uFleetManager Guide

Raphael Segal

June 16, 2017

1 Purpose

This guide is intended to explain to a member of the PAVLAB how to use and improve the fleet manager app. Those members are expected to have a basic familiarity with MOOS-IvP, C++, and Bash.

2 Usage

2.1 Dependencies

`uFleetManager` was developed for Mac. It is in principle compatible with Linux, but that has never been demonstrated.

Currently, the only dependency is `ncurses`. On a Mac, use Macports or Homebrew like so:

```
port install ncurses
```

The usage is slightly more complicated on Linux. Without having gotten it working, it's hard to say for sure, but it looks like `libncurses5-dev` is the correct version. So on Ubuntu, the usage would be

```
apt-get install libncurses5-dev
```

2.2 Installation

`uFleetManager` is bundled in the `moos-ivp-aquaticus` tree. Assuming you haven't already, install `moos-ivp-aquaticus` in your home directory.

2.2.1 Download ARO

Most users will use the Anonymous Read Only version of `moos-ivp-aquaticus`:

```
svn co https://oceanai.mit.edu/svn/moos-ivp-aquaticus-aro-trunk/trunk
moos-ivp-aquaticus
```

2.2.2 Download for Editing

A few users will have edit and commit privileges; speak to Dr. Benjamin.

2.2.3 Enable

Open ~/moos-ivp-aquaticus/src/CMakeLists.txt and in the BUILD_ALL section, find the line ADD_SUBDIRECTORY(uFleetManager) and uncomment it. Remember to recomment it before committing code, and check it after pulling down new code.

2.3 Running the Fleet Manager

Build the fleet manager with the aquaticus build script:

~/moos-ivp-aquaticus/build.sh

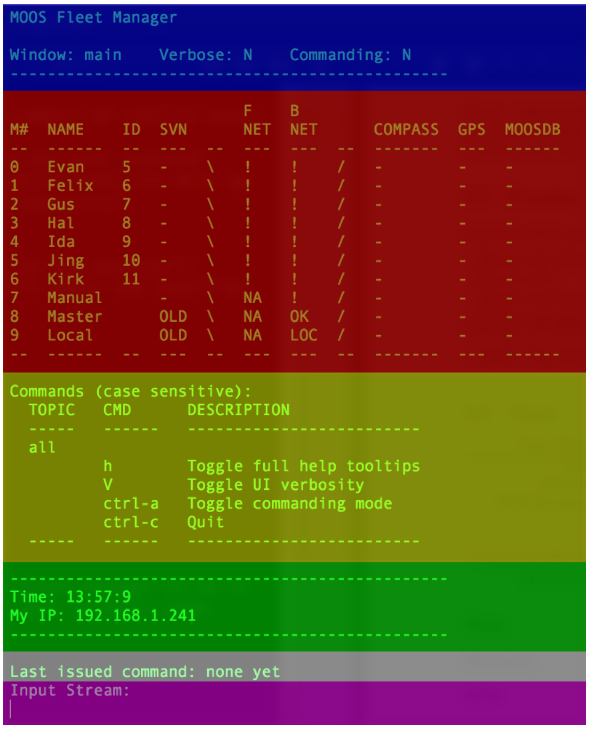
Run it with

~/moos-ivp-aquaticus/bin/uFleetManager

With no arguments, the fleet manager will monitor all known machines, but not be able to launch missions. To add a mission, write a config file (see Config Files) and include it with --file:

~/moos-ivp-aquaticus/bin/uFleetManager --file /path/to/foobar.moos

2.4 Layout



Blue Header; displays the state of the app

Red Window; displays the view indicated in the header (see below for details on each view).

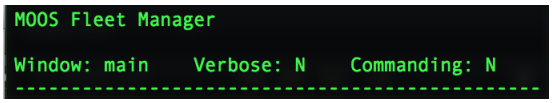
Yellow Help; displays the currently available command set.

Green My Machine; displays own computer information. 'Time' is the one topic that is expected to change frequently and consistently, and therefore can be used to determine if the app has crashed.

Grey Last command; displays the last command issued.

Purple Input; shows currently input characters

2.4.1 Header



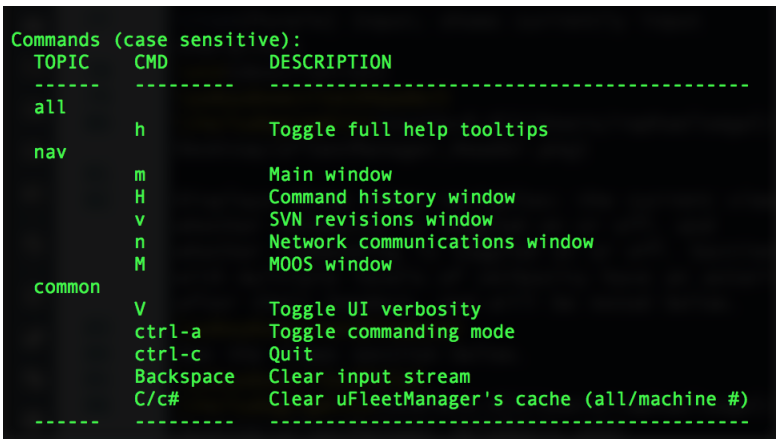
Displays three state variables: the current view, whether verbosity is toggled on or off, and whether commanding is toggled on or off. Sections with multiple levels of verbosity have an asterisk after their headers, and will be noted below.

2.4.2 Windows

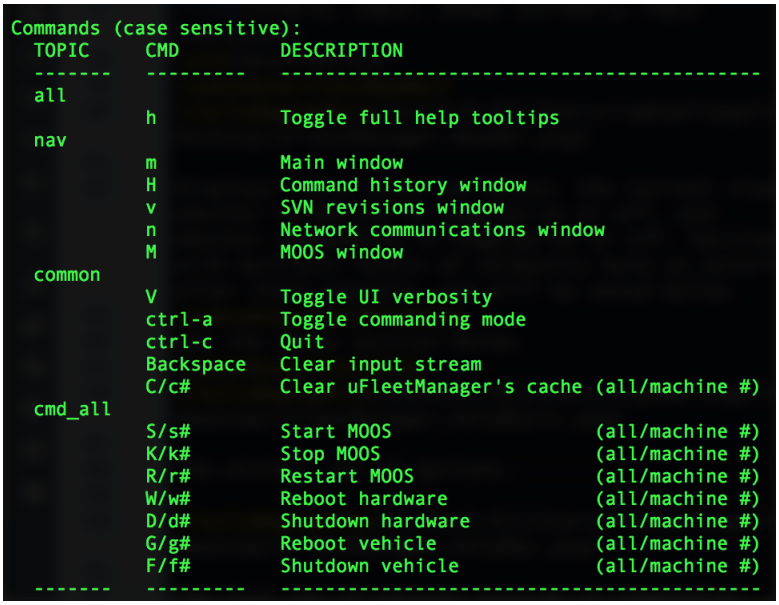
See the Views section below.

2.4.3 Help

The minimal set of options.



The full set of options outside of commanding mode; contains the common set, navigation commands, and the command to clear the local cache of information requests.

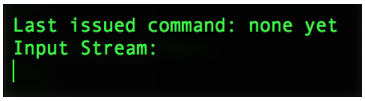


The full set of options, including those in commanding mode.

2.4.4 My Machine

Stats about your own machine. `Time` serves as a responsive UI element, demonstrating that the app is actually refreshing. `MY IP` is helpful if you're running the shoreside on your computer and need to update the machine's UI, but it also indicates which wifi network you're on; if the first block is 10 you're probably on MIT-GUEST, if the wifi is 192.168.1.X, you're probably on kayak-local.

2.4.5 Footer

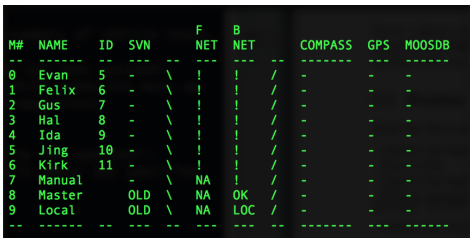


Information about keys you're currently inputting, and the executive summary of the command you've most recently input.

2.5 Views

View Name	Nav Key	Description
Main	m	Main window, provides a ready/not ready summary of vehicle state.
Network	n	Vehicle addresses and whether ping and ssh test succeed
SVN Revisions	v	Lists revisions and summarizes which trees are most up-to-date for `moos-ivp`, `moos-ivp-aquaticus`, `moos-ivp-colregs`, `pablo-common`, and `mokai-common`
Command History	H	Lists the commands dispatched by the operator
MOOS-IvP	M	Lists mission configuration and details about the specified mission

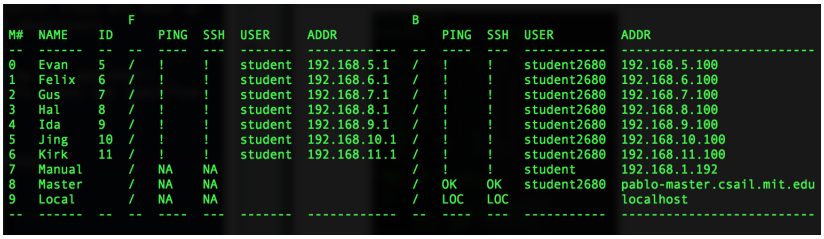
2.5.1 Main



M#	NAME	ID	SVN	F NET	B NET	COMPASS	GPS	MOOSDB
0	Evan	5	- \	!	! /	-	-	-
1	Felix	6	- \	!	! /	-	-	-
2	Gus	7	- \	!	! /	-	-	-
3	Hal	8	- \	!	! /	-	-	-
4	Ida	9	- \	!	! /	-	-	-
5	Jing	10	- \	!	! /	-	-	-
6	Kirk	11	- \	!	! /	-	-	-
7	Manual		- \	NA	! /	-	-	-
8	Master		OLD \	NA	OK /	-	-	-
9	Local		OLD \	NA	LOC /	-	-	-

Topic	Explanation	Comments
M#	Machine #; the # in the Commands section	Limited to $0 <= M\# < 10$
Name	Vehicle Name	List hard coded in Configuration class
ID	Lab vehicle id system, alpha=1, bravo=2, ...	
SVN	Summary; worst status from all its svn trees	OLD and NEW are relative amongst vehicles e.g. if even one of your trees is out of date, then your summary will be OLD. See SVN view for more detail
F NET	Front Seat network summary	ssh and ping; see Network view for more detail
B NET	Back Seat network summary	same as F NET. Single-computer robots are back seats
COMPASS	Reports if vehicle's compass is up	M300 common failure mode is NaNs Mokai common failure mode is disconnects
GPS	Reports GPS status	M300 reports PDOP Mokai only reports connectedness
MOOSDB	Counts the MOOSDB processes running	1 is the only sane value Also lists the vehicle's team, if one is given; see the MOOS-IvP section

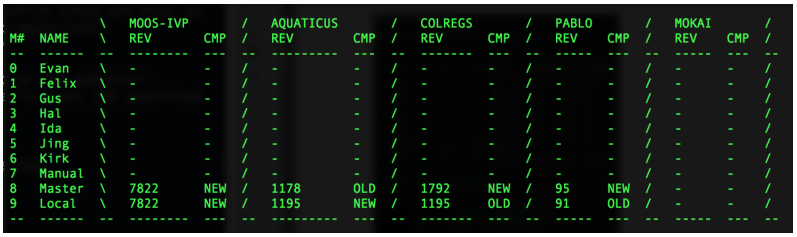
2.5.2 Network



M#	NAME	ID	F	PING	SSH	USER	ADDR	B	PING	SSH	USER	ADDR
0	Evan	5	/	!	!	student	192.168.5.1	/	!	!	student2680	192.168.5.100
1	Felix	6	/	!	!	student	192.168.6.1	/	!	!	student2680	192.168.6.100
2	Gus	7	/	!	!	student	192.168.7.1	/	!	!	student2680	192.168.7.100
3	Hal	8	/	!	!	student	192.168.8.1	/	!	!	student2680	192.168.8.100
4	Ida	9	/	!	!	student	192.168.9.1	/	!	!	student2680	192.168.9.100
5	Jing	10	/	!	!	student	192.168.10.1	/	!	!	student2680	192.168.10.100
6	Kirk	11	/	!	!	student	192.168.11.1	/	!	!	student2680	192.168.11.100
7	Manual		/	NA	NA			/	!	!	student	192.168.1.192
8	Master		/	NA	NA			/	OK	OK	student2680	pablo-master.csail.mit.edu
9	Local		/	NA	NA			/	LOC	LOC		localhost

Topic	Explanation	Comments
M#		See the Main view
Name		See the Main view
ID		See the Main view
F	Front Seat block	
PING	Is ADDR reachable by ping	NA indicates no front seat expected
SSH	If USER@ADDR can run a simple test command	NA indicates no front seat expected
USER	The front seat username	
ADDR	The front seat address	
B	Back Seat block	Single-computer vehicles are considered back seats
PING	Is ADDR reachable by ping	
SSH	If USER@ADDR can run a simple test command	
USER	The back seat username	
ADDR	The back seat address	

2.5.3 SVN

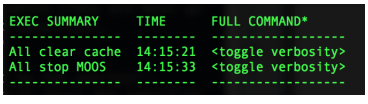


M#	NAME	MOOS-IVP REV	MOOS-IVP CMP	AQUATICUS REV	AQUATICUS CMP	COLREGS REV	COLREGS CMP	PABLO REV	PABLO CMP	MOKAI REV	MOKAI CMP
0	Evan	-	-	-	-	-	-	-	-	-	-
1	Felix	-	-	-	-	-	-	-	-	-	-
2	Gus	-	-	-	-	-	-	-	-	-	-
3	Hal	-	-	-	-	-	-	-	-	-	-
4	Ida	-	-	-	-	-	-	-	-	-	-
5	Jing	-	-	-	-	-	-	-	-	-	-
6	Kirk	-	-	-	-	-	-	-	-	-	-
7	Manual	-	-	-	-	-	-	-	-	-	-
8	Master	7822	NEW	1178	OLD	1792	NEW	95	NEW	-	-
9	Local	7822	NEW	1195	NEW	1195	OLD	91	OLD	-	-

Topic	Explanation	Comments
M#		See the Main view
Name		See the Main view
ABC REV	Revision number of the copy of ABC	
ABC CMP	ABC tree is comparatively OLD or NEW(est)	Contacting a new machine may change who is newest

The tracked trees are `moos-ivp`, `moos-ivp-aquaticus`, `moos-ivp-colregs`, `pablo-common` and `mokai-common`. The PABLO and Mokai trees tend to not coexist, so they are special cased on the Main view such that having one but not the other will not bubble up an error.

2.5.4 History

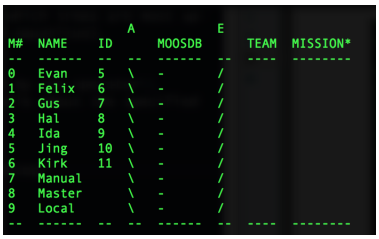


EXEC SUMMARY	TIME	FULL COMMAND*
All clear cache	14:15:21	<toggle verbosity>
All stop MOOS	14:15:33	<toggle verbosity>

Topic	Explanation	Comments
EXEC SUMMARY	Explains the command	Most recent is is displayed in the footer
TIME	Time command was dispatched	Local computer time
Full Command	Full command as sent over the wire	Often very large; toggle verbosity to read

Only the last ten commands are displayed.

2.5.5 MOOS-IvP



Topic	Explanation	Comments
M#		See the Main view
Name		See the Main view
ID		See the Main view
A	Actual results block	Values here read off the target machine
MOOSDB		See the Main view
		Note that team isn't included here, unlike in the Main view
E	Expected results block	Values here are what uFleetManager would dispatch
Team	Team that the machine is on	Read from config
Mission	Launch file and args	If this is blank, startMOOS doesn't dispatch anything
		Toggle verbosity to see full path

2.6 Commands

Note: many of these commands require the operator's fleet manager to be in "commanding mode"; they will be indicated by a star next to their name in this list.

Command	Key Feed	Description
Quit	ctrl-c	Close uFleetManager
Help	ctrl-h	Toggle help text; default is most hidden
CMD mode	ctrl-a	Toggle command mode; default is not in command
Verbose mode	V	Toggle verbose mode; default is terse
Clear	Backspace	Clear key feed
Start MOOS*	S	Start MOOS on each available machine, if possible
	s#	Start MOOS on machine #, if possible
Stop MOOS*	K	Stop MOOS on all available machines (aka `ktm`)
	k#	Stop MOOS on machine #
Restart MOOS*	R	Equivalent to the sequence `K S`
	r#	Equivalent to the sequence `k# s#`
Reboot Machine*	W	Reboot all the machines (back seats)
	w#	Reboot machine #'s back seat
Shutdown Machine*	D	Shutdown all the machines (back seats)
	d#	Shutdown machine #'s back seat
Reboot Vehicle*	G	Reboot each of the machines' front seats, if they have them
	g#	Reboot machine* #'s front seat, if it has one
Shutdown Vehicle*	F	Shutdown each of the machines' front seats, if they have them
	f#	Shutdown machine #'s front seat, if it has one

2.7 Config files

3 Modifying the Fleet Manager

3.1 Adding Views

There are four places in `ui.cpp` that need to be modified to add a view; the help text, the table formatting, the navigation character handlers, and the view render block.

Help Text Find the block in `UI::setTableFormats()` of additions to `m_help["nav"]`; the syntax is a struct of three strings;

{`view name, navigation character, help text description`}

Table Formats Find the blocks in `UI::setTableFormats()` like

```
foo.push_back("BLAH")

foo.push_back("BLAH BLAH")

m_headers["foobar"].push_back(foo)
```

The map `m_headers` stores the headers for each view. A header is a vector of vectors of strings; the outer vector stores rows to feed to ACTables, and the inner vector stores the strings to put in each column of the table. Sections that have multiple header rows should be specified as

```
foo1.push_back("BLAH"); foo2.push_back("DUH")

foo1.push_back("BLAH BLAH"); foo2.push_back("DUH DUH")

m_headers["foobar"].push_back(foo1);

m_headers["foobar"].push_back(foo2);
```

the first block would result in a table formatted like

BLAH	BLAH BLAH
...	...

while the second block would result in a table formatted like

BLAH	BLAH BLAH
DUH	DUH DUH
...	...

Usually one or two lines is sufficient; the first line to delineate sections and the second line for column headers. Add your own section, consistent with what you put in the Help Text section. You will revisit this in Adding Topics to a View.

Character Handlers Find the block in `UI::actOnKeyPress()` with sequences like

```
else if (m_key_feed=="M") {

 m_view = "MOOS";

 command_match = true;

}
```

and add you own, consistent with the information you put in the Help Text section

View Render Find the block in `UI::printWindow()` that looks like

```
if (m_view=="FOO") {

 view_table << something

}

else if (m_view=="BAR") {

 view_table << something else

}

...
```

and add a similar block checking for your new view. See the next section for how to fill out that block.

3.2 Adding Topics to a View

There are two places in `ui.cpp` that need to be modified to add a column to a view; the table formatting block and the view rendering block.

Table Formats Find your block in `UI::setTableFormats()`, the same as your Table Formats block from Adding Views. Add a string to all the inner vectors. Disallowed[1] strings include "\n" and "|", and allowable strings include "", "\", "/", and "#".

View Render Find your block in `UI::printWindow()`, the same as your View Render block from Adding Views. The `n`th line such as `view_table << something` will fill the `n`th column of the table as ordered in Table Formats.

3.3 Adding Commands

The interface for the UI to call vehicle commands, to get information or to take action, is public ManagedMoosMachine methods.

Commands are fired off into the void, with a file to write results back to. These files are opened and read synchronously with the local machine, with a small proability[2] of reading partially written

[1] Used by ACTables for formatting.

[2] Determined by the duty cycle of file IO

data[3]. This architecture approximates threading[4], but does not require maintainers to understand threading per se.

3.3.1 Dispatching

3.3.2 Standard PAVLAB Commands

3.3.3 Special Commands

3.3.4 Reading Mail

3.3.5 Required Variables and Caching

3.4 Miscellaneous

3.4.1 Tips and Tricks

4 Configuring Machines to work with the Fleet Manager

4.1 SSH Keys

4.2 Shell Startup and Sources

4.3 Software

4.4 Permissions

5 Debugging

Symptom	Issue	Resolution
Semicolons in command (see History)	Needs to be \n	Replace in relevant ManagedMoosMachine dispatcher function.

[3]uFleetManager's networking layer is written such that in that case, the message ID is the last thing written, and only once it is complete will the app do anything with that data. In formal terms, this satisfies only the Consistency pillar of CAP. If the user clears the cache agggressively, it also weakly satisfies Partition Tolerance.

[4]This architecture was selected in keeping with Dr. Benjamin's standing instructions that any user with basic C++ and Bash experience should be able to understand any code in the lab. The PAVLAB considers threading a non-basic feature.