ACS'ing the Bugs

roadmap to release for OTREC

ACS

Daemon

- Known as ‘acsd’
- Controls Hardware
- Spews data into database
- Exposes API for IPC

ncar.github.io/ACS/API

Web / User Interface

- v1 is web-based (HTTP)
- Working on Alpha
- What most people care about
- Most straightforward

Daemon

- 95% complete: can log data, control equipment, etc, start / stop soundings
- System tuning (fixing known CPU hogs)
- Per-Flight startup parameter entry: remove need to hand-modify config files
- Release Management / Release Tooling - source tree to published “packages”
- Data Handling:
  - QC Algorithms to apply to soundings data
  - ‘Raw’ to NetCDF / HDF
- Documentation for Internal, External, End Users, & QC use cases.

API

- Hardware listens for API interactions via HTTP.
- github.com/NCAR/ACSd/docs/API
- RAML files that gets compiled into a HTML
- Currently no automated testing of RAML spec vs API, but could be done.
- Missing some endpoints dealing with data export

API

General

The /channel route interacts with constructs that surround individual channels. This is the main entry point to start, stop, read status, or otherwise manipulate a channel. In general, a *channel* is anything that records sondes data, stores it in the database, and keeps track of necessary accounting associated with the sonde. This can be thought of as nested routes, a master channel wrangler which can start, stop, reinit, pause channels, and the actual channels themselves which do all the work.

Caching explicitly disabled

All data emited from any GET routes located here purposefully has caching disabled due to the transient nature of everything emitted. GETs usually return status of some variety, and the status should be thought of as invalid the second it is received. Most of the state machines are retriving data at multi-Herz rates.

/api/{version}/channel	PUT DELETE GET
/api/{version}/channel/ops/{cid}	PUT DELETE POST
/api/{version}/channel/{cid}	GET
/api/{version}/channel/{cid}/control	POST GET
/api/{version}/channel/{cid}/settings	POST GET

Launcher Ops

- Heartbeat & Versioning
- Auto-Initiate Soundings
- Authentication
- **Channel Ops**
- Launcher Ops
- Requesting Arbitrary Data
- Metadata Extraction and Recording
- Sounding Data
- Spectrum Sweep Data
- System Time
- Log Monitoring
- User Interface Goodies
- Mission => Sounding Table
- Stats for Nerds

Web / UI

- Technically straightforward (state maps to UI)
- Modern HTML5 & Js: ReactJS, D3js, Semantic-UI
- Being worked on now.
- People care only about UI

α-level Components

- Auth & Login Dialogs
- Channel Status
- ACS instance info
- Various Timestamps
- Spectrum Analyzer
- List of Mission Soundings
- Sounding Selector
- Metadata viewer
- Basic Launcher Display
- Realtime table of incoming data

To Be Done

- Sounding Realtime X-Y plots
- Skew-T sounding plot
- Advanced Launcher Controls (GV & GH only)

- UI Refinements
- UI Feedback
- UI Layouts
  - Per-Role: Operator vs Scientist
  - Per-Org: NOAA vs AF

Documentation

- Internal: Source Code documentation and design methodology
- External: REST API (build your own tool)
- End User - End users installing ACS.
- QC - Document what ACS does to the data

SSID: acs-demo
Password: auc-2018

http://acs-demo:3000
http://192.168.1.42:3000
Operator Shared Secret: *avaps*

Stay for the Demo

... or consume nourishing victuals ...

Input Needed

- Buttons that do not do what you expect
- Layout Preferences
- Content organization
- Color Meanings / Indications