

MOMA Python Packages

This page lists useful Python (and some shell) scripts that facilitate data extraction and analysis, and links to install packages containing these scripts. A more detailed description of library capabilities can be found [here](#).

To determine which version of 699util you have installed, you can type `pip list` in your terminal.

Installation Instructions

For installation instructions, go to the [Python Tool Installation](#) page, or contact [MOMA software developer](#).

If you already have a package installed and just want to upgrade to a new package, you can:

- download the latest DMG file from this page
- double click the DMG file to mount it
- Open a terminal
- type `cd /Volumes/699util_Installer`
- type `./install699util.sh`
- restart your terminal.

List of Scripts

Script Name	Description
c_apv_ops.py	count the number of APV operations
c_emcount.py	count ions against the multipliers
c_emruntime.py	determine how long EMs were on
c_expectedvaluechk.py	report actual vs. expected housekeeping values
c_extract_hkid_928.py	stream HKID 928 (the 1 KHz RSIM current) data to a file (this HKID is too dense for c_tmfields.py)
c_filament_cycles.py	count filament cycles
c_ioncount.py	performs various ion counting calculations for MOMA scans
c_lasercount.py	count laser pulses
c_look_for_missing_sci_data.py	print errors for missing science data
c_maifruntime.py	returns the amount of time the WRP was on for the telemetry file
c_momaconsume.py	run various consumable scripts
c_momafilTime.py	determine how long filaments were on
c_momascan.py	plot voltage vs. ion counts

Script Name	Description
c_pktsummary.py	print a table of packet types and number of occurrences
c_plotdatarate.py	plot the instrument's data rate from the Rover's perspective
c_print_meta_markers.py	print metamarkers that apply to the tm.mom file
c_pwm_cycles.py	count PWM cycles
c_sync_moma_model_data.py	sync new lab data to SVN
c_t0.py	print t0 for various time systems
c_tid_to_mzml.py	output science scans in MZML format
c_tidssummary.py	output a JSON file summarizing TID data
c_tmfields.py	output housekeeping values
c_tmmarker.py	print markers in the tmfile
c_tmmsg.py	print the message log
c_tmplot.py	plot housekeeping data
c_tmsequence.py	check the packet sequence numbers and report anomalies
c_tmsummary.py	print a summary of each packet in the telemetry file
c_valve_2_open_time.py	determine how long valve 2 was open
c_valve_cycles.py	count the number of valve cycles across a range of TIDs
add_gse_pkts.py	add or modify the GSE packets at the beginning of telemetry files
boxcar.py	determine boxcar average of two columns of data
checksum.py	calculate the Fletcher checksum of files or input from stdin
evref	copy an expected value reference file to current directory and run expectedvaluechk.py
extract_science_data.py	write science data to files
filterExpectedValues.py	filter out ExpectedValue.txt lines for which there are no markers in the tm file
fixPacket1.py	creates tm.mom.fix such that packet 1 is modified to match the cwd
iniget.py	parse an INI file and print a specific value
make_metafile.py	make a tm.meta file for a TID
output_patched_database.py	apply patch files, and print out the resulting database
playtm.py	replay a telemetry file packet by packet
plotref	copy a plot reference file to the CWD and execute it
print_sequence_numbers_and_timestamps.py	what a descriptive name
pump_down_trend.py	trend APV pressure pulse information in LDI mode
rf_chk.py	output RF monitor values for a given bin
runall.py	execute a command across all (or a range of) TIDs
threshold_timestamp.py	print the time that an HKID went above (or below) a threshold
tids.sh	defines convenient functions to jump to TID folders

Script Name	Description
time_between_msgs.py	print the amount of time between messages in a message log (or between two occurrences of the same message)
tmdump.py	generate a hex dump of a telemetry file, separated by packets
tmexcerpt.py	filter data from a telemetry file to a smaller telemetry file
tmkeys.py	parse and print the telemetry database

What's with all this "c_" stuff?

Scripts that begin with "c_" are actually using MOMA Data Viewer code to examine MOMA telemetry files. In many cases, you can omit the "c_" (for example, by running `tmfields.py` instead of `c_tmfields.py`) to instead use the pure-Python version of the script. However, we no longer support the pure-Python telemetry extraction library, so its output will become less reliable as time goes on.

I want to write my own script!

MOMA Python packages will create the directory `"${HOME}/custom_699_utils"` and add it to your PATH. So, users can write a Python script that uses [the MOMA Python telemetry extraction library](#), stick it in this directory, make it executable, and then use it like any other script. Future package installs will not modify the contents of this directory, so your scripts will persist.

Bug Reports

Please do not report bugs on this wiki. Instead, use the [Bugzilla server](#) we have set up for bug tracking and reporting. Once you have set up an account, you may file a bug for the python software at the page for [Command-line Utils](#).

Releases

3.03.12

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util-3.03.12.dmg

C++ Revision: 4942

Python Revision: 2278

Release Date: 2017-03-16

- ``import c699util`` now resolves `_c699util.so`'s shared library dependencies

3.03.11

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util-3.03.11.dmg

C++ Revision: 4934

Python Revision: 2274

Release Date: 2017-03-15

- Added support for new laser decode types
- Adds new marker_relative_timestamp() method to PythonStyleMomaScan()

3.03.09

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util-3.03.09.dmg

C++ Revision: 4693

Python Revision: 2252

Release Date: 2017-01-30

- Fixed a bug where MAIF datapoint timestamps were incorrect if they had a smaller timestamp than their packet's timestamp.
- Implemented c699util.get_tmfile()
- Implemented c699util.get_science_data_cache()

3.03.08

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util-3.03.08.dmg

C++ Revision: 4666

Python Revision: 2247

Release Date: 2017-01-24

- Fixed a bug where the NEXT_MARKER and MARKER end conditions could randomly fail.

3.03.06

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util-3.03.06.dmg

C++ Revision: 4608

Python Revision: 2239

Release Date: 2017-01-12

- Added support for TVAC Status packet (type 42).

3.03.05

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util-3.03.05.dmg

C++ Revision: 4592

Python Revision: 2234

Release Date: 2017-01-10

- c_expectedvaluechk.py now supports temperature dependent values.

3.03.04

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util-3.03.04.dmg

C++ Revision: 4592

Python Revision: 2233

Release Date: 2016-12-16

- Released c_bus_energy.py.

3.03.03

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util-3.03.03.dmg

C++ Revision: 4592

Python Revision: 2231

Release Date: 2016-12-15

- Fixed an issue found by Brad where c_wrp_cycles.py was the wrong name, causing install699util.sh to fail.
- Included c_print_meta_markers.py in the package release.

3.03.02

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util-3.03.02.dmg

C++ Revision: 4592

Python Revision: 2229

Release Date: 2016-12-15

- Fixed an issue found by Brad where regular expression matching in the message log (which happens with the new meta marker "message" trigger) wasn't stripping away the timestamp and whitespace before checking for a match.

3.03.01

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util-3.03.01.dmg

C++ Revision: 4591

Python Revision: 2227

Release Date: 2016-12-14

- The "message" trigger now has an offset_in_seconds field
- The "message" trigger can now be used as an end condition

3.03.00

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util-3.03.00.dmg

C++ Revision: 4589

Python Revision: 2222

Release Date: 2016-12-14

- c699util objects now have docstrings, which are available through the help() function.
- meta markers are no longer generated when TMFile and Science Caches are constructed, which speeds things up. Now, meta markers are only generated if one of the get_meta_markers() methods is called.
- implemented "message" meta marker start condition
- added optional script configuration fields to the "marker" start condition
- the "marker" and "next_marker" end conditions will now only end a meta marker if the timestamp of the marker packet is greater than or equal to the meta marker's start time

3.02.03

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util-3.02.03.dmg

C++ Revision: 4509

Python Revision: 2215

Release Date: 2016-11-15

- Implemented decode type 230, to support HKID 408 -- SUM PKT TIC

3.02.02

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util-3.02.02.dmg

C++ Revision: 4483

Python Revision: 2200

Release Date: 2016-11-04

- MOMA Python tools are now aware of the metamarkers defined in momagse/MetaMarkerRules.
- `c_print_meta_markers.py` can be used to see what metamarkers apply to a tm.mom file, and to print highly specific metamarker rule file errors
- Metamarkers can be used for filtering wherever a normal marker would be used. For example:
`c_tmfields.py 811 --mkid 50050`
- bogus SWTS1 and SWTS2 packet timestamps are now corrected

3.01.01

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util-3.01.01.dmg

Revision: 2175

Release Date: 2016-10-14

- Markers are now assigned to packets based on timestamp, rather than packet index. **Note that many scripts still extract datapoints from packets, and assume that those datapoints always have the same markers as their parent packets.** MOMA marker assignment is a work in progress.

3.01.00

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util-3.01.00.dmg

Revision: 2167

Release Date: 2016-10-03

- This release includes the new MOMA Data View timestamp resolver, which properly handles timestamp resolution when resets occur. Using the same time resolution code for MOMA Data View, Python scripts, and XINA should eliminate time consistency issues.
- evref output is now sorted first by MKID, and next by HKID

3.00.02

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util/699util-3.00.02.dmg

Revision: 2156

Release Date: 2016-08-30

- Fixed a bug where (c_)sync_moma_model_data.py didn't have access to (c_)emcycles.py.

3.00.01

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util-3.00.01.dmg

Revision: 2155

Release Date: 2016-08-30

- c_expectedvaluechk.py and evref now read XINA exported limits. They are no longer compatible with old limits files.

3.00.00

DMG (Mac) Installation Package: https://s3.amazonaws.com/699_bin/699util/699util-3.0.0.dmg

Revision: 2151

Release Date: 2016-08-22

- We are now releasing MOMA Python packages.

Revision #1

Created 22 March 2023 20:38:33 by Nick Dobson

Updated 24 March 2023 13:47:06 by Nick Dobson