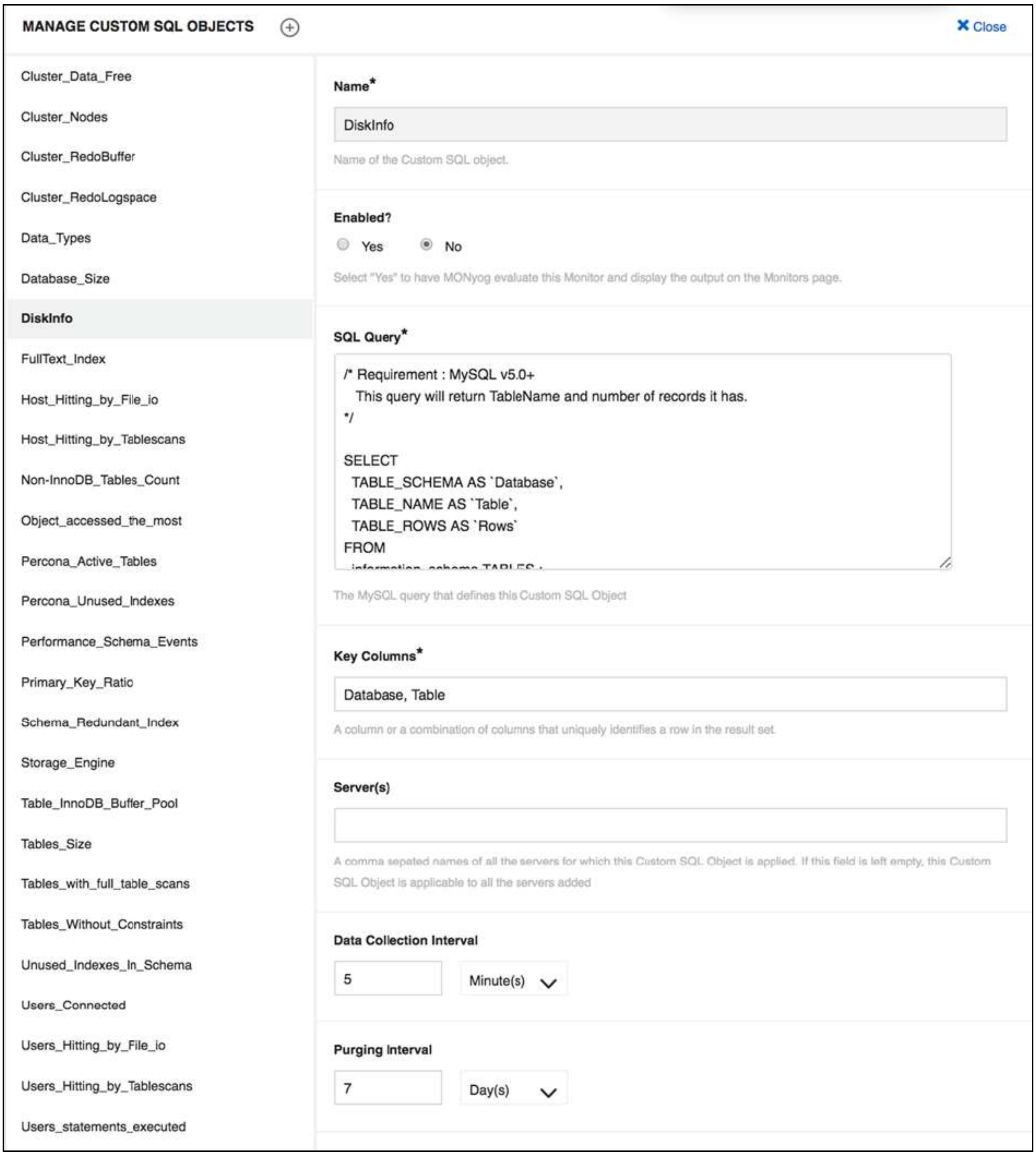
A CSC is based on any user-defined SQL query returning a result set. The array returned by MySQL from the SQL query populates a SQL DM for MySQL Object (a ”Custom SQL Object” (CSO) in this case). This is exposed as a javascript array that may be referenced in SQL DM for MySQL counter definitions like any SQL DM for MySQL object.

4.4.10.1 Enabling pre-defined CSCs and CSOs:

In order to monitor CSOs you need to create a Custom SQL Counter(CSC). SQL DM for MySQL comes shipped with a bunch of pre-defined CSCs with their respective CSOs. By default, all pre-defined CSOs and CSCs are disabled. To enable some of these samples follow these steps:

1. Click the drop-down icon beside the title Monitors -> Manage CSO. The twenty-eight pre-defined CSOs display in the left menu.
   As an example select the **DiskInfo** item. The User Defined SQL-query displays in the SQL box. Sample interval and retention timeframe specific for this CSO may be changed as per your preference and you may specify for which MySQL server(s) this particular CSO should be collected. Also note that one or more Key columns are defined. It must be a column or a set of columns returning (a) unique (set of) value(s) (similar to a UNIQUE KEY in MySQL). Without defining a Key Column, the result monitors might now show proper

values.



MANAGE CUSTOM SQL OBJECTS ⊕

✖ Close

- Cluster_Data_Free
- Cluster_Nodes
- Cluster_RedoBuffer
- Cluster_RedoLogspace
- Data_Types
- Database_Size
- **DiskInfo**
- FullText_Index
- Host_Hitting_by_File_io
- Host_Hitting_by_Tablescans
- Non-InnoDB_Tables_Count
- Object_accessed_the_most
- Percona_Active_Tables
- Percona_Unused_Indexes
- Performance_Schema_Events
- Primary_Key_Ratio
- Schema_Redundant_Index
- Storage_Engine
- Table_InnoDB_Buffer_Pool
- Tables_Size
- Tables_with_full_table_scans
- Tables_Without_Constraints
- Unused_Indexes_In_Schema
- Users_Connected
- Users_Hitting_by_File_io
- Users_Hitting_by_Tablescans
- Users_statements_executed

Name*

DiskInfo

Name of the Custom SQL object.

Enabled?

○ Yes ● No

Select "Yes" to have MONyog evaluate this Monitor and display the output on the Monitors page.

SQL Query*

```
/* Requirement : MySQL v5.0+
   This query will return TableName and number of records it has.
*/

SELECT
  TABLE_SCHEMA AS `Database`,
  TABLE_NAME AS `Table`,
  TABLE_ROWS AS `Rows`
FROM
  information_schema.TABLES ;
```

The MySQL query that defines this Custom SQL Object

Key Columns*

Database, Table

A column or a combination of columns that uniquely identifies a row in the result set.

Server(s)

A comma sepated names of all the servers for which this Custom SQL Object is applied. If this field is left empty, this Custom SQL Object is applicable to all the servers added

Data Collection Interval

5 Minute(s)

Purging Interval

7 Day(s)

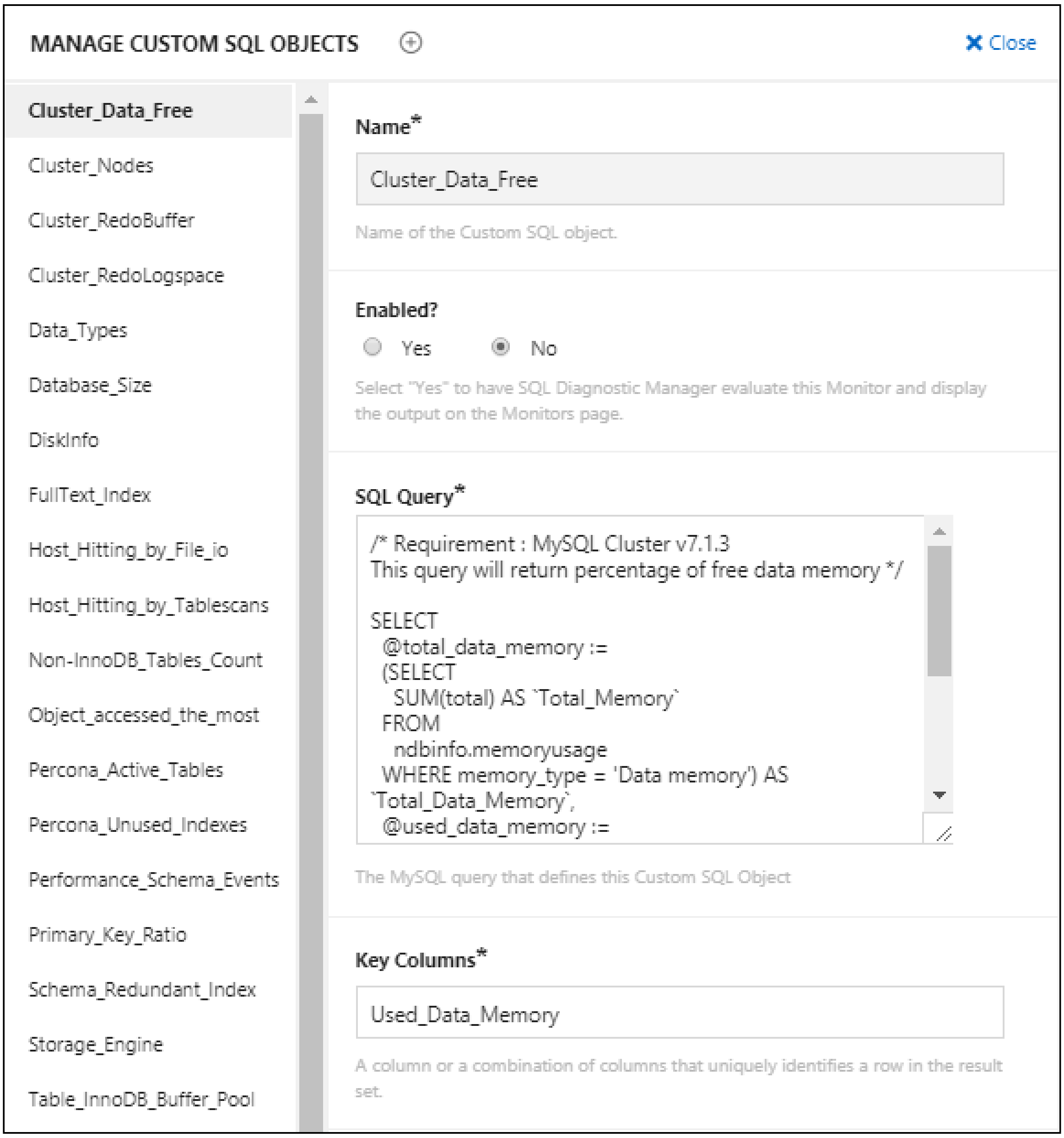
2. Go to Monitors page, select **Manage Monitor Groups**, enable the **DiskInfo** Group, and **Save** the changes. This pre-defined group contains pre-defined CSC’s using the CSOs you enabled in the previous step.
3. Go to Monitors page, select the **DiskInfo** group that now displays at the bottom. You can see five new counters in that group that in various ways reference the CSOs that we just enabled (click the counter name and next ‘Customize’ as usually to see the javascript code). Customize those further as you want to do with any counter in SQL DM for MySQL.

4.4.11 New CSOs and CSCs

4.4.11.1 What makes a CSO?

Choose Manage CSO from the drop-down. This opens up a form with the following details:

- **SQL**: Any user defined SQL which returns a result set.
- **Key Columns:** CSOs work with a result set which has unique rows. A combination of one or more columns of the result set can be made as a key column as long as long as this key column identifies a unique row in the result set.
- **Server(s):** Comma separated names of the servers for which this SQL needs to be queried every Collection Interval.
- **Data Collection Interval:** Interval in which this SQL is queried periodically. We recommend five minutes which also happens to be the default value.
- **Purging Interval:** The data retention time & we recommend seven days.



MANAGE CUSTOM SQL OBJECTS ⊕ ✖ Close

- Cluster_Data_Free
- Cluster_Nodes
- Cluster_RedoBuffer
- Cluster_RedoLogspace
- Data_Types
- Database_Size
- DiskInfo
- FullText_Index
- Host_Hitting_by_File_io
- Host_Hitting_by_Tablescans
- Non-InnoDB_Tables_Count
- Object_accessed_the_most
- Percona_Active_Tables
- Percona_Unused_Indexes
- Performance_Schema_Events
- Primary_Key_Ratio
- Schema_Redundant_Index
- Storage_Engine
- Table_InnoDB_Buffer_Pool

Name*

Cluster_Data_Free

Name of the Custom SQL object.

Enabled?

○ Yes ● No

Select "Yes" to have SQL Diagnostic Manager evaluate this Monitor and display the output on the Monitors page.

SQL Query*

```
/* Requirement : MySQL Cluster v7.1.3
This query will return percentage of free data memory */

SELECT
 @total_data_memory :=
 (SELECT
  SUM(total) AS `Total_Memory`
 FROM
  ndbinfo.memoryusage
 WHERE memory_type = 'Data memory') AS
`Total_Data_Memory`,
 @used_data_memory :=
```

The MySQL query that defines this Custom SQL Object

Key Columns*

Used_Data_Memory

A column or a combination of columns that uniquely identifies a row in the result set.

4.4.11.2 Making its CSC

We create a CSC like any other monitor. Go to monitors page and select **Add new monitor** from the drop-down

- Enter the name of the counter being added.
- Type in the name of the group to which this counter is being added.
- Choose the type of counter as Custom SQL.
- **Formula**: A MySQL server parameter that is needed to compute the value of this counter.
- **Value**: This defines a function that computes the value. Below, you can find a template:

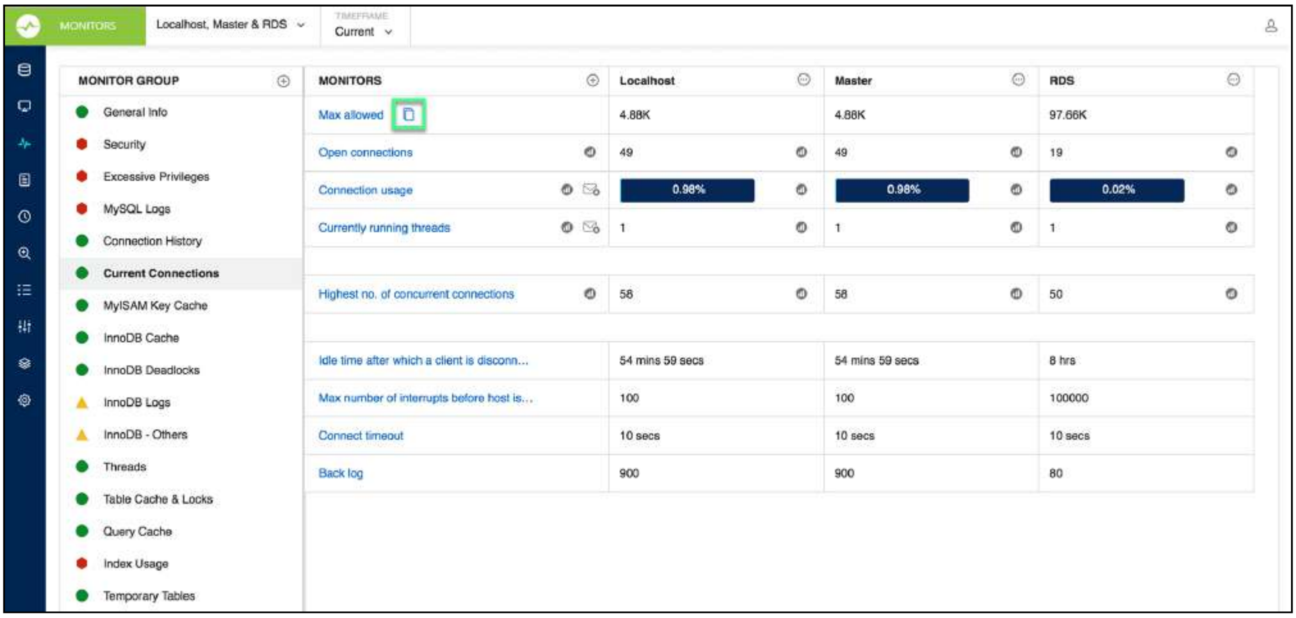
```
function() {
```

```
var sqlObject = MONyog.UserObject('<Name of your Custom SQL Object>');
if (!sqlObject || !sqlObject.isEnabled() || !MONyog.MySQL.Custom.Available)
return '(n/a)';
/* You will have to call select here to fetch the resultset. */
var resultSet = sqlObject.select();
var results = ''; /* results holds the resultset in the form of array of row(s).*/
/*Get column(s) for each row from the result set */
for (i in resultSet) {
if (resultSet.length > 0)
results += '<br>';
results += resultSet[i].<Column name in resultset> +
'.' + resultSet[i].<Column name in resultset>;
}
if (results.length == 0)
results = 'None';
return results;
}
```

- **Description**: Description of Monitor/Advisor.
- **Advice Text**: Advise text to the Monitor/Advisor being added.

4.4.12 Duplicate Monitors

You can duplicate a Monitor and build on that template to make a new Monitor. This is possible by clicking the **Duplicate Monitor** icon adjacent to the respective Monitor name. Refer to Adding or Editing Monitors[58] for more information on adding new Monitors.



MONITOR GROUP

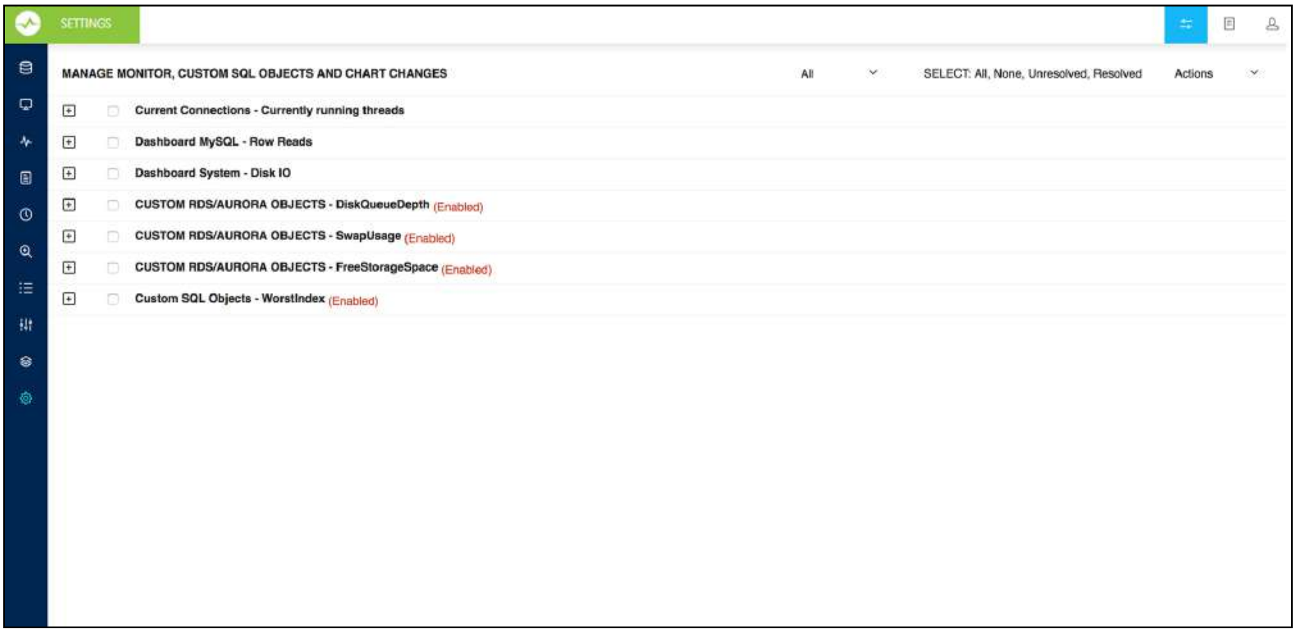
- ● General Info
- ● Security
- ● Excessive Privileges
- ● MySQL Logs
- ● Connection History
- ● Current Connections
- ● MyISAM Key Cache
- ● InnoDB Cache
- ● InnoDB Deadlocks
- ▲ InnoDB Logs
- ▲ InnoDB - Others
- ● Threads
- ● Table Cache & Locks
- ● Query Cache
- ● Index Usage
- ● Temporary Tables

MONITORS	Localhost	Master	RDS
Max allowed	4.88K	4.88K	97.66K
Open connections	49	49	19
Connection usage	0.98%	0.98%	0.02%
Currently running threads	1	1	1
Highest no. of concurrent connections	58	58	50
Idle time after which a client is disconn...	54 mins 59 secs	54 mins 59 secs	8 hrs
Max number of interrupts before host is...	100	100	100000
Connect timeout	10 secs	10 secs	10 secs
Back log	900	900	80

[58] http://wiki.idera.com/x/mAEvvgE

4.4.13 Managing your changes

To review the applied changes in Monitors and/or in Charts, go to Settings and select **Manage changes**.

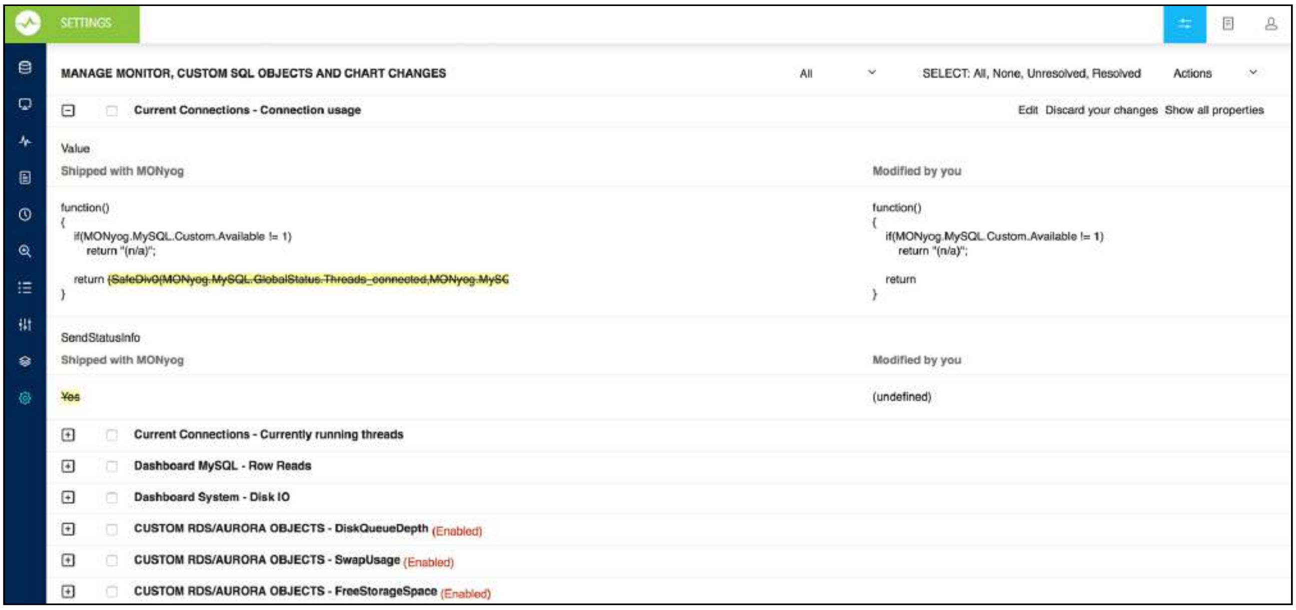


MANAGE MONITOR, CUSTOM SQL OBJECTS AND CHART CHANGES — All — SELECT: All, None, Unresolved, Resolved — Actions

- ⊞ □ Current Connections - Currently running threads
- ⊞ □ Dashboard MySQL - Row Reads
- ⊞ □ Dashboard System - Disk IO
- ⊞ □ CUSTOM RDS/AURORA OBJECTS - DiskQueueDepth (Enabled)
- ⊞ □ CUSTOM RDS/AURORA OBJECTS - SwapUsage (Enabled)
- ⊞ □ CUSTOM RDS/AURORA OBJECTS - FreeStorageSpace (Enabled)
- ⊞ □ Custom SQL Objects - WorstIndex (Enabled)

The Manage Changes page lists all the applied changes you made. If the modification is in Monitors or in Dashboard Charts shipped with SQL DM for MySQL, just click it and it shows you the differences between your definition and the original.

In the event that you need to revert any of your changes to existing Monitors or Charts you can just click **Discard your changes**. SQL DM for MySQL reverts to the original definition of the Monitors or Charts supplied with it. Discarding Monitors or Charts that you have created can cause a permanent deletion.



MANAGE MONITOR, CUSTOM SQL OBJECTS AND CHART CHANGES — All — SELECT: All, None, Unresolved, Resolved — Actions

⊟ □ Current Connections - Connection usage — Edit Discard your changes Show all properties

Value

Shipped with MONyog	Modified by you

function()
{
if(MONyog.MySQL.Custom.Available != 1)
return "(n/a)";
 	
return ~~{SafeDiv0(MONyog.MySQL.GlobalStatus.Threads_connected,MONyog.MySC~~
}
	function()
	{
	if(MONyog.MySQL.Custom.Available != 1)
	return "(n/a)";
	
	return
	}

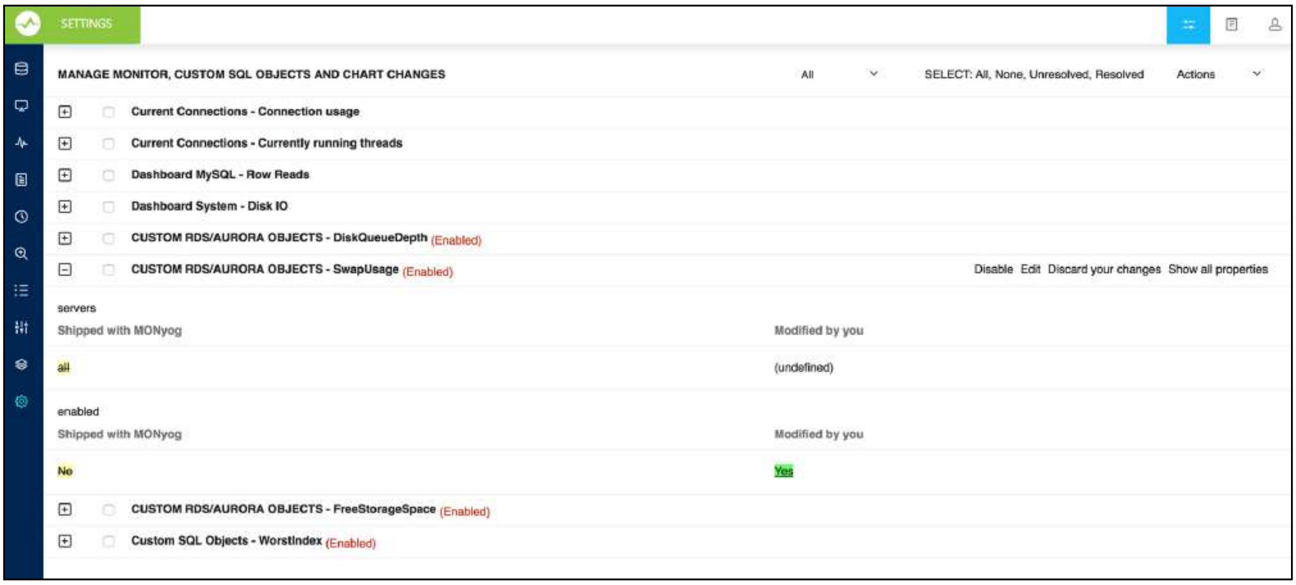
SendStatusInfo

Shipped with MONyog	Modified by you
~~Yes~~	(undefined)

- ⊞ □ Current Connections - Currently running threads
- ⊞ □ Dashboard MySQL - Row Reads
- ⊞ □ Dashboard System - Disk IO
- ⊞ □ CUSTOM RDS/AURORA OBJECTS - DiskQueueDepth (Enabled)
- ⊞ □ CUSTOM RDS/AURORA OBJECTS - SwapUsage (Enabled)
- ⊞ □ CUSTOM RDS/AURORA OBJECTS - FreeStorageSpace (Enabled)

4.4.13.1 Manage Custom SQL Objects

Review the applied changes of the Custom SQL Objects, go to Settings and select **Manage changes**.



MANAGE MONITOR, CUSTOM SQL OBJECTS AND CHART CHANGES   All   SELECT: All, None, Unresolved, Resolved   Actions

- Current Connections - Connection usage
- Current Connections - Currently running threads
- Dashboard MySQL - Row Reads
- Dashboard System - Disk IO
- CUSTOM RDS/AURORA OBJECTS - DiskQueueDepth (Enabled)
- CUSTOM RDS/AURORA OBJECTS - SwapUsage (Enabled)   Disable Edit Discard your changes Show all properties

servers

Shipped with MONyog	Modified by you
all	(undefined)

enabled

Shipped with MONyog	Modified by you
No	Yes

- CUSTOM RDS/AURORA OBJECTS - FreeStorageSpace (Enabled)
- Custom SQL Objects - WorstIndex (Enabled)

4.4.13.2 **Upgrading SQL DM for MySQL**

Before explaining how SQL DM for MySQL manages users changes when upgrading, we need to go over two terms:

- **Resolved:** Monitors or Charts or Custom SQL Objects marked as resolved (displayed in white rows in the Settings -> Manage changes page) are currently in use by SQL DM for MySQL. You can see these Monitors or Charts or Custom SQL Objects either on the Monitors page or the Charts page as the case may be.
- **Unresolved**: Monitors or Charts or Custom SQL Objects marked as unresolved (displayed in red rows in the Settings -> Manage changes page) are neither loaded nor used by SQL DM for MySQL. You have to explicitly resolve such conflicts. Resolving a conflict is as simple a task as deciding whether to keep your changes or discarding them.

SQL DM for MySQL customization framework allows users changes to propagate from one version to another while upgrading. There are certain cases that you need to keep in mind:

- Integrating a Monitor or a Charts or Custom SQL Object which is shipped with SQL DM for MySQL but has been modified by you in another version causes a conflict, i.e. SQL DM for MySQL does not know which definition to use. To resolve this conflict click on **Settings -> Manage changes**, select the unresolved rows, and either select **Use your changes** to keep your changes or **Discard your changes** to revert back to original definition of the Monitor or a Charts.

> You can perform the same action for an individual conflicted Monitor or Charts by clicking **Discard your changes**, by expanding that particular conflicted Monitor or Charts or Custom SQL Object.

- When integrating a Monitor or a Charts or Custom SQL Objects created by you, SQL DM for MySQL automatically assimilates, and start using it without your intervention.

4.4.14 Monyog object model

The Monyog Object model (MOM) abstracts all OS/MySQL values required for calculating all performance metric. It relieves the user from performing low level tasks like connecting to the servers, executing SQL statement, checking return codes, etc.

Values from MOM are used to calculate and display metric, check thresholds, send notification mails, etc.

All output returned from the "SHOW GLOBAL STATUS" is available as MONyog.MySQL.GlobalStatus.*. For example: to get the value of Uptime status variable, you can use MySQL.GlobalStatus.Uptime.

Similarly the following Object Models are defined as follows:

Variable	MOM Object
`SHOW GLOBAL VARIABLES`	`MONyog.MySQL.GlobalVariables.*`
`SHOW GLOBAL STATUS`	`MONyog.MySQL.GlobalStatus.*`
`SHOW SLAVE STATUS`	`MONyog.MySQL.Slave.*`
OS-level counters	`MONyog.System.*`

⚠ Operating system-level counters measure CPU, memory usage, and so on.

MOM variable for SQL DM for MySQL tags and server names(see page 193)

4.4.14.1 AlertCondition:

This attribute expects an expression that should evaluate to one the following 3 values: Critical, Warning, or None. Generally, a JS expression or function is specified which compares some of the MOM values to the Warning or Critical threshold values defined above. (Optional) Consider two examples of a monitor to receive alerts based on an alert condition.

```
function()
{
if(ToInt(MONyog.MySQL.GlobalStatus.Select_scan) > 1000)
return GetWarnStatusInt(this.Value, this.Critical, this.Warning, true);
else
return "None";
}
```

```
function()
{
```

```
if(this.Value != "(n/a)" && this.Value > 1)
return "Critical";
else
return "None";
}
```

WarmUpRequired: A value of Yes specifies that this metric makes sense only if the server is running for a minimum period of time. If the server is not running for the minimum period, AlertCondition is not evaluated and no alerts would be displayed or notified. The default value is 3 hours, if you want to change this value; go to Settings, select **General** you find MYSQL WARMUP PERIOD.

MailAlert: Specifies whether the user wants mail alerts for this metric in case the thresholds are crossed. (Optional)

Graph: This value defines whether real-time graphs are shown for this metric. (Optional)

Bargraph: This value defines whether Percentage type value should be plotted as Bar Graph. (Optional)

Uptime: This value determines whether this metric contain cumulative values. Cumulative values are those values which always increase continuously since server startup (or since last FLUSH STATUS). For example: Connection Attempts. The value of Connection Attempts is always incremented by the MySQL server. Cumulative values are treated differently from "point in time" values like "Currently Running Threads".

If a metric is Cumulative you should always set the value of this attribute to a constant expression: "MONyog.MySQL.GlobalStatus.Uptime"

Format: The display format of various counters. The only possible value currently applicable is - NumCounterWithSeconds. This specifies whether the metric values should also be displayed in "per second" value.

AdviceText: The advice text that is shown to a user whenever any AlertCondition evaluates to "Critical" or "Warning". This text is also shown as the tool tip when the user points the mouse over the alert icons.

RetryOverride: A MOM variable that takes an integer value and overrides the server-level "Send notification when alert-able" setting at the counter-level. Note: It does not take the value "0".

NotifyStableOverride: A MOM variable that takes either "Yes" or "No" as a value and overrides the server-level "Notify when stable" setting at the counter-level.

4.4.14.2 MOM variables for SQL DM for MySQL tags and server names

Tags and server names that are used in SQL DM for MySQL are exposed as Monyog Object Model (MOM) variables. MONyog.Connections.TagName returns an array of tags for that server.

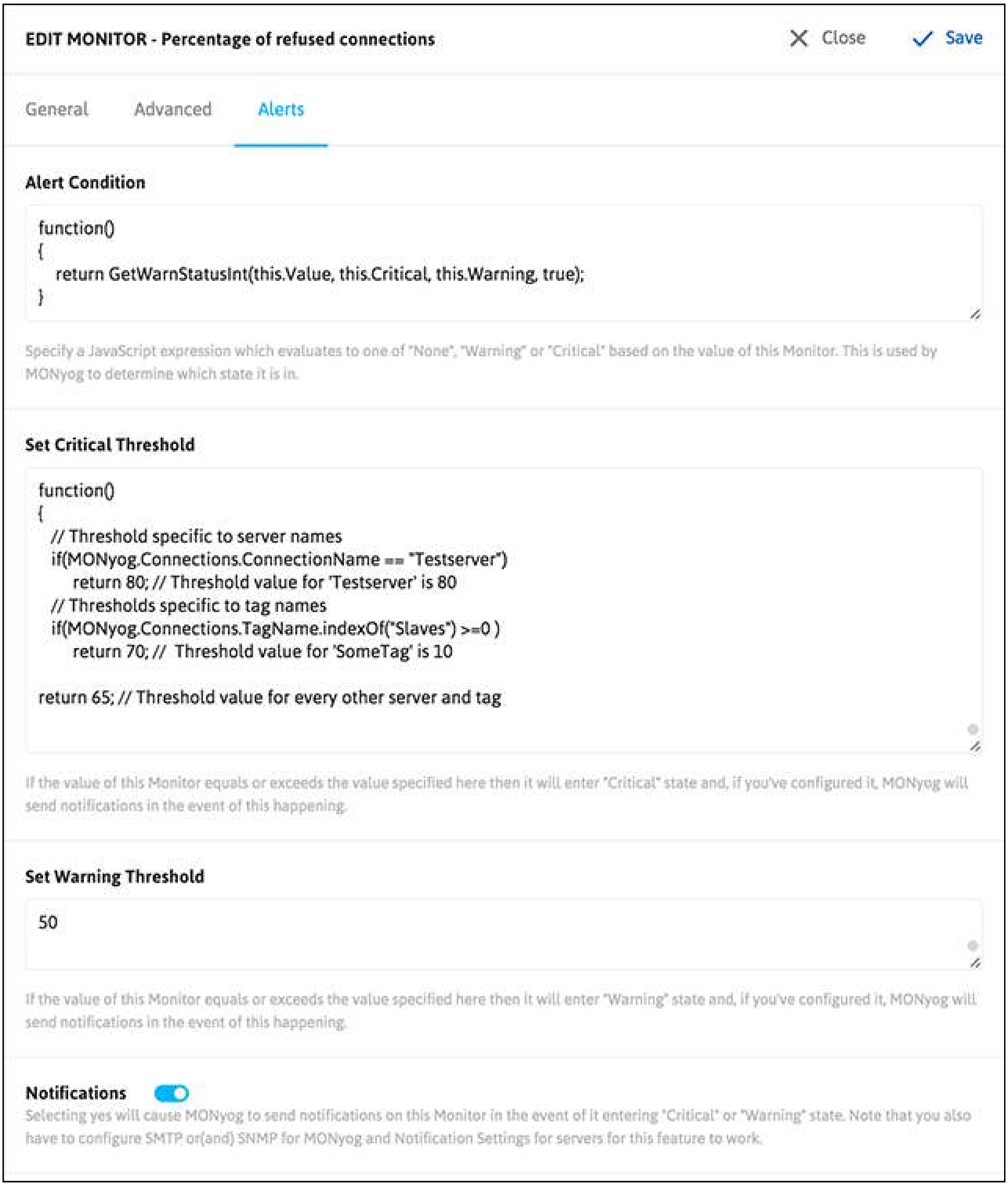
`MONyog.Connections.ConnectionName` Gives the name of that server. This can be extremely useful while setting different threshold levels (based on tags or server names) for a monitor. For example:

The following can be added to the critical/warning field in **Add/Edit server->View Advanced** to set server and tag specific thresholds.

```
// Threshold based on server names
if(MONyog.Connections.ConnectionName == "Testserver")
return 80; // Threshold value for 'Testserver' is 80
if(MONyog.Connections.ConnectionName == "Productionserver")
return 50;
// Thresholds based on tag names
```

```
if(MONyog.Connections.TagName.indexOf("SomeTag") >=0 )
return 10; // Threshold value for 'SomeTag' is 10
if(MONyog.Connections.TagName.indexOf("SomeOtherTag") >=0 )
return 39;
```

Example of a Monitor(Percentage of max allowed reached) that is customized to receive alerts based on server names and tags.



Alert Condition

```
function()
{
   return GetWarnStatusInt(this.Value, this.Critical, this.Warning, true);
}
```

Specify a JavaScript expression which evaluates to one of "None", "Warning" or "Critical" based on the value of this Monitor. This is used by MONyog to determine which state it is in.

Set Critical Threshold

```
function()
{
  // Threshold specific to server names
  if(MONyog.Connections.ConnectionName == "Testserver")
      return 80; // Threshold value for 'Testserver' is 80
  // Thresholds specific to tag names
  if(MONyog.Connections.TagName.indexOf("Slaves") >=0 )
      return 70; //  Threshold value for 'SomeTag' is 10

return 65; // Threshold value for every other server and tag
```

If the value of this Monitor equals or exceeds the value specified here then it will enter "Critical" state and, if you've configured it, MONyog will send notifications in the event of this happening.

Set Warning Threshold

50

If the value of this Monitor equals or exceeds the value specified here then it will enter "Warning" state and, if you've configured it, MONyog will send notifications in the event of this happening.

Notifications ●

Selecting yes will cause MONyog to send notifications on this Monitor in the event of it entering "Critical" or "Warning" state. Note that you also have to configure SMTP or(and) SNMP for MONyog and Notification Settings for servers for this feature to work.

4.4.14.3 SQL DM for MySQL Attribute Reference - Charts Interface

- **Chart Name:** The name of the new Chart.
- **ChartType:** The type of chart can be MySQL or System.

- **SeriesCaption:** Array containing labels for every series in a graph.
- **SeriesValues:** Array containing the values of each series in a graph.
- **ChartValue:** The type to plot the actual values of the seconds_behind_master where it is the difference between the 2 intervals. It can plot the values in 2 ways:
  - **Delta**
  - **Current**

  And the default is **Delta**.
- **This chart shows boolean values:** Possible values are "OnOff" only. This is a special time of Y-Axis plotting that has only 2 possible values - "On" or "Off". This type of graph is useful for plotting Availability status of MySQL/OS across a timeframe. (Optional)
- **Unit:** Specify y-axis unit in the chart. Ex: ['','KB','MB','GB','TB']
- **Unit Factor:** The limit when the unit should be incremented. Ex: if this value is 1024 then, 1024 = 1KB, 1024KB=1MB and so on.
- **Uptime:** See the section for "SQL DM for MySQL Attribute Reference - Monitors Interface". (Optional)

4.4.14.4 System Information Populated by MOM

The System Information is populated by MOM is divided into the following categories:

General

- `MONyog.System.General.version` : The Linux kernel version.

Physical Memory (in Kilobytes)

- `MONyog.System.Mem.sys_mem_total` : Total physical memory.
- `MONyog.System.Mem.sys_mem_free` : Available physical memory.
- `MONyog.System.Mem.proc_mem_vmrss` : Physical memory being used by MySQL.

Swap memory (in Kilobytes)

- `MONyog.System.Swp.sys_swp_total` : Total swap memory.
- `MONyog.System.Swp.sys_swp_free` : Free Swap memory.
- `MONyog.System.Swp.proc_swp_vmsize` : Swap memory being used by MySQL.

CPU

Below are the CPU related metrics. Each gives the number of jiffies spent in various modes, since the last capture.

- `MONyog.System.Cpu.sys_cpu_user` : User mode
- `MONyog.System.Cpu.sys_cpu_nice` : Nice mode
- `MONyog.Systeem.Cpu.sys_cpu_system` : System/Kernel mode
- `MONyog.System.Cpu.sys_cpu_idle` : Spent Idly
- `MONyog.System.Cpu.sys_cpu_iowait` : Spent in waiting for IO
- `MONyog.System.Cpu.sys_cpu_hi` : Spent in hardware interrupts
- `MONyog.System.Cpu.sys_cpu_si` : Spent in Software interrupts

I/O

Below are the block devices related metrics. Each gives the number of blocks read and written to the devices attach to the system.

- `MONyog.System.Io.blocks_in` : Total number of blocks read from the devices.
- `MONyog.System.Io.blocks_out` : Total number of blocks written to the devices.

Custom

- `MONyog.System.Custom.Available` : If the system is available to Monyog or not.

> ⚠ Currently "Timeframe" does not have any effect on system related values.

Disk

- `MONyog.System.Disk.sys_disk_free_mysql` : Amount of free space left on the volume where MySQL data resides.
- `MONyog.System.Disk.sys_disk_freepercent_mysql:` Percentage of free space left.
- `MONyog.System.Disk.sys_disk_total_mysql` : Total size of the volume where MySQL data resides.
- `MONyog.System.Disk.sys_disk_used_mysql` : Space being used by various files on the volume.
- `MONyog.System.Disk.sys_disk_free_innodb` : Amount of free space left on the volume where InnoDB data resides.
- `MONyog.System.Disk.sys_disk_freepercent_innodb` : Percentage of free space left.
- `MONyog.System.Disk.sys_disk_total_innodb` : Total size of the volume where InnoDB data resides.
- `MONyog.System.Disk.sys_disk_used_innodb` : Space being used by various files on the volume.

Connection:

Connection name, MySQL user, SSH user, SSH tunneling user which are saved in connection details are exposed for customization in Monitors. (For instance 'connection name' can be accessed using MONyog.Connections.ConnectionName, 'MySQL user' as-MONyog.connections.MySQLUser etc.)

- `MONyog.connections.ConnectionName` : Name of that server
- `MONyog.connections.TagName` : Returns an array of tags for that server
- `MONyog.connections.MySQLUser` : Using this MySQL user can be accessed
- `MONyog.connections.MySQLHost` : MySQL host can be accessed
- `MONyog.connections.MySQLPort` : MySQL port can be accessed

> ⚠ To retrieve system counters from Linux and access log files from remote system on all platforms, SSH server uses below variables:

- `MONyog.connections.SSHHostSystem` : To access SSH host
- `MONyog.connections.SSHPortSystem` : To access SSH port
- `MONyog.connections.SSHUserNameSystem` : To access SSH username

> ⚠ If you have used SSH tunneling to your MySQL server below variables can be used:

- `MONyog.connections.SSHHostTunnel` : To access SSH host
- `MONyog.connections.SSHUserNameTunnel` : To access SSH username
- `MONyog.connections.SSHPortTunnel` : To access SSH port

4.4.14.5 MySQL Information Populated by MOM:

MySQL Error log

- `MONyog.MySQL.ErrorLog.Is_accessible` : To access the MySQL error log
- `MONyog.MySQL.ErrorLog.Total_size` : Size of MySQL error log
- `MONyog.MySQL.ErrorLog.Size_changed` : To determine any new entry is there or not in MySQL error log
- `MONyog.MySQL.ErrorLog.Last_error` : Last error in the MySQL error log

Innodb Deadlock

- `MONyog.MySQL.InnodbStatus.Deadlock_detected` : Any new Innodb deadlock is found
- `MONyog.MySQL.InnodbStatus.Last_detected_time` : Period in which last deadlock is detected
- `MONyog.MySQL.InnodbStatus.Latest_deadlock` : Latest deadlock is detected

4.4.15 Customizing Monitors

Everything that you see in the Monitors interfaces is defined as JavaScript (JS) objects using Monyog Object Model (MOM) and are editable through a simplified interface from within SQL DM for MySQL.

The SQL DM for MySQL daemon/web server has an embedded JS execution engine. Whenever a request for a report is received by SQL DM for MySQL, a new JS runtime context is created. This runtime compiles and evaluates the JS objects defined by various scripts generated from user input, formats the result, and sends the formatted report to the client. The following sections get into the details of each and every aspect of SQL DM for MySQL JS environment.

Each of the SQL DM for MySQL Advisor rules allows the MySQL DBA to customize the thresholds that are acceptable for specific MySQL servers. As an example, a DBA using the supplied Advisor Rule ''MyISAM Key Cache Misses'' may use lower threshold values for their MySQL servers running OLTP applications, while higher thresholds may be acceptable for OLAP applications.

SQL DM for MySQL customization framework empowers users with the ability to add new advisors, modify existing advisors, and even disable some of the predefined advisors supplied by IDERA.

4.4.16 Adding or Editing Monitors

To add a new Monitor do the following:

- Click the **Add new Monitor** link from the drop-down in Monitors page.
- Choose the type of counter - MySQL, System, or Custom SQL.
- Type in the name of the group to which this monitor is being added.
- Enter the name of the counter being added.

For editing a monitor, click **Edit Monitor** icon beside the monitor in Monitors page.

4.4.16.1 Formula:

A MySQL server parameter that is needed to compute the value of this counter.

Value: Defines a function that computes the value.

Example:

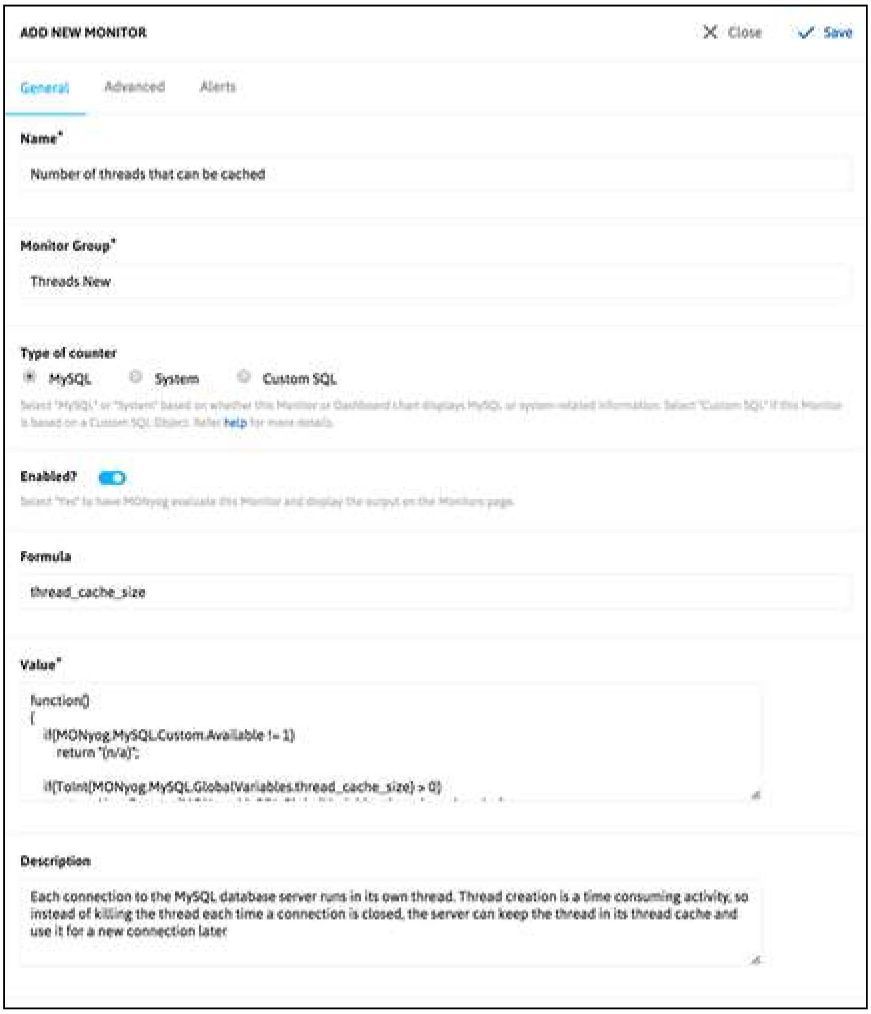
```
function()
{
if(MONyog.MySQL.Custom.Available != 1)
return "(n/a)";

return MONyog.MYSQL.GlobalVariables.max_allowed_packet;
}
```

For a detailed description of values, see Monyog Object Model[59] for more information.

DocText: Description of Monitor.

Advise Text: Advise text to the Monitor being added.

[59] http://wiki.idera.com/x/lgEvvgE



Name*

Number of threads that can be cached

Monitor Group*

Threads New

Type of counter

● MySQL ○ System ○ Custom SQL

Select "MySQL" or "System" based on whether this Monitor or Dashboard chart displays MySQL or system-related information. Select "Custom SQL" if this Monitor is based on a Custom SQL Object. Refer help for more details.

Enabled?

Select "Yes" to have MONyog evaluate this Monitor and display the output on the Monitors page.

Formula

thread_cache_size

Value*

```
function()
{
  if(MONyog.MySQL.Custom.Available != 1)
    return "(n/a)";

  if(ToInt(MONyog.MySQL.GlobalVariables.thread_cache_size) > 0)
```

Description

Each connection to the MySQL database server runs in its own thread. Thread creation is a time consuming activity, so instead of killing the thread each time a connection is closed, the server can keep the thread in its thread cache and use it for a new connection later

In this customization form each field corresponds to a valid MOM property. Some fields take on valid JavaScript expressions while others take on text/HTML as values. The values entered in these fields are used to generate JS objects which are basically name-value pairs.

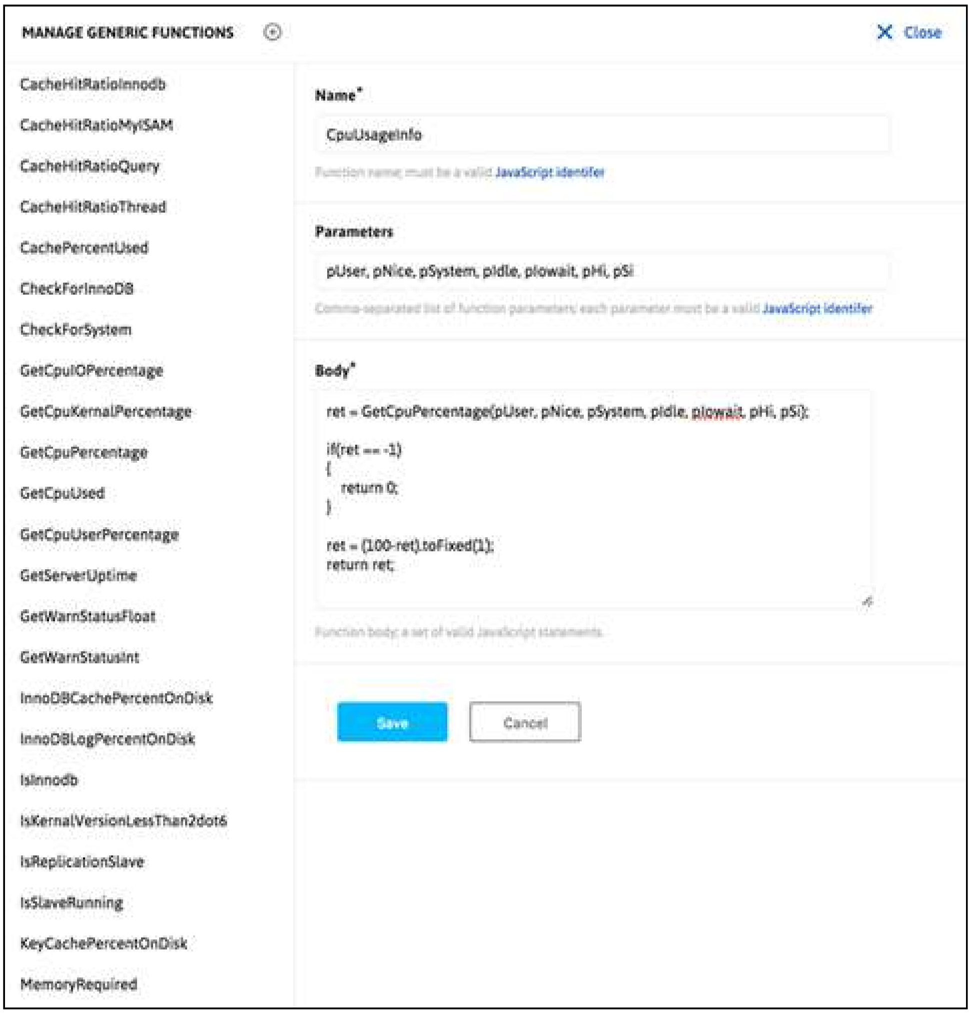
The General tab displays properties which are basically necessary to define a Monitor. Properties with asterisk(*) are mandatory. To customize the behavior of the Monitor further, click the **Advanced** that displays all MOM properties.

For more information, see New CSO's and CSC's[60]

4.4.16.2 Add generic functions

Common functions required across multiple Monitors can be put into generic functions. The generic functions shipped with SQL DM for MySQL cannot be edited. Only the user added generic functions will be editable. For example, the function StorageUnits defined in generic functions takes a numeric value as a parameter and returns a string which is the value specified in terms of K(ilo), M(ega), B(ytes), etc. To add a generic function click the icon adjacent to the title.

[60] http://wiki.idera.com/x/kwEvvgE



Name*

CpuUsageInfo

Function name must be a valid JavaScript identifer

Parameters

pUser, pNice, pSystem, pIdle, pIowait, pHi, pSi

Comma-separated list of function parameters; each parameter must be a valid JavaScript identifer

Body*

```
ret = GetCpuPercentage(pUser, pNice, pSystem, pIdle, pIowait, pHi, pSi);

if(ret == -1)
{
   return 0;
}

ret = (100-ret).toFixed(1);
return ret;
```

Function body; a set of valid JavaScript statements.

4.4.16.3 Enable/Disable notification for a particular Monitor.

SQL DM for MySQL allows user to disable notification for a monitor for a particular server. When notification is enabled for a monitor it generates alerts for all the servers whose values are alertable which may include servers that are not of interest for that monitor. Hence, receiving unwanted alerts is completely irritable. Now, you can avoid notification for a particular server or group of servers by using the below API.
The API action is to disableNotification and enableNotification.

Example API:

To disable notifications for “Percentage of max allowed reached” monitor under “connection history” monitor group for the servers under the tag “Prod”.

```
$ Curl "http://192.168.1.1:5555/?
_object=MONyogAPI&_action=disableNotification&_groupid=5&_counterid=10&_tag=Prod"
```

To enable notifications for “Percentage of max allowed reached” monitor under “connection history” monitor group for a particular server named “Staging-DB”.

```
$ Curl
"http://192.168.1.1:5555/?
_object=MONyogAPI&_action=enableNotification&_groupid=5&_counterid=10&_server=Staging
-DB"
```

You can check the Group ID and Counter ID of a monitor by hovering over the name of the monitor on the monitors page.

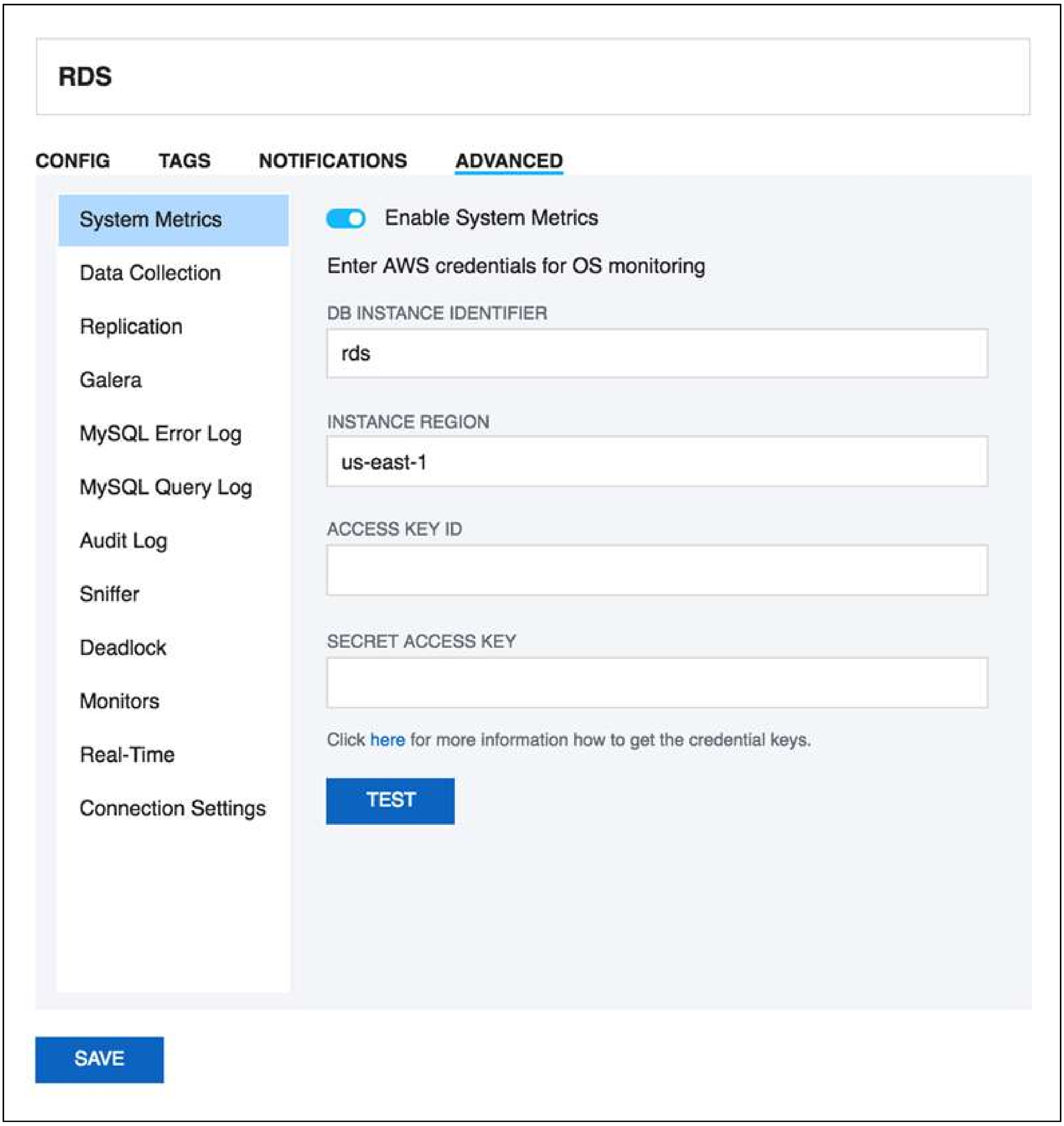
4.4.17 How to RDS_Aurora OS monitoring in SQL DM for MySQL

RDS/Aurora OS monitoring feature introduced with Monyog-8.1.0. SQL DM for MySQL makes use of the CloudWatch API and uses the different OS metrics available with the API to fetch and display the data. All the RDS/Aurora OS monitors are shown under the monitor group "RDS/Aurora Instance Metrics" in Monitors page and the corresponding charts are available on the Dashboard page. In order to be able to see the OS data, you should first enable system metric for the RDS/Aurora instance

4.4.17.1 Enabling System Metrics

Go to **Servers**, select **Edit server**, and choose **Advanced** of the RDS/Aurora instance and **Enable System Metrics**. Once enabled, the user should enter the following four parameters:

1. **DB instance identifier:** A unique name to identify your RDS/Aurora instance.
2. **Instance region:** The region in which your instance is hosted, for e.g: us-east-1
3. **Access key ID:** It is a twenty character long key ID which can be created from the AWS Management Console. It is used to make a programmatic request to AWS.
4. **Secret access key:** It is a forty characters long and can be created from the AWS Management Console. You can refer to the documentation, on how to generate credential keys in the following page: Getting your Access Key ID and Secret Access Key(see page 202).



RDS

Enable System Metrics

Enter AWS credentials for OS monitoring

DB INSTANCE IDENTIFIER: rds

INSTANCE REGION: us-east-1

ACCESS KEY ID:

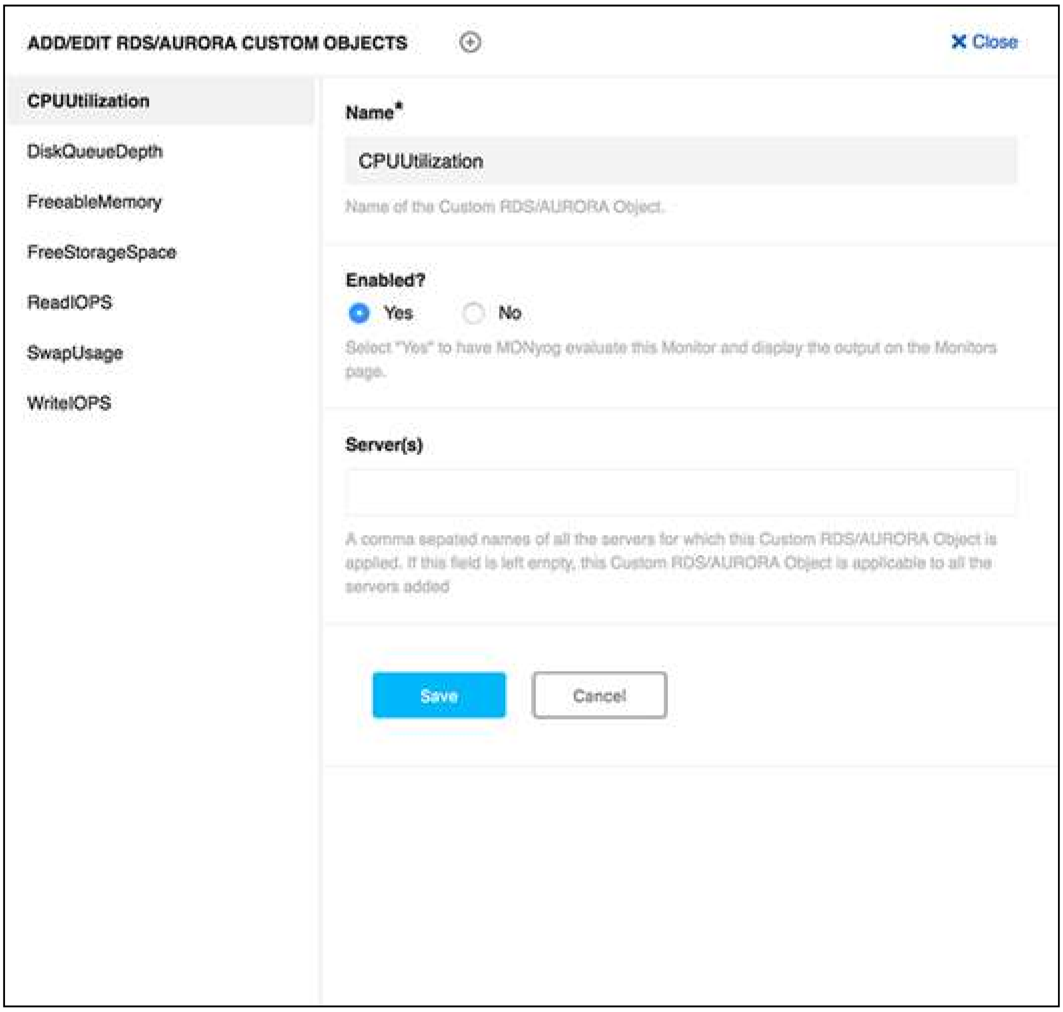
SECRET ACCESS KEY:

Click here for more information how to get the credential keys.

4.4.17.2 Enabling RDS/Aurora Custom Objects

Once the System metrics is enabled, you start getting OS data in the monitor group **RDS/Aurora Instance Metrics.** The custom objects CPU Utilization, Freeable memory, Read IOPS, and Write IOPS are enabled by default, while rest are disabled. In order to enable the others:

Go to **Monitors,** select **RDS/Aurora Instance Metrics**, click the ⊕ icon, select **Manage RDS/Aurora Custom Objects,** and select **Yes** for the Enabled option for any of the listed custom objects.



ADD/EDIT RDS/AURORA CUSTOM OBJECTS

Name*

CPUUtilization

Name of the Custom RDS/AURORA Object.

Enabled?

● Yes ○ No

Select "Yes" to have MONyog evaluate this Monitor and display the output on the Monitors page.

Server(s)

A comma sepated names of all the servers for which this Custom RDS/AURORA Object is applied. If this field is left empty, this Custom RDS/AURORA Object is applicable to all the servers added

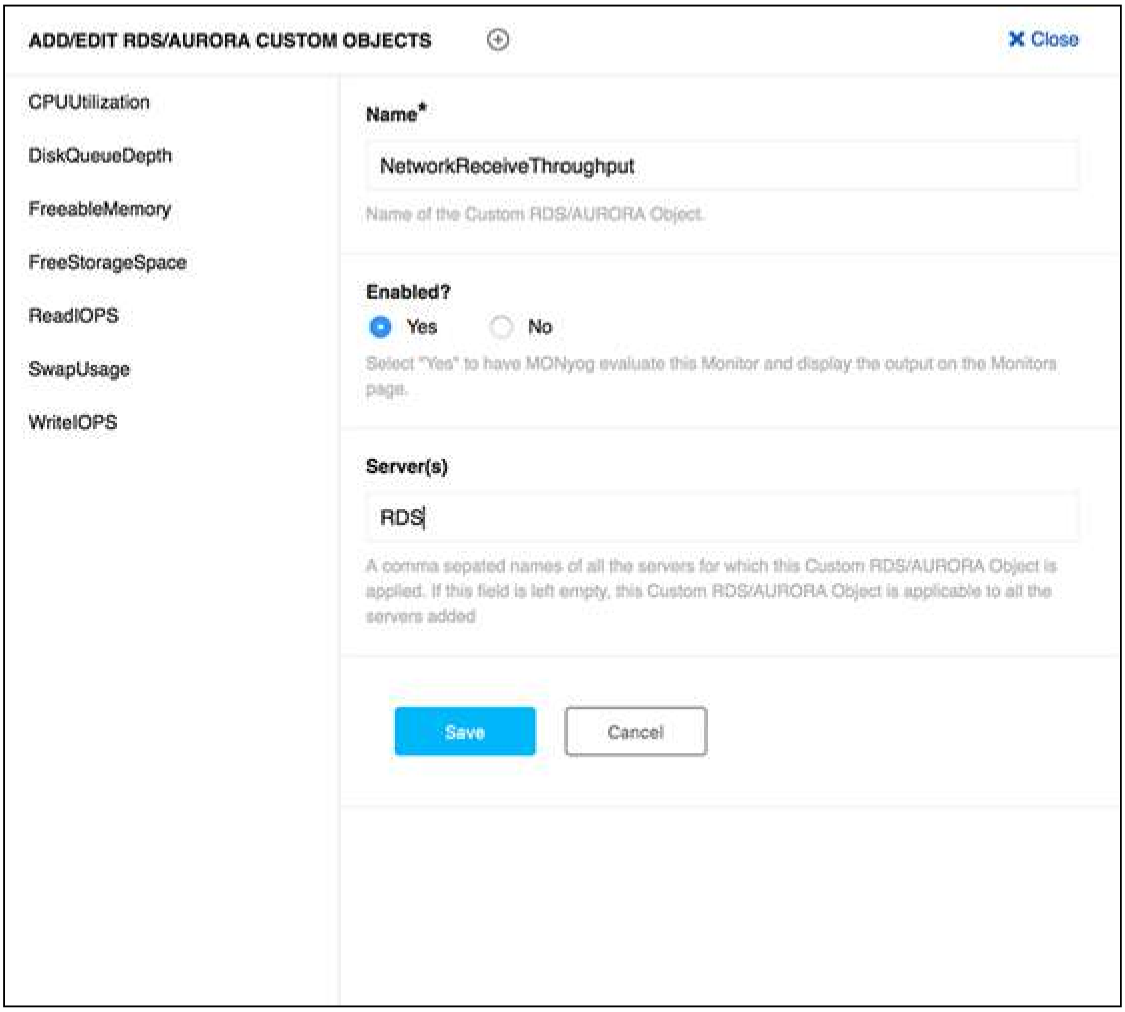
4.4.17.3 Adding New RDS/Aurora Custom Objects

Apart from the default metrics shipped with SQL DM for MySQL, any user can also add an RDS/Aurora Custom Object from the list of available CloudWatch metrics (Refer here for the CloudWatch(see page 202)

metrics available for RDS and here for the Aurora instances(see page 202)). Adding RDS/Aurora Custom Objects is a 2-step process:

Adding a Custom Object

1. Go to Monitors, select **RDS/Aurora Instance Metrics**, click the ⊕ icon, select **Manage RDS/Aurora Objects**, and click **Add new RDS/Aurora custom object (+)** .
2. Enter the name of the CloudWatch Metric which you want to add under the Name field. You can also give the name of the servers comma separated for which you want this metric to be evaluated for.



ADD/EDIT RDS/AURORA CUSTOM OBJECTS ⊕ ✖ Close

Name*

NetworkReceiveThroughput

Name of the Custom RDS/AURORA Object.

Enabled?

● Yes ○ No

Select "Yes" to have MONyog evaluate this Monitor and display the output on the Monitors page.

Server(s)

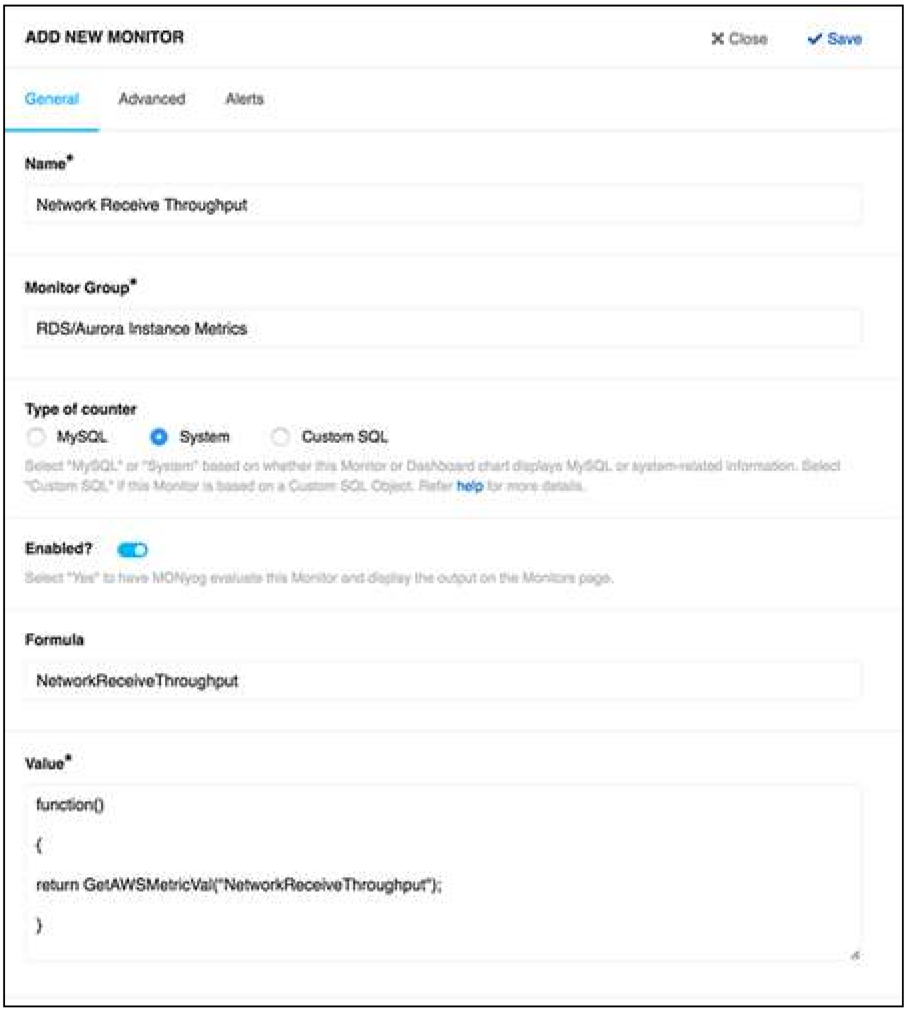
RDS

A comma sepated names of all the servers for which this Custom RDS/AURORA Object is applied. If this field is left empty, this Custom RDS/AURORA Object is applicable to all the servers added

Adding the Monitor

1. Go to Monitors, select **RDS/Aurora Instance Metrics**, click the ⊕ icon, and select **Add new monitor.**
2. Enter Monitor name and the Monitor group name in which you want to add this new monitor to (use "RDS/Aurora Instance metrics" if you want to add it in this group). Select **System** as "Type of counter" if you are adding a system metric.
3. Enter a simple JavaScript function in the Value field using the Cloudwatch metric like:

```
function()
{
return GetAWSMetricVal("NetworkReceiveThroughput");
}
```



ADD NEW MONITOR

Name*

Network Receive Throughput

Monitor Group*

RDS/Aurora Instance Metrics

Type of counter

○ MySQL ● System ○ Custom SQL

Select "MySQL" or "System" based on whether this Monitor or Dashboard chart displays MySQL or system-related information. Select "Custom SQL" if this Monitor is based on a Custom SQL Object. Refer help for more details.

Enabled? ●

Select "Yes" to have MONyog evaluate this Monitor and display the output on the Monitors page.

Formula

NetworkReceiveThroughput

Value*

```
function()
{
return GetAWSMetricVal(“NetworkReceiveThroughput”);
}
```

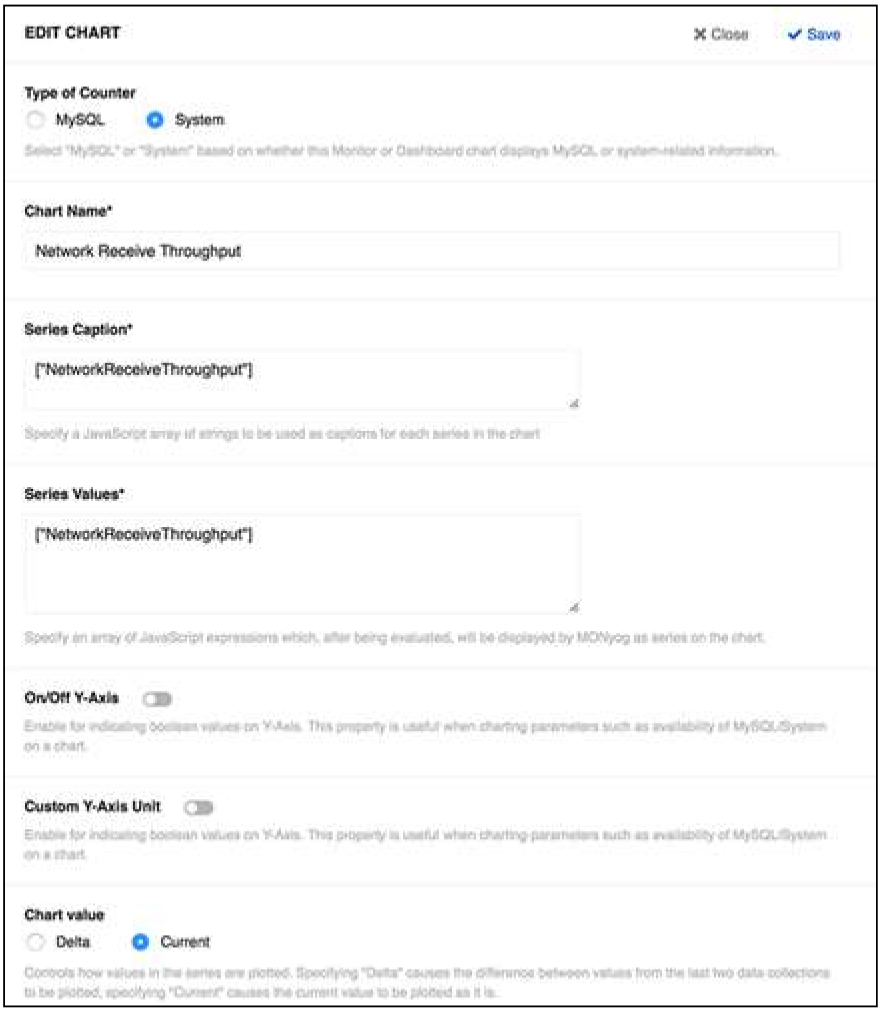
4.4.17.4 Enabling RDS/Aurora Dashboard charts

Go to **Dashboard**, select **Manage Dashboard**, and **Enable the listed RDS/Aurora charts** under System Charts. The Dashboard page gives the flexibility to create a dashboard with a particular set of charts, so a user can create a dashboard with only RDS OS metrics charts for ease of monitoring.

Adding New RDS/Aurora Dashboard charts

To add a RDS/Aurora chart in the dashboard page, the corresponding RDS/Aurora custom objects should be defined and enabled at **Monitors -> RDS/Aurora Instance Metrics -> click** the ⊕ icon **-> select Manage RDS/Aurora Objects**. Once enabled, follow the steps below to add the chart:

1. Go to **Dashboard** , select **Manage Dashboard**, and click **Add new chart** ⊕ .
2. Select **System** as the "Type of counter" and give a proper chart name.
3. Enter the **RDS/Aurora custom object** in the series caption field and the corresponding MOM (Monyog Object Model) variable in Series values field.



EDIT CHART

Type of Counter

○ MySQL ● System

Select "MySQL" or "System" based on whether this Monitor or Dashboard chart displays MySQL or system-related information.

Chart Name*: Network Receive Throughput

Series Caption*: ["NetworkReceiveThroughput"]

Specify a JavaScript array of strings to be used as captions for each series in the chart

Series Values*: ["NetworkReceiveThroughput"]

Specify an array of JavaScript expressions which, after being evaluated, will be displayed by MONyog as series on the chart.

On/Off Y-Axis

Enable for indicating boolean values on Y-Axis. This property is useful when charting parameters such as availability of MySQL/System on a chart.

Custom Y-Axis Unit

Enable for indicating boolean values on Y-Axis. This property is useful when charting parameters such as availability of MySQL/System on a chart.

Chart value

○ Delta ● Current

Controls how values in the series are plotted. Specifying "Delta" causes the difference between values from the last two data collections to be plotted, specifying "Current" causes the current value to be plotted as it is.

4.5 Threads

The Threads page shows you the number of threads currently being executed by MySQL. Each query sent to MySQL is executed in a thread. The Threads feature is used to check which queries are being executed presently. It gives you a general sense of what is keeping your server busy at that very moment.

However, this feature is not the right tool to review statistical information about the queries executed on the MySQL server over a period of time. For finding problem SQL, for example, you should use the Query Analyzer which takes a snapshot of queries, their execution times, etc. over a period of time and presents you with a consolidated report. You can then use the report for further analysis; you can even export it as CSV and run it through other tools.

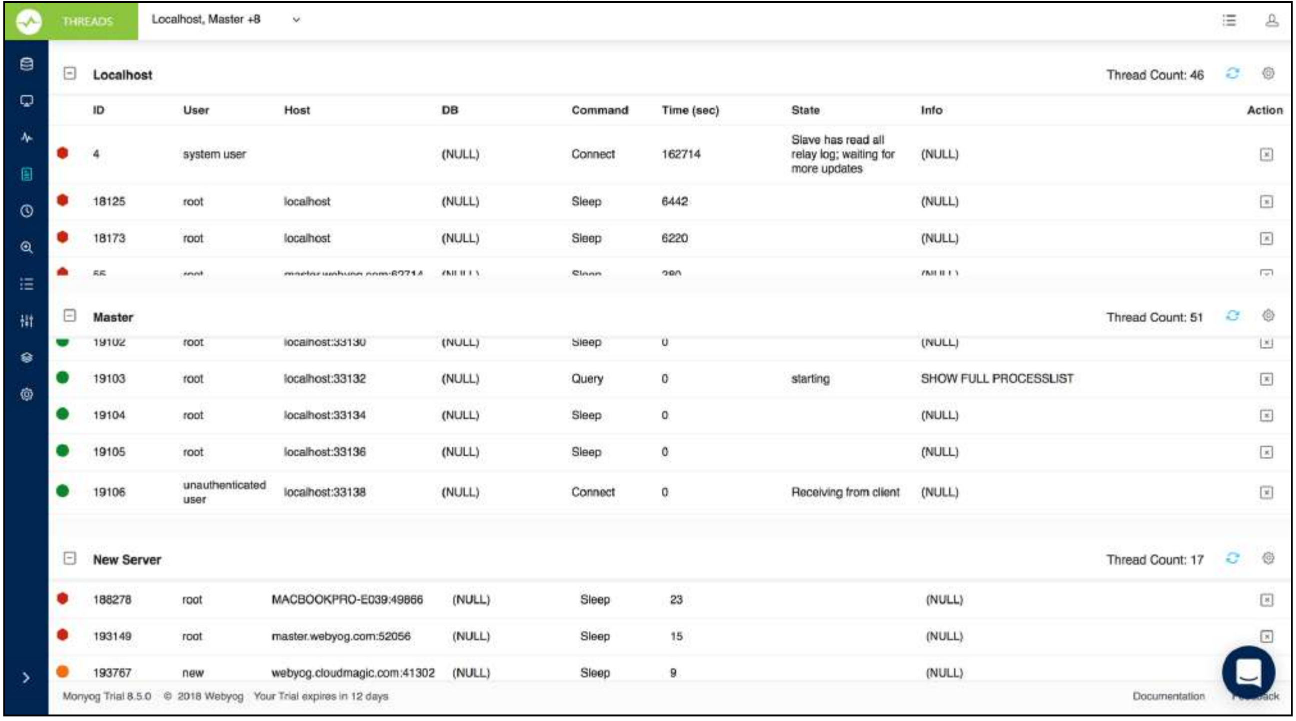
The view can be filtered and sorted as described subsequently. Columns include the connection ID, the connection status, and the text of the current query. From this display you can select a query to view, copy, explain, or kill a query. The **Settings** icon for the respective server thread table allows you to change the PROCESSLIST query to display information you want. This gives you unprecedented flexibility to filter data.

4.5.1 Layout_View of Threads

4.5.1.1 View Queries in real time

The rows returned by the `SHOW FULL PROCESSLIST` query to the MySQL server are stored in a table (named `processlist`) in the embedded database - one column for every column returned plus one additional column named `action`. The Action column contains an HTML string that is used for the various options that are provided by SQL DM for MySQL. This table is stored in a `MEMORY` type database (no disk storage) and only the data returned by the latest `SHOW FULL PROCESSLIST` for each server is saved.

Also, `PROCESSLIST` data is only retrieved when the corresponding page in the web interface is showing in one or more browser windows connected to SQL DM for MySQL.



Localhost — Thread Count: 46

	ID	User	Host	DB	Command	Time (sec)	State	Info	Action
●	4	system user		(NULL)	Connect	162714	Slave has read all relay log; waiting for more updates	(NULL)	☒
●	18125	root	localhost	(NULL)	Sleep	6442		(NULL)	☒
●	18173	root	localhost	(NULL)	Sleep	6220		(NULL)	☒
●	55	root	master.webyog.com:62714	(NULL)	Sleep	280		(NULL)	☒

Master — Thread Count: 51

	ID	User	Host	DB	Command	Time (sec)	State	Info	Action
●	19102	root	localhost:33130	(NULL)	Sleep	0		(NULL)	☒
●	19103	root	localhost:33132	(NULL)	Query	0	starting	SHOW FULL PROCESSLIST	☒
●	19104	root	localhost:33134	(NULL)	Sleep	0		(NULL)	☒
●	19105	root	localhost:33136	(NULL)	Sleep	0		(NULL)	☒
●	19106	unauthenticated user	localhost:33138	(NULL)	Connect	0	Receiving from client	(NULL)	☒

New Server — Thread Count: 17

	ID	User	Host	DB	Command	Time (sec)	State	Info	Action
●	188278	root	MACBOOKPRO-E039:49866	(NULL)	Sleep	23		(NULL)	☒
●	193149	root	master.webyog.com:52056	(NULL)	Sleep	15		(NULL)	☒
●	193767	new	webyog.cloudmagic.com:41302	(NULL)	Sleep	9		(NULL)	

Monyog Trial 8.5.0 © 2018 Webyog Your Trial expires in 12 days

4.5.2 Actions on currently running queries

4.5.2.1 Query Details and Kill

SQL DM for MySQL enables you to do the following actions on your threads:

- `VIEW` the current query for the selected process.
- `EXPLAIN` the query currently executed by the selected process (EXPLAIN works for DML queries only) and this shows the optimized query returned by EXPLAIN EXTENDED if MySQL server supports.
- `KILL` a process.

4.5.2.2 Privileges required for optimal use of this feature

You should notice that to retrieve information about processes of other users, you need the `PROCESS` (or `SUPER`) privilege. However, to kill a process you need the SUPER privilege. SQL DM for MySQL notifies you of the limitations of your permission level (that is, if you do not have full privileges to use all the options provided).

4.5.3 Filtering

4.5.3.1 Advanced filtering options

The display of the Threads is implemented as a `SELECT` - query against the embedded database. The initial query is as follows:

```
SELECT *
FROM processlist
ORDER BY time DESC;
```

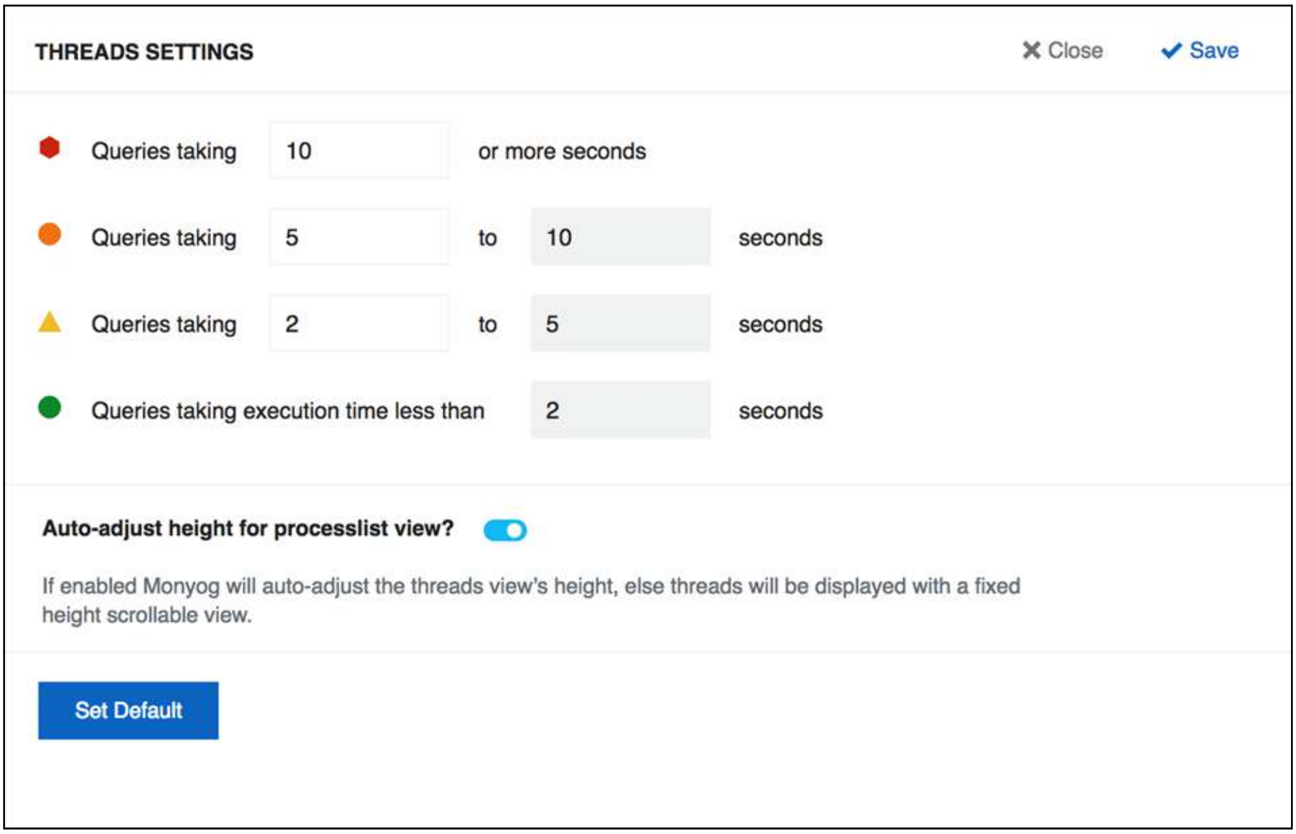
The ORDER BY clause, here, will display the slowest queries at the top (which is convenient for further investigation). You can define your own sort and filter criteria by editing this query using the `WHERE` , `ORDER BY` , and even `GROUP BY` clauses and `AGGREGATES` as you prefer. To change the query simply click on the 'Settings' icon.

For example a user query is as follows:

```
SELECT user, count(user) as no_of_queries, max(time) as running_longest
FROM [processlist]
WHERE ((user <> 'Monyog_user') and (command <> 'Sleep'))
GROUP BY user;
```

ⓘ Note that string comparison is case-sensitive in SQLite and also note the use of [square brackets] in SQLite similar to backquotes in MySQL.

Notice that implementing support for the `SHOW FULL PROCESSLIST` command using a SELECT statement gives you much more flexibility over the output than the command does by itself.



THREADS SETTINGS

🟥 Queries taking 10 or more seconds

● Queries taking 5 to 10 seconds

▲ Queries taking 2 to 5 seconds

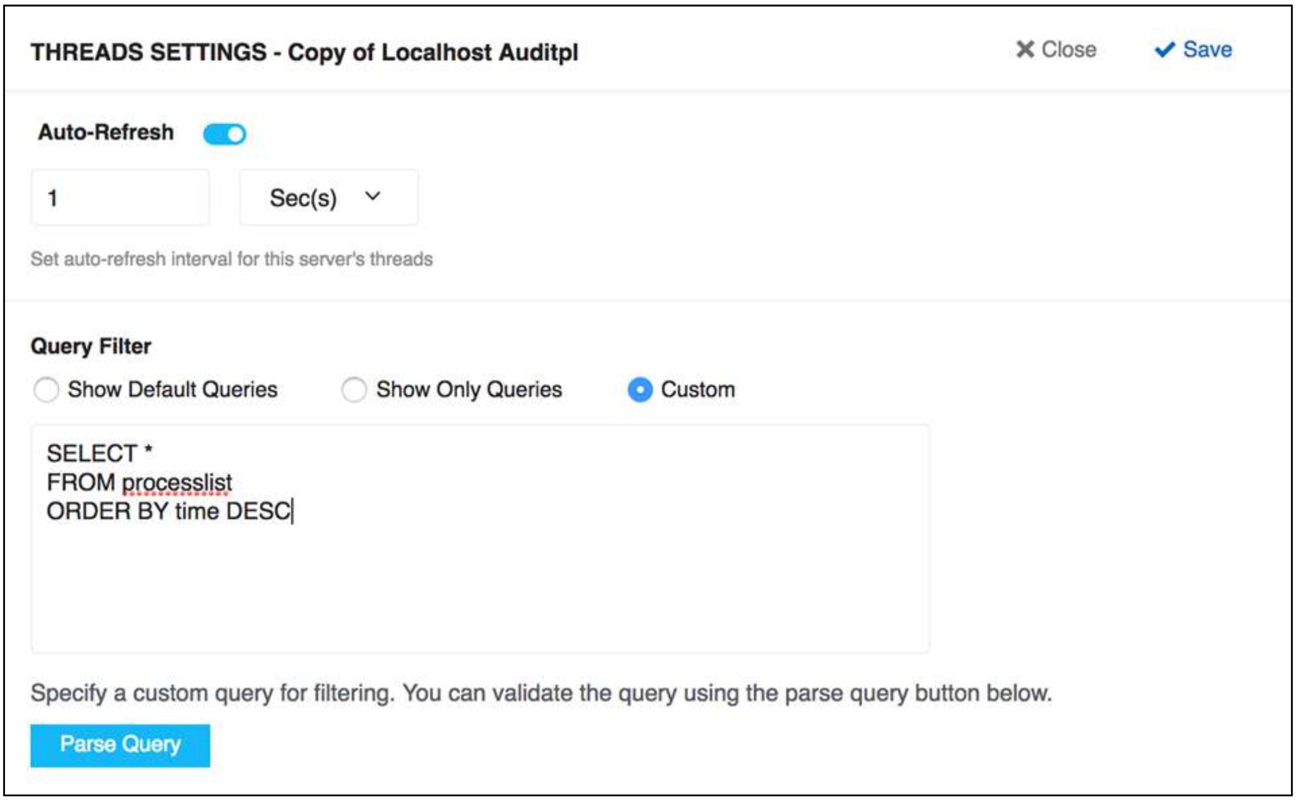
● Queries taking execution time less than 2 seconds

Auto-adjust height for processlist view?

If enabled Monyog will auto-adjust the threads view's height, else threads will be displayed with a fixed height scrollable view.

4.5.4 Customize Threads

SQL DM for MySQL provides a novel and unique way to customize its features. For instance, you can write your own query to filter the output of the processlist. You can issue a SELECT query on this table to filter your Processlist display. This gives you unprecedented flexibility to filter data. The table name is 'processlist' and the column names are Id, User, Host, Db, Command, Time, State, Info, and Action.



THREADS SETTINGS - Copy of Localhost Auditpl

Auto-Refresh ●

1 Sec(s)

Set auto-refresh interval for this server's threads

Query Filter

○ Show Default Queries ○ Show Only Queries ● Custom

```
SELECT *
FROM processlist
ORDER BY time DESC
```

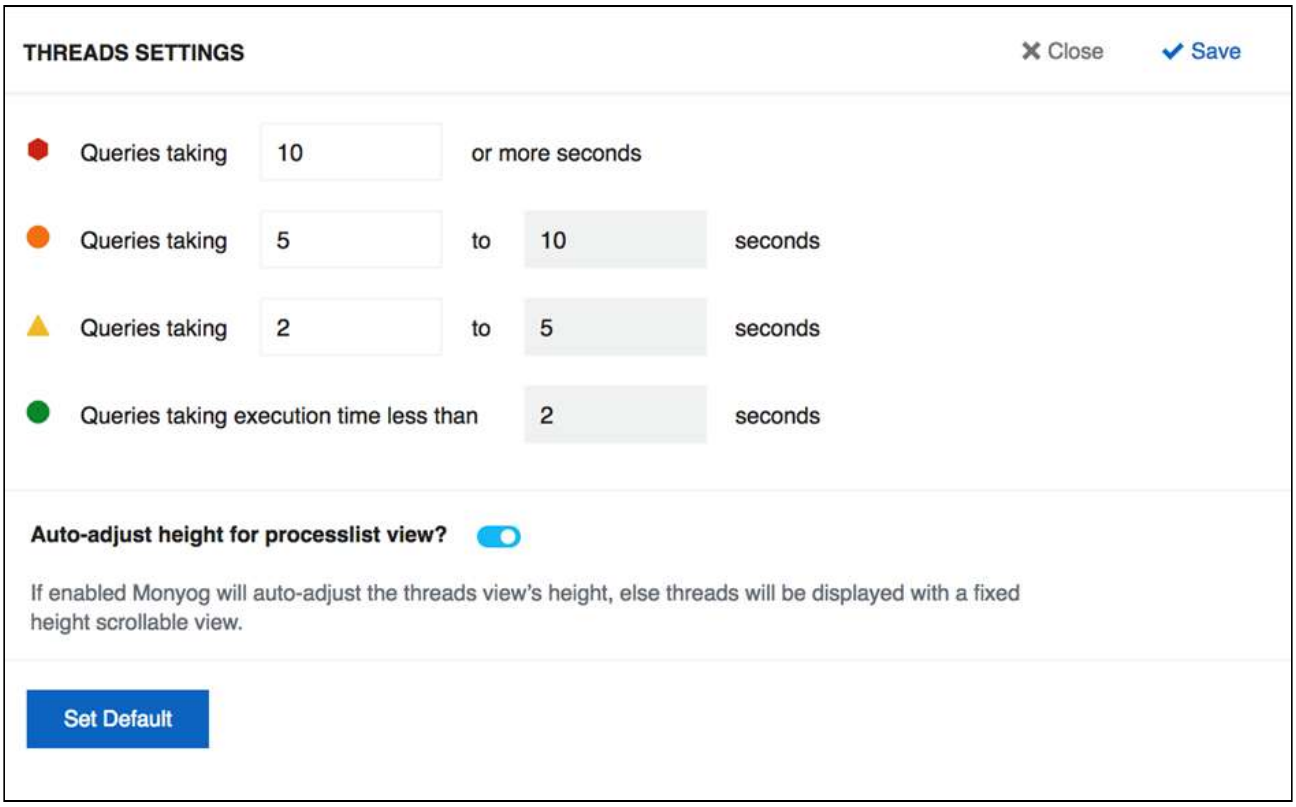
Specify a custom query for filtering. You can validate the query using the parse query button below.

4.5.5 Threads Settings

Using the Threads option, you can customize the timeframe for the currently running queries based on their execution time.

4.5.5.1 How To Change?

Users can change the Threads option by clicking the Threads Settings icon. A new window displays where we can set the time for currently running queries in decreasing order.



⬢ Queries taking 10 or more seconds

● Queries taking 5 to 10 seconds

▲ Queries taking 2 to 5 seconds

● Queries taking execution time less than 2 seconds

Auto-adjust height for processlist view? ●

If enabled Monyog will auto-adjust the threads view's height, else threads will be displayed with a fixed height scrollable view.

An option is given to select the view for the processlist window. You can either select to have a fixed height or dynamic (full screen) height for your processlist window.

4.6 Real-Time

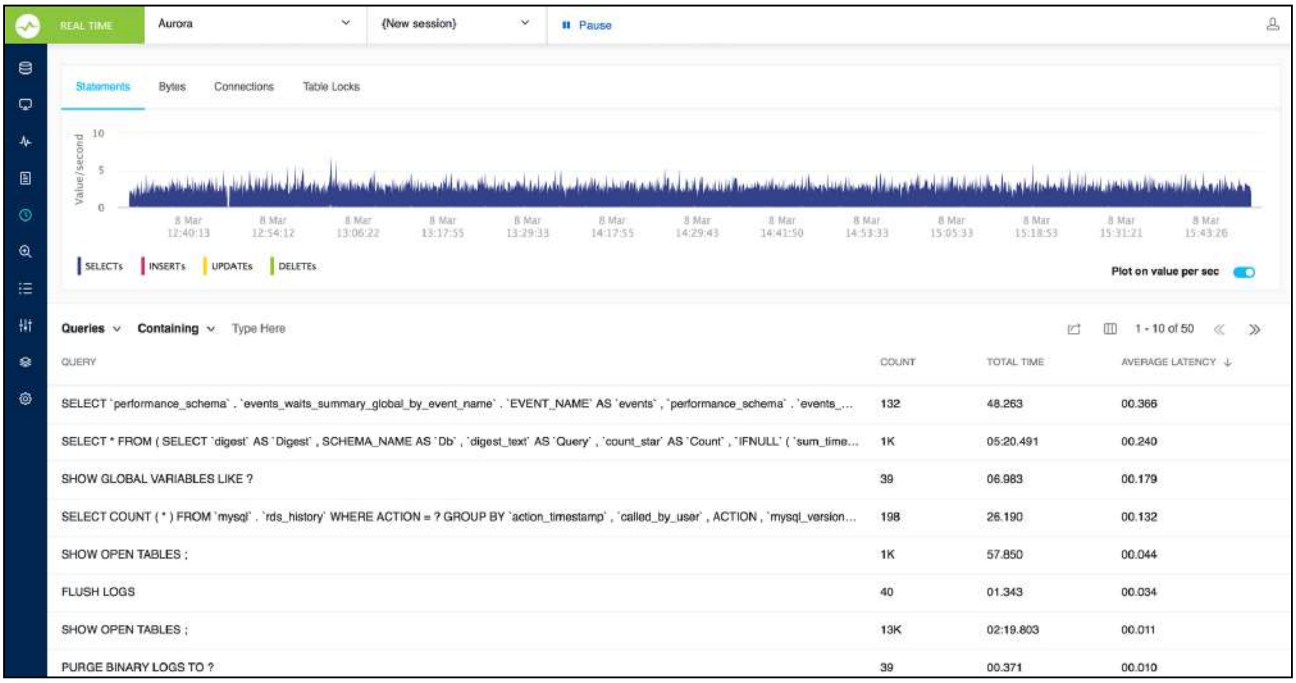
Real-Time continuously executes a bunch of queries on a server and fetches information on the top queries, tables, databases, user, hosts, queries that are locked, queries that are locking, etc. Since the information is retrieved in real time, you get to see what your server is up to at any point in time. You have to start **Recording** a session in order to see your server activity. This action can be stopped and saved for later analysis.

> ✓ Real-Time is what you need to use if you do not have Slow or General query log or sniffer, and still want to know what is happening on your server in real time.

4.6.1 Real-Time collection mode

Real-Time monitoring can be done in two modes: Processlist and Performance Schema.

- Processlist mode, SQL DM for MySQL executes the query `SHOW FULL PROCESSLIST` every second to fetch the queries. Performance schema based Real-time makes use of the performance_schema database of the MySQL server. Performance schema database logs each and every query in its table. SQL DM for MySQL queries the performance_schema and retrieves the queries, including the short-lived ones.
- Performance schema based Real-time, you can get extra information like Number of full table scans done by the query, Success, Error, and Warning count for each query.

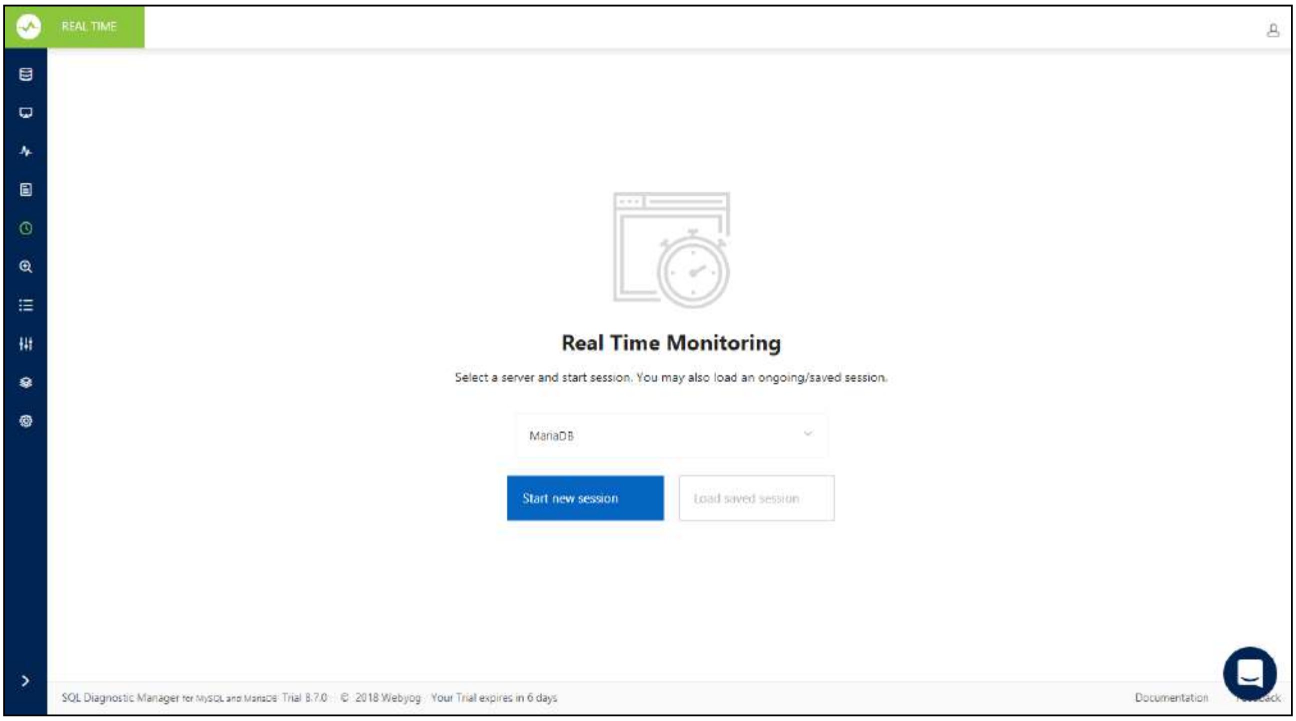


QUERY	COUNT	TOTAL TIME	AVERAGE LATENCY ↓
SELECT `performance_schema` . `events_waits_summary_global_by_event_name` . `EVENT_NAME` AS `events` , `performance_schema` . `events_...	132	48.263	00.366
SELECT * FROM (SELECT `digest` AS `Digest` , SCHEMA_NAME AS `Db` , `digest_text` AS `Query` , `count_star` AS `Count` , `IFNULL` (`sum_time...	1K	05:20.491	00.240
SHOW GLOBAL VARIABLES LIKE ?	39	06.983	00.179
SELECT COUNT (*) FROM `mysql` . `rds_history` WHERE ACTION = ? GROUP BY `action_timestamp` , `called_by_user` , ACTION , `mysql_version...	198	26.190	00.132
SHOW OPEN TABLES ;	1K	57.850	00.044
FLUSH LOGS	40	01.343	00.034
SHOW OPEN TABLES ;	13K	02:19.803	00.011
PURGE BINARY LOGS TO ?	39	00.371	00.010

4.6.2 Using Real-Time

You can pause recording and save a session for later analysis. The charts are zoomable.

To start using Real-Time, select a server from the dropdown, choose a saved session, or select **Start new session.**

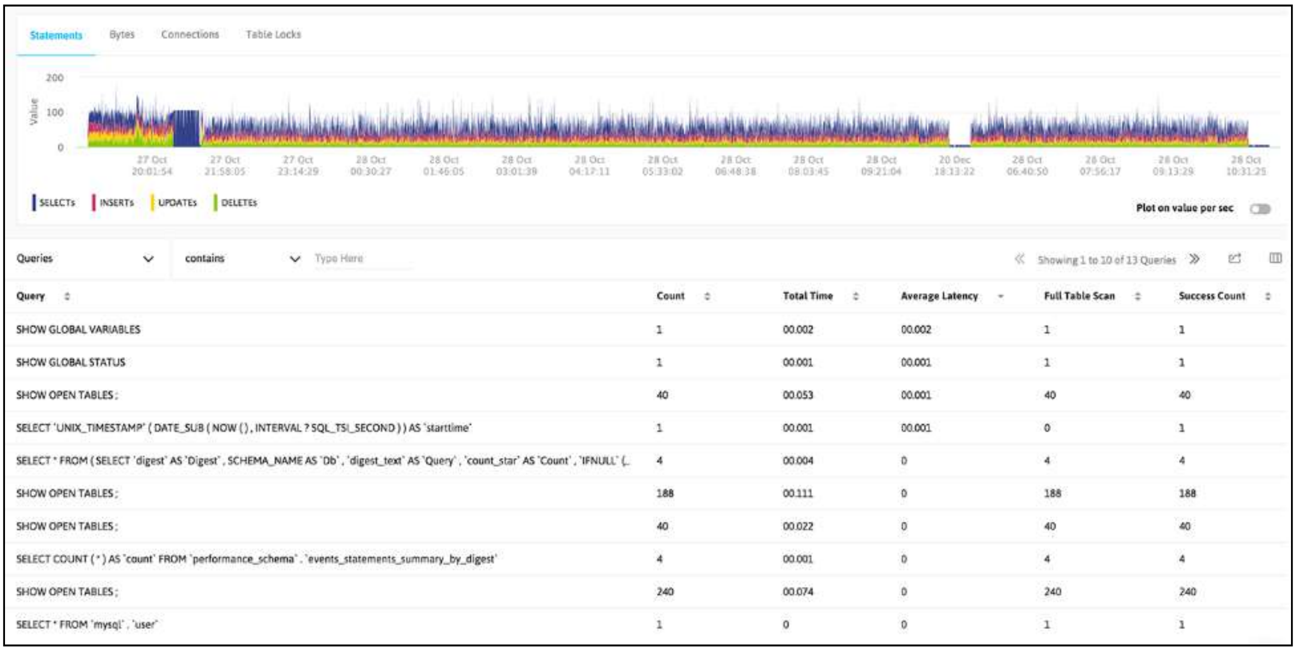


Real Time Monitoring

Select a server and start session. You may also load an ongoing/saved session.

SQL Diagnostic Manager for MySQL and MariaDB Trial 8.7.0 © 2018 Webyog Your Trial expires in 6 days

SQL DM for MySQL stores Real-Time data gathered in a session in SQLite. You can find them in the `data` folders under `realtimedata` . Real-Time data collection for the server stops and save after three days if no activity happens inside that session.

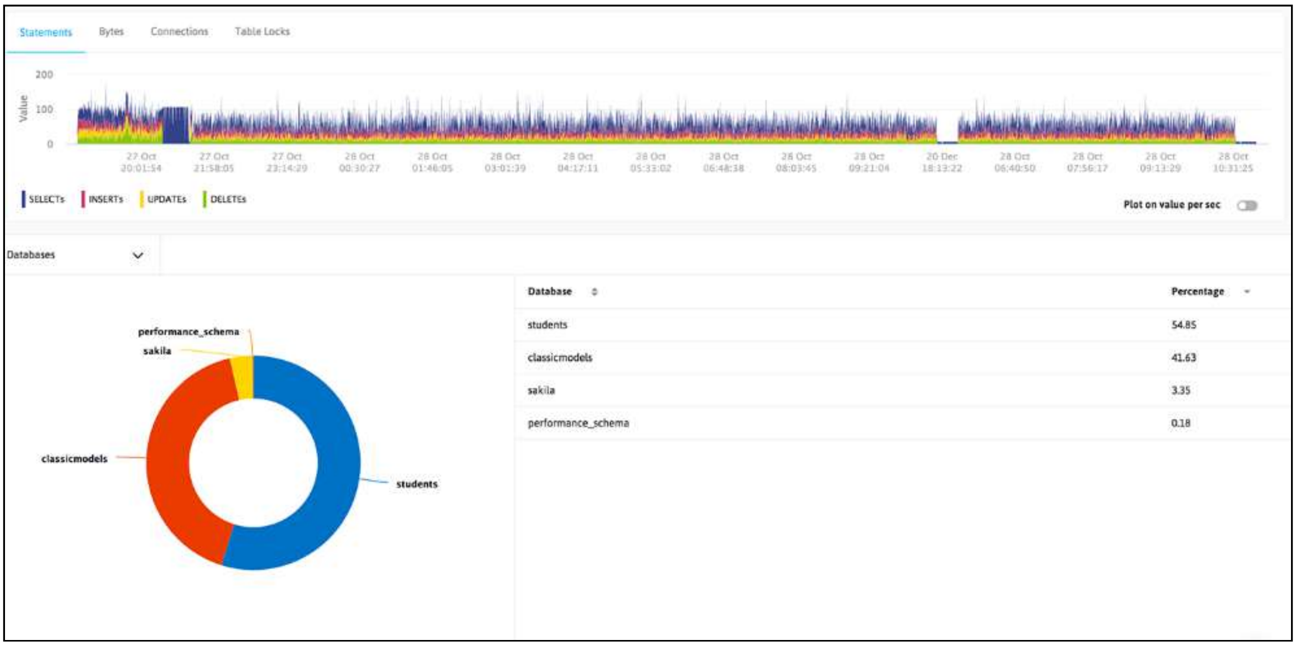


Query	Count	Total Time	Average Latency	Full Table Scan	Success Count
SHOW GLOBAL VARIABLES	1	00.002	00.002	1	1
SHOW GLOBAL STATUS	1	00.001	00.001	1	1
SHOW OPEN TABLES ;	40	00.053	00.001	40	40
SELECT 'UNIX_TIMESTAMP' (DATE_SUB (NOW ( ) , INTERVAL ? SQL_TSI_SECOND)) AS 'starttime'	1	00.001	00.001	0	1
SELECT * FROM (SELECT 'digest' AS 'Digest' , SCHEMA_NAME AS 'Db' , 'digest_text' AS 'Query' , 'count_star' AS 'Count' , 'IFNULL' (...	4	00.004	0	4	4
SHOW OPEN TABLES ;	188	00.111	0	188	188
SHOW OPEN TABLES ;	40	00.022	0	40	40
SELECT COUNT (*) AS 'count' FROM 'performance_schema' . 'events_statements_summary_by_digest'	4	00.001	0	4	4
SHOW OPEN TABLES ;	240	00.074	0	240	240
SELECT * FROM 'mysql' . 'user'	1	0	0	1	1

Real-Time gives you details of your server activity.

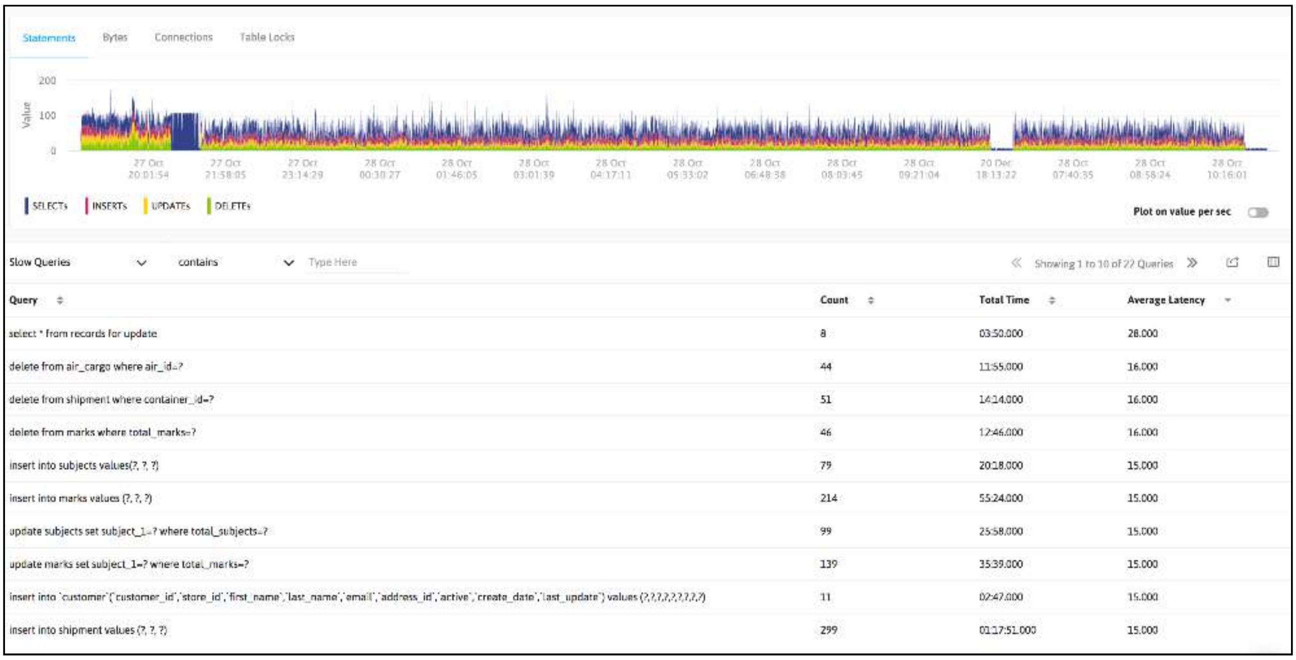
The Queries tab gives you an aggregated view of the queries that are executed in a session. This information is obtained from the `SHOW FULL PROCESSLIST` . You get additional information like execution time, user/host information, and the total count for each query. You do not have to enable general log to retrieve this information.

The Tables and Databases tabs detail information retrieved using `SHOW OPEN TABLES` :



Database	Percentage
students	54.85
classicmodels	41.63
sakila	3.35
performance_schema	0.18

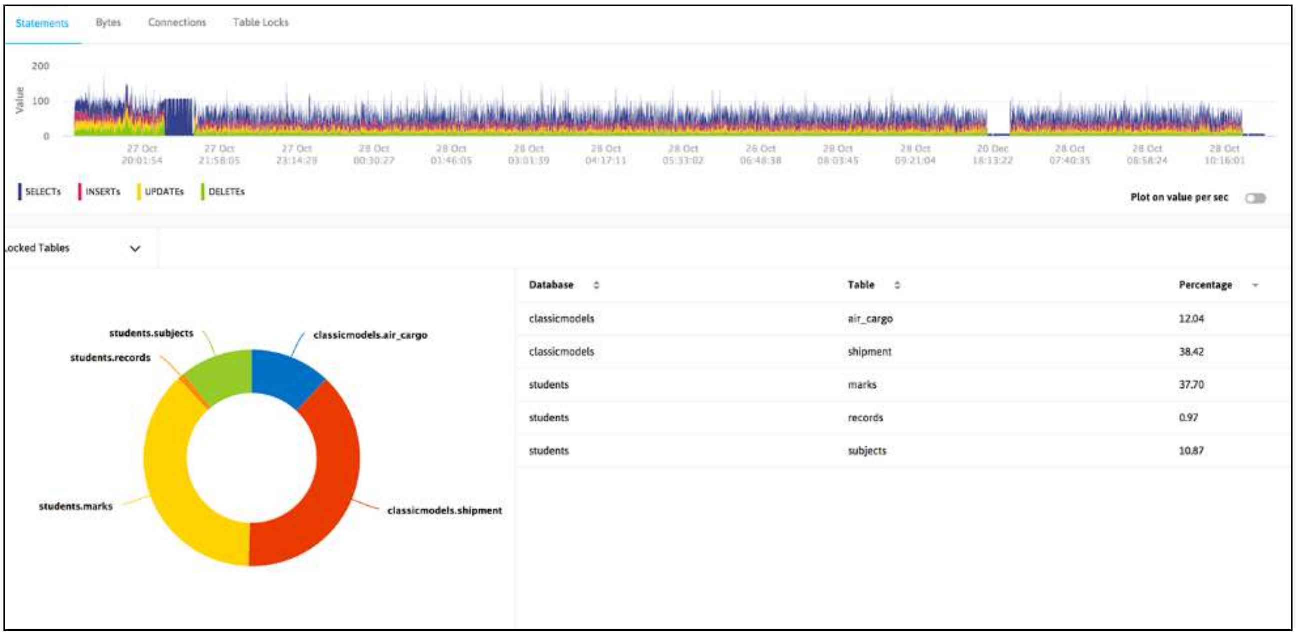
The Hosts, Users, and Slow queries are again fetched from the `SHOW FULL PROCESSLIST` and aggregated information is displayed. There is an option to change the slow query time of Real-Time profiler. On selecting Slow queries, Real-Time considers the queries taking more than the specified time as slow queries.



Query	Count	Total Time	Average Latency
select * from records for update	8	03:50.000	28.000
delete from air_cargo where air_id=?	44	11:55.000	16.000
delete from shipment where container_id=?	51	14:14.000	16.000
delete from marks where total_marks=?	46	12:46.000	16.000
insert into subjects values(?, ?, ?)	79	20:18.000	15.000
insert into marks values (?, ?, ?)	214	55:24.000	15.000
update subjects set subject_1=? where total_subjects=?	99	25:58.000	15.000
update marks set subject_1=? where total_marks=?	139	35:39.000	15.000
insert into `customer`(`customer_id`,`store_id`,`first_name`,`last_name`,`email`,`address_id`,`active`,`create_date`,`last_update`) values (?,?,?,?,?,?,?,?,?)	11	02:47.000	15.000
insert into shipment values (?, ?, ?)	299	01:17:51.000	15.000

The Locked queries, Locking queries, and Locked tables tabs are fetched from InnoDB transaction tables.

⚠ For MySQL versions above 5.1, InnoDB plugin has to installed to retrieve locked and locking query information. But for 5.5 onwards MySQL comes with InnoDB storage engine. Refer Innodb Introduction (see page 213) for more details.



Database	Table	Percentage
classicmodels	air_cargo	12.04
classicmodels	shipment	38.42
students	marks	37.70
students	records	0.97
students	subjects	10.87

4.7 Query Analyzer

The Query Analyzer feature of SQL DM for MySQL helps you identify problem SQL. SQL DM for MySQL can find problem SQL by one or more of the following methods:

- Taking `SHOW PROCESSLIST` snapshots at regular intervals (using SQL DM for MySQL Sniffer)
- Using MySQL Proxy to collect profiling data (using SQL DM for MySQL Sniffer)
- Utilising Performance schema tables (`events_statements_summary_by_digest` , `events_statements_history_long`)
- Parsing Slow Query Log and General Query Log (using SQL DM for MySQL Log Parser)

There are several advantages and disadvantages of each approach.

`SHOW PROCESSLIST` is available in all MySQL versions and it is the easiest to setup. However, taking a snapshot of `SHOW PROCESSLIST` does not guarantee that all queries are captured. Many short-lived queries can be missed between two successive snapshots. It is a quick and easy way to find long running queries.

Log parsing requires some additional setup. Also, switching on the General Query Log puts a significant amount of load on the server. You should always keep the Slow Query Log switched on. Parsing the Slow Query Log is an effective way to find bad queries.

Using MySQL Proxy gives you the most accurate information on profiling SQL. However, during profiling you have to configure your clients to connect to MySQL Proxy, which in turn connects to MySQL server. Using MySQL Proxy ensures that all queries are profiled. It helps you to find problematic queries that don not take much time, but are executed thousands of times. Eliminating such queries can significantly improve the performance of your application.

Performance schema based sniffer makes use of the performance_schema database of the MySQL server. SQL DM for MySQL queries the performance schema database and collects snapshots at regular interval. Since each and every query is logged in the Performance schema database, SQL DM for MySQL displays even the short lived queries using this method.

To use the SQL DM for MySQL Query Analyzer functionality for a specific server, the server General query log or Slow query log details must be configured in **Connection Settings** or a **Query Sniffer** must be enabled for that server.

Using the above tools to find problem SQL is almost always a post-mortem excercise. In certain situations you may want real-time notifications for long-running queries. SQL DM for MySQL can continuously monitor queries in real-time and send notifications (on mail or SNMP) for queries that take more than a specified amount of time to execute. You can also specify an option to kill such queries instantly.

⚠ The SQL DM for MySQL Sniffer taking snapshots of `SHOW PROCESSLIST` is different from the the Processlist feature in that the Sniffer retains the information retrieved in a database for generation of reports and further analysis, whereas the Processlist feature just displays that information as is, without manipulating or storing it.

4.7.1 Different Types of Logs Supported

4.7.1.1 MySQL query log settings

SQL DM for MySQL retrieves (completely or partially) the General query log and the Slow query log from the MySQL servers it connects to, and analyzes them. Here, you see how to set up details for the connection, so that log analysis are available with SQL DM for MySQL. You have to set up details for the general query log and the slow query log independently. Enabling slow query log 'log queries not using indexes' instead needs `SUPER` privilege. Refer to the MySQL Documentation(see page 217) on how to enable and configure logging. MySQL server logs can be written to files on the server machine or to tables in the MySQL database itself.

The MySQL server (since version 5.0) has an option to log (in the slow log) queries that do not use an index. Such queries need not be slow if there are only a few hundred or few thousand records in the table(s) involved. But they are 'potentially slow' and should be identified if they access tables, which continue to grow. You can enable and disable it, as well (SQL DM for MySQL sends the appropriate `SET` of statements to MySQL).

⚠ Only DML and DDL queries are recorded in the slow query log.

Logs written to files

First, consider the situation where server logs are stored as files on the server machine. This is the most common situation and the only one available with MySQL servers before version 5.1.

If it is the first time you configure a server with this option, click the **Fetch query log details** button. MySQL server detects (it is stored in server variables) what logs are enabled and how logging is configured. Click **Test Path** to verify the path. SQL DM for MySQL connects and verify the existence of the file (but not its content).

The log files can be accessed from the local file system (if SQL DM for MySQL and MySQL is running on the same computer) or by using SFTP (if SQL DM for MySQL and MySQL is running on different computers). Note that you must use the file and path syntax of the machine where the logs are.

If the log files can be accessed from a shared drive, over a network, or from a network enabled file system (like NFS on Linux), then SQL DM for MySQL can access them as if they were local files. No additional SSH/SFTP configuration is required in this case: the operating system takes care of the file transfer transparently.

When **Via SFTP** option is chosen, then SSH server details as defined in SSH server details settings are used to read the file from the remote system.

✓ The SSH user must have read access to the log files.

If MySQL server version is greater than 5.1.6 then all the fields mentioned in log analyzer would be editable i.e. if a user changes and saves the settings by clicking **Save** a pop up is displayed where a user can set the new value to corresponding MySQL Server.

By default, MONyog(SQL DM for MySQL) service runs under Local System Account. If you have Slow query or General query logs in a Mapped Network Drive, SQL DM for MySQL is not able to reach it. You need to use UNC notation for SQL DM for MySQL to be able to access them. See FAQ 31(see page 304) for details.

Logs written to MySQL tables

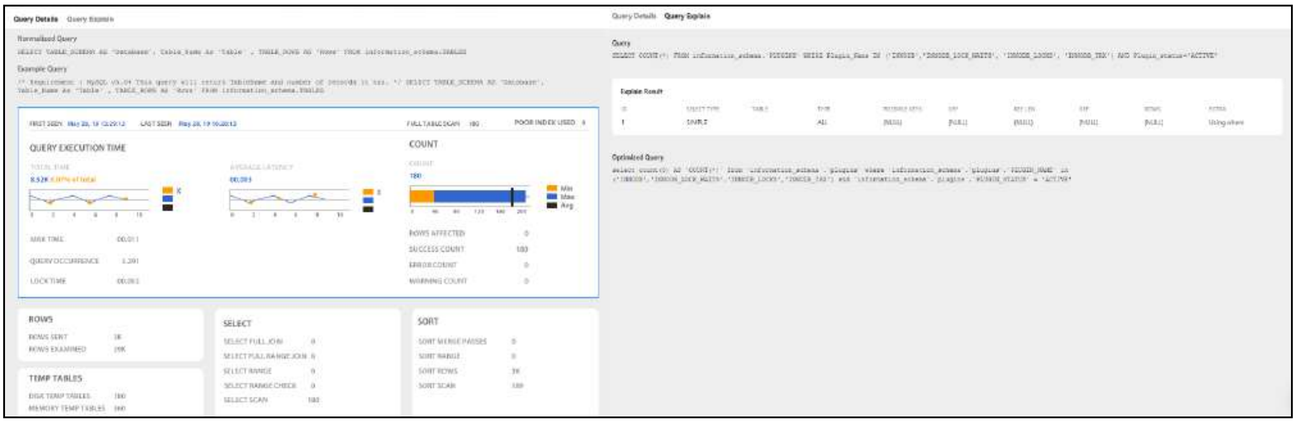
It is supported by MySQL from version 5.1. Also, SQL DM for MySQL supports when this option is available. Here, click the **Fetch Log Details From MySQL** button. When this option is used there is no file path to configure and no SSH details to consider. SQL DM for MySQL can retrieve the server log by sending simple `SELECT` statements. Only the MySQL user used by SQL DM for MySQL to connect to MySQL must have `SELECT` privileges to the tables.

In the Query Analyzer tab select the MySQL server, the type of log (including the 'pseudo log') you want to analyze (Slow Query Log, General Query Log, or Sniffer[61]), and click **Analyze** to start the analysis**:**

[61] http://wiki.idera.com/display/SQLDMYSQL/Sniffer

> ⓘ SQL DM for MySQL includes an improved Top Query Overview to ease the query performance overtime analysis.

After getting the results of your Analysis, click a query to obtain the Query Details and Query Explain tabs with detailed information about the selected query:



> ⓘ Explain plan is available in Query analyzer for Slow_log table based logging, Processlist based sniffer and performance schema based sniffer.

> With the General Query Log there are few specific problems:
>
> 1. With multi-line queries only record the first line of the statement. The reason is that, as the log does not record the statement DELIMITERs, there is no way to tell where a multi-line statement ends. Even the option to 'Show full' is not displayed more than one line as SQL DM for MySQL has only stored one line. Refer to FAQ 23(see page 302).
> 2. It is not always possible to tell what user executed a specific query. When this is the case the User column displays empty in the Query Analyzer output. It is not a bug in SQL DM for MySQL but a limitation with the general log itself.
>
> You can sort the display by clicking on the column header. Note that statement grouping/counting and sorting is case insensitive.

4.7.1.2 Filter settings

There is an option to **Replace literals from the query.** The purpose of this option is to eliminate small differences between almost identical queries. Currently, quoted strings and numbers are replaced with the dummy string '?' only. The filtering settings are stored for that particular session which is not permanent.

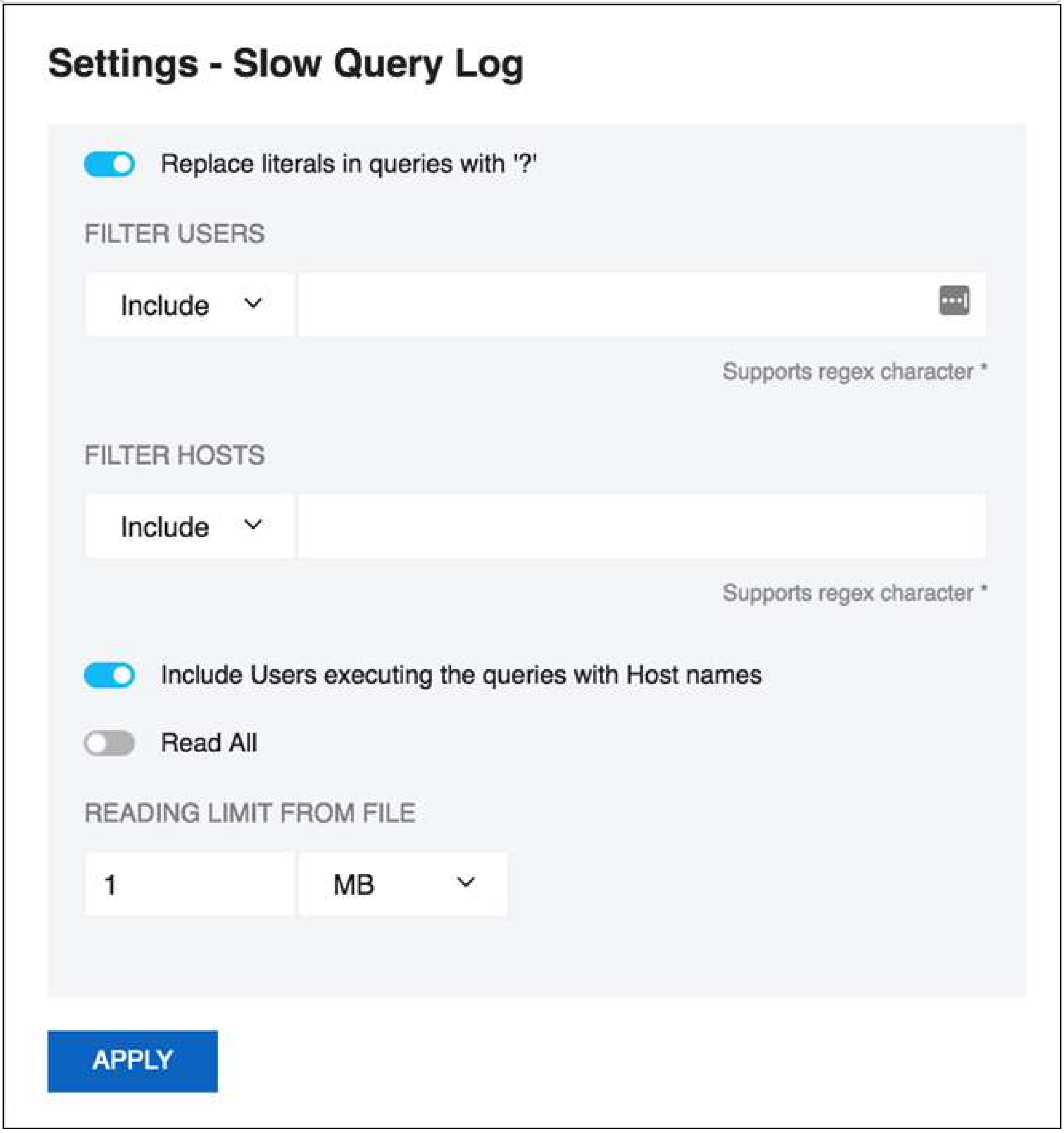
For example:

```
SELECT * FROM customer_master
WHERE cust_id = 23 AND address = 'r;#23 fleet street';
```

becomes,

```
SELECT * FROM customer_master
```

```
WHERE cust_id = ? AND address = ?;
```



Settings - Slow Query Log

Replace literals in queries with '?' (on)

FILTER USERS

Include

Supports regex character *

FILTER HOSTS

Include

Supports regex character *

Include Users executing the queries with Host names (on)

Read All (off)

READING LIMIT FROM FILE

1 MB

The reading limit **All** is selected it considers the whole file for analyzing but if the option is **Last**, it reads the last specified chunk in KB, MB, or bytes out of the whole log file. Also, you can define a timeframe to be analyzed and the size of the 'log chunk' (in KB, MB and Bytes for file based logs and in rows for table-based logs) to be transferred to SQL DM for MySQL.

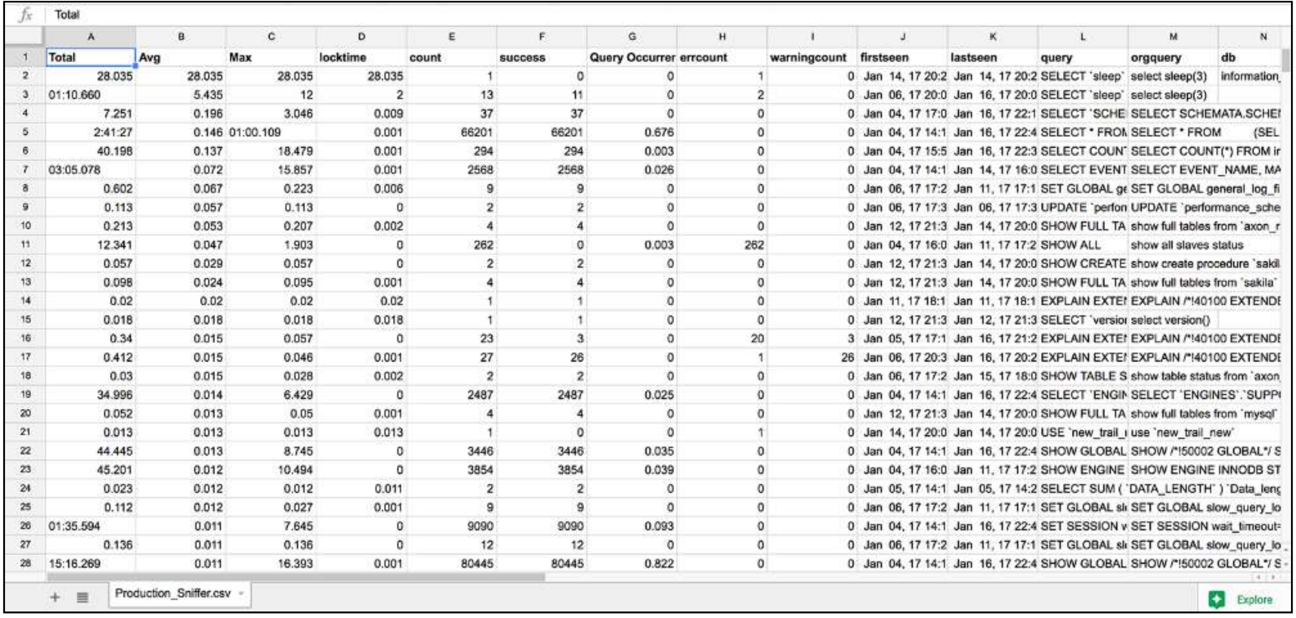
If **All** is selected in the list, is not considering any timeframe and just displays all queries within the specified size/ chunk. Also note, it is the smallest of those two settings that have effect for the analysis. For analyzing the sniffer pseudo log there is no chunk size to be defined as the complete pseudo log as stored in the SQL DM for MySQL database that is considered. The selected log chunk needs to have statements for the selected period. If not, then SQL DM for MySQL of course only display data from the first log record available.

Include User and Host Information: If this option is selected it displays the User and Host of that particular query and it groups the query analyzer table based on user@host[62] and query.

> ⚠ If SQL DM for MySQL is already installed in the machine and Sniffer is enabled, it only displays the User info in Query Analyzer table because old SQL DM for MySQL never used to store the Host information in sniffer.data table. Also note, that this option is not supported by MySQL Proxy. In General query log, if connect string is not included in the specified chunk, it is not display the user@host[63] information which is just left as an empty space.

4.7.1.3 Export As CSV

The option to define the field delimiter is provided because some localized Windows programs for LOCALEs where comma " , " is used as a decimal sign and requires a semicolon " ; " as field separator. This includes Microsoft Office programs (Excel and Access) and Microsoft text-ODBC driver on such localized Windows. On Linux, the situation is more non-uniform but also such localized OpenOffice Calc (spreadsheet) requires semicolon " ; " as field separator.



fx Total

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Total	Avg	Max	locktime	count	success	Query Occurrer	errcount	warningcount	firstseen	lastseen	query	orgquery	db
2	28.035	28.035	28.035	28.035	1	0	0	1	0	Jan 14, 17 20:2	Jan 14, 17 20:2	SELECT `sleep`	select sleep(3)	information_
3	01:10.660	5.435	12	2	13	11	0	2	0	Jan 06, 17 20:0	Jan 16, 17 20:0	SELECT `sleep`	select sleep(3)	
4	7.251	0.196	3.046	0.009	37	37	0	0	0	Jan 04, 17 17:0	Jan 16, 17 22:1	SELECT `SCHE	SELECT SCHEMATA.SCHE	
5	2:41:27	0.146	01:00.109	0.001	66201	66201	0.676	0	0	Jan 04, 17 14:1	Jan 16, 17 22:4	SELECT * FRO	SELECT * FROM (SEL	
6	40.198	0.137	18.479	0.001	294	294	0.003	0	0	Jan 04, 17 15:5	Jan 16, 17 22:3	SELECT COUN	SELECT COUNT(*) FROM ir	
7	03:05.078	0.072	15.857	0.001	2568	2568	0.026	0	0	Jan 04, 17 14:1	Jan 14, 17 16:0	SELECT EVENT	SELECT EVENT_NAME, MA	
8	0.602	0.067	0.223	0.006	9	9	0	0	0	Jan 06, 17 17:2	Jan 11, 17 17:1	SET GLOBAL ge	SET GLOBAL general_log_fi	
9	0.113	0.057	0.113	0	2	2	0	0	0	Jan 06, 17 17:3	Jan 06, 17 17:3	UPDATE `perfon	UPDATE `performance_sche	
10	0.213	0.053	0.207	0.002	4	4	0	0	0	Jan 12, 17 21:3	Jan 14, 17 20:0	SHOW FULL TA	show full tables from `axon_r	
11	12.341	0.047	1.903	0	262	0	0.003	262	0	Jan 04, 17 16:0	Jan 11, 17 17:2	SHOW ALL	show all slaves status	
12	0.057	0.029	0.057	0	2	2	0	0	0	Jan 12, 17 21:3	Jan 14, 17 20:0	SHOW CREATE	show create procedure `sakil	
13	0.098	0.024	0.095	0.001	4	4	0	0	0	Jan 12, 17 21:3	Jan 14, 17 20:0	SHOW FULL TA	show full tables from `sakila`	
14	0.02	0.02	0.02	0.02	1	1	0	0	0	Jan 11, 17 18:1	Jan 11, 17 18:1	EXPLAIN EXTE	EXPLAIN /*!40100 EXTEND	
15	0.018	0.018	0.018	0.018	1	1	0	0	0	Jan 12, 17 21:3	Jan 12, 17 21:3	SELECT `versio	select version()	
16	0.34	0.015	0.057	0	23	3	0	20	3	Jan 05, 17 17:1	Jan 16, 17 21:2	EXPLAIN EXTE	EXPLAIN /*!40100 EXTEND	
17	0.412	0.015	0.046	0.001	27	26	0	1	26	Jan 06, 17 20:3	Jan 16, 17 20:2	EXPLAIN EXTE	EXPLAIN /*!40100 EXTEND	
18	0.03	0.015	0.028	0.002	2	2	0	0	0	Jan 06, 17 17:2	Jan 15, 17 18:0	SHOW TABLE S	show table status from `axon	
19	34.996	0.014	6.429	0	2487	2487	0.025	0	0	Jan 04, 17 14:1	Jan 16, 17 22:4	SELECT `ENGIN	SELECT `ENGINES`.`SUPP	
20	0.052	0.013	0.05	0.001	4	4	0	0	0	Jan 12, 17 21:3	Jan 14, 17 20:0	SHOW FULL TA	show full tables from `mysql`	
21	0.013	0.013	0.013	0.013	1	0	0	1	0	Jan 14, 17 20:0	Jan 14, 17 20:0	USE `new_trail_	use `new_trail_new`	
22	44.445	0.013	8.745	0	3446	3446	0.035	0	0	Jan 04, 17 14:1	Jan 16, 17 22:4	SHOW GLOBAL	SHOW /*!50002 GLOBAL*/ S	
23	45.201	0.012	10.494	0	3854	3854	0.039	0	0	Jan 04, 17 16:0	Jan 11, 17 17:2	SHOW ENGINE	SHOW ENGINE INNODB ST	
24	0.023	0.012	0.012	0.011	2	2	0	0	0	Jan 05, 17 14:1	Jan 05, 17 14:2	SELECT SUM (`DATA_LENGTH`) `Data_leng		
25	0.112	0.012	0.027	0.001	9	9	0	0	0	Jan 06, 17 17:2	Jan 11, 17 17:1	SET GLOBAL sl	SET GLOBAL slow_query_lo	
26	01:35.594	0.011	7.645	0	9090	9090	0.093	0	0	Jan 04, 17 14:1	Jan 16, 17 22:4	SET SESSION v	SET SESSION wait_timeout=	
27	0.136	0.011	0.136	0	12	12	0	0	0	Jan 06, 17 17:2	Jan 11, 17 17:1	SET GLOBAL sl	SET GLOBAL slow_query_lo	
28	15:16.269	0.011	16.393	0.001	80445	80445	0.822	0	0	Jan 04, 17 14:1	Jan 16, 17 22:4	SHOW GLOBAL	SHOW /*!50002 GLOBAL*/ S	

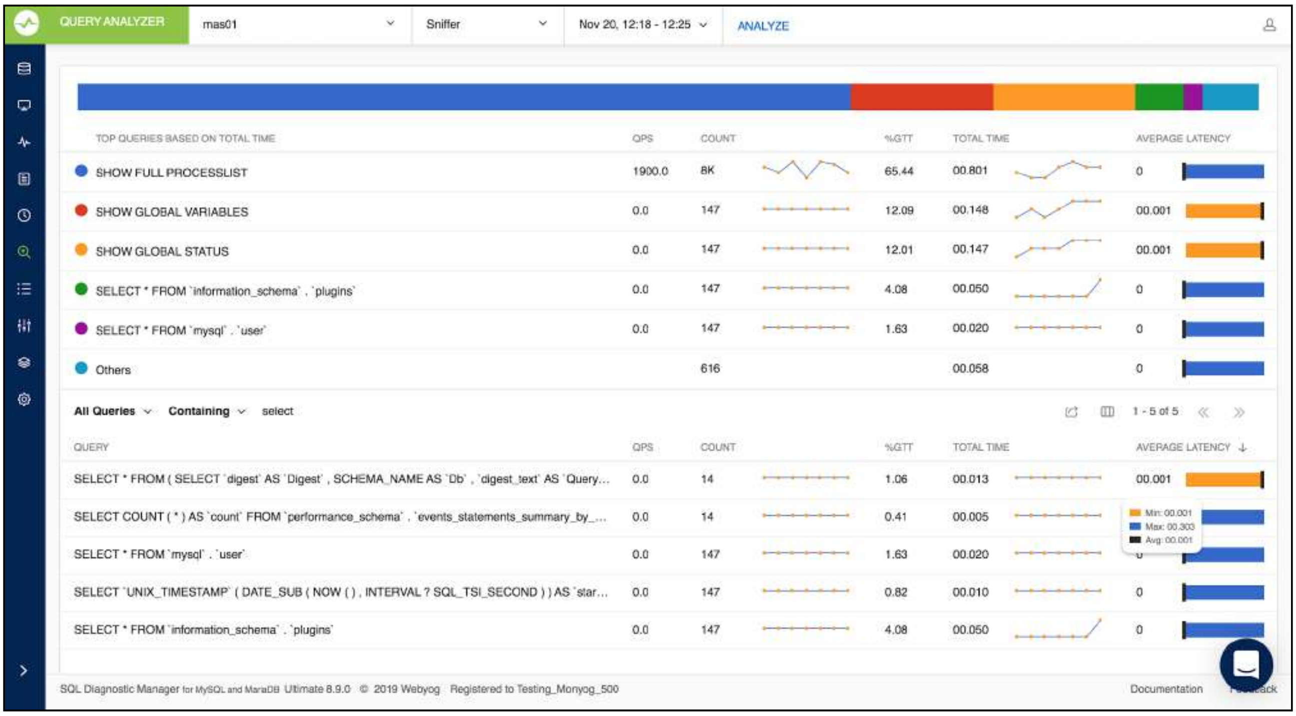
4.7.2 Sniffer

SQL DM for MySQL Query Sniffer is a functionality that records a pseudo server log and stores it in the SQL DM for MySQL embedded database. With **Query sniffer** enabled, SQL DM for MySQL can populate the pseudo server log in three different ways at the intervals you specify:

- By utilizing Performance Schema tables (`events_statements_summary_by_digest` , `events_statements_history_long`) and collecting snapshots at regular intervals.
- By sending the query `SHOW FULL PROCESSLIST` to the MySQL server.

[62] mailto:user@host
[63] mailto:user@host

- Or, by connecting to a running instance of the MySQL-Proxy program that is used by one or more clients to connect to a MySQL server.



TOP QUERIES BASED ON TOTAL TIME	QPS	COUNT	%GTT	TOTAL TIME	AVERAGE LATENCY
SHOW FULL PROCESSLIST	1900.0	8K	65.44	00.801	0
SHOW GLOBAL VARIABLES	0.0	147	12.09	00.148	00.001
SHOW GLOBAL STATUS	0.0	147	12.01	00.147	00.001
SELECT * FROM `information_schema` . `plugins`	0.0	147	4.08	00.050	0
SELECT * FROM `mysql` . `user`	0.0	147	1.63	00.020	0
Others		616		00.058	0

QUERY	QPS	COUNT	%GTT	TOTAL TIME	AVERAGE LATENCY ↓
SELECT * FROM (SELECT `digest` AS `Digest` , SCHEMA_NAME AS `Db` , `digest_text` AS `Query...	0.0	14	1.06	00.013	00.001
SELECT COUNT (*) AS `count` FROM `performance_schema` . `events_statements_summary_by_...	0.0	14	0.41	00.005	
SELECT * FROM `mysql` . `user`	0.0	147	1.63	00.020	0
SELECT `UNIX_TIMESTAMP` (DATE_SUB (NOW ( ) , INTERVAL ? SQL_TSI_SECOND)) AS `star...	0.0	147	0.82	00.010	0
SELECT * FROM `information_schema` . `plugins`	0.0	147	4.08	00.050	0

SQL Diagnostic Manager for MySQL and MariaDB Ultimate 8.9.0 © 2019 Webyog Registered to Testing_Monyog_500

For MySQL 5.6.14 and above you can use Performance schema, Proxy, and Processlist for query analysis. If using MySQL version less than 5.6.14 then only Proxy or Processlist can be used in SQL DM for MySQL.

Performance Schema on MySQL contains queries executed on server along with other information:

- Number of rows sent and examined
- Number of temporary tables created on disk
- Number of temporary tables created on memory
- Number of joins performed and the type of join
- Whether sorting happened and the type of sort
- Whether index used
- Whether good index used

SQL DM for MySQL uses performance schema statement digest `table(events_statements_summary_by_digest)` to get the above information and is dependent on the statements_digest in setup_consumers table. By default, this option is enabled, you can disable it by executing the following:

```
UPDATE performance_schema.setup_consumers
SET enabled = 'YES'
WHERE name = 'statements_digest';
```

Example query is available in `events_statements_history_long` table and has to be enabled and is dependent on the `events_statements_history_long` in setup_consumers table. By default, this is not enabled and should be enabled by executing the following:

```
UPDATE performance_schema.setup_consumers
SET enabled = 'YES'
WHERE name = 'events_statements_history_long';
```

The `performance_schema.events_statements_summary_by_digest` table size is dependent on `performance_schema_digests_size global` variable. By default, this size is set to 5000 rows. Once it reaches this limit you may lose the queries. SQL DM for MySQL provides an option to truncate the performance schema digest table when it reaches 80% of `performance_schema_digests_size`.

Performance schema based sniffer comes with different filters like: Queries with errors, Queries with warnings, Queries with missing indexes, and Queries with poor indexes.

> ⚠ `performance_schema` truncates queries after 1024 characters and always replaces literals with a wildcard (in other words: P_S contains a summary/an aggregation only). So query listing not replacing literals is not possible with this option. And finally also observe that no other tool (or user) should be writing (including deleting or truncating) to `events_statements_summary_by_digest` and `events_statements_history_long` tables if this option is used as there is only one of each table for all users (it is not a temporary table or a materialized view or similar private for the user). This is a design limitation with the tables in P_S as such and not a Monyog issue.

If using MySQL version less than 5.6.14 then only Proxy or Processlist can be used in SQL DM for MySQL. Although, configuring a Proxy instance is a little more complicated, the PROXY-based sniffer has several advantages over the PROCESSLIST-based, including the following:

1. All queries that was handled by the Proxy are recorded by SQL DM for MySQL sniffer when PROXY option is used. When PROCESSLIST option is used very fast queries may execute completely between two SHOW FULL PROCESSLIST queries andis not recorded.
2. You can choose to analyze queries from specific client(s)/application(s) only. Simply let (only) the clients that you want to focus on at the moment connect through the Proxy.
3. When using the PROXY option you can distribute part of the load generated by the sniffer on the machine that fits best in your deployment scenario (like on the one that has most free resources available) by deciding where to have the PROXY: The MySQL machine, the SQL DM for MySQL machine (if not the same), or quite another machine. The machine running MySQL have no additional load due to the sniffer if the Proxy is not running on that machine.

Also, if more SQL DM for MySQL instances use the same PROXY they use the same data collected, when the Proxy Sniffing is enabled by the first SQL DM for MySQL instance. To work with SQL DM for MySQL sniffer the MySQL Proxy instance must be started with the name of a LUA script called **MONyog.LUA** (LUA is a scripting/programming language) as argument and is distributed with SQL DM for MySQL. You find it in the Monyog program folder after installing (Windows and Linux RPM) or unpacking (Linux .tar.gz) the SQL DM for MySQL program package as downloaded from the IDERA website. The MySQL Proxy program however you need to download from MySQL website (we cannot include it for license reasons). SQL DM for MySQL works with Proxy versions from 0.61 to 0.81(latest currently) with the exception of 0.7x versions for windows and Mac due to a bug in those specific builds. For more information on Proxy MySQL Proxy(see page 221).

To start a Proxy instance for use with SQL DM for MySQL use the command:

- **For Older version:**

```
Proxy installation folder>mysql-proxy --proxy-backend-addresses=192.168.x.x:3305 --
proxy-address=192.168.y.y:4045 --proxy-lua-script=SQL DM for MySQL.lua
```

- **For v0.81 and later:**

```
Proxy installation folder>mysql-proxy --proxy-backend-addresses=192.168.x.x:3305 --
proxy-address=192.168.y.y:4045 --admin-username=root --admin-password=root --admin-
lua-script=MONyog.lua --proxy-lua-script=MONyog.lua
```

(It is assumed that the 'MONyog.LUA' was copied to the folder where the PROXY binary is). Also, if no port is specified the PROXY listens on port 4040. Now, you can connect to the Proxy from one or more clients/applications. The Proxy sends queries to MySQL and results back to the client. But when started with the LUA script for SQL DM for MySQL sniffer it also send information to SQL DM for MySQL that SQL DM for MySQL uses to populate the sniffer 'pseudo log'.

Once this 'pseudo log' has been recorded (in either of the two ways described: PROCESSLIST or PROXY-based) the SQL DM for MySQL log analysis functionalities can operate on the 'pseudo log' as well as the 'real logs'. The data recorded in the pseudo log is purged automatically based on the 'data retention timeframe' option set by you.

Further some filtering options are provided. This filtering happens before storing to the SQL DM for MySQL database. It prevents the sniffer database to grow out of control. The filtering options are as follows:

- **User and host**: You can choose to store queries executed only by a specific combination of users and/or hosts.
- **Minimum time taken:** For every PROCESSLIST returned to SQL DM for MySQL, the queries are recorded in the embedded database only if they have been executing for a time greater than the specified minimum execution time. Furthermore, if a query with the same structure and details (like process ID) as one already recorded is encountered, the embedded database is UPDATED, and the statement is recorded only once. This setting should be somewhat larger than the sample interval (and also consider the latency of the connection).
- **Queries starting with:** Enter any string and only queries starting with that string is recorded. Examples: `SELECT *`, `UPDATE Customer_Base`.

Also note, in the `PROCESSLIST` Sniffer there is an option 'Long Running Query Options' where you can monitor the long running queries by notifying or killing a query which takes more than a time specified by you. You can also specify users whose queries are ignored (i.e. queries) by such user are not killed by SQL DM for MySQL and never raise an alert even if they take a longer time to execute than the alert/kill setting time you specified.

Clicking the **monitor only locked queries** would only monitor those long queries that are locked.

> ⚠ Note that the query sniffer is never a complete General Log. Very fast statements may or may not be recorded as they may or may not finish executing between two `PROCESSLIST` 's generated. The time interval between subsequent data collections for the 'pseudo log' depends on the connection to the MySQL server.

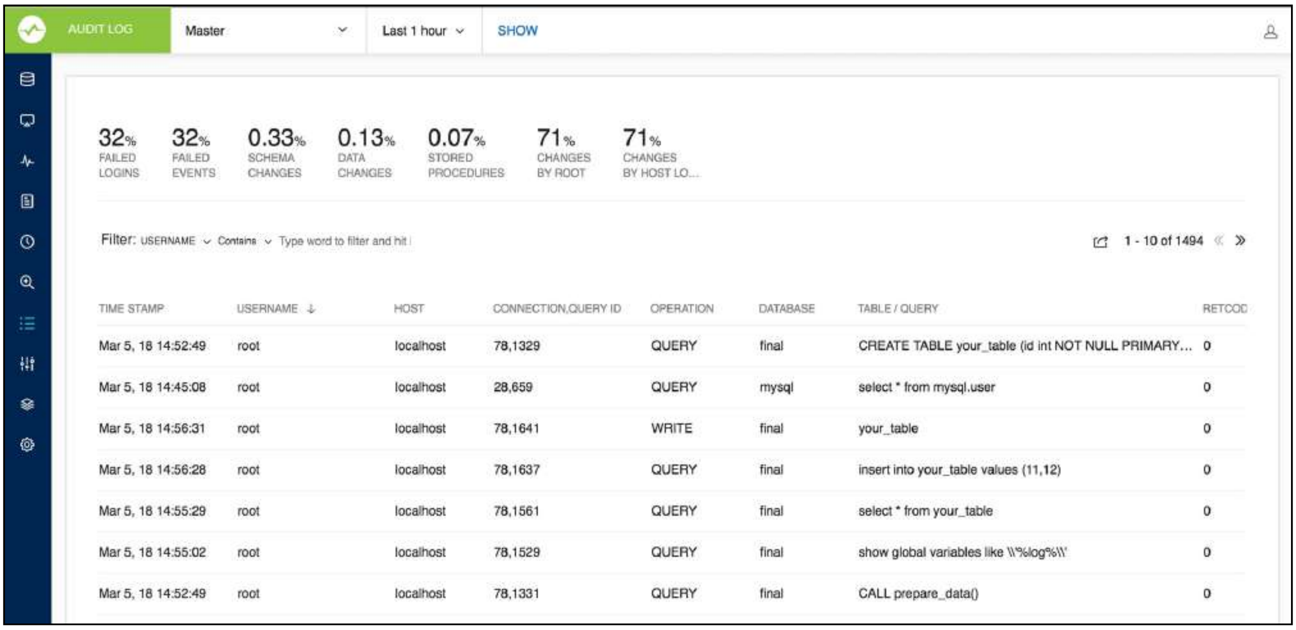
The identical queries are only listed once and the Count column tells how many times this query was executed.

4.8 Audit Log

This feature parses the audit log maintained by the server and displays the content in clean tabular format. SQL DM for MySQL supports both MySQL Enterprise and MariaDB audit logs.

SQL DM for MySQL accesses the audit log file, the same way it does for other MySQL log files: Slow Query, General Query, and Error log. Please review Advanced Settings[64] to learn how to enable the audit log on your server, and configure SQL DM for MySQL.

After selecting the server and the time-frame for which you want the audit log to be seen from, click **SHOW AUDIT LOG** to get the content of the log. The limit on the number of rows which can be fetched in one time-frame is 10000 (This limit is calculated from the beginning of the audit log), so in case you have more than 10000 entries in your audit log for the selected time frame, then the remaining entries are not displayed.



32%	32%	0.33%	0.13%	0.07%	71%	71%
FAILED LOGINS	FAILED EVENTS	SCHEMA CHANGES	DATA CHANGES	STORED PROCEDURES	CHANGES BY ROOT	CHANGES BY HOST LO...

Filter: USERNAME Contains Type word to filter and hit

1 - 10 of 1494

TIME STAMP	USERNAME ↓	HOST	CONNECTION,QUERY ID	OPERATION	DATABASE	TABLE / QUERY	RETCOD
Mar 5, 18 14:52:49	root	localhost	78,1329	QUERY	final	CREATE TABLE your_table (id int NOT NULL PRIMARY...	0
Mar 5, 18 14:45:08	root	localhost	28,659	QUERY	mysql	select * from mysql.user	0
Mar 5, 18 14:56:31	root	localhost	78,1641	WRITE	final	your_table	0
Mar 5, 18 14:56:28	root	localhost	78,1637	QUERY	final	insert into your_table values (11,12)	0
Mar 5, 18 14:55:29	root	localhost	78,1561	QUERY	final	select * from your_table	0
Mar 5, 18 14:55:02	root	localhost	78,1529	QUERY	final	show global variables like \\'%log%\\'	0
Mar 5, 18 14:52:49	root	localhost	78,1331	QUERY	final	CALL prepare_data()	0

The section on the top gives you quick summary of the audit log in percentage like Failed Logins, Failed Events, Schema changes, Data Changes, and Stored Procedure. All these legends are clickable and shows the corresponding audit log entries.

Furthermore, you can use the filter option to fetch audit log based on Username, Host, Operation, Database, and Table/Query, use the Contains or Does not contain options to have more specific Audit log results.

> ⓘ Export the fetched audit log content in CSV format.

4.9 Server Configuration

Maintaining server configuration and tracking changes, plays a vital role in the maintenance of MySQL servers. DBAs may be responsible for hundreds of servers and keeping an eye on the configuration settings for all of them could be difficult to say the least.

[64] http://wiki.idera.com/display/SQLDMYSQL/Advanced+Settings

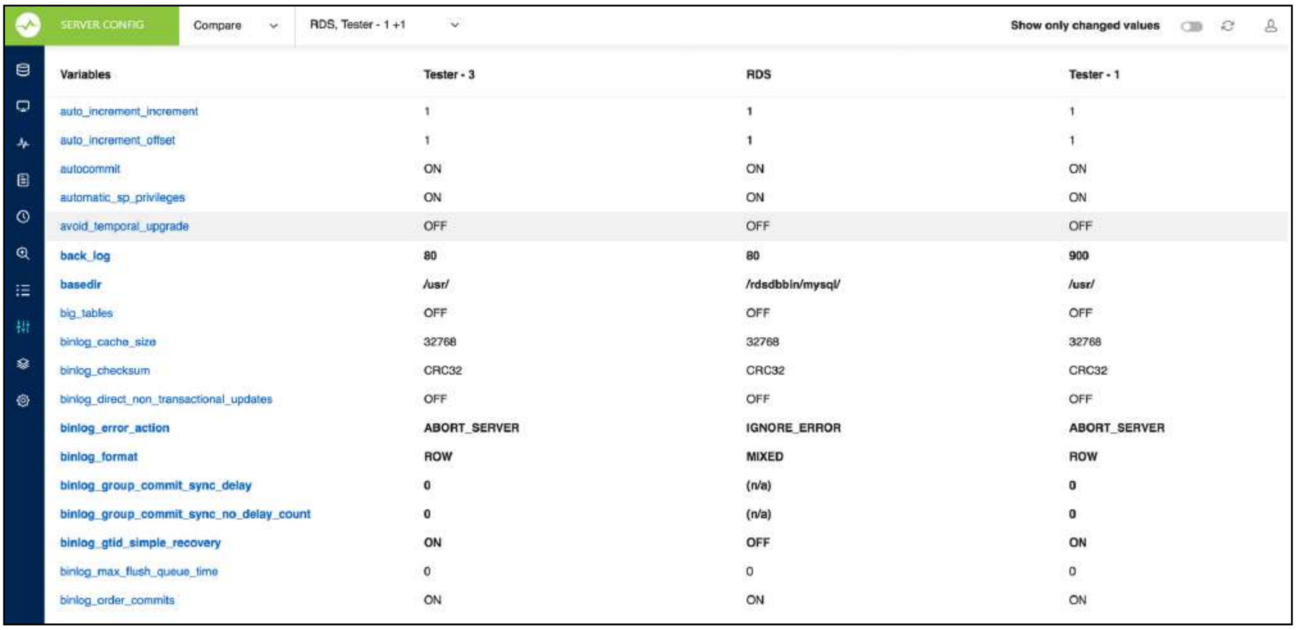
Server Config allows you to compare MySQL configurations of multiple servers side-by-side, with all changes highlighted so that differences are visually discernible at a glance.

> ✓ Wondering why server A is not performing as well as server B when they share the exact same load? The answer could lie in the configuration files.

Server Config also allows you to track changes to your server configuration files over a period of time. Now you are in full control of what goes into those files and the impact they have on your MySQL server

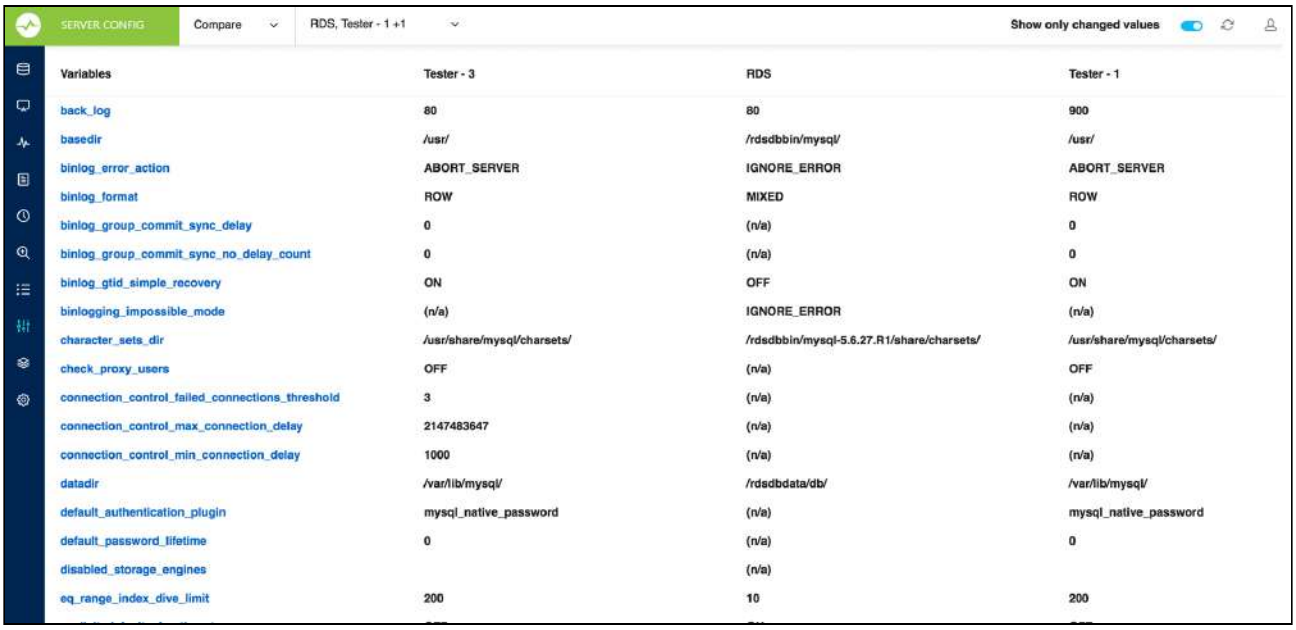
4.9.1 Compare Configuration

Compare Configuration allows you to compare MySQL Global Variables of multiple servers side-by-side. This comes in handy when trying to determine why one MySQL server is not performing as well as another one with similar conditions of load. Differences in the Global Variables are highlighted.



Variables	Tester - 3	RDS	Tester - 1
auto_increment_increment	1	1	1
auto_increment_offset	1	1	1
autocommit	ON	ON	ON
automatic_sp_privileges	ON	ON	ON
avoid_temporal_upgrade	OFF	OFF	OFF
back_log	**80**	**80**	**900**
basedir	**/usr/**	**/rdsdbbin/mysql/**	**/usr/**
big_tables	OFF	OFF	OFF
binlog_cache_size	32768	32768	32768
binlog_checksum	CRC32	CRC32	CRC32
binlog_direct_non_transactional_updates	OFF	OFF	OFF
binlog_error_action	**ABORT_SERVER**	**IGNORE_ERROR**	**ABORT_SERVER**
binlog_format	**ROW**	**MIXED**	**ROW**
binlog_group_commit_sync_delay	**0**	**(n/a)**	**0**
binlog_group_commit_sync_no_delay_count	**0**	**(n/a)**	**0**
binlog_gtid_simple_recovery	**ON**	**OFF**	**ON**
binlog_max_flush_queue_time	0	0	0
binlog_order_commits	ON	ON	ON

To view only changed values, enable the **Show only changed values** toggle-bar on the tool bar at the top.

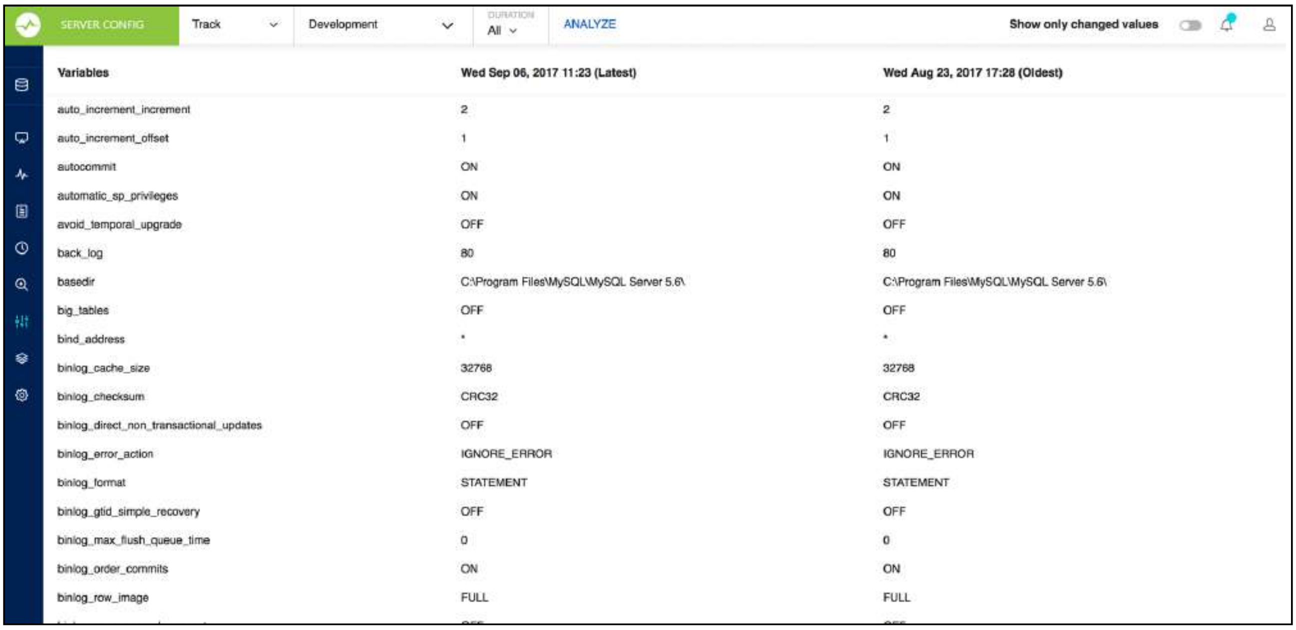


Variables	Tester - 3	RDS	Tester - 1
back_log	80	80	900
basedir	/usr/	/rdsdbbin/mysql/	/usr/
binlog_error_action	ABORT_SERVER	IGNORE_ERROR	ABORT_SERVER
binlog_format	ROW	MIXED	ROW
binlog_group_commit_sync_delay	0	(n/a)	0
binlog_group_commit_sync_no_delay_count	0	(n/a)	0
binlog_gtid_simple_recovery	ON	OFF	ON
binlogging_impossible_mode	(n/a)	IGNORE_ERROR	(n/a)
character_sets_dir	/usr/share/mysql/charsets/	/rdsdbbin/mysql-5.6.27.R1/share/charsets/	/usr/share/mysql/charsets/
check_proxy_users	OFF	(n/a)	OFF
connection_control_failed_connections_threshold	3	(n/a)	(n/a)
connection_control_max_connection_delay	2147483647	(n/a)	(n/a)
connection_control_min_connection_delay	1000	(n/a)	(n/a)
datadir	/var/lib/mysql/	/rdsdbdata/db/	/var/lib/mysql/
default_authentication_plugin	mysql_native_password	(n/a)	mysql_native_password
default_password_lifetime	0	(n/a)	0
disabled_storage_engines		(n/a)	
eq_range_index_dive_limit	200	10	200

4.9.2 Track Configuration Changes

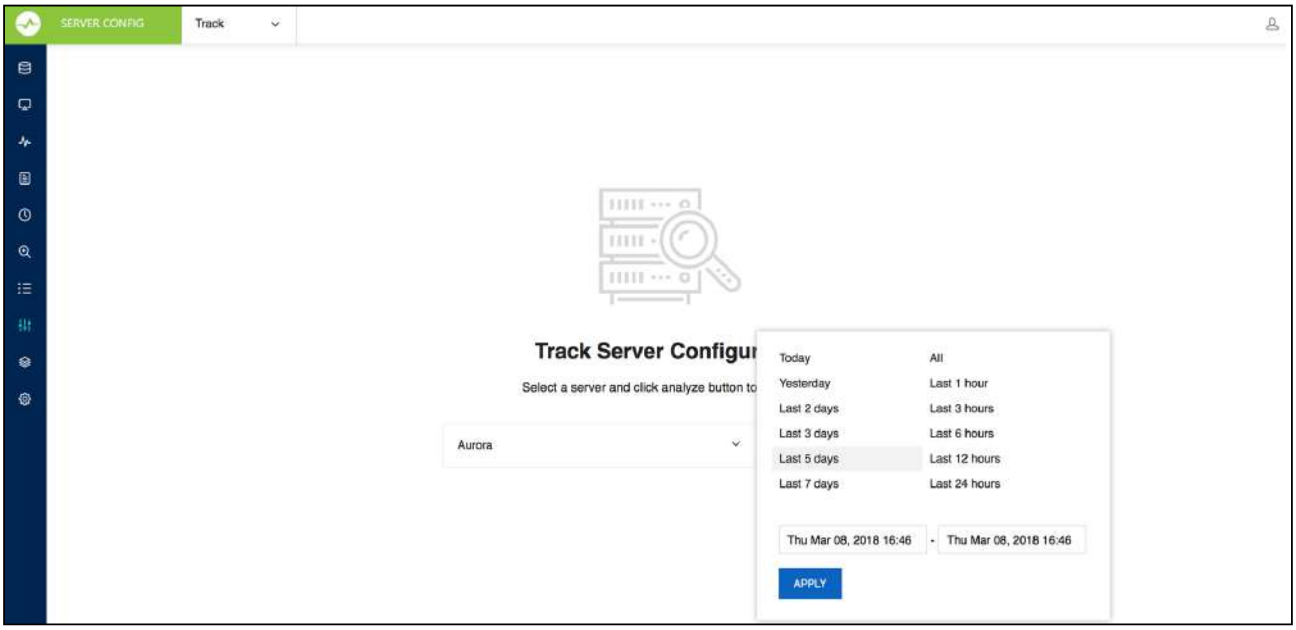
Track Configuration Changes is version control for your MySQL server Global Variables. SQL DM for MySQL tracks changes to global configuration no matter if the configuration parameters were specified in my.ini/my.cnf, are server defaults or if somebody with SUPER privilege has executed a SET GLOBAL statement. If you notice your MySQL server performance degraded from the last time you checked, you can track, and compare changes in the configuration file to determine whether a change in the Global Variables has brought about this degradation.

To view data for a server, select the server from the drop-down menu and click **ANALYZE**. All timestamps when changes were detected are listed in the drop-down with the latest timestamp and oldest timestamp compared by default.



Variables	Wed Sep 06, 2017 11:23 (Latest)	Wed Aug 23, 2017 17:28 (Oldest)
auto_increment_increment	2	2
auto_increment_offset	1	1
autocommit	ON	ON
automatic_sp_privileges	ON	ON
avoid_temporal_upgrade	OFF	OFF
back_log	80	80
basedir	C:\Program Files\MySQL\MySQL Server 5.6\	C:\Program Files\MySQL\MySQL Server 5.6\
big_tables	OFF	OFF
bind_address	*	*
binlog_cache_size	32768	32768
binlog_checksum	CRC32	CRC32
binlog_direct_non_transactional_updates	OFF	OFF
binlog_error_action	IGNORE_ERROR	IGNORE_ERROR
binlog_format	STATEMENT	STATEMENT
binlog_gtid_simple_recovery	OFF	OFF
binlog_max_flush_queue_time	0	0
binlog_order_commits	ON	ON
binlog_row_image	FULL	FULL

The time-frame can be changed and the respective changes to the configuration are displayed. To track changes between two timestamps select the original timestamp on the left drop-down menu. Next, select the timestamp to compare with from the right drop-down menu, enable the **Show only changed values** toggle-bar, which hides all values that have not changed between the two timestamps.



4.10 Replication

This interface shows the replication graph and relationship of all registered MySQL servers, as well as `SLAVE STATUS` and `MASTER STATUS` where it applies on hovering over each of the server block. The display gets updated after every one minute.

The servers are color-coded to represent the different states of the replication servers: green denotes stable, red disconnected servers and yellow not in sync. The yellow color is decided if any of the following variables return the below value:

```
'SLAVE IO RUNNING' = No
```

or

```
'SLAVE SQL RUNNING' = No
```

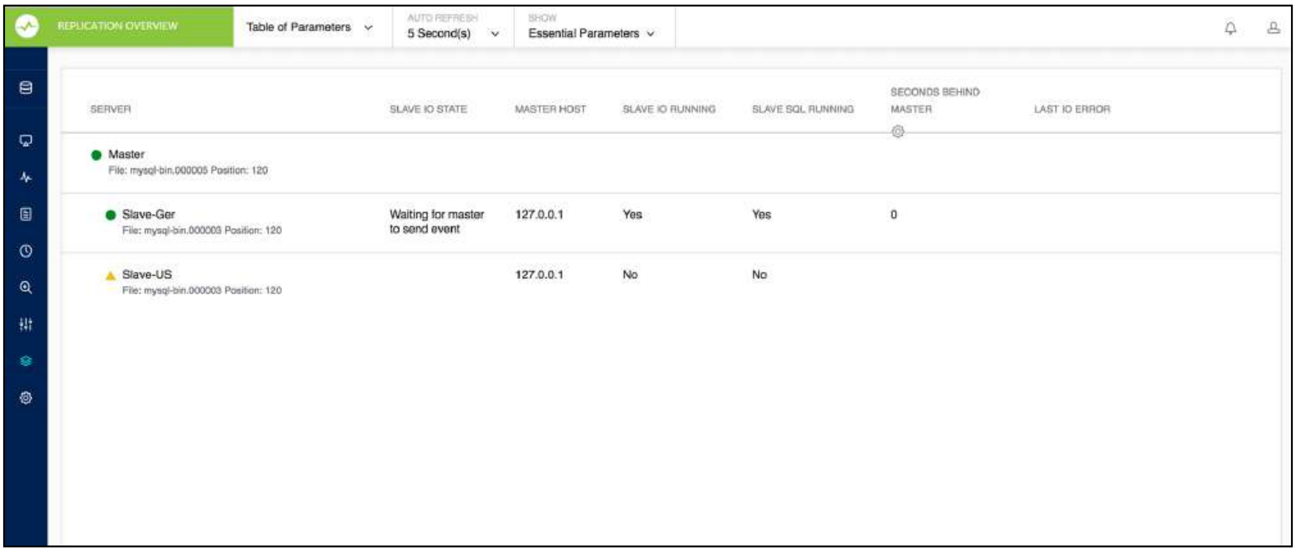
or

```
'SECONDS BEHIND MASTER' >= 450 Secs (this is the default global value for all the
servers. A user can also set a value according to their MySQL environment)
```

From Monyog 8.1.0, the user can set the **Seconds Behind Master** threshold value for all the servers in SQL DM for MySQL by clicking the icon in the **Seconds Behind Master** column or in the dialog pop-up upon clicking the respective server in the table/topology.

There are two interfaces that the user can choose to see the information about the replication setup:

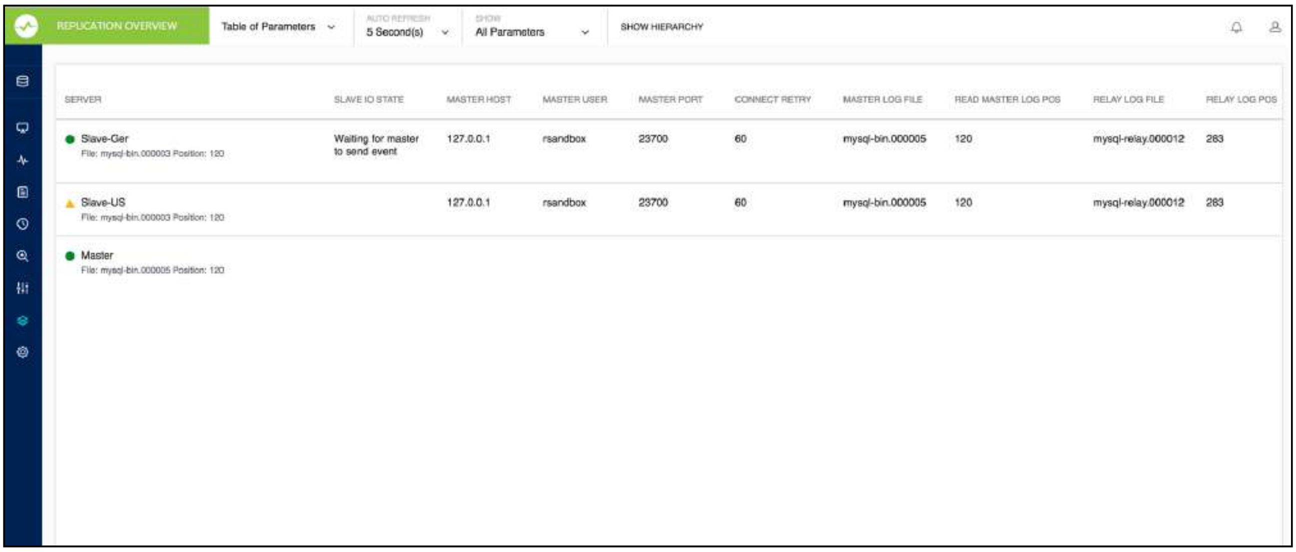
- **Table of Parameters:** This table contains the result set of SHOW [ALL] SLAVE STATUS for servers which have been marked as replication slaves in SQL DM for MySQL. The first time the number of columns in the table is condensed to only display 'Essential Parameters'. To get the detailed view, you can select the option ‘All Parameters’ from the drop-down menu in the header.



SERVER	SLAVE IO STATE	MASTER HOST	SLAVE IO RUNNING	SLAVE SQL RUNNING	SECONDS BEHIND MASTER	LAST IO ERROR
● Master File: mysql-bin.000005 Position: 120						
● Slave-Ger File: mysql-bin.000003 Position: 120	Waiting for master to send event	127.0.0.1	Yes	Yes	0	
▲ Slave-US File: mysql-bin.000003 Position: 120		127.0.0.1	No	No		

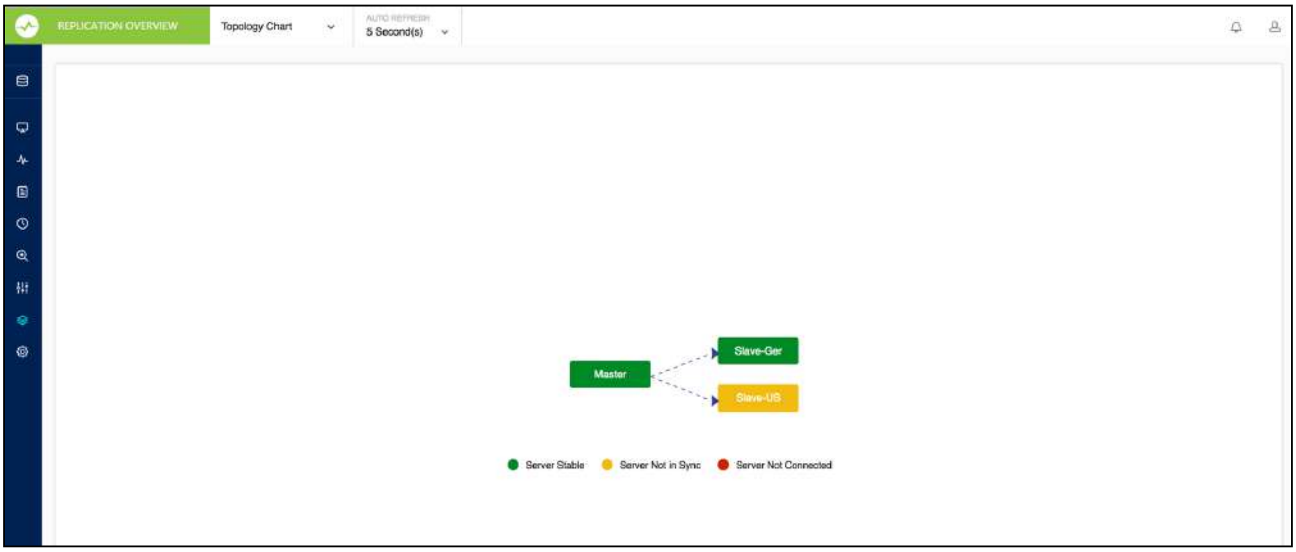
You can sort any column which removes the hierarchy listing of servers. To view slaves in the hierarchy, click the **SHOW HIERARCHY** link.

Also, if the I/O thread (‘Slave_IO_Running’) is not running, it dynamically brings out the Last IO Error column for a quick view on what went wrong with the Slave IO Thread. It works similarly for SQL thread (‘Slave_SQL_Running’) as well.



SERVER	SLAVE IO STATE	MASTER HOST	MASTER USER	MASTER PORT	CONNECT RETRY	MASTER LOG FILE	READ MASTER LOG POS	RELAY LOG FILE	RELAY LOG POS
● Slave-Ger File: mysql-bin.000003 Position: 120	Waiting for master to send event	127.0.0.1	rsandbox	23700	60	mysql-bin.000005	120	mysql-relay.000012	283
▲ Slave-US File: mysql-bin.000003 Position: 120		127.0.0.1	rsandbox	23700	60	mysql-bin.000005	120	mysql-relay.000012	283
● Master File: mysql-bin.000005 Position: 120									

- **Topology Chart:** This interface shows the replication graph and relationship of all registered MySQL servers which are marked as master/slave, as well as SLAVE STATUS and MASTER STATUS while hovering over each of the server blocks.The display gets updated at the user-specified refresh interval which is a browser-specific setting. The default interval is 5 seconds.



The server blocks on clicking gives the result set of SHOW [ALL] SLAVE STATUS for the slave and SHOW MASTER STATUS for the master servers.

5 SQL DM for MySQL Database Schema

Database Schema reveals the SQLite schema details which are used for managing SQL DM for MySQL connections (copying connections/creating new connections etc.). Also, with this information you get the basic understanding of how data is stored so that you get to query the database for information without using SQL DM for MySQL.

> If multiple applications use the same database one issue `LOCKS` that cause the other to wait, so generally suggested that you use a copy of the database.

5.1 What is SQL DM for MySQL Data?

SQL DM for MySQL uses SQLite for storing all the data. With 'data' - we are referring to:

- Data collected from MySQL servers (stored in the database file mysql.data).
- Data collected from the operating system (currently available for Linux only, stored in the database file system.data).
- Data captured from 'sniffing' (stored in the database file sniffer.data).
- Data captured from 'CSO's (stored in database file udo.data).
- Data captured for events (stored in events.data).
- Data captured when a Real-Time session is started and saved (stored in the realtimedata folder: rt_name.data).
- Data related to MySQL server connection (stored in connection.data).
- Also, for faster retrieval in future, SQL DM for MySQL stores the cached data (stored in cache.data).

There is one of each of those database files for every connection except for events.data that is common to all the servers.

5.1.1 Where can you find this data?

Below, you can find the default paths for the data collected by SQL DM for MySQL for the first connection created by SQL DM for MySQL:

5.1.1.1 In windows systems

```
C:\ProgramData\Webyog\MONyog\Data\0001
```

5.1.1.2 In Linux systems

```
- RPM: /usr/local/MONyog/data/0001
- Tar: In the same directory where MONyog was 'untarred'.
```

5.1.2 How to view existing schema and data?

You can view the existing schema and data by using a SQLite client. In addition to the official SQLite command-line client there are simple GUI clients available SQLite Manager (see page 231) (This is a plugin for the Firefox browser and works on all platforms, but there are more GUI clients available for download - mostly for Windows. Most Linux distributions ship with some database client software that handles SQLite).

> Schemas may be subject to change. We may add/remove columns, change data types, change indexes etc. with new releases. When we do that, you can check-in release notes and you can open the database with the tools mentioned to see the columns.

5.2 Preferences and Connection Settings

5.2.1 Preferences and global settings

In addition to the data stored on a per-server basis; SQL DM for MySQL has a database storing global user preferences (preferences.config database) and also a very tiny text file (MONyog.ini), that only has what minimal information is required for SQL DM for MySQL to start.

5.2.1.1 connection.data

There is a Preferences table in connection.data which is used for storing the default processlist query.

```
CREATE TABLE IF NOT EXISTS [preferences] (
[name] VARCHAR(50) DEFAULT '' NOT NULL PRIMARY KEY UNIQUE,
[value] TEXT DEFAULT '');
```

The server_names table is used for storing all the connection details.

```
CREATE TABLE [server_names]
(id INTEGER DEFAULT 0 NOT NULL PRIMARY KEY UNIQUE,
key VARCHAR(255) DEFAULT '' NOT NULL,
value VARCHAR(255) DEFAULT '' NOT NULL);
```

5.3 MySQL variables and System data

5.3.1 **mysql.data and system.data**

These two databases have a completely identical structure:

```
CREATE TABLE IF NOT EXISTS [metric_master] (
[metric_id] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
[metric_desc] TEXT ASC UNIQUE )
```

```
CREATE TABLE IF NOT EXISTS [snapshot_master] (
[timestamp_id] INTEGER NOT NULL,
[metric_id] INTEGER NOT NULL,
[metric_now] TEXT,
[metric_diff] TEXT,
PRIMARY KEY (metric_id, timestamp_id))
```

```
CREATE INDEX IF NOT EXISTS [timestamp_id_index] ON [snapshot_master] ([timestamp_id])
```

```
CREATE TABLE IF NOT EXISTS [timestamp_master] (
[timestamp_id] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
[server_timestamp] INTEGER,
[server_start_time] INTEGER,
[server_uptime] INTEGER,
[server_uptime_diff] INTEGER,
[server_is_connected] INTEGER)
```

```
CREATE INDEX IF NOT EXISTS [server_timestamp_idx] ON [timestamp_master]
([server_timestamp])
```

What is most important to understand here is the [timestamp_id] column occurring in both [snapshot_master] and [timestamp_master] tables. Actually, with MySQL and InnoDB you would probably create a Foreign Key from [snapshot_master] to [timestamp_master] for constraining and clarity. To get meaningful results you need to `JOIN` the two in the query or use a `SUBQUERY`.

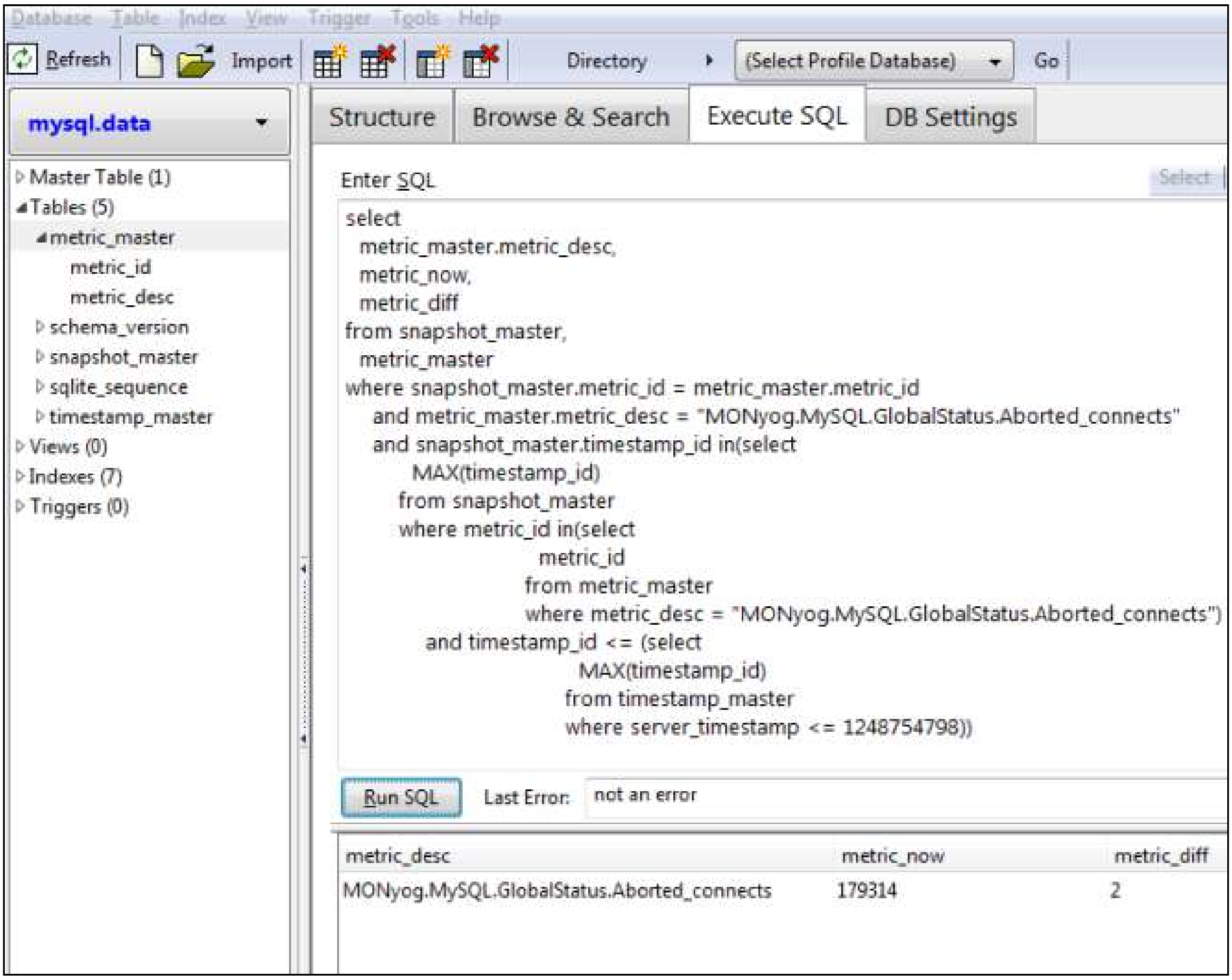
This is basically how they work:

- Everytime SQL DM for MySQL sends a statement like `SHOW VARIABLES/STATUS` of some kind (or fetching a OS metric from Linux/proc folder) one row is INSERTED into [timestamp_master] table with information about current time and the metrics retrieved for each such statement is INSERTED into [snapshot_master]. The [snapshot_master] table contains the metrics details. The [timestamp_id] column identifies when metric details were like that. Also, that timestamps in SQL DM for MySQL databases are unix_timestamps.
- And actually, we do not always `INSERT` into [snapshot_master]. Only if the particular metric was changed since last time something was INSERTED for that metric we will INSERT again. So if you want to find the value of a metric at some particular time you need to find the most recent value stored before that particular time for that particular metric.
- Finally, the [snapshot_master] does not have the names of the metrics. It is not possible to know in advance what metrics the server returns as it depends on server details (version and configuration). And actually a server may be upgraded.
  Saving each textual description only once which would save disk space. So [snapshot_master] only contains a number in the [metric_id] column referring to the textual description the [metric_master] table. So, if the query returns the name of the metric or the metric name should be used in a WHERE-clause also

[metric_master] table must be referenced in the query. If you are familiar with the SHOW statements and what information they return you identify the meaning of each row in [metric_master] table.

> The term Metric here refers to the discrete values returned for `SHOW` statements themselves (SHOW GLOBAL VARIABLES; SHOW GLOBAL STATUS; SHOW SLAVE STATUS etc.). Whatever calculations SQL DM for MySQL does in its web interface are done after and not before storage. But we do one calculation before storing however: whenever a metric is INSERTED we also retrieve that latest stored value for the same metric and calculate the difference. Both the current value and this difference is stored (in [metric_now] and [metric_diff] columns respectively).

An example of an easily understandable query doing all this could look like:



```
select
  metric_master.metric_desc,
  metric_now,
  metric_diff
from snapshot_master,
  metric_master
where snapshot_master.metric_id = metric_master.metric_id
    and metric_master.metric_desc = "MONyog.MySQL.GlobalStatus.Aborted_connects"
    and snapshot_master.timestamp_id in(select
          MAX(timestamp_id)
        from snapshot_master
        where metric_id in(select
                    metric_id
                  from metric_master
                  where metric_desc = "MONyog.MySQL.GlobalStatus.Aborted_connects")
          and timestamp_id <= (select
                      MAX(timestamp_id)
                    from timestamp_master
                    where server_timestamp <= 1248754798))
```

Last Error: not an error

metric_desc	metric_now	metric_diff
MONyog.MySQL.GlobalStatus.Aborted_connects	179314	2

An example of a query that we actually execute (optimized for large SQLite databases) to populate a graph is as follows:

```
SELECT metric_now
FROM snapshot_master
WHERE snapshot_master.metric_id = my_metric_id
AND snapshot_master.timestamp_id IN(
SELECT MAX(timestamp_id)
FROM snapshot_master
WHERE metric_id = my_metric_id
AND timestamp_id <= (
SELECT MAX(timestamp_id)
FROM timestamp_master
WHERE server_timestamp <= my_metric_timestamp)
)
```

Actually SQLite support has recommended using SUBQUERIES and not JOINS in most cases with SQLite for best performance with big databases. That is also the experience we have ourselves when profiling different queries returning same results.

5.3.2 udo.data

This is the database where we store data from the Custom SQL Objects(CSOs). This database has 3 different tables- Snapshot_master, Column_master and timestamp_master

```
CREATE TABLE [column_master]
(id INTEGER NOT NULL PRIMARY KEY,
timestamp_id INTEGER NOT NULL,
udo_id INTEGER NOT NULL,
key_column_value VARCHAR(255),
column VARCHAR(255),
UNIQUE([udo_id], [key_column_value], [column], [timestamp_id]));
```

```
CREATE TABLE [snapshot_master] ([timestamp_id] INTEGER NOT NULL,
[metric_id] INTEGER NOT NULL,
[metric_now] TEXT,
[metric_diff] TEXT,
PRIMARY KEY (metric_id, timestamp_id));
```

```
CREATE TABLE [timestamp_master] (
[timestamp_id] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
[server_timestamp] INTEGER,
[server_start_time] INTEGER,
[server_uptime] INTEGER,
[server_uptime_diff] INTEGER,
[server_is_connected] INTEGER, [udo_id] INTEGER);
```

5.3.3 events.data

This database holds information related to events in SQL DM for MySQL.

```
CREATE TABLE [events] (id INTEGER PRIMARY KEY AUTOINCREMENT,
first_seen INTEGER(5) NOT NULL,
last_seen INTEGER(5) NOT NULL,
down_count INTEGER(5) NOT NULL DEFAULT 0,
server_id VARCHAR(255) NOT NULL,
server_name VARCHAR(255) NOT NULL,
group_id INTEGER(5) NOT NULL DEFAULT 0,
counter_id INTEGER(5) NOT NULL DEFAULT 0,
grp VARCHAR(255),
name VARCHAR(255),
sampling_time_frame VARCHAR(255),
type INTEGER NOT NULL DEFAULT 0,
threshold VARCHAR(255),
value VARCHAR(255),
advice TEXT,
mail_alert VARCHAR(10) NOT NULL DEFAULT '',
down_count_override INTEGER(5) NOT NULL DEFAULT 0,
notify_stable_override VARCHAR(10) NOT NULL DEFAULT '',
smtp_alert_cnt INTEGER(5) NOT NULL DEFAULT 0,
snmp_alert_cnt INTEGER(5) NOT NULL DEFAULT 0,
ignored_timestamp INTEGER NOT NULL DEFAULT 0);
```

```
CREATE TABLE [timestamp_master](
[timestamp_id] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
[server_timestamp] INTEGER UNIQUE);
```

5.4 Query Analyzer and Processlist data

5.4.1 sniffer.data

```
CREATE TABLE IF NOT EXISTS [query_master](
[id] INTEGER PRIMARY KEY AUTOINCREMENT,
[query] TEXT,
UNIQUE([query]))

CREATE TABLE IF NOT EXISTS [query_snapshot] (
[pkeyid] INTEGER PRIMARY KEY AUTOINCREMENT,
[id] INTEGER,
[threadid] INTEGER,
[user] TEXT,
[querytime] INTEGER,
[uptime] INTEGER,
[host] TEXT DEFAULT)
```

Here, you see the same pattern as above: the [id] column in the [query_snapshot] table identifies a row in the [query_master] where the actual/textual query is saved. Also note, that a UNIQUE KEY is defined on the [query] column so that we can use an INSERT ... ON DUPLICATE KEY construction and thus ensure that the [query_master] table only has the same query stored once. But in [query_snapshot] table there is one row for every instance of the query.

Actually, with general/slow query log analyzers we use identical tables. The log CHUNK retrieved from the server isparsed and the tables populated like - you see in your sniffer.data database. The tables used for log analysis, however, are MEMORY tables and they are only available in SQL DM for MySQL and only for as long as they are needed.

5.4.2 Processlist

Also, SQL DM for MySQL processlist feature uses a SQLite MEMORY table (for every server). The table structure is as follows:

```
CREATE TEMPORARY TABLE IF NOT EXISTS [processlist](
[Id] INTEGER NOT NULL PRIMARY KEY,
[User] TEXT,
[Host] TEXT,
[Db] TEXT,
[Command] TEXT,
[Time] INTEGER,
[State] TEXT,
[Info] TEXT,
[Action] TEXT)
```

So, that is how the MySQL processlist displays in SQL DM for MySQL - unlike when connected to MySQL directly - it can be filtered, sorted etc. by using WHERE, ORDER BY, GROUP BY etc. with a SELECT query against the SQL DM for MySQL [processlist] table. But as it is a MEMORY table you can only query it from inside the SQL DM for MySQL processlist interface.

5.4.3 Information about the SQL DM for MySQL database schema itself

There is a schema_version table in all databases created by SQL DM for MySQL. Every time SQL DM for MySQL starts it checks here if the database is up to date with the current program version. If it is not, SQL DM for MySQL will perform the necessary schema upgrades at start-up. Schema definition reads as follows:

```
CREATE TABLE IF NOT EXISTS [schema_version] (
[schema_desc] TEXT,
[schema_major_version] TEXT,
[schema_minor_version] TEXT,
PRIMARY KEY ([schema_major_version], [schema_minor_version]))
```

5.5 SQL DM for MySQL Data Maintenance

You can rebuild the embedded SQLite database for a server by clicking the arrow icon on the left panel next to the server name and selecting the **Rebuild Database** option. Using this option periodically may result in better performance - including shorter startup time when OS is rebooted. Basically using this option defragments the database including indexes. It is not possible to provide an absolute advice on how this option should be used. But the larger the databases and the shorter the sample interval the faster there is a chance of fragmentation occurring after a huge amount of INSERTs and DELETEs to the database (Only with databases files of around 1 GB and larger we have seen the need for this. A general advice could be to execute monthly with large database files).

Also note, if the error "Database or Disk is full" (on Windows) prompts when VACUUM is executing, it refers to the file system hosting the temporary files location. Try setting the TEMP environment variable to a location with more free space, restart your system and it works.

Location in Windows 2008: `C:\ProgramData\Webyog\MONyog\Data\`

5.6 Real-Time data

The session data that are recorded by SQL DM by MySQL Real-Time are stored in SQLite in Monyog directories data folder.

The Real-time database has many tables. Schema details are as follows:

5.6.1 inndob_locks:

```
CREATE TABLE `innodb_locks` (
`row_id` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
`lock_id` VARCHAR (81) NOT NULL DEFAULT '',
`lock_type` VARCHAR (32) NOT NULL DEFAULT '',
`lock_table` VARCHAR (1024) NOT NULL DEFAULT '',
`lock_index` VARCHAR (1024) DEFAULT NULL,
`lock_data` VARCHAR (8192) DEFAULT NULL,
`lock_mode` VARCHAR (8192) DEFAULT NULL)
```

5.6.2 innodb_transactions:

```
CREATE TABLE `innodb_transactions` (
`row_id` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
`trx_id` VARCHAR (18) NOT NULL DEFAULT '',
`trx_state` VARCHAR (13) NOT NULL DEFAULT '',
`trx_start_time` INTEGER NOT NULL,
`trx_query_id` INTEGER (128) NOT NULL DEFAULT '',
`trx_query_starttime` INTEGER NOT NULL,
`trx_query_endtime` INTEGER NOT NULL,
`trx_user_host` VARCHAR (20) NOT NULL DEFAULT '',
`trx_db` VARCHAR (64) NOT NULL DEFAULT '',
`blocking_trx_id` VARCHAR (18) NOT NULL DEFAULT '',
`blocking_query_id` INTEGER NOT NULL DEFAULT '')
```

5.6.3 metric_master:

```
CREATE TABLE [ metric_master ] (
[ metric_id ] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
[ metric_desc ] TEXT ASC UNIQUE)
```

5.6.4 profiler_timestamps:

```
CREATE TABLE `profiler_timestamps` (
[ timestamp_id ] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
[ server_timestamp ] INTEGER UNIQUE)
```

5.6.5 query_master:

```
CREATE TABLE 'query_master' (
'id' INTEGER PRIMARY KEY AUTOINCREMENT,
'query' TEXT,
UNIQUE ('query'))
```

5.6.6 query_snapshot:

```
CREATE TABLE 'query_snapshot' (
'pkeyid' INTEGER PRIMARY KEY AUTOINCREMENT,
'id' INTEGER,
'threadid' INTEGER,
'user' TEXT,
'querytime' INTEGER,
'uptime' INTEGER,
'host' TEXT,
'state' TEXT,
'db' TEXT)
```

5.6.7 schema_master:

```
CREATE TABLE [ schema_master ] (
[ schema_id ] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
[ schema_name ] TEXT UNIQUE)
```

5.6.8 schema_version:

```
CREATE TABLE [ schema_version ] (
[ schema_desc ] TEXT,
[ schema_major_version ] TEXT,
[ schema_minor_version ] TEXT,
PRIMARY KEY (
[ schema_major_version ],
[ schema_minor_version ]))
```

5.6.9 snapshot_master:

```
CREATE TABLE [ snapshot_master ] (
[ timestamp_id ] INTEGER NOT NULL,
[ metric_id ] INTEGER NOT NULL,
[ metric_now ] TEXT,
[ metric_diff ] TEXT,
PRIMARY KEY (metric_id, timestamp_id))
```

5.6.10 sqlite_sequence:

```
CREATE TABLE sqlite_sequence(name,seq)
```

5.6.11 table_master:

```
CREATE TABLE [ table_master ] (
[ table_id ] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
[ schema_id ] INTEGER,
[ table_name ] TEXT,
UNIQUE (schema_id, table_name))
```

5.6.12 table_snapshot:

```
CREATE TABLE \[ table\_snapshot \] (
\[ timestamp\_id \] INTEGER NOT NULL, \[ table\_id \] INTEGER NOT
NULL, \[ COUNT\] INTEGER, PRIMARY KEY (timestamp\_id, table\_id))
```

5.6.13 timestamp_master:

```
CREATE TABLE \[ timestamp\_master \] (
\[ timestamp\_id \] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT
UNIQUE, \[ server\_timestamp \] INTEGER UNIQUE, \[
server\_start\_time \] INTEGER, \[ server\_uptime \] INTEGER, \[
server\_uptime\_diff \] INTEGER, \[ server\_is\_connected \]
INTEGER)
```

5.7 SQL DM for MySQL ini parameters

The MONyog.ini is the configuration file used by SQL DM for MySQL to read the configuration and is created when MONyog(SQL DM for MySQL) service is started for the first time. It consists of different parameters and can be used to customize a few settings as well. MONyog.ini consists of the following parameters:

- **Port:** The port at which SQL DM for MySQL listens to the HTTP requests. The default port is 5555, but you can change it to any open port available on the system where SQL DM for MySQL is running.
- **Password:** This field stores the obfuscated form of the admin password for SQL DM for MySQL UI. Hence, you can not change/edit the admin password from here, you can do it from the SQL DM for MySQL UI ("User profile" icon at the top right corner)
- **Data_path:** This gives the default data path at which SQL DM for MySQL stores the collected data from all the registered servers. A user can change the data path from here in order to store data to the desired location.
- **Registration_name:** The registration name used while registering the license keys in SQL DM for MySQL.
- **Key:** The license key used to register SQL DM for MySQL.
- **Install_ID:** This gives the unique ID associated with each SQL DM for MySQL installation.
- **JSRuntime_size:** It gives the amount of memory allocated to run all the Javascript (Monitors and Dashboard page) in SQL DM for MySQL. The default value is 1024 MB, it may need to be changed to higher values like 2048, 4096 when monitoring 100+ servers
- **Overview_Available:** It determines whether the Overview page is available in SQL DM for MySQL or not. The default value is 1, indicating Overview page will be available. User can make it 0, if they do not want this page to be displayed.
- **MONyogLogPath(Not present in the default MONyog.ini file):** It can be used to change the default MONyog.log path, for e.g: MONyogLogPath=/home/ubuntu/MONyogLog/

Please stop MONyog(SQL DM for MySQL) service before making any changes to any of the below parameters.

Depending on the mode of SQL DM for MySQL installation, you can find the MONyog.ini file in the below locations:

- **Windows**

```
<MONyog installation drive>\ProgramData\Webyog\MONyog\MONyog.ini
```

- **Linux System**

```
- for RPM: /usr/local/MONyog/MONyog.ini
- for .tar:<MONyog extracted directory>/MONyog/MONyog.ini
```

It is created when MONyog(SQL DM for MySQL) service is started for the first time. A default MONyog.ini looks like this:

```
[GENERAL]

Port=5555

Password=

Data_path=/home/pankaj/Downloads/MONyog/bin/..//data

Registration_name=Webyog

Key=7AD04B55-8898-4DEFG90-8D37-D88243971EF3

Install_id=b11c77ad-e5ea-439e-99bc-8f551a1380b7

JSRuntime_size=1024

Overview_Available=1

Intercom_Available=1
```

Apart from the above default parameters, you can also use MONyog.ini file to change the default MONyog-bin.pid file location from "/var/run/" to some other directory. A user can give the path in the MONyog.ini file like: “Pid_file_path=/abc/xyz".

6 The SQL DM for MySQL API

SQL DM for MySQL API is an application programming interface that enables SQL DM for MySQL to interact with other software like a command shell or a script or application. To use the SQL DM for MySQL API you send a HTTP request simply (specifying the SQL DM for MySQL URL with parameters) and thus even the address line of any browser can be used. Configure SQL DM for MySQL and perform other actions (like you will be able to from GUI). The APIs are capable of managing the servers (Adding, Editing, and Removing) along with managing data collection and alert settings.

6.1 Using the SQL DM for MySQL API

You can access the API by passing parameters to Monyog through its base URL.

For example, if Monyog is running on a system with IP 192.168.1.1, then the parameters need to be passed to: `http://192.168.1.1:5555/`

You can use either of the HTTP methods GET and POST.

6.1.1 The Parameters

The parameters that you will need to pass are:

- `_object` : This basically addresses the logical object in Monyog that you want to direct your request to.The only acceptable value is MONyogAPI.
- `_action` : This specifies the part of the object specified above that you want to direct your request to. The acceptable values are:
  - `Alerts`
  - `DataCollection`
  - `Sniffer`
  - `LongRunningQueries`
  - `LockedQueries`
  - `LongRunningQueryAction`
  - `AddServer`
  - `EditServer`
  - `RemoveServer`
- `_value` : The operation that you want to perform for the action specified in the _action field. Acceptable values include for:
  - `Alerts` , `DataCollection` , `Sniffer` , `LongRunningQueries` , `LockedQueries` : `enable` and `disable`
  - `LongRunningQueryAction` : `notify` , `kill` and `notifyandkill`
- `_user` : It may be Monyog user, LDAP user or LDAP Group user. In case, no user is supplied, admin account is used by default.
- `_password` : The password for the specified _user.

- `_server` : Name or data directory number of the servers separated by a comma(‘,’) for which the operation to be performed.
- `_tags` : Name of the tag separated by a comma(‘,’) for which the operation to be performed for all the servers under the specified tag.

6.1.2 API's for server management

To manage servers in Monyog via API, the following parameters need to be passed:

- `_server` : The name or data directory number of the server to be registered
- `_mysqlhost` : MySQL host/ip address
- `_mysqluser` : MySQL user name
- `_mysqlport` : MySQL port
- `_mysqlpassword` : MySQL user password

For example, to add a server:

```
$ curl "192.168.1.1:5555/?_object=MONyogAPI&_action=addserver&_user=admin
&_server=Production&_mysqlhost=127.0.0.1&_mysqluser=admin
&_mysqlport=3306&_mysqlpassword=adminpassword"
```

Additional parameters for registering servers are listed here.

For example, suppose you have a server named Production001 registered with Monyog. To stop data collection for this server using the HTTP GET method, the URL would look like:

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=DataCollection
&_value=disable&_password=mypassword&_server=Production001"
```

In summary, the various URLs that you can use with curl:

6.1.2.1 Change LDAP bind user password:

```
$ curl "127.0.0.1:5555/?_object=MONyogAPI&_action=changeldapbindpassword&_user=<sqldm
user name>&_password=<sqldm user password>&_currentpassword=<current ldap user
password>&_newpassword=<new ldap user password>"
```

6.1.2.2 Change user password:

```
$ curl "127.0.0.1:5555/?_object=MONyogAPI&_action=changepassword&_user=<sqldm
username>&_password=<currentpassword>&_newpassword=<newpassword>"
```

6.1.2.3 Starts data collection for <server name>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=DataCollection
&_value=enable&_user=admin&_password=Password&_server=Production001"
```

6.1.2.4 Starts data collection for <multiples servers>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=DataCollection
&_value=enable&_user=admin&_password=Password&_server=Production001,Test"
```

6.1.2.5 Stops data collection for <server name>(Slave Of Production)

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=DataCollection
&_value=disable&_user=admin&_password=Password&_server=Slave+Of+Production"
```

6.1.2.6 Starts data collection for <tag>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=DataCollection
&_value=enable&_user=admin&_password=Password&_tag=Production"
```

6.1.2.7 Stops data collection for <tag>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=DataCollection
&_value=disable&_user=admin&_password=Password&_tag=Production"
```

6.1.2.8 Stops data collection globally for all the servers (Maintenance)

```
$ curl "http://192.168.1.1:5555/?
_object=MONyogAPI&_action=datacollection&_value=enable&_user=admin&_password="
```

6.1.2.9 Enables alerts for <server name>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=Alerts
```

```
&_value=enable&_user=admin&_password=Password&_server=Production001"
```

6.1.2.10 Disables alerts for <server name>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=Alerts
&_value=disable&_user=admin&_password=Password&_server=Production001"
```

6.1.2.11 Enables alerts for <tag>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=Alerts
&_value=enable&_user=admin&_password=Password&_tag=Production"
```

6.1.2.12 Disables alerts for <tag>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=Alerts
&_value=disable&_user=admin&_password=Password&_tag=Production"
```

6.1.2.13 Disables alerts globally for all the servers(Maintenance)

```
$ curl "http://192.168.1.1:5555/?
_object=MONyogAPI&_action=Alerts&_value=disable&_user=admin&_password="
```

6.1.2.14 Enables alerts globally for all the servers(Maintenance)

```
$ curl "http://192.168.1.1:5555/?
_object=MONyogAPI&_action=Alerts&_value=enable&_user=admin&_password="
```

6.1.2.15 Enables Sniffer for <server name>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=sniffer
&_value=enable&_server=Production001"
```

6.1.2.16 Disables Sniffer for <server name>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=sniffer
&_value=disable&_server=Production001"
```

6.1.2.17 Add Server

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=addserver
&_mysqluser=msandbox&_mysqlhost=127.0.0.1&_mysqlport=3306&_tags=Production
&_server=Test&_mysqlpassword=msandbox&_connectontype=direct
&_user=admin&_password=Password"
```

6.1.2.18 Add Server with SSH Tunnel

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=addserver
&_mysqluser=msandbox&_mysqlhost=127.0.0.1&_mysqlport=3306
&_tags=Production&_server=Test&_mysqlpassword=msandbox
&_connectiontype=ssh&_sshhost=192.168.1.86&_sshuser=username
&_sshpassword=sshpassword&_sshport=22&_user=admin&_password=Password"
```

6.1.2.19 Edit Server

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=editserver
&_mysqluser=msandbox&_mysqlhost=127.0.0.1&_mysqlport=3306
&_tags=Production&_server=Test&_mysqlpassword=msandbox
&_connectontype=direct&_user=admin&_password=Password"
```

6.1.2.20 Delete Server

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=removeserver&_server=Test"
```

6.1.2.21 Delete all the servers under <tag>

```
curl "http://192.168.1.1:5555/?
_object=MONyogAPI&_action=removeserver&_tag=Production"
```

6.1.2.22 Delete multiple servers

```
curl "http://192.168.1.1:5555/?
_object=MONyogAPI&_action=removeserver&_server=Server1,Server2"
```

6.1.3 Return Codes

Assuming that the connection to SQL DM for MySQL was successful, it returns a text message. The message is in the JSON format:

```
{"STATUS": "SUCCESS/FAILURE", "RESPONSE" : "<Response text>"}
```

Your application can parse this message and determine whether the operation was successfully carried out or not.

⚠ Since version 5.21 we have deprecated the API calls to `_object=ConnectionMgr` . Instead use `_object=MONyogAPI` .

6.1.4 Applications

The Monyog API is very flexible and can be accessed from other programming languages including scripting languages such as Perl, VBScript, etc. Here is a very generic Perl script that accepts the required parameters from the command line and executes the specified action:

```
#! /usr/bin/perl
use LWP 5.64;
# USAGE: MONyog.pl <hostname>:<port> <user> <password> <connection_name/ID> <action>
<value>
# $ARGV[0] = hostname:port of server running Monyog
# $ARGV[1] = Monyog user
# $ARGV[2] = Monyog password
# $ARGV[3] = connection name
# $ARGV[4] = action
# $ARGV[5] = value
my $numArgs = $#ARGV + 1;
if($numArgs < 5) {
die 'USAGE: MONyog.pl <hostname>:<port> <user> <password> <connection_name/ID>
<action>';
}

my $browser = LWP::UserAgent->new;

# The request URL
my $url = URI->new('http://' . $ARGV[0] . '/');
```

```
# The form data pairs:
$url->query_form(
'_object' => 'MONyogAPI',
'_action' => $ARGV[4],
'_user' => $ARGV[1]
'_password' => $ARGV[2],
'_server' => $ARGV[3],
'_value' => $ARGV[5]
);

# The response object
$response = $browser->post($url);

if (!$response->is_success) {# Error connecting to MONyog
die $response->status_line . "\n";
} else {

# Successfully connected to MONyog; print MONyog's response
print $response->content . "\n";

}
```

6.2 Additional Parameters for API

6.2.1 SSH tunneling for MySQL

Property name	Description
`_sshhost`	Host machine on which SSH server is running.
`_sshuser`	Username to access the SSH server.
`_sshport`	Port on which SSH server is listening. **Range:** 1 .. 65535 **Defaluts:** "22"
`_sshauthtype`	SSH authentication type. **Permitted values:** `key` , `password` **Default:** `password`
`_sshpassword`	SSH user's password for password based authentication.

Property name	Description
`_sshprivatekey`	SSH user's private 'key' for key based authentication. **Note** The key must be url encoded. For example line endings `\r\n` must be encoded to `%0A` and " " (white space) to `+`.
`_sshpassphrase`	Passphrase for private key file (if any) for key based authentication.

6.2.2 SSL encryption for MySQL

Property name	Description
`_cacertificate`	The digital certificate issued by CA. **Note** The key must be url encoded. For example line endings 'rn' must be encoded to '%0A' and " " (white space) to "+".
`_sslcipher`	Encryption algorithm like DES, AES etc. **Note** The key must be url encoded. For example line endings 'rn' must be encoded to '%0A' and " " (white space) to "+".
`_usesslauth`	SSL client authentication type. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_clientkey`	Private key of the client that is needed for encryption.
`_clientcertificate`	The client certificate.

6.2.3 SSH settings

To enable and use SSH for reading logs via SFTP and/or OS monitoring, the following query parameters are required:

Property name	Description
`_osmonitoring`	Enable or disable SSH settings for OS monitoring/reading MySQL logs via SFTP. **Permitted values:** yes/no; true,false;1/0 **Default:** `no`

Property name	Description
`_ostype`	OS monitoring is available for Linux systems only. Setting to "Linux", SQL DM for MySQL can monitor system related metrics like CPU consumption, Memory Usage, etc. **Permitted values:** `linux` , `rds` .
`_ossameasmysqltunnel`	Uses the same SSH details provided for SSH tunneling. Setting to "yes" will use the details provided for SSH tunneling to MySQL for this registration. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_sshsystemhost`	Host machine on which SSH server is running.
`_sshsystemuser`	Username to access the SSH server.
`_sshsystemport`	Port on which SSH server is listening. **Range:** 1 ..65535 **Default:** "22"
`_sshsystemauthtype`	SSH authentication type to be used.
`_sshsystempassword`	SSH user's password for password based authentication.
`_sshsystemprivatekey`	SSH user's private 'key' for key based authentication. **Note** The key must be url encoded. For example line endings 'rn' must be encoded to '%0A' and " " (white space) to "+".
`_sshsystempassphrase`	Passphrase for your private key file (if any) for 'key' based authentication.

6.2.4 Notification settings

Property name	Description
`_mailalerts`	Setting to "yes" will send alerts/notifications via email. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"

Property name	Description
`_snmpalerts`	Setting to "yes" will trigger SNMP traps on event of alerts. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_mailaddress`	SQL DM for MySQL will send alerts to the email addresses specified under `_mailaddress`. Accepts comma (,) separated list of email addresses.
`_mailaddresscritical`	SQL DM for MySQL will send critical alerts to the email addresses specified under `_mailaddresscritical`.
`_mailaddresswarning`	SQL DM for MySQL will send warning alerts to the email addresses specified under `_mailaddresswarning`
`_mailaddressother`	SQL DM for MySQL will send other alerts such as MySQL server restart and MySQL configuration changes to the email addresses specified under `_mailaddressother`.
`_writetosyslog`	Seting to yes will route the alert notifications to syslog. **Permitted values:** yes/no; true,false; 1/0
`_slackalerts`	Setting to "yes" will route alerts/notifications to the Slack room. **Permitted values:** yes/no
`_slackrulename`	Specify the Slack Rule to be used.
`_pagerdutyalerts`	Setting to "yes" will route alerts/notifications to the PagerDuty. **Permitted values:** yes/no
`_pagerdutyrulename`	Specify the PagerDuty rule to be used.
`_alertableinterval`	Number of times to wait before sending the alerts. **Default:** "1"
`_notifyserverconfigchange`	SQL DM for MySQL will send an alert whenever there is a change in MySQL configuration. **Permitted values:** yes/no; true,false; 1/0 **Default:** "yes"

Property name	**Description**
`_notifyserverrestart`	SQL DM for MySQL will send an alert whenever the server restarts. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_briefemail`	Enable or disable detailed email notification. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_notifystable`	Notify stable alerts when monitor goes into alert-able state and then becomes stable. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_notifytillstable`	Keeps notifying the user until the counter becomes stable. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_reminderinterval`	Defines the number of data collections after which the user receives the notification until the counter becomes stable. **Default:** "5"

6.2.5 Data collection settings

Property name	**Description**
`_datacollection`	Enable data collection for server. SQL DM for MySQL will not collect data for this registration if data collection is disabled. **Permitted values:** yes/no; true,false; 1/0 **Default:** "yes"
`_datacollectioninterval`	The interval for the data collections in seconds. **Default:** "300" seconds (5 minutes)
`_dataretentiontime`	Data purging interval for the server. Timeframe should be specified in seconds. **Default:** "604800" (7 days)

6.2.6 Replication settings

Property name	**Description**
`_replicationslave`	Consider a server as slave server. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_autoregisterslaves`	Register all slave servers. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"

6.2.7 Galera Settings

Property name	**Description**
`_autoregistergaleranodes`	Auto-register all galera nodes. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"

6.2.8 Error log monitoring

Property name	**Description**
`_enableerrorlog`	To enable error log monitoring. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_errorlogreadmode`	Mode to read the error log file. **Permitted values:** sftp, local, rds **Default:** "local"
`_errorlogpath`	MySQL error log path. **Default:** "/var/log/mysql/server-err.log"
`_dbidentifier`	A unique name to identify your RDS/Aurora instance.
`_instanceregion`	The region in which your instance is hosted, e.g: us-east-1.
`_accesskey`	A 20 character long key ID, is generated from AWS Mangement Console.

Property name	Description
`_secretkey`	A 40 character long key ID, is generated from AWS Management Console.

6.2.9 Slow query log and General query log settings

Property name	Description
`_logreadmode`	Mode to read the General and Slow query log files. **Permitted values:** sftp, local, rds **Default:** "local"
`_querylogdestination`	Read logs stored either from FILE or TABLE. **Permitted values:** file, table **Default:** "file"
`_enableslowquery`	To enable slow query log monitoring. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_slowquerylogpath`	Path for slow log. **Default:** /var/log/mysql/server-slow.log
`_enablegeneralquery`	To enable general query log monitoring. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_generalquerylogpath`	Path for general log. **Default:** /var/log/mysql/server-general.log
`_dbidentifier`	A unique name to identify your RDS/Aurora instance.
`_instanceregion`	The region in which your instance is hosted, e.g: us-east-1.
`_accesskey`	A 20 character long key ID, is generated from AWS Management Console.
`_secretkey`	A 40 character long key ID, is generated from AWS Management Console.

6.2.10 Audit log settings

Property name	Description
`_auditlogreadmode`	Mode to read Audit log. **Permitted values:** sftp, local, rds **Default:** "local"
`_enableauditlog`	To enable audit log monitoring. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_auditlogpath`	Path for audit log. **Default:** /var/lib/mysql/server_audit.log
`_dbidentifier`	A unique name to identify your RDS/Aurora instance.
`_instanceregion`	The region in which your instance is hosted, e.g: us-east-1.
`_accesskey`	A 20 character long key ID, is generated from AWS Management Console.
`_secretkey`	A 40 character long key ID, is generated from AWS Management Console.

6.2.11 Sniffer settings

Property name	Description
`_enablesniffer`	Enable or disable sniffer analysis. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_sniffermode`	Specifying a way to populate sniffer data for analysis. **Permitted values:** processlist, performanceschema, proxy **Default:** "processlist"
`_monitorlongrunningqueries`	Monitor only long running queries, executing beyond the specified time. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"

Property name	Description
`_longrunningquerytime`	The time which qualifies a query as long running query (in seconds). **Default:** "10"
`_ignorequeriesbyuser`	A filter to ignore queries by a user.
`_longrunningqueryaction`	The action to be performed for long running queries. **Permitted values:** notify, kill, notifyandkill **Default:** notify
`_monitorlockedqueries`	Monitors for queries in the locked state. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_snifferproxyhost`	MySQL proxy host.
`_snifferproxyport`	MySQL proxy port.
`_sniffinginterval`	This interval specifies how frequently SQL DM for MySQL should "sniff" MySQL server (in seconds). **Default:** "1"
`_snifferpurginginterval`	Specifying a timeframe to purge the data collected (in seconds). **Default:** "259200" (3 days)
`_snifferfilteruser`	A filter to sniff queries only by specified users.
`_snifferfilterhost`	A filter to sniff queries only by specified hosts.
`_sniffquerystartingwith`	To sniff queries starting with the string.
`_snifferlongquerymail`	Setting to "yes" will enable SQL DM for MySQL to send mail alerts for long running queries. **Permitted values:** yes/no
`_snifferlongquerymailto`	SQL DM for MySQL will send long running query alert to the email addresses specified under _snifferlongquerymailto

Property name	Description
`_snifferlongquerytrap`	Setting to "yes" will enable SQL DM for MySQL to send SNMP traps for the long running queries. **Permitted values:** yes/no
`_snifferlongquerywritetosyslog`	Setting to "yes" will enable SQL DM for MySQL to write the alerts for the long running queries to the syslog of the host machine of SQL DM for MySQL. **Permitted values:** yes/no
`_snifferlongqueryslack`	Setting to "yes" will enable SQL DM for MySQL to route the alerts to the Slack room for the long running queries. **Permitted values:** yes/no
`_snifferlongqueryslackrule`	Specify the Slack rule to be used for sending the Long running query alerts.
`_snifferlongquerypagerduty`	Setting to "yes" will enable SQL DM for MySQL to route the alerts to PagerDuty for the long running queries. **Permitted values:** yes/no
`_snifferlongquerypagerdutyrule`	Specify the PagerDuty rule to be used for sending the Long running query alerts.

6.2.12 Deadlock monitoring settings

Property name	Description
`_enabledeadlockmonitoring`	Enable or disable InnoDB deadlock monitoring. Turning this ON will help in tracing deadlocks reported by "SHOW INNODB STATUS". **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"

6.2.13 Manage Monitors settings

Property name	Description
`_disabledmonitorgroups`	This specifies the monitor groups that are to be disabled for the server. Comma-separated Group IDs may be supplied. To know the Group IDs of the various monitor groups, you may hover over the Monitor Groups in Customize -> Manage Monitor Groups. For example, if you want to disable, Binary Log (Group ID: 17), Replication (Group ID : 19) and MySQL Cluster(Group ID: 27), the parameters will be 17,19,27 **Default:** 7,14,22

6.2.14 Real-Time Mode Setting

Property name	Description
`_realtimemode`	For setting up Real-Time monitoring mode. **Permitted values:** processlist/performanceschema **Default:** "processlist"

6.2.15 Connection Settings

Property name	Description
`_mysqlconnecttimeout`	Specify MySQL connection timeout. **Default:** "30" seconds
`_sshconnecttimeout`	Specify SSH tunnel connection timeout. **Default:** "30" seconds
`_sshsystemconnecttimeout`	Specify SSH connection timeout. **Default:** "30" seconds

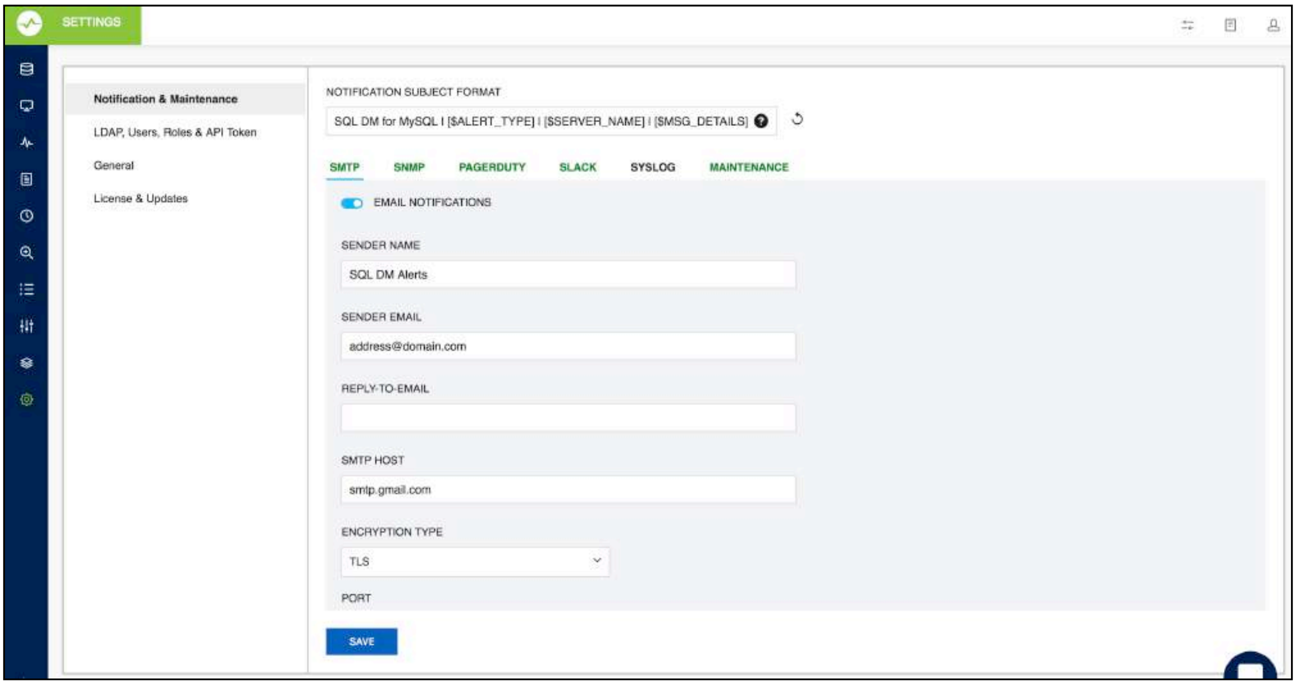
7 Settings

Use the Settings page to configure the following options in SQL DM for MySQL:

- Notification & Maintenance[65]
- LDAP, Users, Roles, & API Token[66]
- General[67]
- License & Updates[68]
- SQL DM for MySQL Log[69]

7.1 Notification & Maintenance

The options here offer you to configure multiple notification channels. To route the alerts generated in SQL DM for MySQL, you have to enable and configure the required channels.



NOTIFICATION SUBJECT FORMAT: SQL DM for MySQL I [$ALERT_TYPE] I [$SERVER_NAME] I [$MSG_DETAILS]

EMAIL NOTIFICATIONS (on)

SENDER NAME: SQL DM Alerts

SENDER EMAIL: address@domain.com

REPLY-TO-EMAIL:

SMTP HOST: smtp.gmail.com

ENCRYPTION TYPE: TLS

PORT

7.1.1 Notification Subject Format

This section provides details to customize the Subject format of the alert notifications that you receive from SQL DM for MySQL. The default format is: **SDM for MySQL | [$ALERT_TYPE] | [$SERVER_NAME] | [$MSG_DETAILS]**

[$ALERT_TYPE]: This describes the severity of the alert - the values can be Critical, Warning, Stable, and INFO. The Alert type is set up based on the type of the alert for which the email was generated. The alert type is set based on the following conditions:

[65] http://wiki.idera.com/x/twEvvgE
[66] http://wiki.idera.com/x/uAEvvgE
[67] http://wiki.idera.com/x/uQEvvgE
[68] http://wiki.idera.com/x/ugEvvgE
[69] http://wiki.idera.com/x/uwEvvgE

- **Critical -** SQL DM for MySQL sends critical alerts to the registered emails in the following scenarios:
    a. When Monitor value crosses the set Critical threshold limit.
    b. When the Long running query action is “Notify And Kill".
    c. Low disk space alert.
    E.g.: SDM for MySQL | Critical | Production | Excessive Privileges: Number Of Users Having Globa…
    SQL DM for MySQL | Critical | SQLDM-Host | Low Disk Space.
- **Warning -** Warning alerts are sent when it meets the following conditions:
    a. When Monitor value crosses the set Warning threshold limit.
    b. When the Long running action is “Notify”
    E.g.: SQL DM for MySQL | Warning | 5.7.18 128 | MySQL Logs: General log - Enabled? - Yes.
- **Stable -** When a Monitor value had crossed the set Critical/Warning threshold limits and is back to the recommended values, user is notified when the alerting monitor is now stable. The Alert Type of this notication is Stable.
  E.g.: SQL DM for MySQL | Stable | Staging | Current Connections: Currently running threads - 1.
- **Info -** Alert type is Info when the mysql server restarts or there is a configuration change. This alert isgenerated when the below conditions are enabled when the server was registered.
    a. Notify when server restarts.
    b. Notify when server config changed.
    E.g.: SQL DM for MySQL | Info | Testing | Server configuration change detected.

[$SERVER_NAME]: Name of the server for which the alert is generated. For Low disk space mails, the server_name is SQLDM-Host.
[$MSG_DETAILS]: This specifies the details of the Alert.
For example, “Linux: CPU usage - 74.4%” OR “Long Running Query: QUERY” OR “Server configuration change detected" Or "Server restarted".
If there are more than one monitor alert in an email then this will list down the number of events in the email, i.e, “n Events” where ‘n’ is the “total no of events”.
E.g.: SQL DM for MySQL | Critical | Production | 2 Events.

✓ You can use the ‘Set Default’ option(the refresh button) to revert back to the default format.

7.1.2 Mail (SMTP)

Use this action to specify the SMTP server address and the "from" address of the e-mail sender.

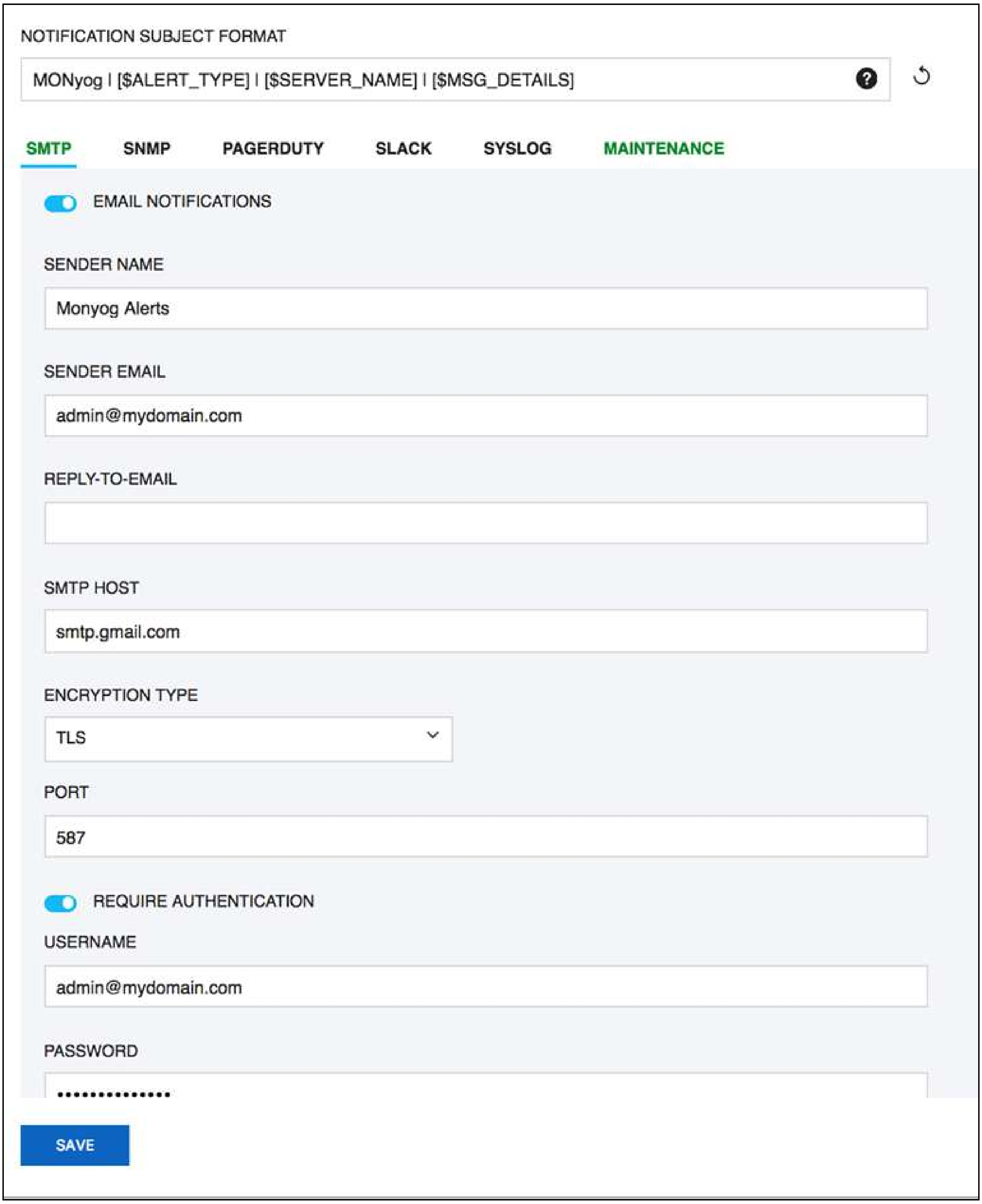
1. Click **Settings**, **Notification & Maintenance**, and select **SMTP**.The SMTP Server Settings window opens. Alternatively, you can click **Configure mail settings** found on the Notifications Settings when registering/editing the details of a server.
2. **From:** Type the from name and e-mail address with the name and address in the fields Your Name and Your Email respectively. This field must be in standard *name@domain.com* address format unless your relay server is set up to accept default domain from addresses. Most problems involving configuration of the email alert are due primarily to an invalid email address format specified in the FROM/TO fields.
3. **Reply-to email:** Enter email address of the recipient (or recipients) of the email message. This field must be in standard *name@domain.com* address format unless your relay server is prepared to accept the default domain from addresses.

> ✓ To add multiple recipients, specify multiple recipients by simply separating the email addresses with a comma ",". Example: *me@here.com, you@there.com, somebody@somewhere.com.*

4. Type the **SMTP** server address. You can enter the host name, including domain, or the TCP/IP address. Example Gmail SMTP address is "smtp.gmail.com".
5. **Encryption:** Select the type of Encryption - SSL/TLS mail encryption which is now supported for mail alerts.
6. **Port:** This field signifies the TCP port on which host/IP addresss should connect in order to deliver the message. By default, this field is set to 25 (SMTP). However, some internal SMTP servers may be setup on non-standard ports which will require that this field be changed to match the listening port of the mail exchange. For example Gmail listens to port 465 for SSL encryption and 587 for TLS encryption.
7. **Username and Password**: When sending mail through an authenticated SMTP server, you can fill in the User and Password fields appropriately. If you are not using a secure SMTP server, an error 5xx unrecognized command may be returned from the server if you enter anything into one of these fields.

> ⚠ Most problems involving configuration of the Email alert are due primarily to an invalid email address format specified in the FROM/TO fields. Please refer Registering servers[70] for more information.

[70] http://wiki.idera.com/x/dgEGBg



NOTIFICATION SUBJECT FORMAT

MONyog | [$ALERT_TYPE] | [$SERVER_NAME] | [$MSG_DETAILS]

EMAIL NOTIFICATIONS

SENDER NAME

Monyog Alerts

SENDER EMAIL

admin@mydomain.com

REPLY-TO-EMAIL

SMTP HOST

smtp.gmail.com

ENCRYPTION TYPE

TLS

PORT

587

REQUIRE AUTHENTICATION

USERNAME

admin@mydomain.com

PASSWORD

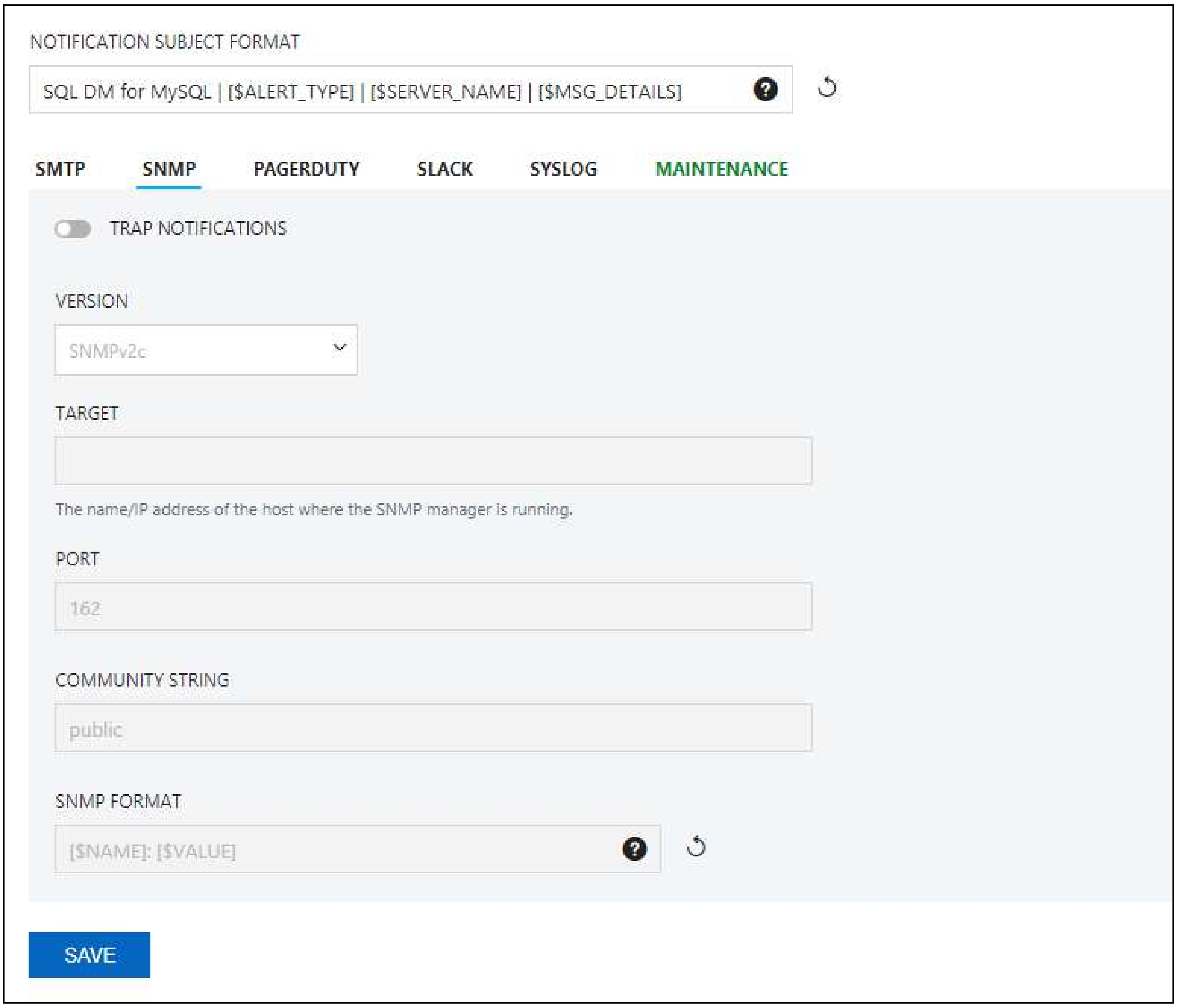
••••••••••••••

7.1.3 SNMP

Use this action to specify the SNMP settings and options available for use with SQL DM for MySQL.

Click **Settings -> Notification & Maintenance -> SNMP** to navigate to the SNMP settings page.

- **Version:** SQL DM for MySQL supports 2 types of SNMP versions: SNMPv1 and SNMPV2c. You can select the trap type from this drop down.
- **Target:** Type in the name or IP address of the host to which you want SQL DM for MySQL to direct traps.
- **Port:** Enter the destination port here; by default, it is 162. This is the port where your SNMP Manager (or Client) listens for traps.
- **Community string:** The read/write community string helps classify your SNMP operations. By default, SQL DM for MySQL sets it to Public, but you can change to whatever suits your need.
- **SNMP Format:** Specify SNMP trap format e.g. [$NAME]:[$VALUE].
- **Enable status traps:** Status traps are sent when SQL DM for MySQL is starting up to indicate just that. If you want to be informed of when SQL DM for MySQL is starting up, select **Yes**.
- **Use the remote MySQL Server host IP as the SNMP trap agent address for Monitor traps**: Check this option if you want the sender IP of the traps sent by SQL DM for MySQL to be that of the host where the monitored MySQL server is running, instead of the host where SQL DM for MySQL is running. This only affects the Monitor traps sent by SQL DM for MySQL.
- Clicking the **Send Test Trap,** results in sending a status trap to the target and port specified, containing a string that indicates that this is a test trap.
- To have your SNMP client decode the arcane digits identifying a trap, you need to load **SQL DM for MySQL's Management Information Base (MIB)**s into your SNMP client. The MIB file is available in the installation directory of SQL DM for MySQL. You can also download the file from the link provided in SQL DM for MySQL browser interface.



NOTIFICATION SUBJECT FORMAT

SQL DM for MySQL | [$ALERT_TYPE] | [$SERVER_NAME] | [$MSG_DETAILS]

TRAP NOTIFICATIONS

VERSION

SNMPv2c

TARGET

The name/IP address of the host where the SNMP manager is running.

PORT

162

COMMUNITY STRING

public

SNMP FORMAT

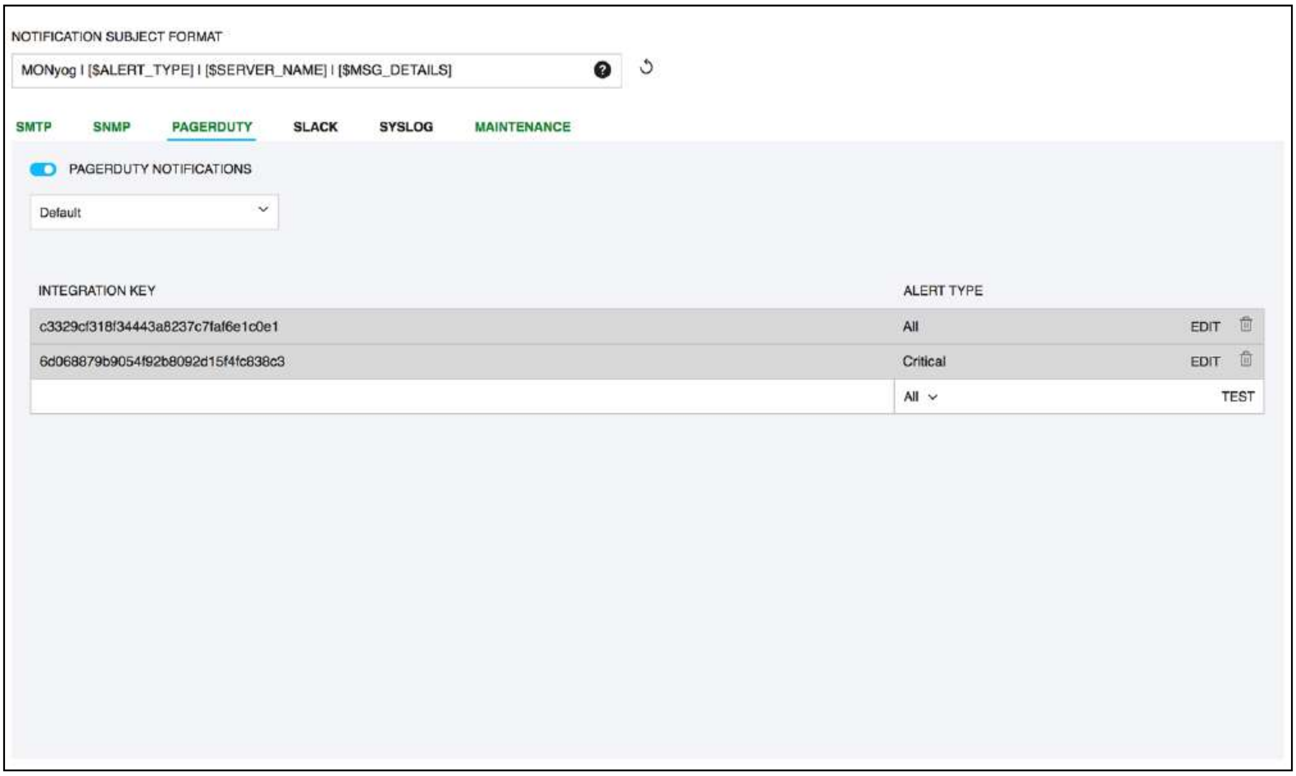
[$NAME]: [$VALUE]

Note more information on types of Traps:

- **Monitor Traps:** Traps sent when one of the Monitors reaches an alert condition. Just like the notification emails, it contains details about the faulting Monitor. Monitor traps are sent when the Send notifications over SNMP option is enabled, and the corresponding counters have mail alert enabled through the MOM.
- **Status Traps**: Sent when SQL DM for MySQL starts up (if status traps are enabled) indicating just that, i.e. "SQL DM is starting up". Also, clicking **Send Test Trap** sends a status trap with the string, "This is a test trap! If you see this message, you have correctly configured SNMP for SQL DM."

7.1.4 PagerDuty

Click **Settings -> Notification & Maintenance -> PAGERDUTY** to navigate to the PagerDuty settings page.



NOTIFICATION SUBJECT FORMAT

MONyog | [$ALERT_TYPE] | [$SERVER_NAME] | [$MSG_DETAILS]

PAGERDUTY NOTIFICATIONS

Default

INTEGRATION KEY	ALERT TYPE	
c3329cf318f34443a8237c7faf6e1c0e1	All	EDIT
6d068879b9054f92b8092d15f4fc838c3	Critical	EDIT
	All	TEST

PagerDuty integration allows you to route alerts to your PagerDuty service.

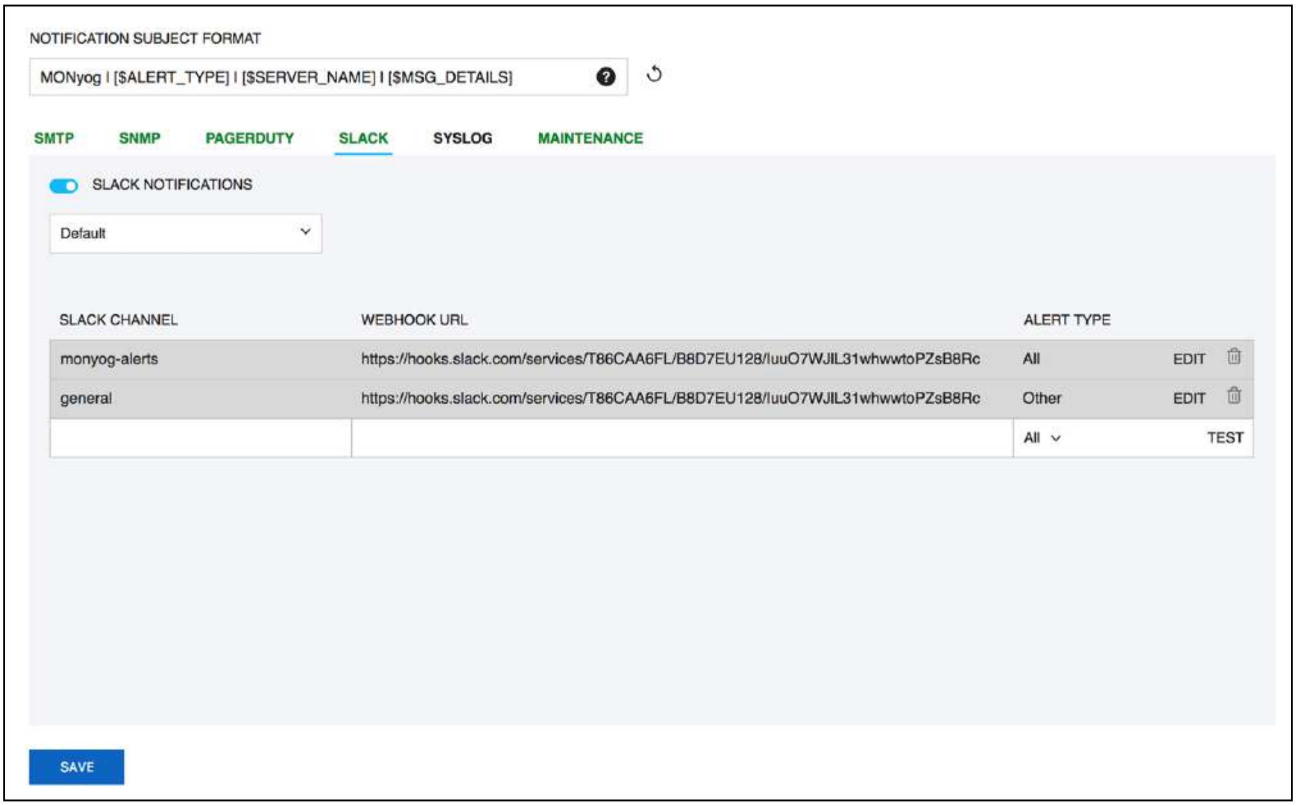
Use the integration key of the service that you want SQL DM for MySQL alerts to be sent to and select the type of alerts to be sent out. Click **Test** to verify whether the integration is configured successfully.

Also, you can create multiple routing rules based on the requirement for different servers registered with SQL DM for MySQL. To create a new rule select **CREATE NEW RULE** option in the drop down.

> ⚠ Make sure that the integration key that you add has the integration type as **API**.

7.1.5 Slack

Click **Settings -> Notification & Maintenance -> SLACK** to navigate to the SLACK settings page.



NOTIFICATION SUBJECT FORMAT

MONyog | [$ALERT_TYPE] | [$SERVER_NAME] | [$MSG_DETAILS]

SLACK NOTIFICATIONS

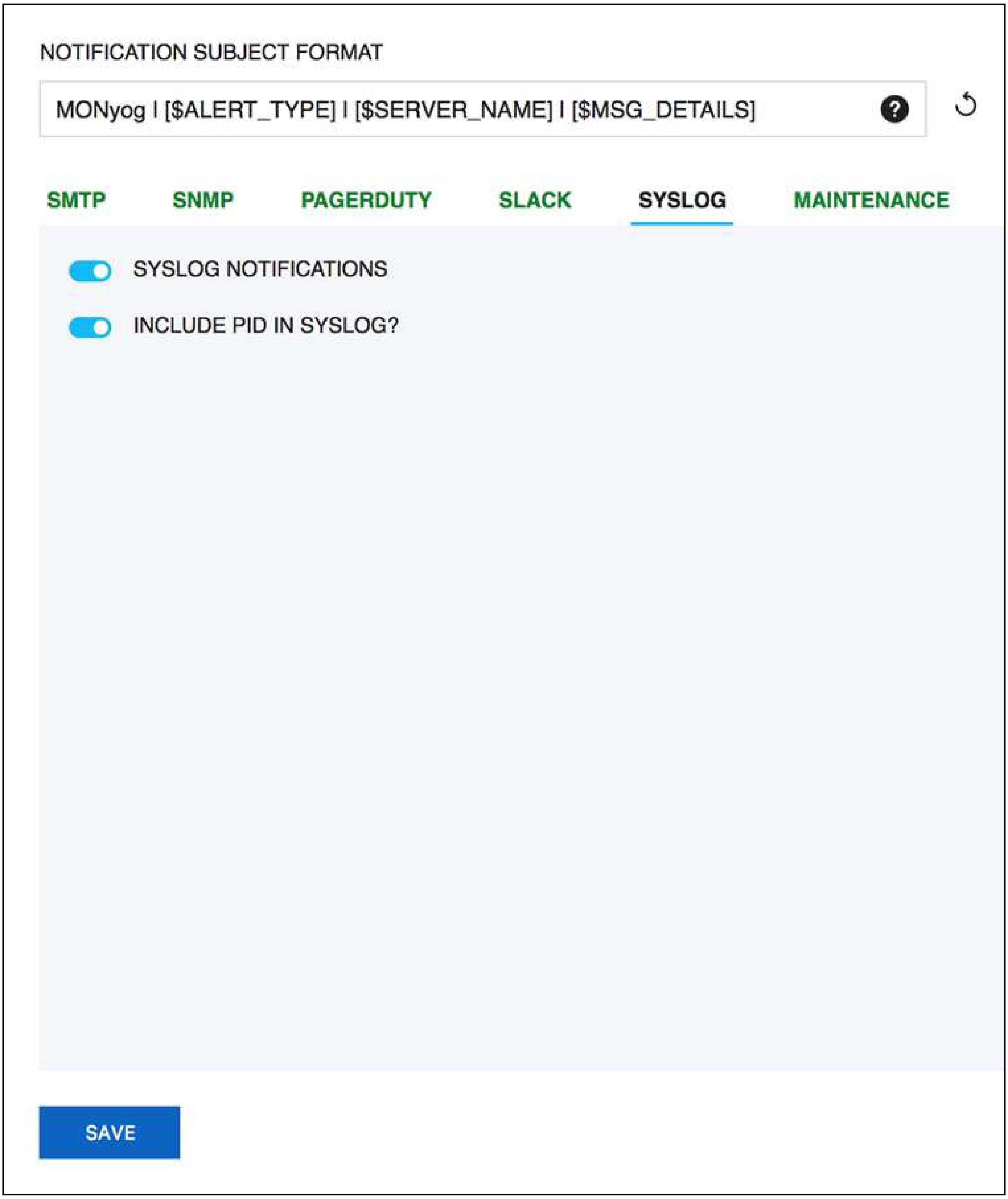
Default

SLACK CHANNEL	WEBHOOK URL	ALERT TYPE		
monyog-alerts	https://hooks.slack.com/services/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	All	EDIT	
general	https://hooks.slack.com/services/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	Other	EDIT	
		All	TEST	

You can Integrate Slack to SQL DM for MySQL to receive the alerts and notifications from SQL DM for MySQL to your Slack channel.
Enter the name of the Channel that you want SQL DM for MySQL alerts to be sent to in the SLACK CHANNEL field, enter the Incoming Webhook in the WEBHOOK URL field and Select the desired Alert Type. To ensure that the rule is configured successfully, try out the rule by clicking **Test**.

7.1.6 SYSLOG

Click **Settings -> Notification & Maintenance -> SYSLOG** to navigate to the SYSLOG settings page.



NOTIFICATION SUBJECT FORMAT

MONyog | [$ALERT_TYPE] | [$SERVER_NAME] | [$MSG_DETAILS]

SYSLOG NOTIFICATIONS ●

INCLUDE PID IN SYSLOG? ●

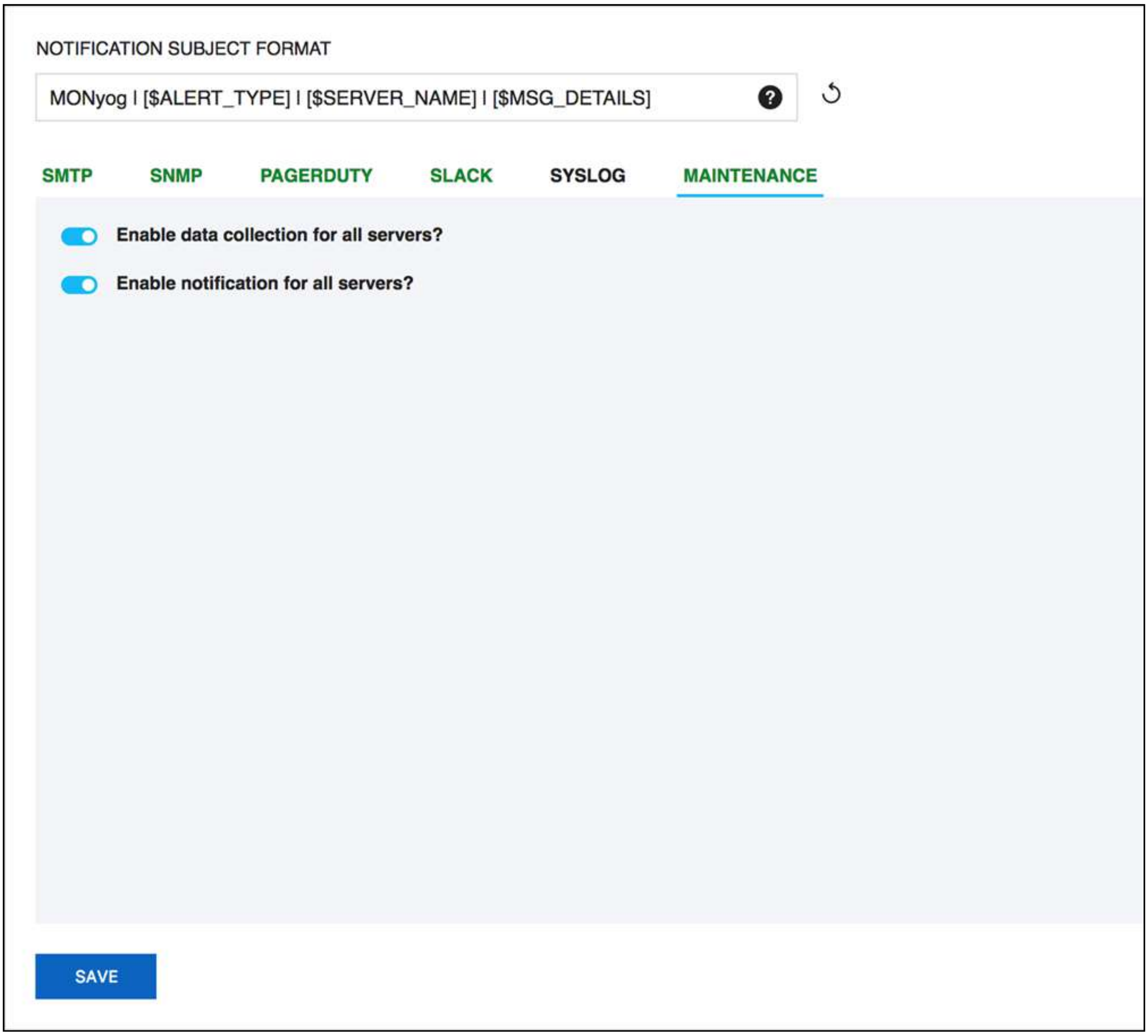
On enabling this option SQL DM for MySQL can write out the alerts and notifications to the Syslog (of the machine where SQL DM for MySQL is installed).

⚠ This is applicable only for Linux installations.

7.1.7 Maintenance

Using this Maintenance option, you can enable/disable data collection from and/or alerting about all servers.

You can enable/disable data collection option and alert for all the registered server using **Settings -> Notification & Maintenance -> Maintenance.**



NOTIFICATION SUBJECT FORMAT

MONyog | [$ALERT_TYPE] | [$SERVER_NAME] | [$MSG_DETAILS]

Enable data collection for all servers?

Enable notification for all servers?

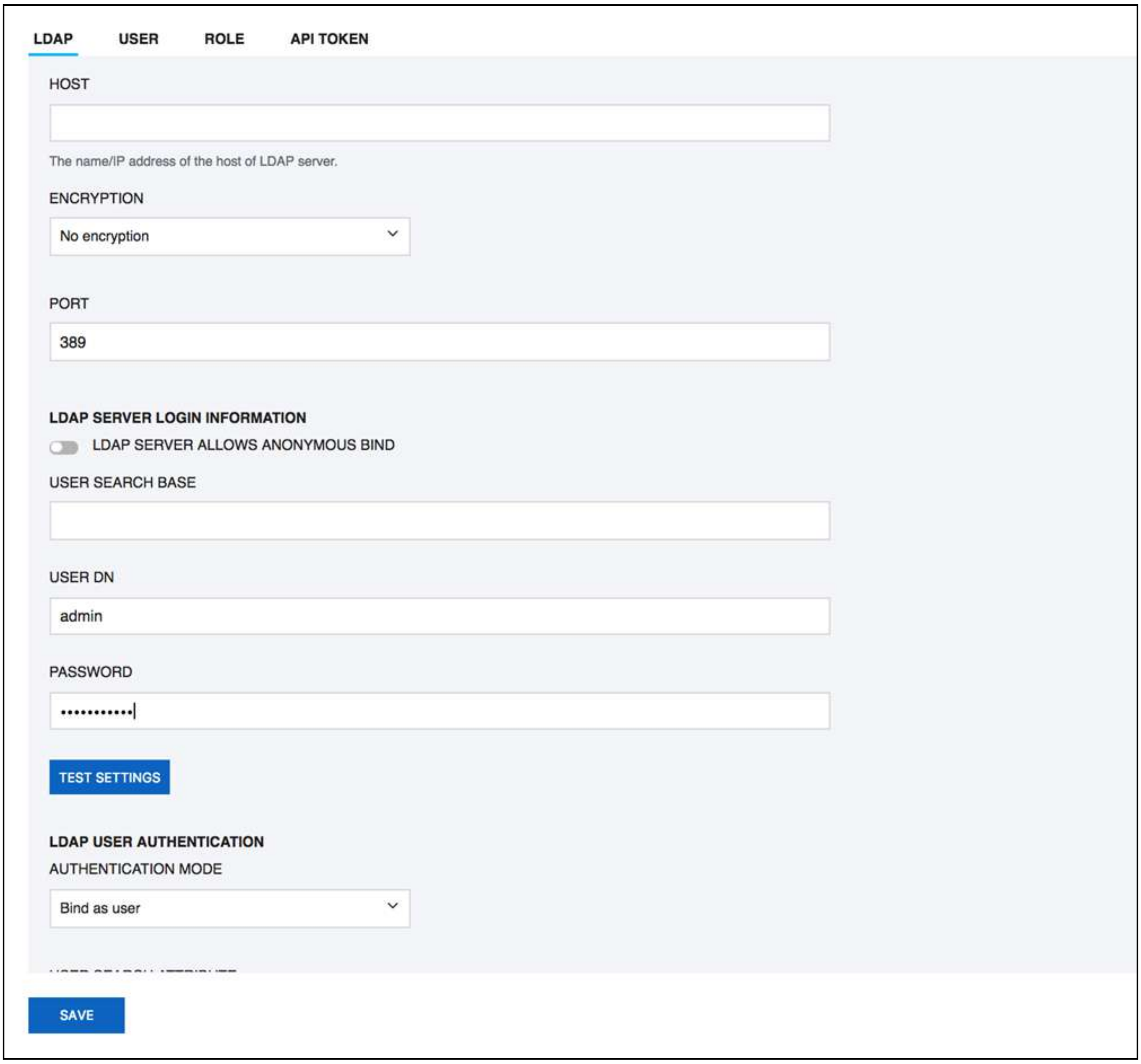
7.2 LDAP, Users, Roles, & API Token

This page allows user to add LDAP authentication, create users, create different roles, and also to generate a new API token.

7.2.1 LDAP Settings

When logging into SQL DM for MySQL from the browser interface you may use authentication provided by LDAP server (including the Microsoft/Windows LDAP 'dialect' known as 'Active Directory'). In this case users need not to know SQL DM for MySQL authentication details directly, but only how to authenticate to the LDAP server. To use LDAP authentication for SQL DM for MySQL, specify settings as below:

1. Click **Settings**
2. Select **LDAP, Users, Roles & API Token,** the LDAP setting page opens, displaying the following options:
    - **Host:** Enter the hostname, IP address or URI (Uniform resource identifier) of your LDAP directory server.
    - **Encryption:** Select the type of encryption required for communication with the LDAP directory server. Supported encryption methods are None, StartTLS, and SSL(ldaps).
    - **CA CERTIFICATE:** If your encryption mode is StartTLS and SSL(Ldaps), then paste the content of your digital certificate issued by CA.
    - **Port:** Type in the port your LDAP directory server uses.
    - **LDAP server allows anonymous binds:** Select this option if your LDAP directory server allows anonymous binds to the server.
    - **User DN:** Enter the distinguished name of the entry to bind to the LDAP directory server.
    - **Password:** Enter the password of the User DN specified for binding the user to LDAP directory server.
    - **Test Settings:** Click **Test Settings** to use the mentioned **User DN**, **Password**, and binds with the LDAP directory server.
    - **Authentication mode:** Select the type of authentication mode to use for authenticating the user with the LDAP directory server. **Bind as User** binds user to LDAP directory with the password provided at login in SQL DM for MySQL interface. Authentication via Comparison is done by comparing the user credentials provided at login with the LDAP directory.
    - **User search base:** Type in the User search base filter for the object class you want to filter your users for authentication.
    - **User search attribute:** Enter the attribute name that contains the user name.
    - **Search entire subtree**: This option controls the search for objects specified in user search base. Selecting this option searchs the entire subtree of **User search base**.



HOST

The name/IP address of the host of LDAP server.

ENCRYPTION: No encryption

PORT: 389

LDAP SERVER LOGIN INFORMATION

LDAP SERVER ALLOWS ANONYMOUS BIND

USER SEARCH BASE

USER DN: admin

PASSWORD: •••••••••••

LDAP USER AUTHENTICATION

AUTHENTICATION MODE: Bind as user

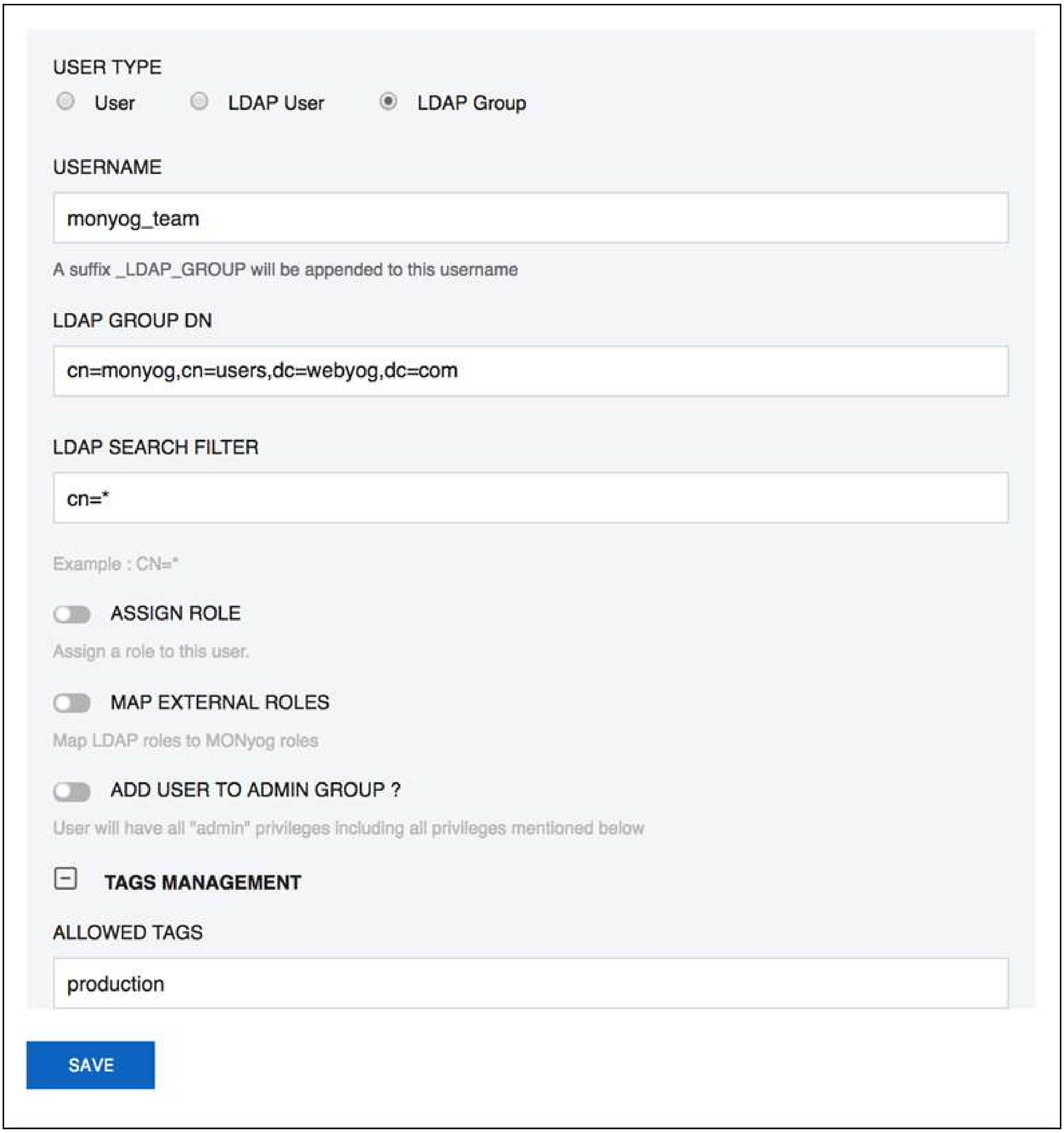
7.2.2 User Management

Using this option User Management, allows you to create, edit, and delete users.

7.2.2.1 How To Create User?

1. Click **Settings,** and select **LDAP, Users, Roles & API Token**. The window opens where you can create and delete users.
2. To create a new user under the USER tab, click **Add user**, add username, and password in the appropriate fields.
3. To add LDAP group, select **LDAP Group** from the options and specify Username, LDAP group DN, and LDAP search filter.
4. **Assign Role:** Select this option to assign SQL DM for MySQL role.

5. **External Roles:** Use this option to Map LDAP roles to SQL DM for MySQL roles.
6. **Add user to Admin group:** You can refer Managing multiple users(see page 273) for further more information.
7. **Action management:** Use this option to give different privileges like server edit, kill query, etc.
8. **T ags management:** You can give the list of allowed/disallowed tags to the user.
9. **Tab management**: An Admin user can create other users with restrictions to individual Custom Dashboards and permissions to create New Dashboards.



USER TYPE

○ User ○ LDAP User ● LDAP Group

USERNAME

monyog_team

A suffix _LDAP_GROUP will be appended to this username

LDAP GROUP DN

cn=monyog,cn=users,dc=webyog,dc=com

LDAP SEARCH FILTER

cn=*

Example : CN=*

ASSIGN ROLE

Assign a role to this user.

MAP EXTERNAL ROLES

Map LDAP roles to MONyog roles

ADD USER TO ADMIN GROUP ?

User will have all "admin" privileges including all privileges mentioned below

⊟ TAGS MANAGEMENT

ALLOWED TAGS

production

7.2.2.2 Managing multiple users

You can manage access to your servers and settings based on your needs using User Management. This feature is useful in creating users who will have limited access to the particular servers - which helps in preventing accidentally killing queries, executing FLUSH STATUS on your MySQL servers, or changing your server settings without your knowledge.

Admin

The SQL DM for MySQL admin user can now create other users having access to a subset of available servers only. Also, the Admin is the only allowed to create, delete server, and user registrations.

The Admin can create users with restrictions to individual Custom Dashboards and permissions to create New Dashboards.

Non-Admin

Following restrictions applies to non-admin users:

- Cannot register a new server.
- Cannot delete a registered server.
- Cannot change tags of a server.
- Can edit a server only if "Server Edit" permission is granted.
- Can kill a query from the 'Processlist' page only if 'Kill Query' permission is granted.
- Can execute 'FLUSH STATUS' from the Monitors page only if 'FLUSH STATUS' permission is granted.
- If no 'Allowed tags' are specified, normal users will have access to servers with no tags only.
- If the same tag is specified in 'Allowed tags' as well as 'Disallowed tags', then the user will not have access to servers with that tag.
- Cannot change user settings (except own password).
- Cannot change Preferences.

Permissions

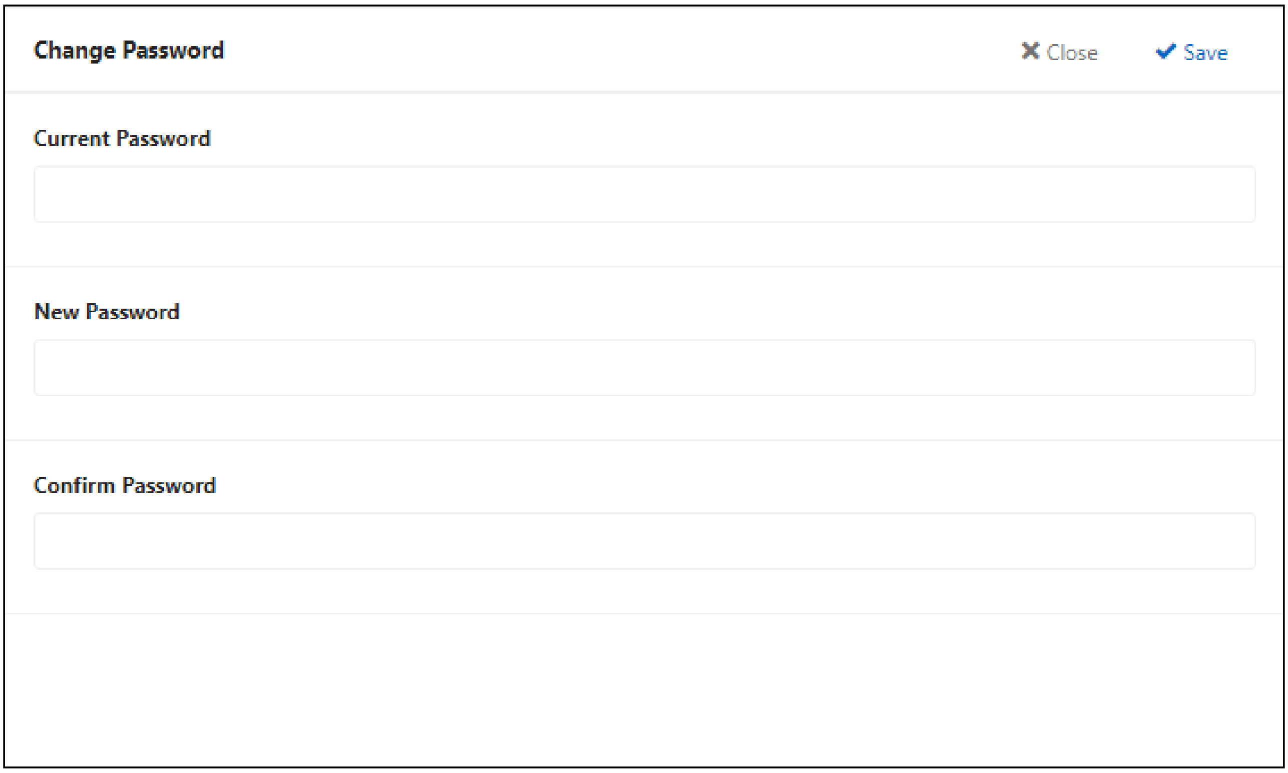
A user can be granted a combination of the following permissions:

- **Server Edit:** Allows the user to edit the settings of servers accessible to him/her.
- **Kill Query:** Allows the user to kill queries through the 'Processlist' page on servers.
- **FLUSH STATUS:** Allows the user to execute the FLUSH STATUS command on servers.
- **View Literals in Queries:** Allows the users to view literals in the Query Analyzer page.
- **Open/Close alert:** Allows the users to open/close alerts through the "Monitors" and "Events" pages.

7.2.2.3 Change Password

Users can change their password by using the Change Password option under the User Profile on the SQL DM for MySQL interface. Click the **User Profile -> Change password**.

Enter your old password in the first field. Enter your new password in the second field, and confirm the new password exactly the same way in the third field and save it.

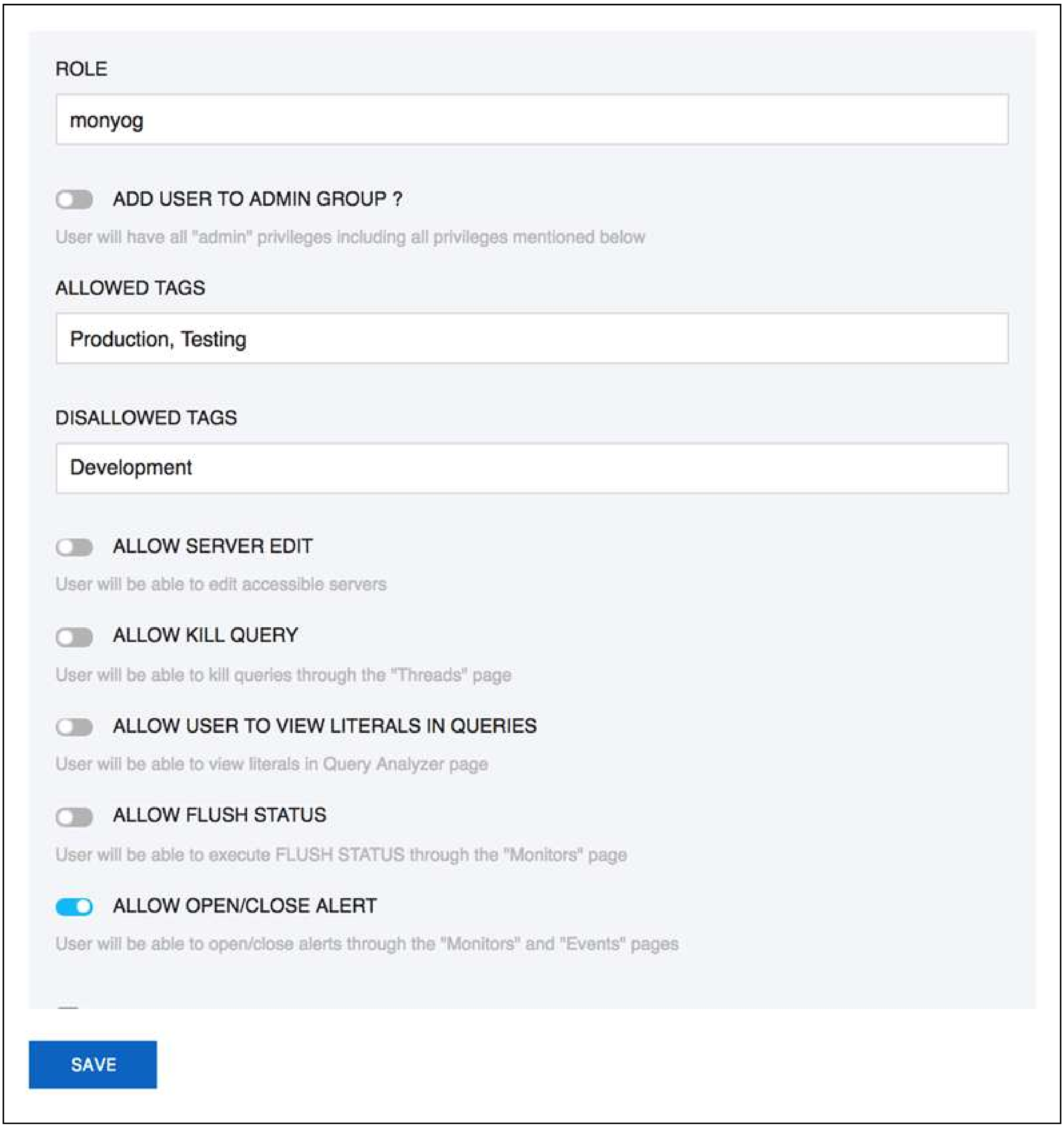


7.2.3 Role Manager

The Role Manager feature allows to create roles in SQL DM for MySQL, which can be then mapped to any users like external LDAP/AD users or the local users created in SQL DM for MySQL. The roles created can then be given different privileges like Allow server edit, Allow kill query, etc. along with the option to restrict access to selected tabs in SQL DM for MySQL.

7.2.3.1 Creating and Assigning Roles

To create a Role in SQL DM for MySQL, go to **Settings**, select **LDAP, Users, Roles & API Token**, and press **Add role.**



ROLE

monyog

○ ADD USER TO ADMIN GROUP ?

User will have all "admin" privileges including all privileges mentioned below

ALLOWED TAGS

Production, Testing

DISALLOWED TAGS

Development

○ ALLOW SERVER EDIT

User will be able to edit accessible servers

○ ALLOW KILL QUERY

User will be able to kill queries through the "Threads" page

○ ALLOW USER TO VIEW LITERALS IN QUERIES

User will be able to view literals in Query Analyzer page

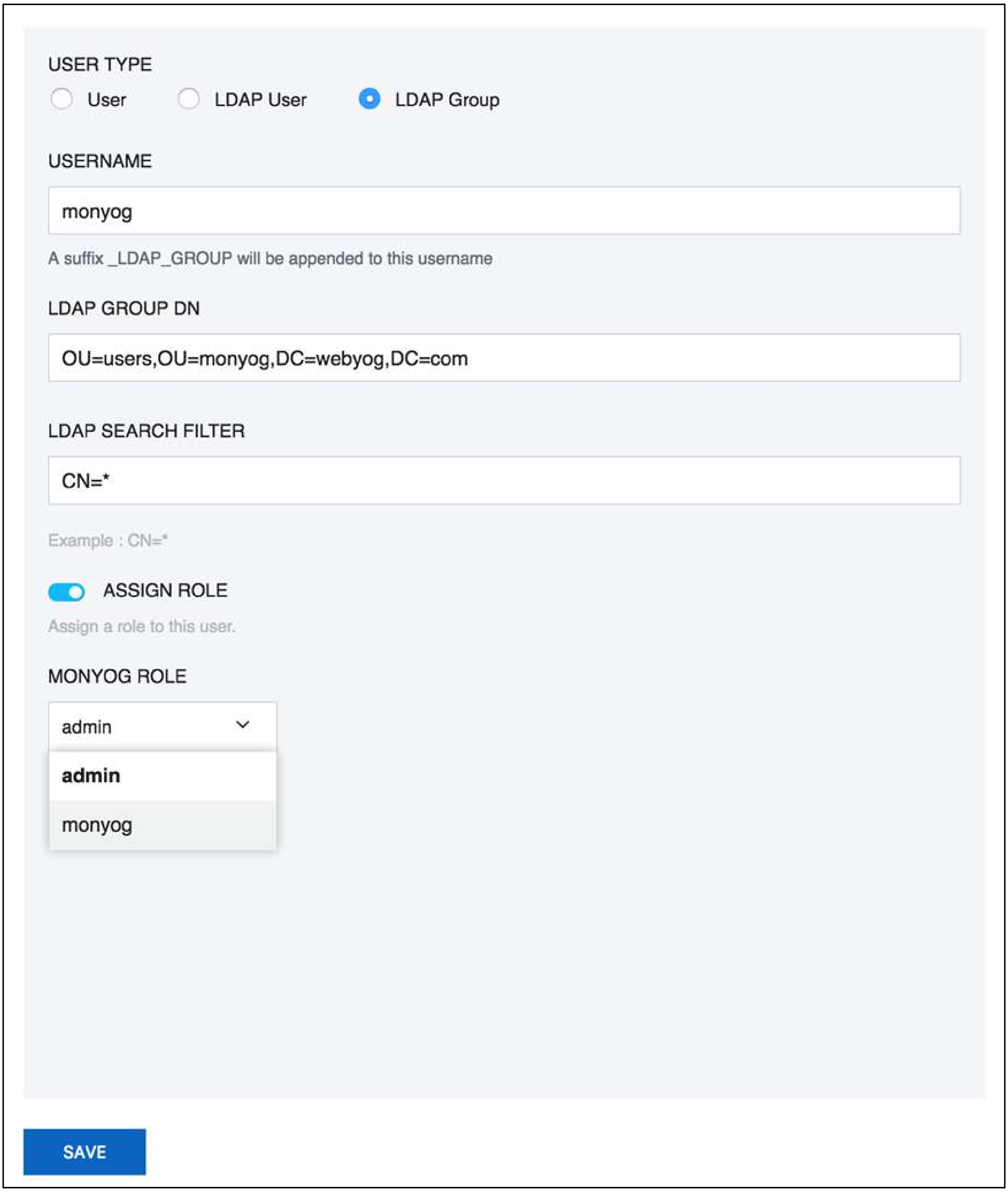
○ ALLOW FLUSH STATUS

User will be able to execute FLUSH STATUS through the "Monitors" page

● ALLOW OPEN/CLOSE ALERT

User will be able to open/close alerts through the "Monitors" and "Events" pages

Go to **Settings,** select **LDAP, Users, Roles & API Token** to create, edit a user, and assign the created role(s). Select the **Assign Role** option in the Create, Edit user pop up page, and select a role to assign from the drop down menu.



USER TYPE

○ User ○ LDAP User ● LDAP Group

USERNAME

monyog

A suffix _LDAP_GROUP will be appended to this username

LDAP GROUP DN

OU=users,OU=monyog,DC=webyog,DC=com

LDAP SEARCH FILTER

CN=*

Example : CN=*

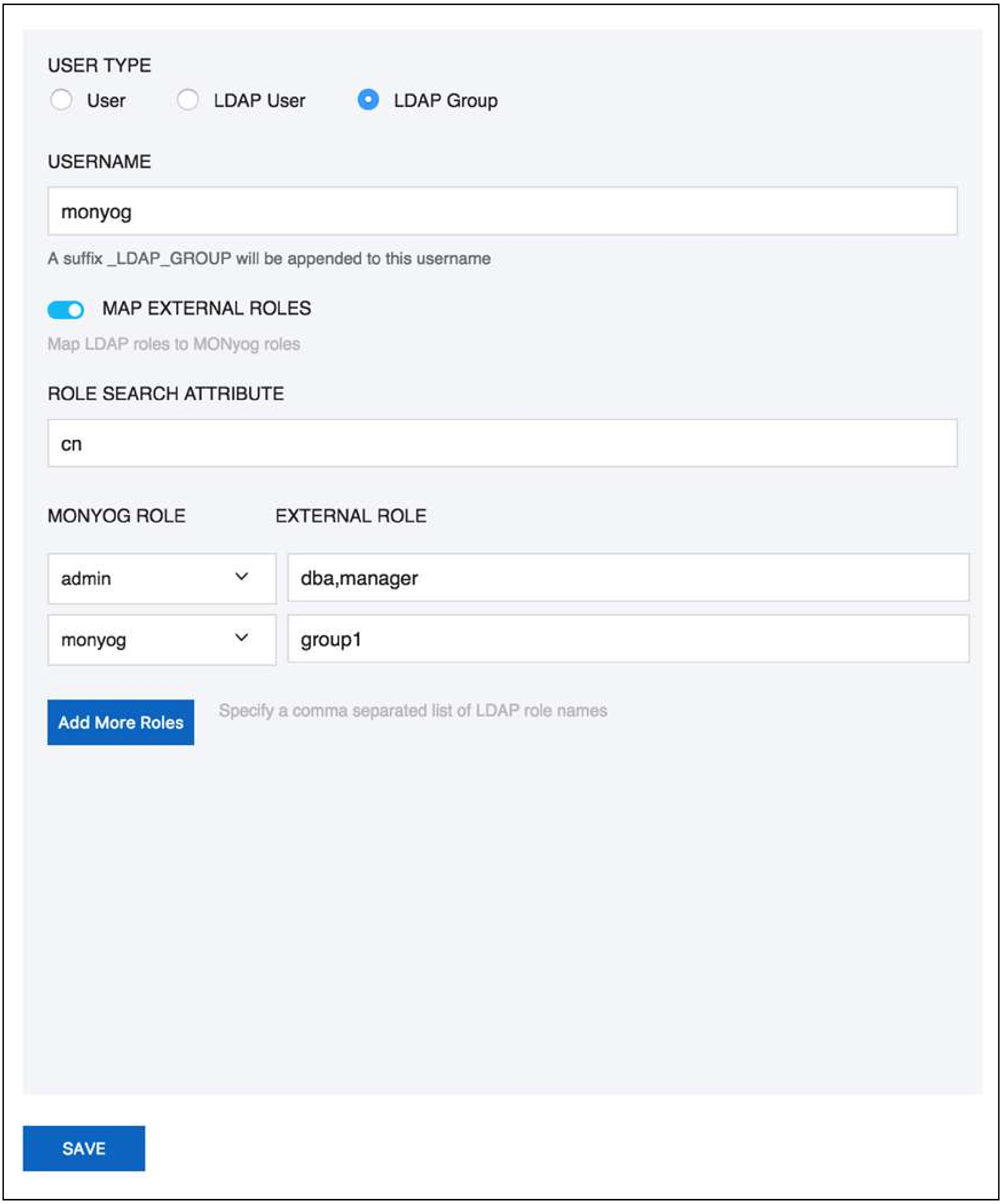
ASSIGN ROLE

Assign a role to this user.

MONYOG ROLE

admin

You can Map the LDAP group to the SQL DM for MySQL role created from the Create, Edit user pop up page, and by selecting the option **Map External Roles**. You can specify the comma separated LDAP group names and select the corresponding SQL DM for MySQL role from the drop-down menu.



USER TYPE

○ User ○ LDAP User ● LDAP Group

USERNAME

monyog

A suffix _LDAP_GROUP will be appended to this username

MAP EXTERNAL ROLES

Map LDAP roles to MONyog roles

ROLE SEARCH ATTRIBUTE

cn

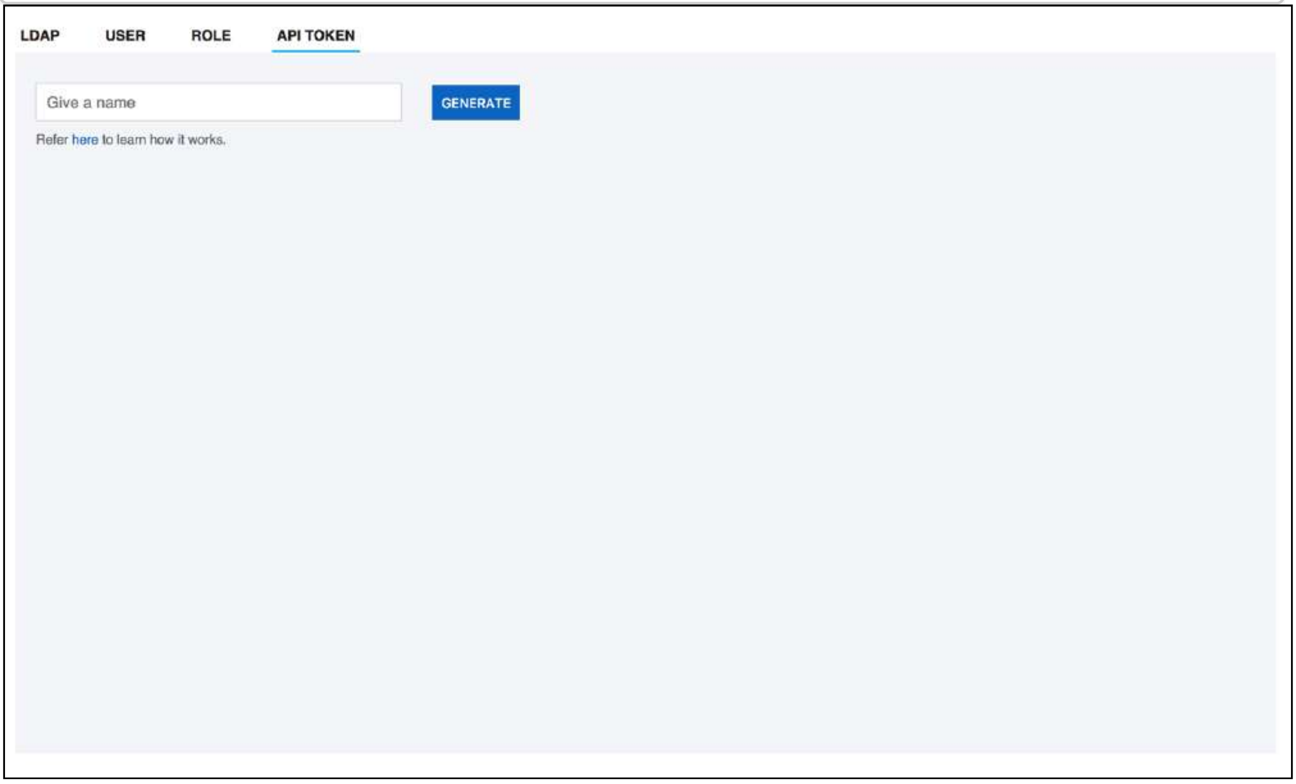
MONYOG ROLE	EXTERNAL ROLE
admin	dba,manager
monyog	group1

Specify a comma separated list of LDAP role names

7.2.4 API Token Manager

This gives an option to generate token in SQL DM for MySQL and use it in API as an alternative to user and password. This feature is available only for Admin users in SQL DM for MySQL. Admin can create multiple tokens for different purposes; like revoke or delete it from inside SQL DM for MySQL whenever required. This also helps to not share the password with anyone else as well as save it from getting logged in some logs. After clicking **GENERATE NEW TOKEN**, the user should give a name which is associated with the generated token. The generated token can be used in SQL DM for MySQL API in the following way:

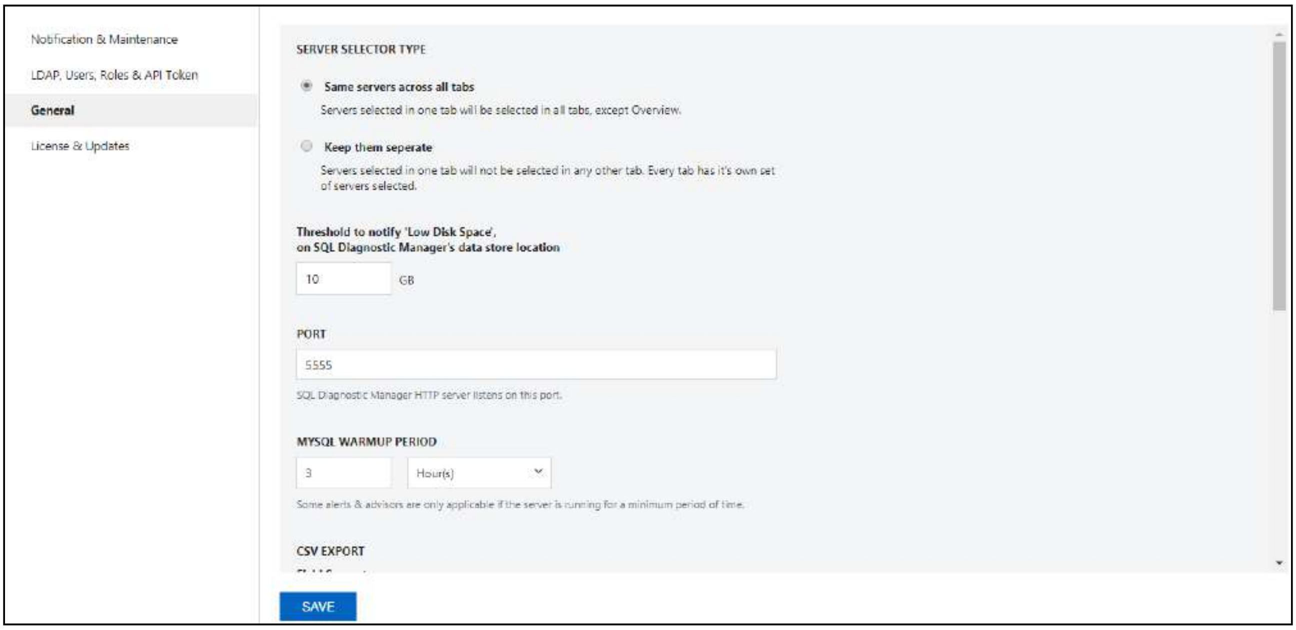
```
curl -H "X-MONYOG-TOKEN: 1234567890abcedfghijklmnopqrstuvwxyz"
"http://192.168.1.1:5555/?_object=MONyogAPI&_action=DataCollection
&_value=enable&_server=Production001"
```



Refer here to learn how it works.

7.3 General

These options provide you to set threshold for the Disk space available to SQL DM for MySQL, choose the server selector behavior in SQL DM for MySQL, and change the port for SQL DM for MySQL GUI. You can also change the settings for MySQL warmup time, exporting CSV files, maximum query length fetched by SQL DM for MySQL, and edit the colors of the charts displayed in SQL DM for MySQL.



SERVER SELECTOR TYPE

● Same servers across all tabs

Servers selected in one tab will be selected in all tabs, except Overview.

○ Keep them seperate

Servers selected in one tab will not be selected in any other tab. Every tab has it's own set of servers selected.

Threshold to notify 'Low Disk Space',
on SQL Diagnostic Manager's data store location

10 GB

PORT

5555

SQL Diagnostic Manager HTTP server listens on this port.

MYSQL WARMUP PERIOD

3 Hour(s)

Some alerts & advisors are only applicable if the server is running for a minimum period of time.

CSV EXPORT

7.3.1 SERVER SELECTOR TYPE

There are two options to switch between the behavior of Server Selector:

- **Same servers across all tabs:** If you enable this option, a server selected on one page makes it the default selected server selected across all pages in SQL DM for MySQL (excluding the Overview page). The Overview page has its own separate server selector applicable only for the page.
- **Keep them separate:** This is the default selected option. If you enable this option, a server selected on one page makes it selected only for that page, i.e, each of the pages in SQL DM for MySQL has its own server selector.

7.3.2 SQL DM for MySQL Disk Monitoring

Low disk space is a critical factor for any application. It can cause the application to perform poorly and make it difficult or unable to install the software upgrades.

SQL DM for MySQL sends SNMP trap, email alerts to all the email-IDs if the disk space where SQL DM for MySQL is installed and is running out. User can free up some disk space before the system goes out of space and shuts down. Thus maintaining high-availability of SQL DM for MySQL.

7.3.2.1 How to monitor disk space with SQL DM for MySQL?

To monitor the disk space, you need to set the threshold for the free disk space. SQL DM for MySQL sends emails to IDs mentioned under the notification settings of all servers. Alerts are also sent to email IDs provided in the SMTP settings.

The threshold limit set denotes is for the disk space available in the Data directory of SQL DM for MySQL. In case the installation directory and data directory are stored in different locations, SQL DM for MySQL alerts if the space available to the Installation directory gets below 500MB.

SQL DM for MySQL monitors the disk space on a 5-minute interval in stable conditions and 1-minute interval when the disk space goes below the threshold value.

The alert notifications are sent every 6 Hours if the disk space remains below the threshold value.

7.3.3 PORT

Using the **Change port** option, you can change the SQL DM for MySQL webserver port. By using this option, you can define on which port the SQL DM for MySQL (web) server listens. This port must be specified when connecting to SQL DM for MySQL from a browser.

Users can change the port from **General > PORT**. Once the port is changed, the user should restart the SQL DM for MySQL service for it to take effect.

7.3.4 MySQL Warmup Period

Using the Change MySQL Warmup Time option, you can change the warm-up time for MySQL servers. Some alerts and advisors are only applicable if the server is running for a minimum period of time.

1. Users can change the MySQL Warmup Time time by using the user settings screen, choose **General > MySQL Warmup Period**.
2. The drop-down can be used to display options like seconds, minutes, hours, days, etc.

7.3.5 CSV Export

Use this option to define the field delimiter for CSV export.

Users can change the CSV export settings by using **General > CSV EXPORT**. You can define the field delimiter for exporting as CSV, and the default is set to comma " , " but you can specify any other delimiters if you need.

7.3.6 Maximum Query Length

Settings -> Maximum Query Length defines the maximum query length displayed. Default is 10000 characters and max Value is 64000 characters.

The maximum query length setting here applies to queries fetched in processlist mode. In Performance Schema mode, the maximum query length is defined by the MySQL global variable performance_schema_max_digest_length. The default value is 1024 bytes and the maximum value allowed is 1048576.

> ⚠ This variable was added in MySQL 5.7.8. In MySQL 5.7.6 and 5.7.7, use max_digest_length instead. Before 5.7.6, the value cannot be changed.

7.3.7 Chart Color

Use this option to set chart colors of your choice.

7.4 License & Updates

This options helps you to review your License Details and enable check for updates automatically.

7.4.1 LICENSE

In the License Manager option you can enter your license details Registration name and License key after purchasing a license. Your license key controls how many servers are able to register in SQL DM for MySQL. If you are using a TRIAL you can register up to 512 servers, but after TRIAL expiration, a license key is required and only the number of MySQL servers that you have license for will be available. So be careful to enter a license key for a proper license before TRIAL expiration. When logging on to a unregistered SQL DM for MySQL instance running in TRIAL period a similar license manager is displayed before you have access to the logon screen. As long as the TRIAL period has not expired you can skip the registration and continue to use SQL DM for MySQL on TRIAL terms.

Enter your License Key by using the **Settings -> License & Updates**. Enter the Registration Name and the key in the appropriate fields.

7.4.2 UPDATES

Using the Update Manager option, you can check for updates manually by clicking the **Check for Updates** button and also you can select 'Yes' so that the SQL DM for MySQL will automatically check for updates once per day.

To receive a notification on SQL DM for MySQL updates automatically, set up in **Settings -> License & Updates**, and enable **Check for updates automatically?** under UPDATES.

7.5 SQL DM for MySQL Log

The SQL DM for MySQL Log has detailed records of all sorts of server errors and messages. This can help you tracking down any problems with SQL DM for MySQL.

The SQL DM for MySQL log records these types of events :

- MySQL server errors received.
- MySQL client/API errors.
- SMTP errors.
- SSH server and client errors.

To read the log just click **Settings -> Show Log** tab in the SQL DM for MySQL interface. The log is a plain text file MONyog.log which is stored in MONyog folder:

- Location in Windows 2008 : `C:\ProgramData\Webyog\Monyog\`
- Location in Windows Vista/7: `{System\_drive}:\ProgramData\Webyog\MONyog)`

7.5.1 Changing SQL DM for MySQL log path

The MONyog.log path can be changed to a desired directory from the MONyog.ini file. You just need to add the parameter “MONyogLogPath” in the MONyog.ini file and give the path to the MONyog.log file, for e.g: MONyogLogPath=/home/ubuntu/MONyogLog/

⚠ Please stop MONyog(SQL DM for MySQL) service before editing the MONyog.ini file.

7.5.2 Enabling log rotation in SQL DM for MySQL

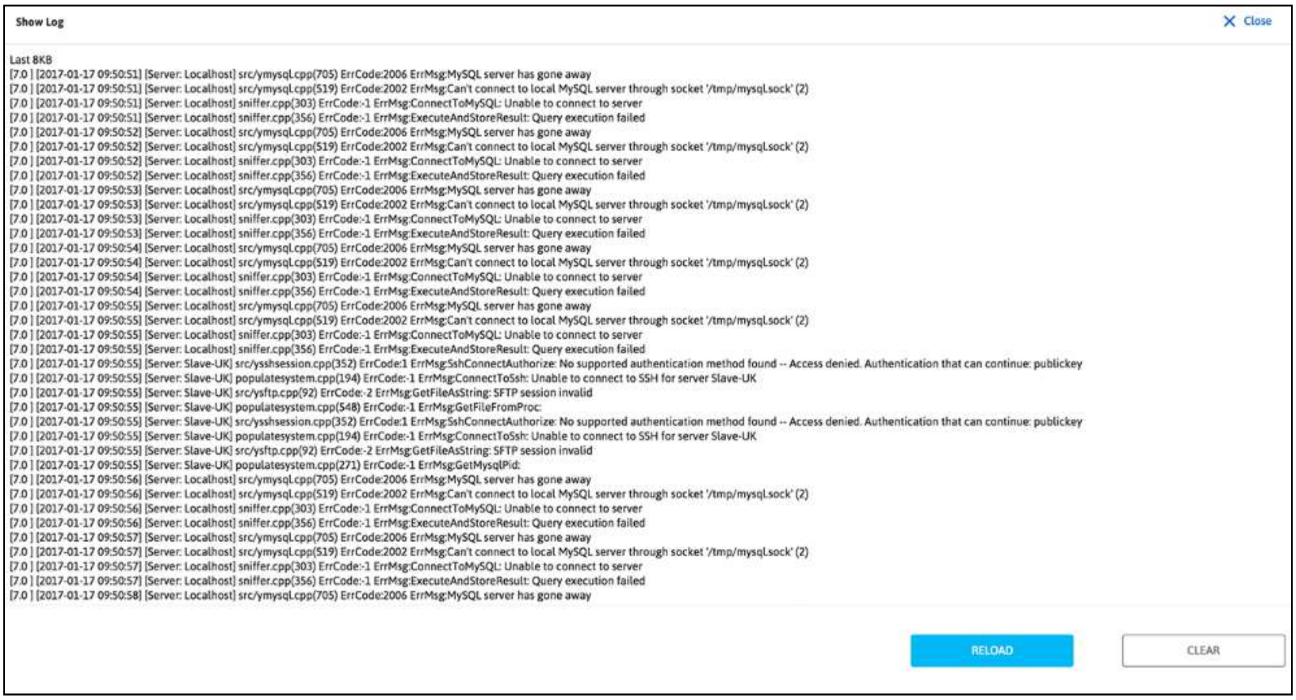
SQL DM for MySQL Linux builds ship with log rotation script to truncate MONyog.log. MONyog-logrotate is available in the installation directory and should be copied to /etc/logrotate.d/ directory.

You can edit and configure the log retention time. By default, the log rotation happens every 7 days.

> ⚠ Log rotation is not available for Windows.

7.5.3 Sample Log

The SQL DM for MySQL log has detailed records of all sorts of server errors and messages. Below, the sample log:



```
Last 8KB
[7.0 ] [2017-01-17 09:50:51] [Server: Localhost] src/ymysql.cpp(705) ErrCode:2006 ErrMsg:MySQL server has gone away
[7.0 ] [2017-01-17 09:50:51] [Server: Localhost] src/ymysql.cpp(519) ErrCode:2002 ErrMsg:Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)
[7.0 ] [2017-01-17 09:50:51] [Server: Localhost] sniffer.cpp(303) ErrCode:-1 ErrMsg:ConnectToMySQL: Unable to connect to server
[7.0 ] [2017-01-17 09:50:51] [Server: Localhost] sniffer.cpp(356) ErrCode:-1 ErrMsg:ExecuteAndStoreResult: Query execution failed
[7.0 ] [2017-01-17 09:50:52] [Server: Localhost] src/ymysql.cpp(705) ErrCode:2006 ErrMsg:MySQL server has gone away
[7.0 ] [2017-01-17 09:50:52] [Server: Localhost] src/ymysql.cpp(519) ErrCode:2002 ErrMsg:Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)
[7.0 ] [2017-01-17 09:50:52] [Server: Localhost] sniffer.cpp(303) ErrCode:-1 ErrMsg:ConnectToMySQL: Unable to connect to server
[7.0 ] [2017-01-17 09:50:52] [Server: Localhost] sniffer.cpp(356) ErrCode:-1 ErrMsg:ExecuteAndStoreResult: Query execution failed
[7.0 ] [2017-01-17 09:50:53] [Server: Localhost] src/ymysql.cpp(705) ErrCode:2006 ErrMsg:MySQL server has gone away
[7.0 ] [2017-01-17 09:50:53] [Server: Localhost] src/ymysql.cpp(519) ErrCode:2002 ErrMsg:Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)
[7.0 ] [2017-01-17 09:50:53] [Server: Localhost] sniffer.cpp(303) ErrCode:-1 ErrMsg:ConnectToMySQL: Unable to connect to server
[7.0 ] [2017-01-17 09:50:53] [Server: Localhost] sniffer.cpp(356) ErrCode:-1 ErrMsg:ExecuteAndStoreResult: Query execution failed
[7.0 ] [2017-01-17 09:50:54] [Server: Localhost] src/ymysql.cpp(705) ErrCode:2006 ErrMsg:MySQL server has gone away
[7.0 ] [2017-01-17 09:50:54] [Server: Localhost] src/ymysql.cpp(519) ErrCode:2002 ErrMsg:Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)
[7.0 ] [2017-01-17 09:50:54] [Server: Localhost] sniffer.cpp(303) ErrCode:-1 ErrMsg:ConnectToMySQL: Unable to connect to server
[7.0 ] [2017-01-17 09:50:54] [Server: Localhost] sniffer.cpp(356) ErrCode:-1 ErrMsg:ExecuteAndStoreResult: Query execution failed
[7.0 ] [2017-01-17 09:50:55] [Server: Localhost] src/ymysql.cpp(705) ErrCode:2006 ErrMsg:MySQL server has gone away
[7.0 ] [2017-01-17 09:50:55] [Server: Localhost] src/ymysql.cpp(519) ErrCode:2002 ErrMsg:Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)
[7.0 ] [2017-01-17 09:50:55] [Server: Localhost] sniffer.cpp(303) ErrCode:-1 ErrMsg:ConnectToMySQL: Unable to connect to server
[7.0 ] [2017-01-17 09:50:55] [Server: Localhost] sniffer.cpp(356) ErrCode:-1 ErrMsg:ExecuteAndStoreResult: Query execution failed
[7.0 ] [2017-01-17 09:50:55] [Server: Slave-UK] src/ysshsession.cpp(352) ErrCode:1 ErrMsg:SshConnectAuthorize: No supported authentication method found -- Access denied. Authentication that can continue: publickey
[7.0 ] [2017-01-17 09:50:55] [Server: Slave-UK] populatesystem.cpp(194) ErrCode:-1 ErrMsg:ConnectToSsh: Unable to connect to SSH for server Slave-UK
[7.0 ] [2017-01-17 09:50:55] [Server: Slave-UK] src/ysftp.cpp(92) ErrCode:-2 ErrMsg:GetFileAsString: SFTP session invalid
[7.0 ] [2017-01-17 09:50:55] [Server: Slave-UK] populatesystem.cpp(548) ErrCode:-1 ErrMsg:GetFileFromProc:
[7.0 ] [2017-01-17 09:50:55] [Server: Slave-UK] src/ysshsession.cpp(352) ErrCode:1 ErrMsg:SshConnectAuthorize: No supported authentication method found -- Access denied. Authentication that can continue: publickey
[7.0 ] [2017-01-17 09:50:55] [Server: Slave-UK] populatesystem.cpp(194) ErrCode:-1 ErrMsg:ConnectToSsh: Unable to connect to SSH for server Slave-UK
[7.0 ] [2017-01-17 09:50:55] [Server: Slave-UK] src/ysftp.cpp(92) ErrCode:-2 ErrMsg:GetFileAsString: SFTP session invalid
[7.0 ] [2017-01-17 09:50:55] [Server: Slave-UK] populatesystem.cpp(271) ErrCode:-1 ErrMsg:GetMysqlPid:
[7.0 ] [2017-01-17 09:50:56] [Server: Localhost] src/ymysql.cpp(705) ErrCode:2006 ErrMsg:MySQL server has gone away
[7.0 ] [2017-01-17 09:50:56] [Server: Localhost] src/ymysql.cpp(519) ErrCode:2002 ErrMsg:Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)
[7.0 ] [2017-01-17 09:50:56] [Server: Localhost] sniffer.cpp(303) ErrCode:-1 ErrMsg:ConnectToMySQL: Unable to connect to server
[7.0 ] [2017-01-17 09:50:56] [Server: Localhost] sniffer.cpp(356) ErrCode:-1 ErrMsg:ExecuteAndStoreResult: Query execution failed
[7.0 ] [2017-01-17 09:50:57] [Server: Localhost] src/ymysql.cpp(705) ErrCode:2006 ErrMsg:MySQL server has gone away
[7.0 ] [2017-01-17 09:50:57] [Server: Localhost] src/ymysql.cpp(519) ErrCode:2002 ErrMsg:Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)
[7.0 ] [2017-01-17 09:50:57] [Server: Localhost] sniffer.cpp(303) ErrCode:-1 ErrMsg:ConnectToMySQL: Unable to connect to server
[7.0 ] [2017-01-17 09:50:57] [Server: Localhost] sniffer.cpp(356) ErrCode:-1 ErrMsg:ExecuteAndStoreResult: Query execution failed
[7.0 ] [2017-01-17 09:50:58] [Server: Localhost] src/ymysql.cpp(705) ErrCode:2006 ErrMsg:MySQL server has gone away
```

7.5.3.1 How helpful it is?

A user does not have to keep his attention fixed to the SQL DM for MySQL interface all the time.

It basically logs the critical places and network interfaces problems faced by SQL DM for MySQL. Like, if for some reason Mail Alert has not been send, or MySQL server is down, etc.

8 Troubleshooting

8.1 Troubleshooting

Having trouble? See if we can help you out with your issue:

8.2 I am not able to connect to MySQL

The error message: `"Error No. 2003: Can't connect to MySQL server on 'localhost' (or some other host)"`

Simply means that connection is not possible for one of the following (or similar) reasons:

- There is no MySQL server running at the specified host.
- Connection to the MySQL server is not allowed using TCP/IP. Check the 'skip-networking' setting in the MySQL configuration file (my.ini on Windows, my.cnf on Unix/Linux). It shall be commented out like '#skip-networking'. If it is not commented out, then do it and restart the MySQL server for the change to take effect. SQL DM for MySQL needs to connect using TCP/IP.
- When trying to connect to a MySQL server at an ISP this error message often indicates that direct connection to MySQL has been blocked. You must then use SSH-tunneling to connect.
- Some networking issue prevents connection. It could be a network misconfiguration or a firewall issue. We have experienced sometimes that some firewalls (ZoneAlarm in particular) is blocking TCP/IP connections even if it claims to be disabled. Most often it will help to uninstall and reinstall the firewall.

8.3 I am not able to view the SQL DM for MySQL home page

Simply means that connection is not possible for one of the following (or similar) reasons:

- Wrong details (port for instance)
- Firewall

8.4 I can connect to MySQL on the host but not able to retrieve OS data

SQL DM for MySQL can retrieve OS data from Linux operations systems if SSH shell access is possible or configured to that system. Note that SQL DM for MySQL can do this no matter on what Operating System SQL DM for MySQL itself is installed.

8.5 In the Opera browser the red and yellow indicators in 'Monitors' interface do not update correctly.

You need to always configure Opera to check if graphics was updated. **Opera Tools > Preferences > Advanced > History**. Opera alone has this option. This is a performance optimization that may be OK with largely static webpages that do not work properly with SQL DM for MySQL webpages. Set Check Images to Always.

8.6 SQL DM for MySQL History/Trend graphics display partly outside visible screen area

On some systems that use a wide-screen monitor, earlier versions of the Opera browser (< 8.5) is not obeying the javascript command to open a new window for the display of a History/TRENDs graph. It opened in an ordinary tab instead. That resulted in the lower part of the graph to be invisible with relatively low screen resolutions (including the popular widescreen resolution 900*1440 (very common with laptops designed for Windows Vista)). You should upgrade Opera.

8.7 Getting mysql.sock error

This error comes up because, if you give 'localhost' as the host, mysql client library trying to use the unix domain socket (file based) instead of the TCP one. And, every mysql client library has one path to the Unix domain socket.

- In SQL DM for MySQL's mysql client library it is: '/var/lib/mysql/mysql.sock'
- In "phpMyAdmin", it is: '/var/run/mysqld/mysqld.sock'

So, if one software works the other breaks. To solve this problem review the following:

- You can create a symbolic link to your original '/var/run/mysqld/mysqld.sock' in '/var/lib/mysql/mysql.sock' '/var/lib/mysql/mysql.sock' -> '/var/run/mysqld/mysqld.sock'

The command to create this is:

```
$ ln -s
/var/run/mysqld/mysqld.sock /var/lib/mysql/mysql.sock
```

This creates a symbolic link '/var/lib/mysql/mysql.sock' to '/var/run/mysqld/mysqld.sock'. So both applications work.

- You can keep '/var/run/mysqld/mysqld.sock' as it is. And, you can force SQL DM for MySQL to use TCP based connection specifying "127.0.0.1" as the host instead of 'localhost'.

8.8 Tunneling works for me but I cannot get system counters

I can connect to my FTP server, but could not use that same user to get system counters. Even, I can use the same user in SQL DM for MySQL to tunnel to MySQL server. To collect system counters, the SSH user should have access to a shell. This is specified usually in '/etc/passwd' file. But to tunnel to MySQL server the SSH user does not need a shell access. You can check the shell access which is given in /etc/passwd file.

For FTP users generally shell access will be blocked. A typical FTP user has an entry something like this in the '/etc/passwd' file:

```
john:x:10009:10001:Jonathan:/var/www/vhosts/yourdomain.com:/bin/false
```

A colon (:) is used to separate the fields. All the fields are explained below:

- **john**: The login name.
- **x**: An x indicates that the encrypted password for this user is kept in /etc/shadow file
- **10009**: The unique User ID for this user.
- **10001**: The primary Group ID for this user.
- **Jonathan**: This is the User ID info. Generally comments, full name of the user, etc.
- **/var/www/vhosts/yourdomain.com**: The home directory for this FTP user.
- **/bin/false**: This field indicates the shell access given to the user. And you can see that no shell access has been given to 'john'. It is /bin/false.

This user 'john' can not use a shell. That is the reason ’john’ can not be used for collecting system counters. But user 'john' can be used for tunneling to MySQL server.

A typical user entry with shell access has an entry in /etc/passwd something like:

```
jenny:x:10002:10003:Jenniffer:/home/jenny:/bin/bash
```

8.9 Key based authentication does not work with SQL DM for MySQL

Please note that for key based authentication SQL DM for MySQL supports only OpenSSH specified standard key format for public and private keys. SQL DM for MySQL does not work with the keys generated by other SSH related products. This list includes but not limited to key pairs generated by:

- Puttygen
- SecureCRT

8.10 MONyog-bin not found on MONyog START command

There may be two reasons for this error:

1. Path is invalid. When using the .gz-compressed build for Linux you start the MONyog(SQL DM for MySQL) service like "{path to} MONyog START". If you are executing from the MONyog folder yourself you need to write "./MONyog START". "./" means 'current folder' and on most Linux 'current folder' is not in PATH environmental variable, so you have to specify it.
2. You are trying to run a 64-bit build on a 32-bit platform or vice-versa. The MONyog-bin file is not recognized by the OS as a valid binary. You should use the 32-bit build on 32-bit OS's and the 64-bit build on 64-bit OS's

(however support for 32 bit binaries may be configured on 64 bit Linux's, but often it is not the case with DEBIAN based Linux distros - including (k/x)Ubuntu's).

8.11 'Test Path' button in Query Analyzer settings keeps throwing 'File Path Invalid' error

Problem: I have a log in a shared folder. I am able to access that log myself, but press **test path** button in Query Analyzer settings keeps throwing 'File Path Invalid' error.

Solution: By default SQL DM for MySQL installs on Windows with the privileges of the local system account. It does not automatically give you access to shared folders located on other systems. In the Windows service manager (Control Panel -> Administrative Tools -> Services) locate the MONyog(SQL DM for MySQL) service and select **Properties** from the context (right-click) menu. In the Log On tab, select an account that has sufficient privileges to access the shared folder on the remote system.

8.12 Authentication problem with key authentication

To resolve an error like the following:

"Failed to connect to SFTP: Error: offering public key failed, access denied, authentications that can continue: ...." or similar, check the following workaround:

1. Ensure that whatever the public key content you pasted in authorized_keys is same 'byte by byte' that you paste in SQL DM for MySQL public key field.
2. Ensure that you are trying with the same username. This is a common mistake to add keys for one user authorized_keys and trying to connect with another user.
3. The authorized_keys file should have permission 600. That is, read/write permission only to the owner.
4. Finally, ensure that the keys are standard OpenSSH keys not any proprietary or application-specific format.

8.13 When I fetched Details from MySQL, Log file Path is not showing?

Log file path could be fetched if the MySQL server you are registering is greater than 5.1.6. If your server is lesser than this, then all fields would be read only and you have entered the slow query log/general query log path yourself.

8.14 CSV file is not displaying correct results for "Query Execution Time" columns

Say for example, in the Query Analyzer Page, the Total column value is 2:16:5.282 but in CSV it is displayed as 16:5.282.

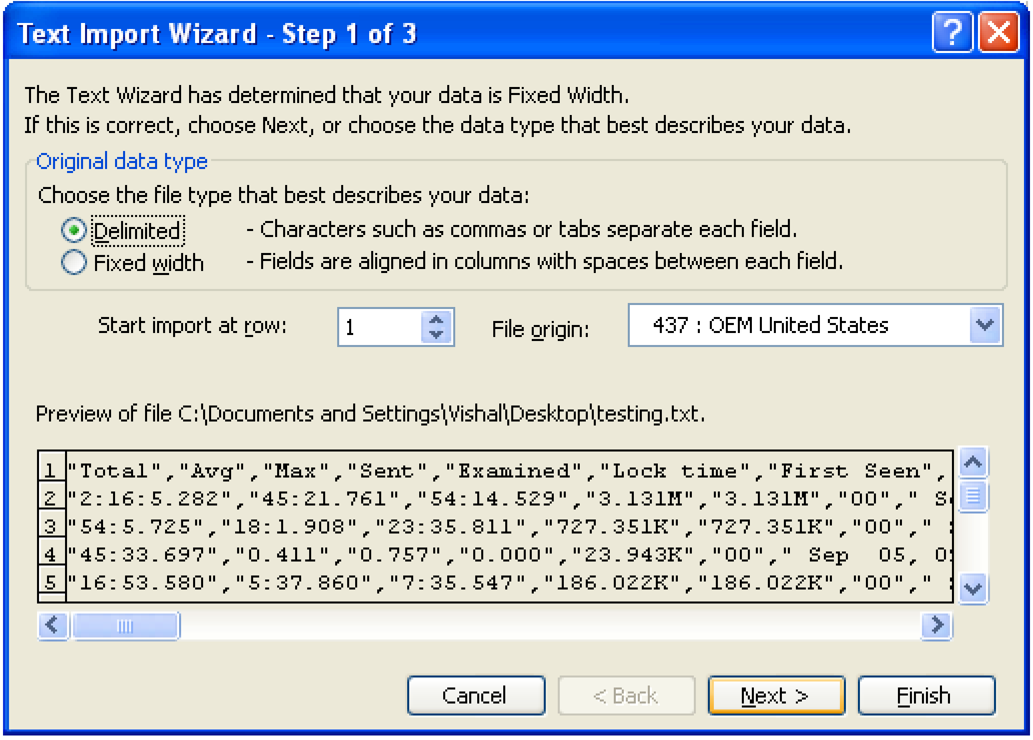
This is a formatting problem in Excel instead if you open the export in Notepad or any other editor this problem will not persist. To view in Excel follow these steps:

- Save the .csv file on disk.
- Import the .csv file in Excel as explained below. (Note: Set the Data Format for columns containing time values to Text).

8.15 How to import a text file into Excel?

After you have started Excel (this FAQ uses Excel 2003), follow these steps:

1. On the Data menu, point to **Import External Data**, and then click **Import Data**.
2. In the Files of type dialog, click **Text Files**.
3. In the Look in list, locate and double-click the text file you want to import.
4. Select **Delimited** option and click **Next**.



The Text Wizard has determined that your data is Fixed Width.
If this is correct, choose Next, or choose the data type that best describes your data.

Original data type

Choose the file type that best describes your data:

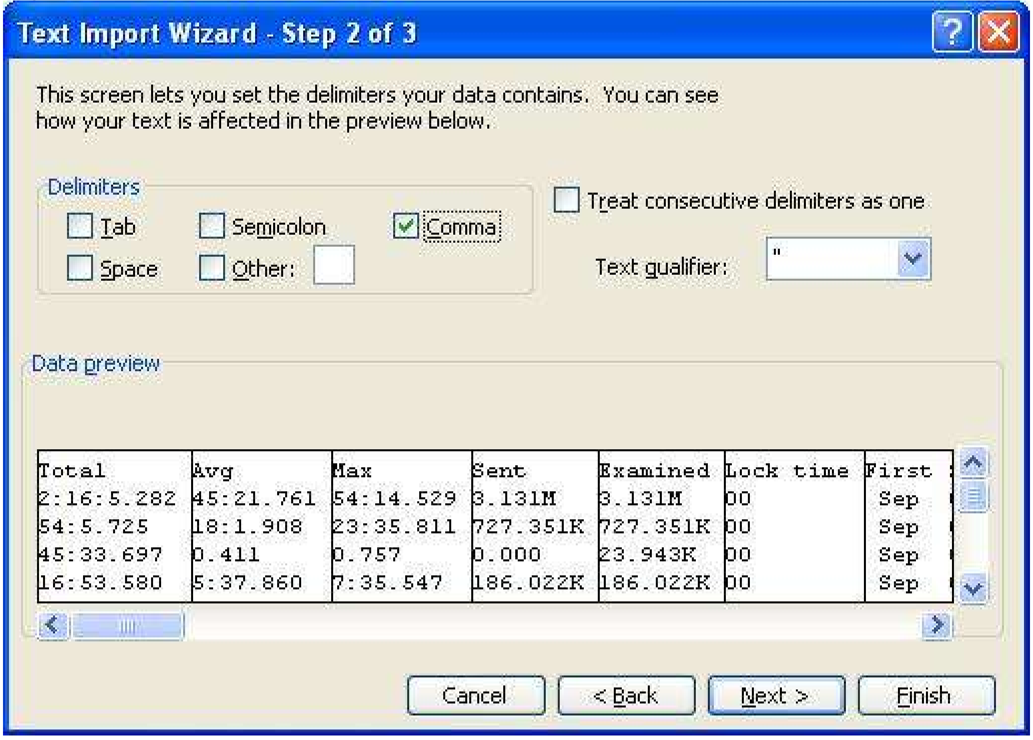
● Delimited - Characters such as commas or tabs separate each field.
○ Fixed width - Fields are aligned in columns with spaces between each field.

Start import at row: 1 File origin: 437 : OEM United States

Preview of file C:\Documents and Settings\Vishal\Desktop\testing.txt.

```
1 "Total","Avg","Max","Sent","Examined","Lock time","First Seen",
2 "2:16:5.282","45:21.761","54:14.529","3.131M","3.131M","00"," S
3 "54:5.725","18:1.908","23:35.811","727.351K","727.351K","00","
4 "45:33.697","0.411","0.757","0.000","23.943K","00"," Sep  05, 0
5 "16:53.580","5:37.860","7:35.547","186.022K","186.022K","00","
```

5. Set Delimiter to your locale-specific delimiter (COMMA " , " for English and SEMICOLON " ; " for most non-English) as you should have already defined in Preferences. Click **Next**.



Text Import Wizard - Step 2 of 3

This screen lets you set the delimiters your data contains. You can see how your text is affected in the preview below.

Delimiters

□ Tab □ Semicolon ☑ Comma

□ Space □ Other:

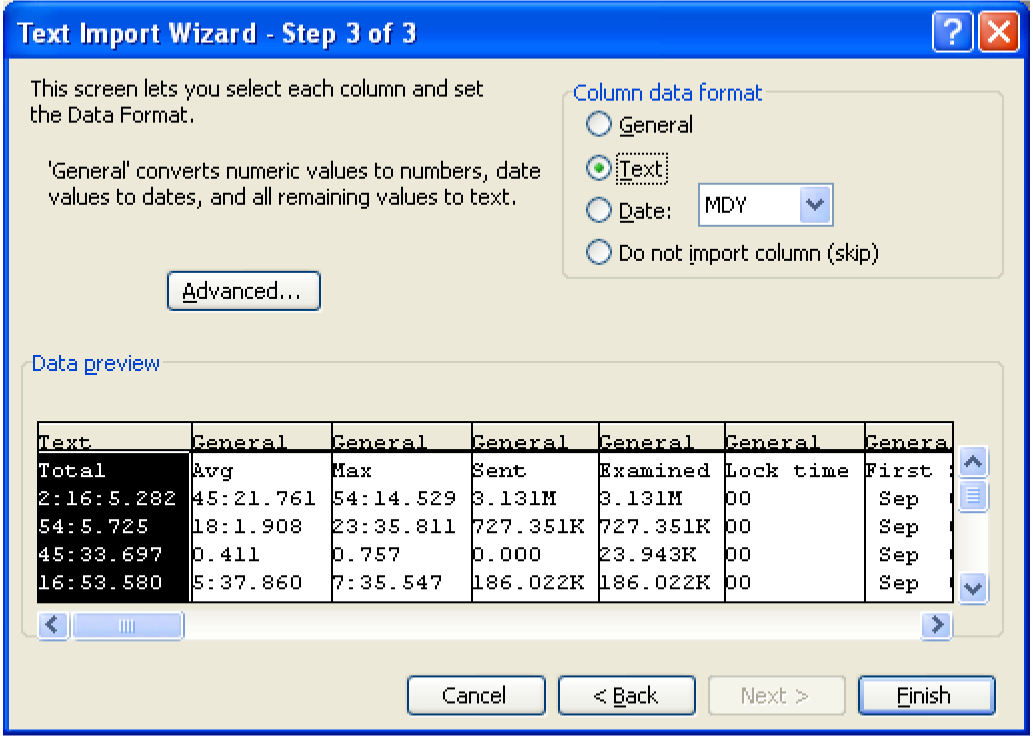
□ Treat consecutive delimiters as one

Text qualifier: "

Data preview

Total	Avg	Max	Sent	Examined	Lock time	First
2:16:5.282	45:21.761	54:14.529	3.131M	3.131M	00	Sep
54:5.725	18:1.908	23:35.811	727.351K	727.351K	00	Sep
45:33.697	0.411	0.757	0.000	23.943K	00	Sep
16:53.580	5:37.860	7:35.547	186.022K	186.022K	00	Sep

6. Select each column and set the Data Format appropriately and then click **Finish**.



Text Import Wizard - Step 3 of 3

This screen lets you select each column and set the Data Format.

'General' converts numeric values to numbers, date values to dates, and all remaining values to text.

Column data format

○ General

● Text

○ Date: MDY

○ Do not import column (skip)

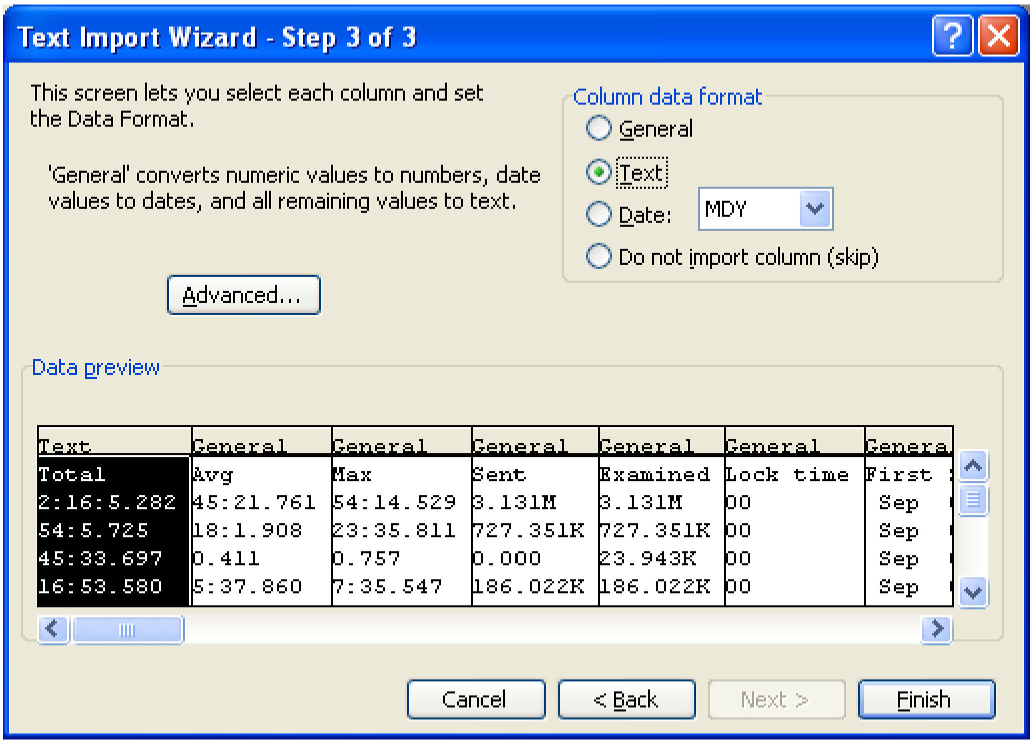
Data preview

Text	General	General	General	General	General	Genera
Total	Avg	Max	Sent	Examined	Lock time	First
2:16:5.282	45:21.761	54:14.529	3.131M	3.131M	00	Sep
54:5.725	18:1.908	23:35.811	727.351K	727.351K	00	Sep
45:33.697	0.411	0.757	0.000	23.943K	00	Sep
16:53.580	5:37.860	7:35.547	186.022K	186.022K	00	Sep

In the Import Data dialog, either one of the following options should be performed:

- To return the data to the location you selected, click **Existing worksheet**, and then click **OK**.

- To return the data to a new worksheet, click **New worksheet**, and then click **OK**. Microsoft Excel adds a new worksheet to your workbook and automatically puts the external data range in the upper-left corner of the new worksheet.



Text Import Wizard - Step 3 of 3

This screen lets you select each column and set the Data Format.

'General' converts numeric values to numbers, date values to dates, and all remaining values to text.

Column data format

○ General

● Text

○ Date: MDY

○ Do not import column (skip)

Data preview

Text	General	General	General	General	General	Genera
Total	Avg	Max	Sent	Examined	Lock time	First :
2:16:5.282	45:21.761	54:14.529	3.131M	3.131M	00	Sep
54:5.725	18:1.908	23:35.811	727.351K	727.351K	00	Sep
45:33.697	0.411	0.757	0.000	23.943K	00	Sep
16:53.580	5:37.860	7:35.547	186.022K	186.022K	00	Sep

9 FAQ

9.1 Table of content:

9.1.1 1. SQL DM for MySQL is licensed per server. Does that mean SQL DM for MySQL servers or MySQL servers?

It means MySQL servers. You may install as many instances of SQL DM for MySQL as you like as long as the total number of MySQL servers monitored does not exceed your license.

TOP(see page 290)

9.1.2 2. Do I need to install SQL DM for MySQL on the same host as MySQL?

MySQL servers, SQL DM for MySQL server(s) and Clients (browser) can be installed independently everywhere where a TCP connection (like an Internet/Intranet connection) is available. Available TCP connections is all that is required. Regarding installing SQL DM for MySQL on the same host as MySQL then SQL DM for MySQL Connects to/ Monitors MySQL running on any platform.

TOP(see page 290)

9.1.3 3. What operating system does SQL DM for MySQL require?

Currently we support Windows (do not support Windows 2003) and Linux operating systems. Those are the Operating Systems where SQL DM for MySQL itself needs to be installed in. The SQL DM for MySQL client functionalities only require an Internet browser and any platform (including platforms for handheld devices like mobile phone, PDA, tablet PC, etc.). There are also no restriction as regards the platforms the MySQL servers that SQL DM for MySQL connects to - it can be any. Additionally, SQL DM for MySQL is able to retrieve OS data from Linux Operation Systems.

TOP(see page 290)

9.1.4 4. How do I upgrade SQL DM for MySQL?

This is actually a license-related question and a technical question as well.

- **License:** SQL DM for MySQL ships with 1 year of free upgrades. After that you will be offered an upgrade with discount. Our website always tells the terms and conditions. Also our website has a Portal for registered users from where you can download free upgrades and purchase upgrades after the expiry of the free upgrade period.
- **Technical:** The automatic installers (the Windows version and the RPM build for Redhat type Linux) handles everything automatically. The gz-compressed build for other Linux's requires that you run execute a few installation scripts from a command shell. We constantly improve and simplify this. After extracting the tar.gz package, you get a file called README. Please refer to that file for details.
- **Need to monitor more servers:** You can upgrade anytime your SQL DM for MySQL installation from monitoring a certain number of servers to higher numbers by opening the License Manager and enter a new license key. If you need this please first contact us through our ticket system. We will consider the value of your existing license and compensate you (details depend on what license you have and what you need and how old your existing license is).

⚠ **Note**

Do not forget to backup whatever JavaScript you have edited, as they get overwritten when you upgrade. You can take a back-up of Counters.def and Udo.def located inside SQL DM for MySQL folder: MONyog for this. Alternatively, you can directly upgrade. After upgrade you get your JS changes as a conflict in SQL DM for MySQL UI, you can resolve those and keep your changes.

TOP(see page 290)

9.1.5 5. Why should I upgrade?

Every new release adds features, fixes bugs, improves performance, stability, the GUI interface etc. Why should you NOT upgrade? Refer to Version History for details.

TOP(see page 290)

9.1.6 6. I have installed SQL DM for MySQL. What now? How do I get the reports?

What you installed was the MONyog(SQL DM for MySQL) service. This service is basically a webserver program. You connect to the service with an Internet browser by specifying the host where it is installed and the port that was specified when installing.

TOP(see page 290)

9.1.7 7. How can SQL DM for MySQL 'know all what it does'?

SQL DM for MySQL queries the MySQL servers about almost anything the server 'knows' except for data stored on those servers. The MySQL servers themselves store and maintain records of server configuration, users, history and much more. SQL DM for MySQL retrieves this information, organizes it, calculates on it and reports.

TOP(see page 290)

9.1.8 8. Where does SQL DM for MySQL store data?

This depends on the Operating system.

- Windows 2008:

Data folder:

```
C:\ProgramData\Webyog\MONyog\Data
```

MONyog.ini + MONyog.log + preferences.config are in:

```
C:\ProgramData\Webyog\MONyog\
```

- Windows Vista:

Data folder:

```
Windows Vista:{System_drive}:\ProgramData\Webyog\MONyog\Data
```

MONyog.ini + MONyog.log + preferences.config are in:>

```
Windows Vista:{System_drive}:\ProgramData\Webyog\MONyog
```

- Linux RPM build:

Data folder: The connection configuration and collected data are kept here. You can find directories named like 0001, 0002, 0003, etc.

```
/usr/local/MONyog/data/
```

Installation file:

```
/usr/local/MONyog/MONyog.ini
```

Log file:

```
/usr/local/MONyog/MONyog.log
```

Configuration file:

```
/usr/local/MONyog/preferences.config
```

- Linux .gz archive:

If you have extracted SQL DM for MySQL package in a directory called MONyog the data stored is in:

Data folder: The connection configuration and collected data are kept here. You can find directories named like 0001, 0002, 0003, etc.

```
MONyog/data/
```

Installation file:

```
MONyog/MONyog.ini
```

Log file:

```
MONyog/MONyog.log
```

Configuration file:

```
MONyog/preferences.config
```

The data folder specified above is the default settings only. You can store in any position on any mapped/mounted drive that is writable.

TOP(see page 290)

9.1.9 9. How does SQL DM for MySQL store its data?

Except for two plain text files: the MONyog.log file and a very small .ini file (that contains information about the port on which SQL DM for MySQL listens, The SQL DM for MySQL administrator password and the path to the data folder), everything is kept in high-performance database files (SQLite format).

TOP(see page 290)

9.1.10 10. Can I move a SQL DM for MySQL installation to another computer while keeping the data stored in SQL DM for MySQL database?

Yes. Just install SQL DM for MySQL on the 2nd machine. After install, stop the running MONyog(SQL DM for MySQL) service and copy the ..\MONyog\Data folder from the old installation. You may also copy the MONyog.log if you want. All the connection configuration and the data is located in 'data' directory. The error log is MONyog.log and settings are stored in 'MONyog.ini' and 'preferences.config'. If you have made any changes to monitors they are stored in 'Counters.def' and for CSO's in 'udo.def'. Copy all of them from your old installation onto new PC. After that start the service again.

TOP(see page 290)

9.1.11 11. Can SQL DM for MySQL be configured as a virtual host in my 'ordinary' Apache webserver?

Yes, at least with Apache this is possible. In your Apache configuration file (httpd.conf) add something like this (where 'ip1.ip2.ip3.ip4' is the IP address you reserve for SQL DM for MySQL).

```
<VirtualHost *:80>
ServerName monyog.mydomain.com
ServerAlias http://monyog.mydomain.com
Redirect permanent / https://monyog.mydomain.com
</VirtualHost>
NameVirtualHost *:443
<VirtualHost *:443>
ServerName monyog.mydomain.com
ProxyPreserveHost On
ProxyPass / http://127.0.0.1:<MONyog-Port>/
ProxyPassReverse / http://127.0.0.1:<MONyog-Port>/
SSLEngine On
SSLCertificateFile <path-to-ssl-certificate.crt>
SSLCertificateKeyFile <path-to-ssl-key.key>
</VirtualHost>
```

And run the following command on the machine running Apache server:

```
/usr/sbin/setsebool httpd_can_network_connect=1
```

After changing the configuration, restart the Apache server.

TOP(see page 290)

9.1.12 12. How can I access SQL DM for MySQL pages proxying through other webservers?

We can also access SQL DM for MySQL using Apache proxy. You need to follow these simple steps to configure your Apache server to support proxy.

Here, we can setup Proxy in system A and we assume that SQL DM for MySQL is installed in system B, now you can access SQL DM for MySQL using, "http:///monyog/".

Configurations on system-A:

1. Please check whether you have libxml2 installed in your system.
2. Download mod_proxy_html.c from http://apache.webthing.com/(see page 290)
3. Now build mod_proxy_html with apxs, apxs -c -I/usr/include/libxml2 -i mod_proxy_html.c
4. You need to load the following modules, so add the following entries in [/etc/httpd/conf/httpd.conf].

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule headers_module modules/mod_headers.so
LoadModule deflate_module modules/mod_deflate.so
LoadFile /usr/lib/libxml2.so
LoadModule proxy_html_module modules/mod_proxy_html.so
```

5. Add the following configuration in your Apache configuration file [/etc/httpd/conf/httpd.conf]

ProxyRequests Off

```
<Proxy *>
Order deny, allow
Allow from all
</Proxy>
ProxyHTMLExtended On
ProxyPass /monyog/ http://<ip-system-B>:5555/
ProxyHTMLURLMap http://<ip-system-B>:5555/ /monyog/
<Location /monyog/>
ProxyPassReverse /
SetOutputFilter proxy-html
ProxyHTMLURLMap / /monyog/
RequestHeader unset Accept-Encoding
</Location>
```

TOP(see page 290)

9.1.13 13. How can I access SQL DM for MySQL pages proxying through nginx?

You can also access SQL DM for MySQL using nginx proxy. You need to follow these simple steps to configure your nginx server to support proxy. Here, we can setup Proxy in system A and we assume that SQL DM for MySQL is installed in system B, now you can access SQL DM for MySQL both over HTTP("http://") and HTTPS("https://").

Configuration of System A:

1. Install nginx on your system.
2. Create directories
   a. /var/log/nginx
   b. /var/www/cache

3. Configure nginx: Open nginx.conf found in /etc/nginx and add the following in the http section:

```
proxy_cache_path /var/www/cache levels=1:2 keys_zone=my-cache:8m max_size=1000m
inactive=600m;
proxy_temp_path /var/www/cache/tmp;
```

A sample nginx.conf would like the following:

```
user nginx;
worker_processes 1;
error_log /var/log/nginx/error.log debug;
pid /var/run/nginx.pid;
events {
worker_connections 1024;
}
http {
include /etc/nginx/mime.types;
default_type application/octet-stream;
log_format main '$remote_addr - $remote_user [$time_local] "$request" '
'$status $body_bytes_sent "$http_referer" '
'"$http_user_agent" "$http_x_forwarded_for"';
access_log /var/log/nginx/access.log main;
sendfile on;
#tcp_nopush on;
keepalive_timeout 0;
proxy_cache_path /var/www/cache levels=1:2 keys_zone=my-cache:8m max_size=1000m
inactive=600m;
proxy_temp_path /var/www/cache/tmp;
include /etc/nginx/conf.d/*.conf;
}
```

4. Put your SSL certificates in /etc/nginx/conf/

5. Create a monyog.conf file inside /etc/nginx/conf.d/ and add the following:

```
server {
```

```
server_name _;
listen 80;
location / {
proxy_pass http://<ip-system-b>:5555;
proxy_redirect off;
proxy_cache my-cache;
proxy_cache_valid 200 302 0m;
proxy_cache_valid 404 0m;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_max_temp_file_size 0;
client_max_body_size 10m;
client_body_buffer_size 128k;
proxy_connect_timeout 9000;
proxy_send_timeout 9000;
proxy_read_timeout 9000;
proxy_buffer_size 4k;
proxy_buffers 4 32k;
proxy_busy_buffers_size 64k;
proxy_temp_file_write_size 64k;
}
}
server {
server_name _;
listen 443;
ssl on;
ssl_certificate /etc/nginx/conf/<certificate_name>.crt;
ssl_certificate_key /etc/nginx/conf/<certificate_key>.key;
location / {
proxy_pass http://<ip-system-b>:5555;
proxy_redirect off;
proxy_cache my-cache;
proxy_cache_valid 200 302 0m;
proxy_cache_valid 404 0m;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_max_temp_file_size 0;
client_max_body_size 10m;
client_body_buffer_size 128k;
proxy_connect_timeout 9000;
proxy_send_timeout 9000;
proxy_read_timeout 9000;
proxy_buffer_size 4k;
proxy_buffers 4 32k;
proxy_busy_buffers_size 64k;
proxy_temp_file_write_size 64k;
}
}
```

TOP(see page 290)

9.1.14 14. Can I access SQL DM for MySQL pages using encrypted connection such as "https"?

Yes, you can access SQL DM for MySQL using "https", you may acquire a certificate from a certificate authority, such as Verisign or you may use the OpenSSL package to create your own certificate and configure your Apache webserver for "https".

Here are the steps you may follow to setup "https" in your Apache webserver.

1. Create a directory

   ```
   mkdir sslcert
   ```

   Now protect the directory,

   ```
   chmod 0700 sslcert
   ```

2. Create two sub-directories

   ```
   mkdir certs private
   ```

3. Create a database to keep track of each certificate

   ```
   echo '100001' >serial
   touch certindex.txt
   ```

4. Create a custom config file for OpenSSL to use similar to openssl.cnf in your /etc/pki/tls folder.

   ```
   dir = .
   [ ca ]
   default_ca = CA_default
   [ CA_default ]
   serial = $dir/serial
   database = $dir/certindex.txt
   new_certs_dir = $dir/certs
   certificate = $dir/cacert.pem
   private_key = $dir/private/cakey.pem
   default_days = 365
   default_md = md5
   preserve = no
   email_in_dn = no
   nameopt = default_ca
   certopt = default_ca
   policy = policy_match
   [ policy_match ]
   countryName = match
   ```

```
stateOrProvinceName = match
organizationName = match
organizationalUnitName = optional
commonName = supplied
emailAddress = optional
[ req ]
default_bits = 1024 # Size of keys
default_keyfile = key.pem # name of generated keys
default_md = md5 # message digest algorithm
string_mask = nombstr # permitted characters
distinguished_name = req_distinguished_name
req_extensions = v3_req
[ req_distinguished_name ]
0.organizationName = Organization Name (company)
organizationalUnitName = Organizational Unit Name (department, division)
emailAddress = Email Address
emailAddress_max = 40
localityName = Locality Name (city, district)
stateOrProvinceName = State or Province Name (full name)
countryName = Country Name (2 letter code)
countryName_min = 2
countryName_max = 2
commonName = Common Name (hostname, IP, or your name)
commonName_max = 64
0.organizationName_default = My Company
localityName_default = My Town
stateOrProvinceName_default = State or Providence
countryName_default = US
[ v3_ca ]
basicConstraints = CA:TRUE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer:always
[ v3_req ]
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
```

5. Create a root certificate. All other certificates you create will be based of this. Since this is not a commercial certificate software may complain when they use your certificates. You may give people the "public" certificate and your certificate works like the commercial ones when they import it. To create, while in the 'sslcert' directory type:

   ```
   openssl req -new -x509 -extensions v3_ca
   -keyout private/cakey.pem -out cacert.pem -days 365 -config ./openssl.cnf
   ```

   You will be prompted for information and a password. Do not lose this password, make sure it is a secure one and back-up the two files that are created.
   The two files that are created are cacert.pem, which is the one you can give to others for import in their browsers, and cakey.pem, which will be in the private directory.
6. Create a key and signing request

```
openssl req -new -nodes -out name-req.pem
-keyout private/name-key.pem -config ./openssl.cnf
```

You will be prompted for information. The critical part is the "Common Name". This must be the server's hostname, such as mail.your.domain or the IP address. If you want to cover all subdomains you can enter *.your.domain. Use the "Organizational Unit" to remind you what the certificate is for, such as "Web Server". This generates two files:

- name-req.pem - the request
- name-key.pem - the private key in the private directory

7. Sign the request. This generates the certificate:

```
openssl ca -out name-cert.pem -config
./openssl.cnf -infiles name-req.pem
```

You will be prompted for the password used when creating the root certificate. Two files are created:

- <number.pem> - a copy of it in the certs directory
- name-cert.pem - which is the certificate

8. Copy to the correct location For Apache 2.x on Red Hat using the default location, the directory is:
   a. For the name-key.pem:

   ```
   cp
   name-key.pem /etc/httpd/conf/ssl.key/
   ```

   b. For the certificate:

   ```
   cp
   name-cert.pem /etc/httpd/conf/ssl.crt/
   ```

9. Create a Virtual Host

```
<VirtualHost ip-system-A>:443> DocumentRoot /var/www/html
ServerName myserver
ErrorLog /etc/httpd/logs/ssl_error_log
TransferLog /etc/httpd/logs/ssl_access_log
SSLEngine On
SSLCertificateFile /etc/httpd/conf/ssl.crt/name-cert.pem
SSLCertificateKeyFile /etc/httpd/conf/ssl.key/name-key.pem
</VirtualHost>
```

10. Configure proxy in Apache described in FAQ 13(see page 296) and restart Apache.
    Edit the Hosts file [/etc/hosts]

```
<ip-system-A> myserver
```

TOP(see page 290)

9.1.15 15. What are the major differences between other major MySQL Monitoring Tool and SQL DM for MySQL?

They have similarities (in which they both differ from server-side scripts used for monitoring): they both make use of a HTTP service, a database and both use a web-browser for reporting. However, important differences are:

- SQL DM for MySQL needs no installation (of **'agents'**) on the server where the MySQL servers are running. Other does.
- SQL DM for MySQL **'has everything in itself'** - the webserver, the database, the MySQL client. It does not depend on the existence of other webservers, runtimes/Virtual Machines (like JAVA) and needs no separate database install. Other monitoring tools requires a full JDK (java), a TOMCAT server and a MySQL server instance for itself. Due to this simplified architecture install, configuration and first of all maintenance and upgrade is much simpler with SQL DM for MySQL. Download packages and disk storage required are much smaller with SQL DM for MySQL.

TOP(see page 290)

9.1.16 16. Can I trust the expertise of SQL DM for MySQL developers?

SQL DM for MySQL is developed by the Webyog Softworks that also created the most popular GUI for data management with the MySQL server - SQLyog Enterprise. We have more than 10 years of experience with designing MySQL related software. We have expanded our team with highly qualified developers ever since we started. We are devoted to constantly extending our knowledge and understanding of MySQL internals.

TOP(see page 290)

9.1.17 17. How does SQL DM for MySQL connect to MySQL?

SQL DM for MySQL uses the most proved and most efficient way of connecting: the native MariaDB Connector/C that is compiled into SQL DM for MySQL. Nothing else required: no separate client instance, no database abstraction layer (like ODBC/JDBC/ADO/.NET) and no webserver extensions (like PHP). Additionally, the connection can be 'wrapped' in a SSH tunnel. Also, SQL DM for MySQL implementation for this does not involve any other program (like Putty) running.

TOP(see page 290)

9.1.18 18. Windows warns after installation that SQL DM for MySQL may not have installed properly.

This can happen on some versions of Windows (Vista and higher) if you install a recent version of SQL DM for MySQL on top of an older version. The reason for this is that recent versions of Windows include a software called "Program Compatibility Assistant" (PCA) which tries to detect if an installer is running. It warns the user that the software might not have been correctly installed if the installer does not register a new uninstaller. The PCA is unable to detect changes made to an existing, registered uninstaller, which is what the new SQL DM for MySQL installers do. And thus, the warning is displayed. You can safely ignore this warning, but if it bothers you, you may just uninstall SQL DM for MySQL before upgrading to 3.5+. All collected data in the SQL DM for MySQL's data folder is still available after reinstall. However, you should:

- Backup the connections.data file before uninstalling
- Restore the old connections.data after the new install. After a restart, SQL DM for MySQL recognizes the connection settings in the old connections.data.

TOP(see page 290)

9.1.19 19. I would like to use SSH-tunnel, but my Windows server does not support it. Can that be fixed?

Yes, SSH support can be installed on Windows. You may install a complete Cygwin (Unix command line implementation for Windows). Alternatively, there are small packages available that support only a small subset of Cygwin (like SSH packages). Installation details depends on the exact Windows version.

TOP(see page 290)

9.1.20 20. SQL DM for MySQL throws an error when trying to connect to MySQL.

Please go through Error when trying to connect to MySQL(see page 290). The same applies to SQL DM for MySQL as the client code is exactly the same in both programs. Observe however that everything related to HTTP-tunneling with SQLyog is not relevant for SQL DM for MySQL.

TOP(see page 290)

9.1.21 21. Failed to connect to MySQL: Can't connect to local MySQL server through socket... What can i do about this?

Ensure that the host specified resolves to an IP-address. This error occurs with some Linux distributions (most important Debian) when specifying 'localhost'. The system maps this to a Unix SOCKET file. SQL DM for MySQL connects through TCP and not to SOCKET. Try the IP '127.0.0.1' instead.

TOP(see page 290)

9.1.22 22. SQL DM for MySQL is taking up too much of system resources with the PROCESSLIST-based sniffer.

You may have noticed that, while using the PROCESSLIST-based sniffer, SQL DM for MySQL increases the load on the CPU as well as the I/O subsystem of the system on which it is installed - even when the MySQL server is idle. Do not panic: it is normal. When using the PROCESSLIST-based sniffer, SQL DM for MySQL continually queries the MySQL server at the end of each time interval, which you can specify. It then retrieves the results and stores them in an internal sniffer database before displaying the results back to you. Now, if you set a short time interval, one that almost approaches 0, then SQL DM for MySQL can get stuck in an infinite loop! Consequently, the load on the CPU and I/O subsystem increases exponentially. We generally recommend an interval of not more than 0.1 seconds times the number of servers for which Processlist-based sniffers are enabled. However, if you are worried that you may miss out on some important queries running on the MySQL server, use the Performance Schem or MySQL Proxy mode. The LUA script supplied with SQL DM for MySQL should handle the task for MySQL proxy. For more information review MySQL Proxy(see page 290).

TOP(see page 290)

9.1.23 23. Why is display of queries truncated in Query Analyzer?

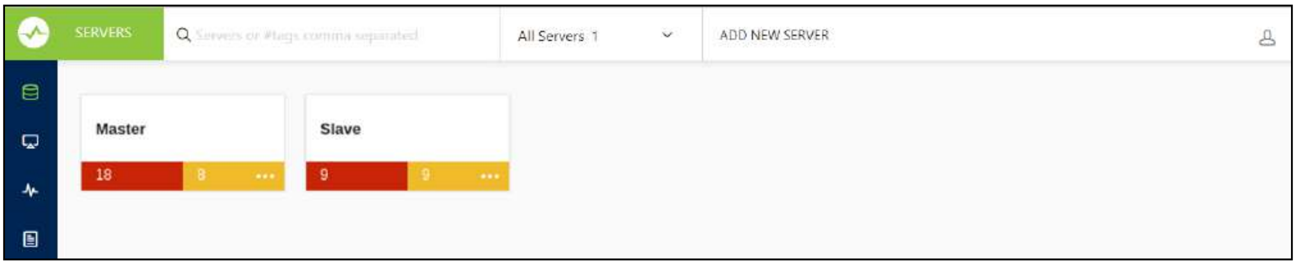
When using Query Analyzer feature, you may notice that sometimes queries displayed in output are incomplete. This may be due to one of the two known causes:

- As a security measure, SQL DM for MySQL extracts only the first 2000 characters of the query.
- MySQL does not record query delimiters in the General Log. Therefore, while analyzing the General Log, SQL DM for MySQL takes into consideration only the first line of the queries, and ignores the rest if they span over multiple lines.

TOP(see page 290)

9.1.24 24. The servers that I have registered do not display. What is wrong?

Check if the server is filtered based on a particular state, change your server filter to **All Servers** and now you can see your server between the servers if you had successfully registered it. You can also use the search bar next to the server filter to search for your server name or tag name to get to your server.



TOP(see page 290)

9.1.25 25. Now, anybody will be able to connect to my SQL DM for MySQL server and retrieve details about MySQL servers.

No, the SQL DM for MySQL authentication system will ensure that only those people that should have access have.

TOP(see page 290)

9.1.26 26. I have the same server registered twice. Metrics are reported different. Why?

For every registered server SQL DM for MySQL collects data independently. That is also the case when a server has been registered twice. Even if they were registered at the same time and even if the chosen sample interval is the same too, the connection and the server have some latency and data is not retrieved simultaneously. For that reason SQL DM for MySQL may retrieve and store slightly different values for each connection. This is most visible in the 'Delta' timeframe and least visible in the 'Current/all' timeframe. For GROUPING with 'History/Trends' the difference for each GROUP depends on the selected grouping interval. Due to laws of statistics the difference is less the longer the time interval (theoretically/statistically they converge more and more the closer time interval and/or the number of samples comes to infinity). Practically, you rarely need more than 20 samples in a GROUP for the difference to be negligible.

TOP(see page 290)

9.1.27 27. Does it affect the performance of a server if SQL DM for MySQL connects to it?

It does not affect on real 'live' servers. The queries sent by SQL DM for MySQL use almost no resources. We do not query data stored on disk and what we do query is stored in memory on the server. However if you are testing SQL DM for MySQL using a server instance that does almost nothing else and if you retrieve data at very short intervals the impact of SQL DM for MySQL may be slightly observable. The special Processlist feature (unique) may take a little more resources if there are lots of processes/client threads running. But SQL DM for MySQL only sends queries related to this when the corresponding SQL DM for MySQL client interface (the SQL DM for MySQL 'processlist' page) is open. Switching to another page or closing the browser stops sending the queries populating the SQL DM for MySQL processlist.

TOP(see page 290)

9.1.28 28. Is it possible to avoid that SQL DM for MySQL itself influences certain counters reported?

SQL DM for MySQL is a client. When it connects, the MySQL server starts a connection thread. And that connection is reported by SQL DM for MySQL. That cannot be avoided. The processlist feature has an option to 'filter out' SQL DM for MySQL connection - as well as other connections from other clients if you want - using a simple SELECT statement.

TOP(see page 290)

9.1.29 29. Can I customize SQL DM for MySQL counters?

Regarding customizing SQL DM for MySQL counters refer to Customization[71]. For scripting examples refer Customization Scripting Examples[72].

TOP(see page 290)

9.1.30 30. I cannot sit watching a browser all the time - Can I get alerts if something goes wrong?

Yes, you can choose to get notifications (Email, SNMP traps, Slack, Pagerduty and Syslog) independently for every server and you can define your own warning levels and select what counters should raise an alert.

TOP(see page 290)

9.1.31 31. SQL DM for MySQL cannot identify if destination of the log file is on a "Mapped Network Drive". Why?

By default MONyog(SQL DM for MySQL) service runs under Local System Account. If you are having Slow query or General query log in a Mapped Network Drive, SQL DM for MySQL is not be able to reach it. If SQL DM for MySQL has to access the file present in a Mapped Network Drive, you have to convert the path into shared path (accessed with UNC notation: \system\share) and then follow these steps:

[71] http://wiki.idera.com/display/SQLDMYSQL/Customizing+Monitors
[72] http://wiki.idera.com/display/SQLDMYSQL/New+CSOs+and+CSCs

1. Click the **Start** menu, then click **Run** and then type,

```
services.msc
```

2. After the Services window pops up with a list of all the services running in your system.
3. Search for Monyog and then right click --> **Properties**.
4. Click the **Log On** tab and then you can see that SQL DM for MySQL is using Local System Account.
5. You need to use **This account** option and then give the credentials that you use to log on to the system with Administrative privilege.
6. Save the settings, restart MONyog (SQL DM for MySQL) service.
7. After following the above steps try to access the file which is shared across network.

Note

The shared path should be accessed with UNC notation (\system\share). SQL DM for MySQL cannot identify if destination of the log file is on a Mapped Network Drive (this is a restriction with services on Windows and not with SQL DM for MySQL).

TOP(see page 290)

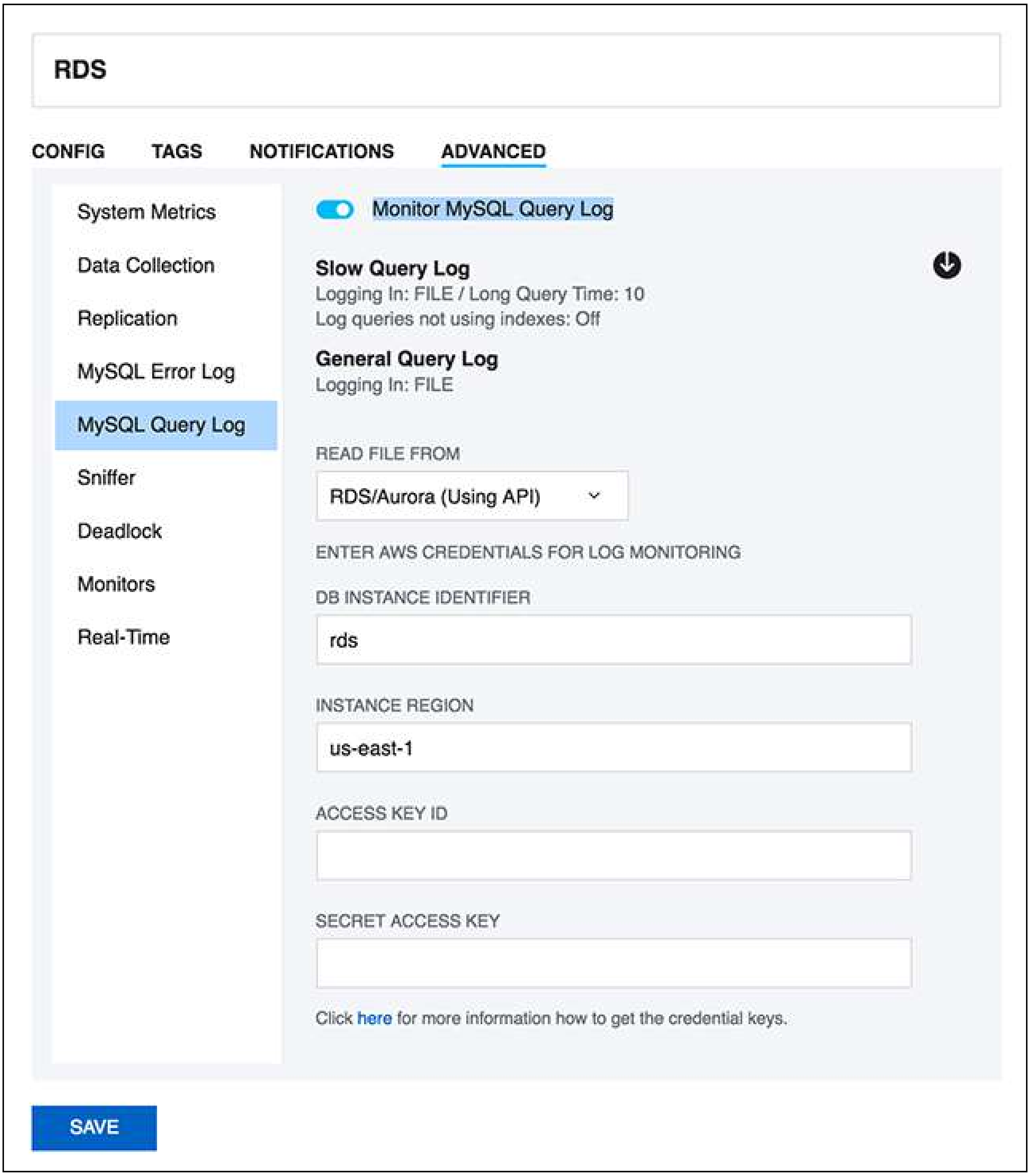
9.1.32 32. Failed to connect to MySQL: Unknown MySQL server host... What can I do about this?

You get this error if SQL DM for MySQL cannot resolve the hostname of a MySQL server. Ensure that other programs like ping, telnet, MySQL shell client are able to resolve the hostname to an IP-address. If yes, check "/etc/nsswitch.conf" of SQL DM for MySQL host. If the hosts section reads "files mdns4_minimal [NOTFOUND=return] dns mdns4", please change it to "files mdns4_minimal dns mdns4" or "files dns". This is introduced in some current Linux distribution. If other programs are not able to resolve the hostname, please check if host to IP resolution is properly defined inside "/etc/host" or in DNS server.

TOP(see page 290)

9.1.33 33. How can I monitor the queries from the file based RDS/Aurora Query logs?

SQL DM for MySQL can fetch the queries from the Slow Query log and General query log on Amazon RDS instance using the RDS REST APIs. SQL DM for MySQL requires the AWS access keys to fetch the file-based logs. Go to the **Edit Server->Advanced->MySQL Query log** and enable the option of "**Monitor MySQL Query Log**". Click the **Fetch logs**(down arrow) button and provide the AWS access key and secret access key to enable SQL DM for MySQL to monitor the log files.



RDS

Monitor MySQL Query Log

Slow Query Log
Logging In: FILE / Long Query Time: 10
Log queries not using indexes: Off

General Query Log
Logging In: FILE

READ FILE FROM: RDS/Aurora (Using API)

ENTER AWS CREDENTIALS FOR LOG MONITORING

DB INSTANCE IDENTIFIER: rds

INSTANCE REGION: us-east-1

ACCESS KEY ID:

SECRET ACCESS KEY:

Click here for more information how to get the credential keys.

TOP(see page 290)

9.1.34 34. What are future plans for SQL DM for MySQL?

SQL DM for MySQL is an important product for us. We plan to add new features as well as to 'refine' existing features. With the latest release we have completed what we originally planned for SQL DM for MySQL.

- It is now possible to get OS metrics from Amazon RDS/Aurora
- Added more notification channels (Slack, Pagerduty and Syslog) for SQL DM for MySQL alerts.

These features have all been requested by users. SQL DM for MySQL development has always been and continues to be very attentive to user requests. We update information here when plans for future developments have been decided.

TOP(see page 290)

9.1.35 35. How do I get help and report problems?

Four ways:

- Website Intercom https://www.idera.com/(see page 290)
- Post in our Forums http://community.idera.com/(see page 290)
- Create ticket: ideramysqlsupport@idera.com[73]
- SQL DM for MySQL UI intercom

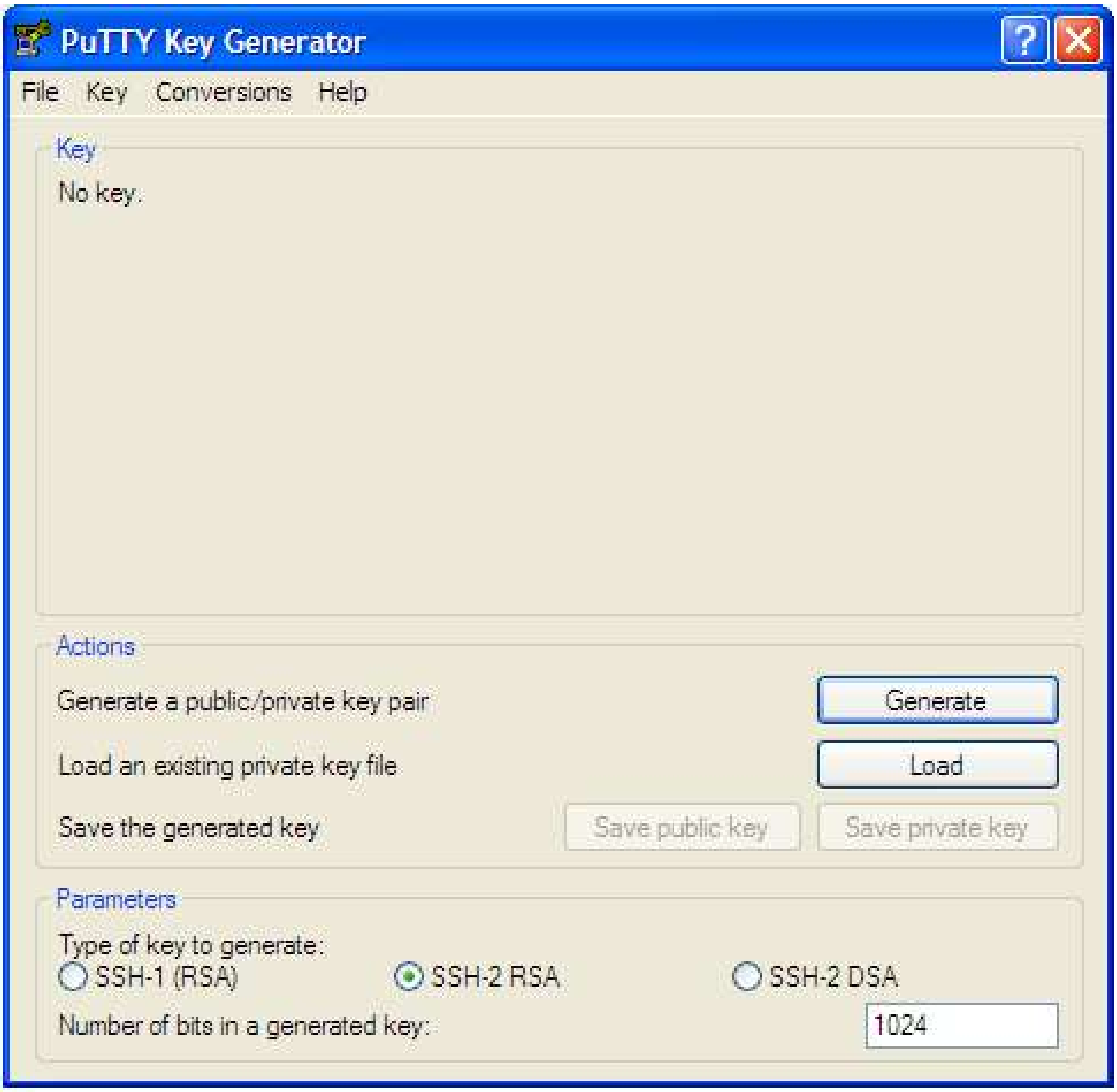
TOP(see page 290)

9.1.36 36. Can I use the keys generated from PuTTY for SSH connection?

SQL DM for MySQL does not support the key generated from PuTTY for SSH connection. However, you can convert the private key generated from "PuTTY key generator" into an open SSH key, and then use this key in SQL DM for MySQL to connect to the server. Here are the steps to follow:

- Go to PuTTY key generator, and generate a public/private key on your local system (refer the screenshot). Click the **Generate** button to generate the keys.

[73] mailto:ideramysqlsupport@idera.com



No key.

Generate a public/private key pair

Load an existing private key file

Save the generated key

Type of key to generate:
○ SSH-1 (RSA) ● SSH-2 RSA ○ SSH-2 DSA

Number of bits in a generated key: 1024

- Copy the public key generated under the "**Key**" space to the **authorized_keys** file, which is located in the **.ssh** directory on the remote host that you want to connect to.
- Go to **"Conversions"** in PuTTY key generator and click "**Export openssh key**" and save the new converted private key in a file.
- Now open the file containing the converted OpenSSH private key, copy this key and paste in the **"Private key"** field in SQL DM for MySQL (**Edit server -> SSH settings -> Private key**).

TOP(see page 290)

9.1.37 37. Steps to auto-start MONyog(SQL DM for MySQL) service with OS reboot in Ubuntu and Debian systems.

Users can make use of the "MONyog" script shipped with the Monyog(SQL DM for MySQL) package to auto-start MONyog(SQL DM for MySQL) service with OS reboot. The SQL DM for MySQL script is located at " /MONyog/bin/". Please follow the steps below:

- Copy the 'Monyog' to "/etc/init.d/" from " /MONyog/bin/"
- Open the 'Monyog' script located at "/etc/init.d/" and edit the variable "curdir" (line number 15) and set it to the path of bin. After editing, it should look like this: curdir="/home/Users/Downloads/MONyog/bin/"
- Make the script executable by 'chmod +x /etc/init.d/MONyog'
- Use debian utility update-rc.d to install the script: update- rc .d MONyog defaults

TOP(see page 290)

9.1.38 38. How to upgrade SQL DM for MySQL without losing your data or configuration?

The SQL DM for MySQL binaries are shipped in 3 packages: .tar, .rpm and .exe. The upgrade process is simple and depends on your package. Follow the steps below to upgrade to the latest version of SQL DM for MySQL:

For .rpm package :

```
rpm -Uvh <MONyog_package>.rpm
```

This command installs the latest build on top of your current installation.

For .tar package:

```
tar -xzvf <MONyog_package>.tar.gz
```

Please untar the package in the directory where the 'MONyog(SQL DM for MySQL)' package was untarred for the previous version to make sure that all your data and settings are intact.

For Windows (.exe) package:

Executing the file installs SQL DM for MySQL on top of the current installation.

All SQL DM for MySQL data and the configuration files are stored in a SQLite repository. In some of the SQL DM for MySQL GA releases, we modify/change the monitor definition in the SQLite files due to some bug or enhancements. In such cases, on upgrading, all the local changes made by the user in the previous version get replaced with the default shipped value and these local changes are shown as a conflict. You can see the conflicts as a notification on the top right-hand corner.

To resolve conflict

You can resolve these conflicts from the "**Settings -> Manage changes**" page, herein you get 2 options for all the listed conflicts: Use your changes/Discard your changes. 1- Use your changes, restores the local modifications which you had made in the previous version. 2- Discard your changes, replaces your changes with the default values.

TOP(see page 290)

9.1.39 39. How to maintain High Availability of SQL DM for MySQL?

Monit tool monitors the server process and can be used to maintain HA for SQL DM for MySQL.

Use the below commands to install monit:

apt:

```
sudo apt-get install monit
```

Yum:

```
sudo yum install monit
```

Once monit is installed, you can add programs and processes to the configuration file:

```
sudo nano /etc/monit/monitrc
```

Uncomment the below lines in the file to enable web interface. You can login to the web interface using the username ‘admin’ and password ‘monit’.

```
set httpd port 2812 and
use address localhost # only accept connection from localhost
allow localhost # allow localhost to connect to the server and
allow admin:monit # require user 'admin' with password 'monit'
```

Also add the below lines in the configuration file to enable the monitoring of MONyog(SQL DM for MySQL) service:

Tar:

```
check process Monyog
matching "MONyog"
start program = "<MONyog extracted directory>/MONyog/bin/MONyog start"
stop program = "<MONyog extracted directory>/MONyog/bin/MONyog stop"
```

Rpm:

```
check process Monyog
matching "MONyog"
start program = "/etc/init.d/MONyogd start"
stop program = "/etc/init.d/MONyogd stop"
```

Once the above configuration changes are made, please use the below command to reload monit:

```
sudo monit reload
```

TOP(see page 290)

10 SQL DM for MySQL PDF

This page contains a direct link to the SQL Diagnostic Manager help in PDF format. This format is suitable for printing and for saving on your local PC for further reference.

The PDF includes all pages from the relevant help published on wiki.idera.com[74].

File	Modified
SQL Diagnostic Manager for MySQL 8.9.x.pdf[75]	Apr 06, 2021 by Fabiola Arias Navia[76]
SQL DM for MySQL 8.9.x.pdf[77]	Nov 06, 2021 by Fabiola Arias Navia[78]

[74] http://wiki.idera.com
[75] https://wiki.idera.com/download/attachments/7485718974/SQL%20Diagnostic%20Manager%20for%20MySQL%208.9.x.pdf?api=v2
[76] https://wiki.idera.com/display/~fabiola.arias@jalasoft.com
[77] https://wiki.idera.com/download/attachments/7485718974/SQL%20DM%20for%20MySQL%208.9.x.pdf?api=v2
[78] https://wiki.idera.com/display/~fabiola.arias@jalasoft.com

11 SQL DM for MySQL Solutions

11.1 How do connection names resolve to IP's?

It does not answer questions about the SQL DM program as such. But we experience quite often that SQL DM user having problems in getting the connection to one ore more MySQL servers working are missing the basic understanding of the TCP protocol that is used for connection. So this is a 'background article' that supplements the more typical FAQ items here.

The TCP protocol needs an IP to connect to a host. An IP is a group of 4 2-digit hexadecimal numbers. Like '0A.1B.2D.3D'. It is common to express this in decimal format 'v.x.y.z' where v,x,y and z are numbers between 0 and 255. To get your own IP in this format just type (on windows) 'ipconfig' on a command-line and the IP is returned.

The IP returned can be a global ip (accessible directly from the internet) or - if you are behind a router - a local ip that only applies behind the router. All computers behind the router share the global ip of the router itself. The router has the functionality to ensure that communication from each machine behind it to and from the internet works.

The TCP-protocol communicates on 'ports'. There are about 65.000 ports possible. A TCP-port is nothing physical but just a number that each 'packet' of information that is sent is coded with. Various ports are reserved for various server programs and programs with sort-of server functionality. Here, you can review the complete list of Service Name and Transport Protocol Port Number Registry(see page 312):

So to access a MySQL server (running on the standard MySQL port 3306) on the ip v.x.y.z the client must ask for connection to the server 'v.x.y.z:3306'. The Internet can handle this. This is the basically how the Internet and the TCP-protocol works! Pretty simple actually! And there is one basic rule more: the ip 127.0.0.1 is always any computer itself!

However you probably never enter internet addresses that way no matter if it is a website, an FTP-server or a MySQL-server that you access. You use NAMES. However these names must be resolved to the above format. If not resolved to ip's the servers on the Internet can't handle the request.

There are basically four ways of resolving a name to an IP(and in that order):

1. Using client hosts' file
2. Using (local) nameserver lookup
3. Using Domain Name Server (DNS) lookup.
4. Using Remote Network routing (using either simple routing based on ports, hosts file or nameserver) on remote network.

1. Client hosts' file

There is an excellent article on the hosts file What is hosts?(see page 312). Try to find your own hosts file. You will find this line in it:

> ***127.0.0.1 localhost***

You might find more lines too - for several reasons: some viruses and spyware add items to the hosts file. Some users and Sys Admins do it on purpose. But since you are able to connect to a MySQL server running on your local machine with the host name **'localhost'** it is because of the above line in the hosts file. The hosts files resolves the name 'localhost' to ip 127.0.0.1. If you prefer to use **'cutie'** instead of **'localhost'** then just add the line

> ***127.0.0.1 cutie***

to your hosts file!
2. A local nameserver.
A local nameserver can be installed explicitly or implicitly. If you have more windows computers connected on a windows' network (using the TCP protocol - not old NETBEUI protocol etc) you already have a local nameserver that was implicitly installed. If you have two machines on your network named 'Bonnie' and 'Clyde', then you can connect to the MySQL server running on 'Bonnie' from the computer 'Clyde' with SQL DM entering 'Bonnie' as the host name. Because the nameserver that was implicitly installed as a part of the Windows network resolves the name 'Bonnie' to 'Bonnie's ip on the network
3. A Domain Name Server
Domain name Servers are part of the Internet structure. Any name/string in the format domain.topleveldomain (ie: school.edu, business.com, whisky.org etc.) can be looked up on the Domain Name Servers of the Internet. The ip returned is then used for creating the connection.

11.1.1 A typical example

You connect to a remote server with the host name *mydb.thisismyisp.com*. What happens here? Simply:

- **First:** local hosts file is checked if there are any matching entry. if not proceed to step 2.
- **Second:** local nameserver (if exists) is checked if there are any matching entry. if not proceed to step 3.
- **Third:** Domain Name Server lookup identifies the ip of domain-host *thisismyisp.com*
- **Fourth:** local network routing systems (router, nameserver or hosts file) on the remote network identifies the local IP mapped to name 'mydb´'

After following the previous steps *mydb.thisismyisp.com* is resolved to ip v.x.y.z on the internet and IP a.b.c.d on the remote network. Once connection is established the routers and other networking gear in the complete communications chain keeps that information until it is closed or times-out due to inactivity (or some error occur).

One more detail, when connecting to the MySQL server at an ISP it is common practice that MySQL user names and MySQL database names must be prefixed with the domain user name (the one that is used for FTP for instance). Like me_username and me_mydb. Simple because other users may have created a MySQL user **username** and a MySQL database **mydb**. But failure to do so does not result in a connection error, but an authentication error. Also note that MySQL usernames can be up to 16 characters long only, review the error "I have a very long username for the MySQL database at my ISP. SQL DM won't let me use it" that is listed below.

Nevertheless, everything can get 'messed up' if it is the first time you try to connect to a remote MySQL server! And here tunneling is not involved.

11.2 What should I enter as 'hostname' when connecting to a MySQL server at an ISP.

It depends on the systems and the network settings at your ISP. The systems that ISP's operate are too numerous to mention. Some operate almost all their server programs on one very powerful (UNIX-) machine, others operate a network of many computers that are basically PC-type hardware. Some use Linux/Unix operating systems other Windows server systems (and some mix it!). And ISP's vary in size from 200 customers to 2.000.000. They can't all operate the same systems!
But the general answer is that probably the ISP operates some name server internally. Such name server will typically be configured with the some name like 'mysqlserver' pointing to the ip of the mysqlserver on the local network.

- **Direct connection:**

If the ISP has the domain name *thisisanisp.com* then with direct connection you will use *mysqlserver.thisisanisp.com*.
The information about the name assigned to the mysql server and thus the exact URL to the MySQL server usually is available as part of your account info from some web-based control panel application. Most likely it is something like 'mydb1.thisisanisp.com' or 'mysql.thisisanisp.com'. If you cannot find the information yourself you will have to ask the support/help desk there.
It also could happen that you could simply use *thisisanisp.com* or the global IP of the ISP, and traffic on port 3306 is routed to the MySQL server - but it will not always work - particularly not if it is a big ISP!. But everything here depends on the local routing systems used at the remote host - whether port-number based routing is active or not.

- **HTTP-tunneling:**

With HTTP-tunneling 'host' can simply be 'mysqlserver' (ie: 'mydb1' or 'mysql' or whatever). *mysqlserver.thisisanisp.com* will work too, but there is no need for an extra domain lookup! If it is the same computer that runs the Web-server (with PHP-interpreter) and the MySQL server it can also be 'localhost' or the ip '127.0.0.1'. PHP will connect to MySQL through a Unix-Socket if 'localhost' is specified and TCP if '127.0.0.1'. Both will normally work. TCP may be a little bit slower on Linux machines than Socket, so you can try 'localhost' first. However there are situations where 'localhost' will return an error like error: **'Error No. 2013: Lost connection to MySQL server during query'**. This is likely because PHP looks for the socket file where it does not exist. The underlying reason could be that MySQL and PHP have been installed from different repositories using different file positions for the socket file and the mysql datadir.

- **SSH-tunneling.**

With SSH-tunneling there are two hosts' settings:

1. The 'MySQL Host Address' on the Server Tab on the connections manager: Here the 'host' is the 'path' to the MySQL-server relative to the SSH-server on the remote local network. Most often 'localhost' will do. If it does not work and 'mysqlserver' does not either you will have to ask the support/Sys Admin for more information. There are lots of possible ways to configure SSH on a network! Also here if the MySQL host as entered here cannot be reached from the SSH-server the MySQL client error: **'Error No. 2013: Lost connection to MySQL server during query'** occurs. Thus this can be caused by an erroneous 'host' entry, but it can also be a network/configuration error on the remote network. Or the MySQL server simply could be down.
2. The 'SSH host' on the Tunnel Tab on the connections manager: This is the 'global URL' of the SHH server at the ISP. The ip will normally work fine, *thisisanisp.com* too. And maybe SSH is available via the internal name server too - then use something like *sshserver.thisisanisp.com.*

11.3 SQL DM gives you the most options for connecting to MySQL

SQL DM connects to MySQL using the native C-API from MySQL - the fastest and most effective way to manage MySQL. This API is compiled into SQL DM code itself.

Even if the TCP-port (3306) normally used by the MySQL server is blocked (as often is the case at ISP's) SQL DM still let you connect using HTTP(s)-tunneling or SSH-tunneling. And if you are behind a proxy SQL DM can handle that too. The client for SSH connection is installed with SQL DM, and a PHP-script for HTTP-tunneling is too. That script must be uploaded to your webhost. The PHP-script uses php_mysql extension that is available practically everywhere where MySQL is available.

SQL DM works with MySQL version 3.23 and upwards.

11.4 SQL DM takes long time to connect when using SSH-tunnel.

If SQL DM is taking long time (more than 5-10 seconds) to connect to your MySQL server using SSH tunneling then there could be problem with the name resolution.
The first thing that the SSH server does before forwarding the connection to the specified MySQL server is to perform a reverse DNS lookup on the client's IP. The SSH server may cause an unnecessary delay during authentication due to incorrect or absent of reverse DNS settings. So in case of any slowness you should check those settings.
You can also disable most of the server-side lookups by setting UseDNS = "no" in SSHD configuration file (/etc/ssh/sshd_config on most systems). But in that case the MySQL host must be specified with an ip and not a hostname.

11.5 What Is SSH and SSH-tunneling?

The Acronym **SSH** stands for **Secure Shell Host**. SSH was originally created to provide a secure way to access server systems at "low level", to be used instead of common (but insecure) **telnet** methods. SSH can use several different forms of encryption, anywhere from 56 to 1024 bit. SSH has been ported to Operating Systems on several platforms including Linux, Microsoft Windows and Macintosh. There are SSH servers and SSH clients available for different types of communication.

Here you may notice this: "OpenSSH includes the ability to forward remote TCP ports over a secure tunnel, allowing that way arbitrary TCP ports on the server side and on the client side to be connected through an SSH tunnel". This is excatly what we make use of.

The term **"SSH tunneling"** in relation to a database server means that in- and outgoing communications to the network that hosts the database server "passes through" the SSH-server and uses the communications port (usually port 22) and the protocol of the SSH-server. The SSH-server then "translates" and "transfers" that in-and outgoing communication to the database server. SSH is actually quite simple to use. However if you are totally unfamiliar with networking terminology you will have to study it somewhat. Actually you may even install an SSH server at your own local machine and use it for connecting to a local MySQL server. Not much use of that, but it will give you an excellent understanding of what SSH is!

There is a built-in SSH-client in SQL DM that lets you connect to a MySQL server using SSH.

Basically there are two benefits of SSH Tunneling:
* SSH can be used to encrypt communications between SQL DM and your remote MySQL server.
* It lets you access the MySQL server even if the MySQL port (3306) is blocked.

Unlike HTTP-tunneling, you cannot take it for granted that SSH-tunneling is available at your webhost. In general the "more professional" and "more expensive" hosting providers offer SSH and the cheaper ones don't.

Refer to the SQL DM help file for instructions how to set up the SQL DM Connections Manager, if you want to use SSH-tunneling with SQL DM. Each SSH-connection occupies a TCP-port at your local machine. With recent SQL DM versions this port is picked automatically from the pool of high-numbered ports not already in use.

A concluding note on the popular 'Putty' program and SQL DM SSH-tunneling.

Sometimes when people are having problems with SSH-connections, we often hear "I can connect with Putty without problems". Maybe so, but it does not tell very much (almost nothing actually!) because the type of connection with Putty or a similar program referred to here by users **is not tunneling** and does not make use of port forwarding. Putty creates a remote (and secure) shell on the client machine, and connects to the 'mysql' client program on the server. So here the MySQL client is the 'mysql' client on the remote server. It is true that Putty can be used for setting up a SSH-tunnel as well, but this is not the simple 'connect with Putty' most often referred to and compared with here. With (SQL DM) SSH-tunnel the MySQL client is the client API that is compiled into SQL DM (and SJA). That is why port forwarding is needed and must be functional with SQL DM SSH-tunneling!

You can review the SSH-related error messages[79].

11.6 Connection Issues

I have a very long username for the MySQL database

11.7 I have a very long username for the MySQL database at my ISP. SQL DM won't let me use it.

Right! Because a MySQL user name is up to 16 characters long. The MySQL docs clearly state:

> ***MySQL usernames can be up to 16 characters long. Operating system usernames might have a different maximum length. For example, Unix usernames typically are limited to eight characters.***

However, it is a bad practice with some ISP's that they generate longer usernames than that. It is typically 'cheaper' hosting providers that offer a single MySQL database as part of a 'personal' or 'small business' subscription plan. They auto-generate the username from the user's domain name and it could be somewhat like **mydb_myveryowndomain** or similar. It is also true that some of our competitors offer support for that. However it is very bad practice! To make it work you will have to:

1. ALTER the TABLE mysql.users and change the mysql.users.users column from a char(16) to some longer value
2. You can no longer GRANT user rights, but must INSERT/UPDATE the mysql system tables directly
3. You must 'patch' (or rather 'hack') the MySQL API/client code

We have had intense discussions with this the MySQL AB on this issue. From the official correspondence we quote:

> ***"This is simply a lucky fluke of sorts*** *(if it works)**. MySQL simply does not support longer usernames .... Altering the system tables, aside from using our own mysql_fix_privilege_tables script to keep up with our changes, is simply unsupportable. There are server and client changes needed to properly handle any sort of modifications here, even though in some cases a quirk (as above) may seem to function .... This is, basically, dangerous behavior. We will attempt to curb it as well as we can.... Luckily, our manual states clearly that in both cases, MySQL will not provide support if any problems arise ... That is , it may work and it may not work, but MySQL will not ponder as to why it works or why it does not work ... We simply do not provide support for such cases."***

We won't play that game as others do! You should convince your ISP that changing the format of the user table is bad and dangerous practice!

And further: MySQL has 'stopped the game'. Again we quote from the above correspondence:

> ***"To make things even more precise, I will now send a server patch to our development management. This patch adds a code that will truncate user column at 16 chars and other columns to their nominal sizes. This will ensure that future 4.1 and 5.0 versions will not work with longer names, whatever changes some application could envisage."***

So with the most recent builds in the MySQL 4.1.x series and with MySQL 5.0 it would not work anyway. There is now code in the server binary itself that truncates any user name to 16 characters.

Error N° 2002

[79] http://wiki.idera.com/display/SQLDMYSQL/SSH-Related+errors

11.8 I get Error Nº 2002. Can't connect to local MySQL server through socket ...

This can occur when connecting using HTTP-tunneling to a MySQL server running on Unix/Linux platforms.

MySQL writes Problems with MySQL sock(see page 312) about this issue.

How to cope with this would depend on which webserver and which php version is used. But here is a workaround that has worked with Apache:
If the directory /var/lib/mysql doesn't exist than create it and chown to user mysql:
"mkdir /var/lib/mysql; chown mysql /var/lib/mysql"
Then edit the /etc/my.cnf file and specify:
[mysqld]
socket=/var/lib/mysql/mysql.socket
[client]
socket=/var/lib/mysql/mysql.socket
And, restart the mysql server (**"etc/init.d/mysql restart"**)
That is enough for MysSQL. However the chances are that PHP was compiled with a different default mysql socket location (e.g. **/tmp/mysql.sock**). In which case you have to edit the php.ini file and find the variable "mysql.default_socket". Set this to the above value
mysql.default_socket = /var/lib/mysql/mysql.socket
And restart Apache to re-read the php.ini file
If you do not have access to the configuration files and system command-line then you must ask your Sys Admin/ support to help with this.

Error Nº 1251

11.9 Error no. 1251: "Client does not support authentication..."

Error no. 1251: "Client does not support authentication protocol requested by server - consider upgrading MySQL client" occurs when the hashing-method for storing password used by the client differs from the one of the server. Typically it occurs when trying to connect to MySQL 4.1 or 5.x with a client compiled for 3.x or 4.0.

The Client as far as SQL DM goes is either the IderaSQLdmforMySQL-8.7.0-0.exe and sja.exe executable files (with its compiled-in MySQL C-API) or - in case you use HTTP-tunneling - your PHP-binary.

SQL DM itself (the IderaSQLdmforMySQL-8.7.0-0.exe -executable) handles all MySQL versions from 3.23.x and upwards automatically, and the error message should not occur with direct connection. In case you experience the error when HTTP-tunneling, you can EITHER replace the PHP-binary OR downgrade the hash-type for the user used for tunneling (and other PHP based connections) with the following command:

SET PASSWORD FOR 'some_user'@'some_host' = OLD_PASSWORD('newpwd');

The command must be written as it is.

At ISP's you should expect the Sys Admin there to have MySQL-installations and PHP-binaries that "fit". If you operate your own MySQL server the hashing method **may** or **may not** change when upgrading the server from 4.0 to a newer version. That depends on the upgrade method. This one is very similar to "Error no. 1045: Connection denied ..", which you can find it in this list. However, this case **connection** is established OK but **access to data** is denied. The user exists but most like he does not have any privilege at all. About user privileges start, please refer to the issue "I am able to connect but can't see the list of databases/tables" listed in this page.

I am able to connect using phpMyAdmin, but SQLyog will not let me connect.

11.10 I am able to connect using phpMyAdmin, but SQL DM will not let me connect.

PhpMyAdmin is running on the server itself so when connecting to MySQL with phpMyAdmin you are NOT connecting from a remote host! With SQL DM you are connecting from a remote host. This is a very important difference as far as user configuration with MySQL is concerned.

The user that you are connecting with maybe has no privilege to connect from remote.
By default, MySQL only gives access for users to connect from localhost. If you want to give some user access to the server from another host you must specify the hostname (an ip or a URL). You can use the SQL wildcards "%" and "_" (but not windows wildcards like "*"). Simply giving permission for a user to access the server from "%" means from everywhere. Read more about the MySQL privileges system in "Error Nº1044".
If the database is at an ISP there usually is some kind of "Control Panel" application available from where to configure users. Often user configuration is only allowed using this tool. It also is very likely that your ISP has blocked direct access to MySQL on port 3306. If that is the case SQL DM offers you the option of using HTTP-tunneling as well as SSH-tunneling.

Error Nº 1044

11.11 Error Nº 1044: "Access denied..."

This one is very similar to "Error no. 1045: Connection denied .."

However this case **connection** is established OK but **access to data** is denied. The user exists but most like he does not have any privilege at all. Read more about user privileges in "I am able to connect but cannot see the list of databases/tables" issue listed.

I am able to connect but cannot see the list of databases/tables

11.12 I am able to connect but can't see the list of databases/tables.

You must have at least SELECT privilege (as a global privilege or a SCHEMA/COLUMN privilege, to be able to see any data. More information on the MySQL privilege system can be found The MySQL Access Privilege System(see page 312).

If the database is hosted at an ISP/hosting provider it is likely that you must use some web-based "Control Panel" application to set up user privileges.

I am getting "Protocol Mismatch; Server Version 9; Client Version 10"

11.13 I am getting "Protocol Mismatch; Server Version 9; Client Version 10"

The protocol version 9 is used by MySQL 3.22 and earlier. The current version of SQL DM supports 3.23.x and above. To use it with 3.22 we have to provide you with a special build of SQL DM. Registered users can request such copy from ideramysqlsupport@idera.com[80]. We can not guarantee that it will be the latest version of SQL DM.

The opposite error message appears if you try to connect from an application with a connector (for instance an ODBC-driver) that is too old too be used with the actual MySQL version.

[80] mailto:ideramysqlsupport@idera.com

I have an account with Yahoo. Can I use SQLYog

11.14 I have an account with Yahoo. Can I use SQL DM…

Yes. But several users have had problems getting connection parameters right. Here is what Yahoo say themselves:

> *Why can't I access my database?*
>
> *First, make sure that you are using the hostname "mysql" and not "localhost" in any of your PHP or Perl configuration files that require a MySQL hostname.*

You will need to use HTTP Tunneling. First upload SQL DMTunnel.php (available with the SQL DM installer). Put the correct URL in the HTTP Tunneling field and use the same credentials as you use in your PHP pages. The Hostname should be "mysql" (case-sensitive).

Error Nº 1130

11.15 I get error 1130 "Host is not allowed to connect ..." or "Access denied ..." or "Could not connect ..."

Error 1130 is a networking error. The server cannot resolve the hostname of the client. Or the host is not allowed to connect to the MySQL server.

There are basically 2 categories of possible reasons:

- **The simple one:**

  In MySQL a user a user is specified using BOTH the user name and the host from where the user may connect. If no user has been created where the host-part (using wildcards or not) mathces the host of the client trying to connect MySQL returns this error.

- **Specific for MySQL 5.7:**

  When upgrading to MySQL 5.7.3 from a previous version this may occur due to changes to the user table introduced in 5.7.3. There is a good blog about it, Upgrade and Resolving ERROR 1130 Host 'localhost' is Not Allowed to Connect(see page 312).

- **The tricky ones:**

1. Your hosts file is damaged or invalid. Various vira and spyware attack and alter the host file in various ways. For instance if the hosts file does not contain the line
   > ***127.0.0.1 localhost***

   'Localhost' can not be resolved as pointing to ip 127.0.0.1. On some larger corporate network it is widely used to "roll out" host files to all clients with symbolic names (like 'mysqlserver', 'mailserver' etc.) for important machines on the network and the corresponding IP's. Check with your admin that you got the right file!
2. If you use the Windows network name as hostname, it may be a network configuration problem. Try using the ip instead.
3. On Unix/Linux systems the hosts files sometimes reads
4. Is a variation of number 3. With a complex server setup (involving more IP's, domains, subdomains and/or virtual hosts) a similar issue can occur.
   In this situation HTTP-tunneling will normally work for all users, but often/sometimes direct connection and SSH will not work with the 'root' user - everything depending on the server configuration. Try another and 'ordinary' user account. You may mirror the privileges of 'root' to a 'superadmin' user.
   > ***127.0.0.1 localhost.localdomain***

This causes a problem with MySQL. MySQL docs at Access Control, Stage 1: Connection Verification(see page 312) say:

> *A Host value may be a hostname or an IP number, or 'localhost' to indicate the local host.*

No mention of 'localhost.localdomain'. This means that MySQL cannot resolve that 'automatically'! This can affect SQL DM when tunneling and SJA for Linux. Workarounds for this include:
I: A workaround that has worked is to give the user access from 'localhost.%'

II: You can add localhost to ip 127.0.0.1 in host file like
a. Make sure your '/etc/hosts' file reads as follows:
127.0.0.1 localhost //localhost *MUST* be first (notice separate entries)
127.0.0.1 localhost.localdomain
127.0.0.1 . . .
b. Make sure you reference the local server in the SJA.XML file as:
127.0.0.1 or localhost.localdomain
This is known to work in some situations where the MySQL configuration file contains:
bind-address = 127.0.0.1
III: It has sometimes worked to add a 'dummy' ip to my.cnf like:
bind-address = 10.10.10.10
.. supposed that 10.10.10.x is also the ip of the local machine. Then you use this 'dummy ip' as the host specification when connecting with SQL DM and SJA.
It seems to be something special for some DEBIAN distributions to use this 'bind-address' construction with MySQL. Solution II) and III) both are solutions that users have contributed at our Forums. Both situations involved DEBIAN.
IV: You can try any host name that the host file maps to ip 127.0.0.1. There might be several! Even a SAMBA NetBIOS alias might work!
V: You can use the local ip (ie. 10.0.0.1 or whatever) of the actual machine or a name server alias for this. But in this case normal TCP-connections must be enabled in MySQL configuration - that is 'skip-networking' must be disabled/commented out and no 'bind-address' may be there. Of course then an additional DNS lookup will have to take place for the connection to be established. This is of no practical importance.
Finally you could test if this connection issue is the same with 'MySQL Administrator'. It uses the same client code (the C-API) and connects exactly as SQL DM and SJA do.

Error Nº 1045

11.16 Error no. 1045: "Connection denied..."

The error message: Error No. 1045: Connection denied for 'someuser@somehost' (using password: YES/NO)

It is a user authentication error. The user details specified do not "match" the user tables of the specified MySQL server. Common situations are:

- No such user.

NOTE: MySQL does not use the OS's or domain's user's management. It operates its own user accounts. With a fresh MySQL installation the user ROOT is created with NO PASSWORD. When working with MySQL databases at ISP's an admin user most often must be activated from some web based Control Panel Application before connection to the MySQL server is possible. There could be more "rules" applying here (database and user naming conventions etc). Refer to the docs/support at the ISP for details on that. We can't give them!

- User is not allowed to connect from the actual host. Note that MySQL by default only allows connection from 'localhost'. To specify from where a user may connect SQL wildcards (% and _) can be used. Simply 'someuser@%' means that user 'someuser' may connect from everywhere.
- Wrong password, missing password or password specified where it should not

- If you are upgrading MySQL from an old version (4.0.x or lower) to a more recent (4.1.x or highere) and if you are still using a rather old PHP version you may need to execute this command from command-line client

  SET PASSWORD FOR some_user@localhost = OLD_PASSWORD('newpwd');

  (where some_user@localhost is the user used for this connection) since the format for storing passwords has changed between 4.0.x and 4.1.x versions.

Error Nº 2005

11.17 Error No. 2005: Unknown MySQL server host...

The error message: Error No. 2005: Unknown MySQL server host 'some_URL_or_ip'

Simply means that connection is not possible for the following (or similar) reasons:

A protocol is specified in the "MySQL host address" field of the SQL DM Connection Manager that does not support MySQL connection. It is a common mistake among beginners to use "http://...", instead of just "www.myveryowndomain.com" or "sales.myowncompany.biz" or "localhost" (if the webserver and the MySQL server is running on the same computer). When connecting to a remote network you may need to ask the Sys Admin there for the correct URL to use for addressing the MySQL server.

However if you use HTTP-tunneling the URL-field on the Tunnel -tab of the SQL DM Connection Manager should be a complete URL with the "http://" protocol specified. This is because the tunneling script must be addressed through a webserver (that is the idea of HTTP-tunneling!).

Error Nº 2003

11.18 Error no. 2003: Cannot connect...

The error message: Error No. 2003: Can't connect to MySQL server on 'localhost' (or some other host)

Simply means that connection is not possible for one of the following (or similar) reasons:

- There is no MySQL server running at the specified host
- Connection to the MySQL server is not allowed using TCP-IP. Check the 'skip-networking' setting in the MySQL configuration file (my.ini on Windows, my.cnf on Unix/Linux). It shall be commented out like '#skip-networking'. If it is not commented out, then do it and restart the MySQL server for the change to take effect. SQL DM needs to connect using TCP-IP.
- Some networking issue prevents connection. It could be a network malconfiguration or a firewall issue. We have experienced sometimes that some firewalls (ZoneAlarm in particular) is blocking TCP-IP connections even if it claims to be disabled. Most often it will help to uninstall and reinstall the firewall.
- When trying to connect to a MySQL server at an ISP this error message often indicates that direct connection to MySQL has been blocked. You must then use HTTP-tunneling or SSH-tunneling to connect.
- Also, you can find more relevant information in Error No. 1130 above. It describes some more special situations when connection to MySQL on Linux.

11.19 SSH-Related errors

11.19.1 SQLyog takes long time to connect when using SSH-tunnel.

If SQLyog is taking long time (more than 5-10 seconds) to connect to your MySQL server using SSH tunneling then there could be problem with the name resolution.
The first thing that the SSH server does before forwarding the connection to the specified MySQL server is to

perform a reverse DNS lookup on the client's IP. The SSH server may cause an unnecessary delay during authentication due to incorrect or absent of reverse DNS settings. So in case of any slowness you should check those settings.
You can also disable most of the server-side lookups by setting UseDNS = "no" in SSHD configuration file (/etc/ssh/sshd_config on most systems). But in that case the MySQL host must be specified with an IP and not a hostname.

11.19.2 SSH - related

- "FATAL ERROR: Unable to authenticate". Occurs in case of a SSH Username/Password Mismatch or in case that you are using a username or a host IP which is listed in "DenyUsers" in the SSHD configuration file (/etc/ssh/sshd_config on most systems).
- "FATAL ERROR: Network ERROR: connection timeout". Network TimeOut (as it says) or simply could not establish contact to a SSH daemon on the URL specified. You even get this if the computer you are trying to contact is not available at all or offline. Also this error happens if you are using public/private key authentication and the keys are not valid. Keys used by SQLyog must be in .ppk -format (same as used by the 'Putty' program).
- "FATAL ERROR: Network ERROR: connection refused". Occurs typically in case of wrong SSH port. (Note that the term 'refused' implies an **active refusal** from the server!)
- The above error may also result from a hardware error on the remote network.
  Also those:
- FATAL ERROR: network error: Software caused connection abort.
- FATAL ERROR: Network ERROR: no route to host
  ... may be hardware related (they both can both be triggered by disconnecting the network cable). But likely that there are more reasons.
- "SSH ERROR: Unable to open connection; Host does not exist"
  ...occurs in case SSH Host does not exist (for instance in case of unsuccessful DHCP or DNS lookup - here there is no **active refusal** from the server, as there was not even an attempt to connect it!)
- **Also see** the note to the MySQL client error 2013 "Lost connection ...".

11.19.3 MySQL - related

As a result of this improvement we can now also retrieve more meaningful MySQL server and client errors than before - for instance:

- If MySQL user/pw is wrong result is MySQL server error 1045: "Access denied ..." as with any type of connection.
- If MySQL port is wrong result is MySQL client error 2013 "Lost connection ...".
  Note that this error also occurs if port forwarding is disabled in SSH configuration (the configuration parameter 'AllowTcpForwarding' is set to 'no' in the 'sshd_config' file). It (here) simply tells that there is no connection from SSH to MySQL for some reason. But the mySQL client API 'thinks' there was **one** connection and that is why is says 'Lost connection ...' and **not** 'Can't connect...'. There was one successful connection - but not to the MySQL server - to the SSH daemon only! But the MySQL client API is not designed to 'see' the difference!
- If MySQL port is empty or ZERO however result **is** MySQL client error 2003 "Can't connect to mysql server ..."11
- If MySQL host is wrong (or cannot be reached from SSH for some reason) result is MySQL client error 2005 "Unknown MySQL server..."

11.19.4 Unspecified error:

And this one is displayed if SSH connection cannot be established for some other reason and it has not been possible to resolve the error to one of the above.
"Could not establish SSH connection. Make sure that the SSH server is running and you are entering correct values for SSH port forwarding."

A concluding note on the popular 'Putty' program and SQLyog SSH-tunneling.

Sometimes when people are having problems with SSH-connections, we often hear "I can connect with Putty without problems". Maybe so, but it does not tell very much (almost nothing actually!) because a connection with Putty or a similar program **is not tunneling** and does not make use of port forwarding on the remote host. Putty simply creates a remote (and secure) shell on the client machine, and connects to the 'mysql' client program on the server. With Putty the MySQL client is the 'mysql' client on the remote server. With SQLyog SSH-tunnel the MySQL client is the client API that is compiled into SQLyog (and SJA). That is why port forwarding is needed and must be functional with SQLyog SSH-tunneling!

11.20 Debug Facility

The concept of our debug code is that it 'takes over' from or 'substitutes' the normal windows error dump message. There is no additional code executing in the background and no additional disk operations or anything else that slows down the program.

Whenever a crash occurs, SQLyog prompts a Message Box that gives you an option to save a special dump file. The debug dump does not contain any user data or information about his system. It just captures the internal state of the application at the time of crash. In situations where a crash is not DATA-specific this is all we need to tell the exact line in the code where the crash occurred. In case the error is DATA specific we will of course also need STRUCTURE/DATA and the QUERY (or whatever operation) resulting in the crash.

Whenever you experience a crash, we request that you send the file to us with a brief description of how that happened. You can use the Forums or the Tickets system as you prefer. We will then reply back if more information is needed.

This functionality depends on the dbghelp.dll file shipped with recent Windows versions. The version of that file must be 5.1 (native with Windows XP) or higher (as shipped with Windows 2003 and Vista). The dbghelp.dll file shipped with Windows 2000 won't work, but you can copy the file from a XP-machine or download it from the Internet. It must be copied to the SQLyog installation folder and this copy (and not the native version) will then be used by the system with SQLyog.