ACS - Log Location and Retrieval

This document describes how to access and retrieve logs from the aircraft autopilot and payload.

Autopilot

The Pixhawk stores "dataflash" logs on a removable microSD card formatted with a FAT filesystem. The directory structure of the microSD card is given below:

```
APM/
  LOGS/
    1.BIN
    2.BIN
    ... Dataflash logs, increasing number by boot cycle ...
    LASTLOG.TXT  # Contains highest-numbered log file
  TERRAIN/
  MIXER.MIX
```

Each *.BIN file is a dataflash log, representing a single boot cycle of the autopilot; it is important to note that a dataflash log may contain data from zero or more flights.

Logs can be downloaded by removing the microSD card from the autopilot and mounting it on a computer, then copying the *.BIN files onto the computer. It is also possible to download logs over a serial link while the autopilot is powered; see the log download command in mavproxy.py for more information.

To erase logs, either erase all files inside APM/LOGS/ or connect to the powered autopilot using mavproxy.py and use the command log erase. The utility setupPixhawkPlane.sh described in deploy.sh also erases logs.

Payload

Aircraft payloads store two kinds of logs: ROS logs and ROS bags.

ROS logs are stored in `-/.ros/log/` under directories named with GUIDs (unique identifiers). Each directory contains the logs from one launch of the payload software. These log files are textual and contain informational messages from the ROS launch and from individual ROS nodes.

ROS bags are stored on the aircraft payload in `-/bags/`, and are named with the following convention:

```
II_YYYY-MM-DD-hh-mm-ss.bag
```

Where II is the ID of the aircraft (zero-padded to 2 digits) and the timestamp is the time at which the bag started recording. Bags are in a binary format and contain a recording of all ROS topic messages (but not service calls and not parameter changes). Note that in practice some high-throughput topics are explicitly and
intentionally excluded from the ROS bags; see `payload.md` and the code in the `ap_tasks` package in `autonomy-payload` for more details.

Bags can be downloaded via `scp` by booting the payload computer and copying to a local computer:

```
scp odroid@192.168.xxx.yyy::bags/* .
```

The `acs-env` utility `fetchOdroidLogs.exp` automates this for a set of powered aircraft.

To erase payload logs, SSH into the payload and delete the files manually:

```
rm -rf ~/.ros/log/*
rm -/bags/*
```

The utility `updateOdroidPayload.exp` described in `deploy.md` also erases payload logs.

ROS bags may require "reindexing" if the payload is unpowered while the bag is still actively recording. Bag files that were not properly closed will end with the suffix `.active`. To reindex, run `rosbag reindex BAGFILE ...` on those bag files; the resulting (fixed) files will have the same name, and a backup of the original file will be created, replacing the suffix `.active` with `.orig.active`.

Note that, as with autopilot logs, both kinds of payload logs are created per boot cycle, not per flight. Post processing is required to determine which logs contain data from a flight.

**Ground systems**

Not all ground software supports logging at the time of this writing; however, logging does occur where described below.

**Arbiter**

The Arbiter stores a log of its console output in `/tmp/arbiter.log`. This log file is appended to across instances of the Arbiter, and the following precedes any start of a new instance log:

```
# blank line
---
```

It also logs incoming messages on both the "blue" and "red" networks in files

```
/tmp/arbiter_logger.blue.log /tmp/arbiter_logger.red.log
```

respectively. These are binary representations of network traffic, and can be played back using `acs_net_logger.py` in `acs-env`. Log paths can be overridden by command line options `--logger blue PATH` and `--logger red PATH` respectively. Note that these log files are truncated each time the Arbiter starts.