LoxBerry Logging: Tips and tricks

This article collects tips and tricks with LoxBerry's logging capability for both Perl, PHP and Bash, documented in Perl-Modul LoxBerry::Log, PHP Module loxberry_log.php and LoxBerry Logging in Bash.

Logging SDK or own logging?

It is up to you if you do logging and what kind of logging you do.

Without LoxBerry's SDK, the most basic variants are:

- In Perl, print to STDERR (sending your messages to the Apache log for cgi scripts, or to console for normal Perl scripts)
- In PHP, call the error_log function (sending your messages to the Apache log for web requests)
- In Bash, simply pipe to a file.

Using LoxBerry's Logging SDK gives the following advantages:

- You directly can write simple logging commands to your code (e.g. LOGINF, LOGDEB, LOGERR) before you have to tinker about the logging constructor. Output is sent to STDERR. Later, implement the new function to create the file, and all your logging code is working.
- All the stuff of → filtering by loglevel, → setting the loglevel, → generate files and filenames, → show a single logfile, → list all logfiles, etc. is managed by LoxBerry-integrated functions.
- Session-based logfiles are easier to read and handle for users, as files are small and better readable.
- With **Log Manager**, for users it is easier to identify your logfiles, to open your logfiles, to distinguish between different functionalities of your plugin. With the colored status, it is easy to find the log with the error.
- The LoxBerry Logging SDK is more cunning than only put a mass of log lines in a huge file. You should attune to LoxBerry's concept of user-managable logging to get the best benefit.

Tips using the SDK

- The logging of LoxBerry is thought to be session based, not simply file or line based. Every script run is a logging session from start to end. A session should end when the script ends.
- Therefore, the SDK automatically creates a new logfile for every session. All log entries are in that generated file. Using only a single logfile for everything (using the filename parameter) is not the core thought of the Logging SDK and should not be used. It will make forehead wrinkles with strange behaviour if you do so.
- To use the loglevel setting from LoxBerry, you have to set CUSTOM_LOGLEVELS=true in the [SYSTEM] section of your plugin.cfg. See plugin.cfg
- Create the logging object in the main scope of your scripts - and only create it once.
  - Do not create a logging object in a sub function again and again. It will dramatically reduce the overall performance of your script.

    - The Logging SDK uses an object class to handle your logging object, and that also includes to seek and write in the logging database on every creation and destruction of that object. Keep your logging object in the main scope and only create it once, so this operations happen only once. (Creating a logging object in a function, it will be destructed every tome you leave the scope of the function)
- Avoid to override the loglevel in your script. You should not set the loglevel via parameter, as the loglevel is used from the user setting in the Plugin Management or the Log Manager.
- Every logging session must have a LOGSTART event. This persists the creation of a session in the logfile and in the database.
- Every logging session should have a LOGEND event. This persists the finishing of the session and will write the end time of the session to the database.
    - Perl has a "magic" function called END that is always called when a script exits. You could use that to send your LOGEND event.
      Example: https://github.com/mschlenstedt/Loxberry/blob/d16a56984f6a04065f53e1b9b7d8addf31f65676/sbin/log_maint.pl#L385
    - In PHP you also can register a function for shutdown with register_shutdown_function: http://php.net/manual/en/function.register-shutdown-function.php
- Do <u>not</u> filter by loglevel *in your script* - the Logging SDK filters by the loglevel.
- Use the NAME parameter in the thinking of a logging group in your plugin. Example: Your log for the webinterface may use `NAME => "WebUI"`, your REST scripts use `NAME => "REST commands"` and your daemon `NAME => "Daemon"`.
- LoxBerry provides several SDK functions to set the loglevel, to display buttons to the logfiles, and even to list all of your logfiles, from within your plugin. You can use that, or simply do nothing - because loglevel and logfiles are listed in LoxBerry's *Log Manager* automatically.
- LoxBerry automatically does the cleanup of the logfiles. You don't need to pay attention to delete the files.
- BASH Logging: The Bash logging is a bit different in usage and parameters, as in Bash we do not have an object class but only can call functions. Keep a special eye on the mandatory parameters and the order from the examples.
    - FYI: Bash logging interfaces to the PHP Logging SDK (Pre-V1.2.5: Perl Logging SDK) and implements several functions like writing to the logfile itself for performance reasons.
    - It does not provide methods like in Perl/PHP, but only can return variables.
    - It has reduced functionality compared to Perl/PHP.

If you have any issues with the logging functions, please contact us (in LoxForum, or the WhatsApp Developer group) and give us the chance to advise or even fix it.