**Logging Design documentation**

**Introduction**

Logging is a well-known mechanism for developers to debug their code. Python offers a very comprehensive logging system: <https://docs.python.org/3/whatsnew/3.11.html#logging>.

The Oradio logger distinguishes 4 levels: DEBUG, INFO, WARNING and ERROR. Each level has a different color setting for easy recognition.

The default logging level is set in the logging module to DEBUG.

For Oradio we use a single logging entity for all modules. The Oradio logger has 3 handlers: 1 for console logging, 1 for logging to file, and 1 for logging to the Oradio Remote Monitoring Service (ORMS).

We use the concurrent\_log\_handler threaded queue logging handler to perform logging in the background.

**Console logging**

Log messages use the console handler to output to the screen. This is useful when testing the modules stand-alone. The different levels have different colors for easy recognition.

**File logging**

We use the concurrent\_log\_handler to write log events to a log file which is rotated when the log file reaches a certain size.

We do not log to file when the CPU is throttled, as writing when throttled can lead to SD card corruption. Log messages when throttled are lost.

**Remote logging**

When the Oradio has an internet connection WARNING and ERROR messages are posted to the ORMS.

When the Oradio connects to the internet it sends a SYS\_INFO identification message to ORMS.

When the Oradio is connected to the internet it regularly sends a HEARTBEAT message to ORMS.

**Avoiding circular imports**

The remote monitoring module and utils module use the logging module, but also have functionality used by the logging module.

To avoid circular import issues the logging module will only import from the remote monitoring or utils module when the logging module function requires it. This way the import is done after fully loading the module itself, avoiding circular import problems.