

Logging Proposal

All factory and entry logs are from the <logs> section in <log_retention>. Frontends have only <log_retention> for all logs but the logic is similar. I'm using the factory as the example but I think we can apply the same changes to the frontend. This also only applies to the master branch with the new logging.

Current process:

```
<log_retention>
  <condor_logs ..../>
  <job_logs ..../>
  <summary_logs ..../>
  <logs max_days="7.0" max_mbytes="100.0" min_days="3.0"/>
</log_retention>
```

There are 3 logs for the factory and the same 3 logs for each entry. We share the attributes in the <logs> section for all the logs.

- 1) name.err.log with messages of type WARN or WARNING
- 2) name.info.log with messages of type INFO
- 3) name.debug.log with messages that type DEBUG (and a few ERROR)

The logSupport module allows all types of messages and we currently capture all for each log. Filters are then applied using the above message types so only those end up in the log. If you want a message to show in multiple places you have to log for each message type.

The general rule for assigning messages seems to be this:

- Warnings, errors, and exceptions trapped are logged as WARN or WARNING in a short form.
- Errors and exceptions trapped are logged as DEBUG (a few ERROR) with the same message as WARN, possibly with a traceback or other details.
 - Sometimes warnings are logged here too.
- Any message about the code flow of the process is logged in INFO only.
- There are a few debug messages not related to issues or errors in the debug log.

These are the issues I see with the current setup:

- You can't log a message one time and have it show up in multiple logs. If you want it in multiple logs, you must log in multiple places.
- No ability to "turn on" or "turn off" debugging.
- Do we want the ability for the admin to configure the logging they want, such as number and types of logs?
- We manually format tracebacks but Python logging has the ability to automatically do this.

- We may not be consistent in how we log and where we log it to. We've also refactored large amounts of code in master so old messages have gone away and we may not have replaced them with new ones.

Proposed process:

My goal was to decouple the logging in the code from how we configure the logs.

Python logging contains the ability to log the follow types of messages:

- debug, info, warning, error, critical, exception (error with exception info added)

If we do all logging based on these types in the code (independent of the actual log files), we can modify the log files as needed without ever having to change the code.

glideinWMS will define these “types” of logs and specify them in the factory config file as follows:

- INFO, DEBUG, ERROR, ALL (we can add more as needed)

```
<log_retention>
  <condor_logs ..../>
  <job_logs ..../>
  <summary_logs ..../>
  <process_logs>
    <log type="INFO" max_days="7.0" max_mbytes="100.0" min_days="3.0"/>
    <log type="DEBUG" max_days="7.0" max_mbytes="100.0" min_days="3.0"/>
  </process_logs/>
</log_retention>
```

The above configuration will allow flexibility in configuring the number and type of logs. It will also allow the admin to “turn on” and “turn off” debugging. The gwms types do not allow configuring what types of messages go in each log.

Python logging allows us to create filters so only the message types that we want will show up in the appropriate log. This will allow us to keep the code independent of how the logs look. For each gwms type of log, we will have the following Python message types:

- 1) INFO: info messages only
- 2) DEBUG: debug, warning, error (includes exception) messages
- 3) ERROR: warning, error (includes exception) messages
- 4) ALL: all messages

Messages of type info, warning, and error will need to be one-line. Debug and exceptions can be as many lines as needed. This means we won't have to log the same message in multiple locations, such as the current case with errors or warnings. Just log the one line version and if you have any other details, add a debug message. NOTE: Exception messages are of type error which means they will show up in the ERROR log with multiline info. If this is not desirable, we can make the ERROR log only warning messages or write code so that exceptions are only logged in DEBUG.

The rule for logging messages should be the following:

- Info messages should trace normal code flow and be one line.
- Warning messages should show problems that don't cause anything to stop normal execution. Should always be one line. Examples:
 - Finding an invalid request classad
 - Request needs more glideins but we are at the limit for that entry
- Error messages are exceptions that alter normal code flow
 - Condor error querying the current queue info
 - Errors during the submission process so you don't get the glideins that you need
- Exceptions include the full traceback.
- Debug messages

Examples:

To keep existing functionality:

```
<process_logs>
  <log type="INFO" max_days="7.0" max_mbytes="100.0" min_days="3.0"/>
  <log type="DEBUG" max_days="7.0" max_mbytes="100.0" min_days="3.0"/>
  <log type="ERROR" max_days="7.0" max_mbytes="100.0" min_days="3.0"/>
</process_logs/>
```

You may get a few more messages in the DEBUG log than before since warning messages will automatically show up there.

To have only one log with all message types:

```
<process_logs>
  <log type="ALL" fname="/path/to/all_file" max_days="7.0" max_mbytes="100.0" min_days="3.0"/>
</process_logs/>
```

This is useful to the developers who don't necessarily need to have messages separated between different log files.

Other suggestions:

To simplify things further, we could also use a wrapper that allows you to log multiple types of messages at once:

```
log_message(info="info_msg" debug="dbg_msg" error="err_msg" warning="warn_msg")
```

This would further decouple logging in the glideinWMS code from the logging implementation.

Questions:

- What happens if no logs are listed? Do we need to check this on startup to make sure there is at least one log specified? Or if none are specified, we always set up the ALL log?