

# HERA Monitor and Control Subsystem Definition

HERA Team

September 8, 2020

## 1 Introduction

HERA is an international experiment to detect and characterize the Epoch of Reionization (EOR). The telescope is located at the South African SKA site in the Karoo Astronomy Reserve. This note summarizes Monitor and Control (M&C) subsystem for HERA.

Monitor and Control has two main tasks: configuration management and real time metadata logging. Configuration management involves tracking all the physical parts of the telescope and how they are all connected. Real time metadata logging tracks the performance, settings and communications of the subsystems.

The M&C system is built around a database with a well documented table schema and a python software layer to provide a simple developer framework. It also includes various online daemons for monitoring things, and both a front end web-based user interface and a command-line interface to support analysis code.

Software is contained in the repository [https://github.com/HERA-Team/hera\\_mc](https://github.com/HERA-Team/hera_mc).

The organization of this document is as follows: the high-level M&C requirements and the design specifications are in laid out in section 2, the configuration management is described in section 3, the database tables are detailed in section 4 and future plans are sketched out in section 5.

## 2 Requirements and Specifications

### 2.1 Requirements

These were developed and established in early 2016 in discussions among the subsystem leads.

1. Ability to fully reconstruct the historical state of the system.
2. All interactions between subsystems must go through or be logged by M&C.
  - a. Both subsystems in an interaction are responsible for logging communications to M&C.
  - b. Subsystems in an interaction are responsible for logging communications to M&C.
3. Operational metadata (e.g. temperatures, correlator bit occupancies) must be logged to M&C.
4. High availability (M&C must not limit uptime of telescope).
5. M&C is a provider of information about observations to end-users and must be available to them

### 2.2 Design Specification

These were developed and established in 2016 based on the requirements.

1. SQL database
  - a. DB Design principle: every logical sub group has a group of tables. One adds tables to do more things. E.g. different versions of subsystems add new tables. Operations reference which tables they use.
  - b. This document (and appendices) will contain all table definitions.

- c. Use careful dB design to avoid duplicated data, make table links/data relationships clear, use many-to-one and many-to-many links.
  - d. Transactions must be used to ensure DB integrity.
  - e. Must be mirrored in some fashion to observer locations.
2. At least one SW interface layer will be provided.
    - a. It's not required to interact with M&C.
    - b. Must support relational db (i.e. multiple column primary and foreign keys) and transactions.
  3. Hardware
    - a. LOM capabilities
    - b. Multi-teraByte mirrored disk RAID
    - c. Backup machine available on site

### 3 Configuration Management

The HERA array/part configuration management database is a set of five tables within the larger hera\_mc database, which is maintained on-site in the Karoo. The tables are detailed in the Appendix, but they are:

psql table	in python file	with class name
geo.location	hera_mc/geo.location.py	GeoLocation
station_type	hera_mc/geo.location.py	StationType
parts	hera_mc/cm_part_connect.py	Parts
part_info	hera_mc/cm_part_connect.py	PartInfo
connections	hera_mc/cm_part_connect.py	Connections
cm_version	hera_mc/cm_transfer.py	CMVersion
apriori_antenna	hera_mc/cm_part_connect.py	AprioriAntenna
part_rosetta	hera_mc/cm_part_connect.py	PartRosetta

The databases are structured primarily around *parts* and *connections*. *Parts* are meant to be single items that, in theory at least, are a thing that can be replaced as a unit. *Connections* define *ports* on a given part and connect two ports together. By convention, HERA part numbers (**hpn**) are all upper case and ports are all lower case.

All parts and connections are timed in that they have a start and stop time of operation. If stop is **None**, then it is active (it is given a date in the relatively far future). There are currently two special parts (one at each end of the signal chain) that are “geo-located” parts and have entries in the geo.location table: **station** and **node**.

Parts are hooked together via connections of their ports, as defined in the connection table. For the supporting code, architecture dependent parameters are set in the **cm\_sysdef** module.

To capture the known readiness of antennas, an *a priori* antenna table has been defined to facilitate a work flow to incorporate new antennas into the array and capture and known recalcitrant antennas. The table provides an enumerated readiness level for each antenna station.

#### 3.1 Operation Workflow

The work flow in this context relates to antenna migration between one of the *a priori* status states above. It is handled by the operations team, which will review the situation on a roughly monthly cadence. The enumerated states are:

- **dish\_maintenance**: dish is either not completed or being repaired
- **dish\_ok**: dish is ready for use
- **RF\_maintenance**: receiver system is either not installed or being repaired

- **RF\_ok**: receiver system ready for use
- **digital\_maintenance**: digital system is either not installed or being repaired
- **digital\_ok**: digital system is ready for use
- **calibration\_maintenance**: calibration has not been conducted or current
- **calibration\_ok**: ready to use in analysis
- **calibration\_triage**: an issue is being worked on

### 3.2 Hookup Connections

Hookup connections are those that uniquely map the signal chain from the antenna to the node, including the specific correlator input. These are the connections used to generate the correlator hookup. The signal chain hookup for HERA is given below and shown in Fig. 1, shown in black. The list below describes the hookup connections.

- **Station**: geo\_located position. See prefixes in table station\_type (*e.g.* HH for herahex)
- **Antenna**: This is also the correlator number which now matches station number. **A**{#}
- **Feed**: element feed (Vivaldi). **FDV**{#}
- **FEM**: front-end module. **FEM**{#}
- **Node bulkhead plate**: bulkhead plate on node. **NBP**{#}
- **PAM**: post-amp module in node. **PAM**{#}
- **SNAP**: digitizer/channelizer in node. **SNP**{#}
- **Node**: field-deployed node as a part **N**{#}
- **Node**: geo\_located position for node as a station **ND**{#}



<b>geo_location.py</b>	Defines <code>station_type</code> and <code>geo_location</code> . Provides update utility for <code>geo_location</code> . Contains <code>is_in_geo_location</code> and <code>is_in_connections</code> .
<b>geo_handling.py</b>	Contains myriad defs that handle various geo functionalities.
<b>cm_partconnect.py</b>	Defines <code>parts_paper</code> , <code>part_info</code> , and <code>connections</code> . Provides update utilities for <code>parts_paper</code> and <code>connections</code> . Contains <code>_get_part_revisions</code> .
<b>cm_table_info.py</b>	Has mapping and order of tables and classes for initialization.
<b>cm_handling.py</b>	Defines the <code>Handling</code> class to handle various configuration management functionalities.
<b>cm_active.py</b>	Contains modules that load parts/connections active for given date.
<b>cm_transfer.py</b>	Contains myriad defs that help package and initialize the cm tables.
<b>cm_revisions.py</b>	Contains myriad defs that deal with finding revision numbers etc.
<b>cm_hookup.py</b>	Defines the <code>Hookup</code> class to help determine and show the full part hookup.
<b>cm_dossier.py</b>	Contains classes for the part/connection cm information.
<b>cm_utils.py</b>	Contains various defs called by other modules.
<b>cm_redis_corr.py</b>	Modules to get/read relevant cm info from/into redis.
<b>cm_sysutils.py</b>	Various system-wide modules
<b>cm_sysdef</b>	Architecture dependent definitions.

### 3.4 Scripts

This section provides a high-level overview of the high-level scripts: `geo.py`, `dossier.py` and `hookup.py`.

#### 3.4.1 Geographical Information: `geo.py fg.action --arguments`

Has various plotting/printing options for station information. `fg.action` is the “foreground action”, that is the locations that will get printed and shown on top if plotting. See `geo.py -h` for options. Available foregrounds are (note you need the first letter only):

- `a[ctive]`: those antennas that are shown as fully hooked up through the correlator
- `i[nstalled]`: those antennas whose structure is installed
- `p[osition] <csv-list>`: specified antennas in csv-list (e.g. HH1,HH4)
- `c[ofa]`: the center-of-the-array
- `s[ince]`: antennas installed since date/time supplied in arguments
- `n[one]`: no foreground, will just show the background (or nothing if not plotting)

#### 3.4.2 Part/Connection Information: `dossier.py view --arguments`

Has various printing options for part information (part info, connections, hookup, types, etc). The available views are (first letter only needed): parts, connections, notes, revisions. See `dossier.py -h` for options.

#### 3.4.3 Cascaded Connection Information: `hookup.py --arguments`

Displays the signal chain hookup information. See `hookup.py -h` for options.

## 4 Table Definitions

4.1	Observations	7
4.1.1	hera_obs	7
4.2	Common tables	7
4.2.1	server_status (template)	7
4.2.2	subsystem_errors	7
4.2.3	daemon_status	8
4.3	RTP Tables	8
4.3.1	rtp_server_status	8
4.3.2	rtp_status	8
4.3.3	rtp_process_events	8
4.3.4	rtp_process_record	8
4.3.5	rtp_task_resource_record	9
4.4	Librarian Tables	9
4.4.1	lib_server_status	9
4.4.2	lib_status	9
4.4.3	lib_raid_status	9
4.4.4	lib_raid_errors	10
4.4.5	lib_remote_status	10
4.4.6	lib_files	10
4.5	Correlator Tables	10
4.5.1	correlator_config_file	10
4.5.2	correlator_config_status	11
4.5.3	correlator_control_state	11
4.5.4	correlator_control_command	11
4.5.5	correlator_take_data_arguments	11
4.5.6	correlator_config_command	12
4.5.7	correlator_software_versions	12
4.5.8	snap_config_version	12
4.5.9	snap_status	12
4.5.10	antenna_status	13
4.5.11	hera_autos	13
4.5.12	node_sensor	14
4.5.13	node_power_status	14
4.5.14	node_power_command	14
4.5.15	node_white_rabbit_status	14
4.5.16	roach_temperature (deprecated)	15
4.6	QA Info Tables	16
4.6.1	metric_list	16
4.6.2	ant_metrics	16
4.6.3	array_metrics	16
4.7	Site Info Tables	16
4.7.1	weather_data	17
4.8	Configuration Management Tables	17
4.8.1	geo_location	17
4.8.2	station_type	17
4.8.3	parts	17
4.8.4	part_info	18
4.8.5	connections	18
4.8.6	apriori_antenna	18
4.8.7	part_rosetta	19
4.8.8	cm_version	19

The formatting of the tables is as follows:

- **Bold font** = primary key
- *Italics* = foreign\_key.
- \* = NotNull entries

## 4.1 Observations

### 4.1.1 hera\_obs

This is the primary observation definition table. It is written to by the correlator.

Column	Type	Description
<b>obsid</b>	long integer	start time in floor(GPS) seconds. GPS start adjusted to be within 1 second of LST to lock observations to LST for the night
starttime	double	start time in gps seconds. The start time to full accuracy of the beginning of integration of first visibility
stoptime	double	stop time in gps seconds. The stop time to full accuracy of the end of integration of last visibility
jd_start	double	start time in JD. Calculated from starttime, provides a quick way to filter on JD times.
lst_start_hr	double	decimal hours from start of sidereal day. Calculated from start-time, provides a quick search for matching LSTs

## 4.2 Common tables

### 4.2.1 server\_status (template)

Common table structure for server status info. **Note: There is no table named server\_status. This is the structure used for several subsystem tables named <subsystem>\_server\_status.**

Column	Type	Description
<b>hostname</b>	string	name of server
<b>mc_time</b>	long	time report received by M&C in floor(gps seconds)
ip_address*	string	IP address of server (how should we handle multiples?)
mc_system_timediff*	float	difference between M&C time and time report sent by server in seconds
num_cores*	integer	number of cores on server
cpu_load_pct*	float	CPU load percent = total load / num_cores, 5 min average
uptime_days*	float	server uptime in days
memory_used_pct*	float	percent of memory used, 5 min average
memory_size_gb*	float	amount of memory on server in GB
disk_space_pct*	float	percent of disk used
disk_size_gb*	float	amount of disk space on server in GB
network_bandwidth_mps*	float	Network bandwidth in MB/s, 5 min average. Can be null

### 4.2.2 subsystem\_errors

Subsystem errors/issues

Column	Type	Description
<b>id</b>	long	auto-incrementing error id
<b>time*</b>	long	error report time in floor(gps seconds)
<b>subsystem*</b>	string	name subsystem with error (e.g. 'librarian', 'rtp')
<b>mc_time*</b>	long	time report received by M&C in floor(gps seconds)
<b>severity*</b>	int	integer indicating severity level, 1 is most severe
<b>log*</b>	text	TBD on format, either a message or a file with the log

#### 4.2.3 daemon\_status

Status of M&C daemons that monitor various subsystems.

Column	Type	Description
<b>name</b>	string	daemon name
<b>hostname</b>	string	hostname where daemon is running (the same daemon can run on multiple hosts)
<b>jd</b>	integer	Julian Date. This allows for some history without keeping all history.
<b>time*</b>	long	most recent status report time in floor(gps seconds)
<b>status*</b>	string	most recent daemon status. One of 'good' or 'errored'

### 4.3 RTP Tables

#### 4.3.1 rtp\_server\_status

RTP version of the server\_status table, see [4.2.1](#).

#### 4.3.2 rtp\_status

High level RTP status

Column	Type	Description
<b>time</b>	long	status time in floor(gps seconds)
<b>status*</b>	string	status string, options TBD (might become an enum)
<b>event_min_elapsed*</b>	float	minutes elapsed since last event
<b>num_processes*</b>	integer	Number of processes running
<b>restart_hours_elapsed*</b>	float	hours elapsed since last restart

#### 4.3.3 rtp\_process\_events

RTP Processing events (per obsid)

Column	Type	Description
<b>time</b>	long	event time in floor(gps seconds)
<b>obsid</b>	long integer	observation identifier, foreign key into hera_obs table
<b>event*</b>	string	one of: queued, started, finished, error

#### 4.3.4 rtp\_process\_record

RTP record of processed obsids (entry added when processing finished)



Column	Type	Description
<b>time</b>	long	record time in floor(gps seconds)
<b>obsid</b>	long integer	observation identifier, foreign key into hera_obs table
pipeline_list*	text	concatenated list of tasks
rtp_git_version*	string	git version of RTP code
rtp_git_hash*	string	git hash of RTP code
hera_qm_git_version*	string	git version of hera_qm code
hera_qm_git_hash*	string	git hash of hera_qm code
hera_cal_git_version*	string	git version of hera_cal code
hera_cal_git_hash*	string	git hash of hera_cal code
pyuvdata_git_version*	string	git version of pyuvdata code
pyuvdata_git_hash*	string	git hash of pyuvdata code

#### 4.3.5 rtp\_task\_resource\_record

RTP record of start and stop times for a task (e.g., omnical) for an obsid, as well as CPU and memory used (if available)

column	type	description
<b>obsid</b>	long integer	observation identifier, foreign key into hera_obs table
<b>task_name</b>	string	name of specific task (e.g., OMNICAL)
start_time*	long	start time of task in floor(gps seconds)
stop_time*	long	stop time of task in floor(gps seconds)
max_mem*	float	maximum memory, in MB, consumed by the task; nullable column
avg_cpu_load*	float	average CPU load, in number of CPUs, for task (e.g., 2.00 means 2 CPUs used); nullable column

## 4.4 Librarian Tables

#### 4.4.1 lib\_server\_status

Librarian version of the server\_status table, see [4.2.1](#).

#### 4.4.2 lib\_status

High level Librarian status

Column	Type	Description
<b>time</b>	long	status time in floor(gps seconds)
num_files*	long	total number of files in librarian
data_volume_gb*	float	total data volume in gigabytes
free_space_gb*	float	available space in gigabytes
upload_min_elapsed*	float	minutes elapsed since last file upload
num_processes*	integer	number of running background tasks
git_version*	string	git version of Librarian code
git_hash*	string	git hash of Librarian code

#### 4.4.3 lib\_raid\_status

RAID controller status

Column	Type	Description
<b>time</b>	long	status time in floor(gps seconds)
<b>hostname</b>	string	name of RAID server
num_disks*	int	number of disks in RAID server
info*	text	TBD – various info from megaraid controller, may be several columns

#### 4.4.4 lib\_raid\_errors

RAID controller errors/issues

Column	Type	Description
<b>id</b>	long	auto-incrementing error id
time*	long	error report time in floor(gps seconds)
hostname*	string	name of RAID server with error
disk*	string	name of disk with error
log*	text	TBD on format, either a message or a file with the log

#### 4.4.5 lib\_remote\_status

Network bandwidth/health to all remote librarians

Column	Type	Description
<b>time</b>	long	status time in floor(gps seconds)
<b>remote_name</b>	string	name of remote librarian
ping_time*	float	ping time in seconds
num_file_uploads*	int	number of files uploaded in last 15 minutes
bandwidth_mps*	float	bandwidth to remote in Mb/s, 15 minute average

#### 4.4.6 lib\_files

File creation log

Column	Type	Description
<b>filename</b>	string	name of file created
<i>obsid</i>	long integer	observation identifier, foreign key into hera_obs table. Can be null.
time*	long	file creation time in floor(gps seconds)
size_gb*	float	file size in gigabytes

### 4.5 Correlator Tables

The correlator tables are not all defined yet. Notes on future plans are in section [5.1](#).

#### 4.5.1 correlator\_config\_file

List of correlator config files, which specify detailed correlator settings. All files in this table are in the Librarian.

Column	Type	Description
<b>config_hash</b>	string	unique hash for the config
filename*	string	name of the config file in the Librarian

#### 4.5.2 correlator\_config\_status

Config status of the correlator, i.e. which config file is being used by the correlator.

Column	Type	Description
<b>time</b>	long	time of the config status in floor(gps seconds)
<i>config_hash</i>	string	hash for the config in use, foreign key into correlator_config_file table.

#### 4.5.3 correlator\_control\_state

State of control knobs in correlator.

Column	Type	Description
<b>time</b>	long	time of the control state in floor(gps seconds)
<b>state_type</b>	string	type of control state, one of: 'taking_data', 'phase_switching', 'noise_diode'.
state*	boolean	indicator of whether the state_type is true or false

#### 4.5.4 correlator\_control\_command

Commands issued to the correlator. If the command is 'take\_data' or 'update\_config', there will be a matching row in the 'correlator\_take\_data\_arguments' table or the 'correlator\_config\_command' table respectively with the values of the parameters in those commands.

Column	Type	Description
<b>time</b>	long	time the command was sent in floor(gps seconds)
<b>command</b>	string	command sent, one of: 'take_data', 'stop_taking_data', 'phase_switching_on', 'phase_switching_off', 'noise_diode_on', 'noise_diode_off', 'update_config'.

#### 4.5.5 correlator\_take\_data\_arguments

Records the arguments passed to the correlator 'take\_data' command.

Column	Type	Description
<b>time</b>	long	time the command was sent in floor(gps seconds), foreign key into correlator_control_command table
<b>command</b>	string	command sent, always 'take_data', foreign key into correlator_control_command table.
starttime*_sec	long	time to start taking data in floor(gps seconds)
starttime.ms*	integer	milliseconds to add to starttime.sec to set correlator start time
duration*	float	duration to take data for in seconds. After this time, the correlator will stop recording
acclen_spectra*	integer	accumulation length in spectra
integration.time*	float	accumulation length in seconds, converted from acclen_spectra (the conversion is non-trivial and depends on the correlator settings)
tag*	string	tag which will end up in data files as a header entry, one of: 'engineering', 'science'.

#### 4.5.6 correlator\_config\_command

Records the config passed to the correlator ‘update\_config’ command.

Column	Type	Description
<i>time</i>	long	time the command was sent in floor(gps seconds), foreign key into correlator_control_command table
<i>command</i>	string	command sent, always ‘update_config’, foreign key into correlator_control_command table.
<i>config_hash*</i>	string	hash for the config to use, foreign key into correlator_config_file table.

#### 4.5.7 correlator\_software\_versions

Software version numbers for correlator software packages and scripts.

Column	Type	Description
<i>time</i>	long	time of the version report in floor(gps seconds)
<i>package</i>	string	name of the correlator software module or <package>:<script> for daemonized processes (e.g. ‘hera_corr_cm’, ‘udpSender:hera_node_keep_alive.py’).
<i>version*</i>	string	version string for this package or script

#### 4.5.8 snap\_config\_version

SNAP initialization configuration and software versions.

Column	Type	Description
<i>init_time</i>	long	time when the SNAPs were last initialized with the ‘hera_snap_feng_init.py’ script in floor(gps seconds)
<i>version*</i>	string	version string for the hera_corr_f package
<i>init_args*</i>	string	arguments passed to the initialization script at runtime
<i>config_hash*</i>	string	unique hash for the config, foreign key into correlator_config_file table

#### 4.5.9 snap\_status

SNAP status information (reported via the correlator redis DB).

Column	Type	Description
<i>time</i>	long	status time in floor(gps seconds)
<i>hostname</i>	string	SNAP hostname
<i>serial_number</i>	string	SNAP serial number
<i>node</i>	int	node number (derived from config. management tables using SNAP serial number)
<i>snap_loc_num</i>	int	snap location number within the node (derived from config. management tables using SNAP serial number)
<i>psu_alert</i>	bool	true if SNAP PSU (aka PMB) controllers have issued an alert, false otherwise.
<i>pps_count</i>	long	number of PPS pulses received since last programming cycle
<i>fpga_temp</i>	float	reported temperature of FPGA in degrees C
<i>uptime_cycles</i>	long	multiples of $500 \cdot 10^6$ ADC clocks since last programming cycle
<i>last_programmed_time</i>	long	last time this FPGA was programmed in floor(gps seconds)

#### 4.5.10 antenna\_status

Antenna status information from the SNAP (reported via the correlator redis DB).

Column	Type	Description
<b>time</b>	long	status time in floor(gps seconds)
<b>antenna_number</b>	int	antenna number
<b>antenna_feed_pol</b>	string	antenna feed polarization, either 'e' or 'n'.
snap_hostname	string	SNAP hostname
snap_channel_number	int	SNAP ADC channel number (0-7) to which this antenna is connected.
adc_mean	float	mean ADC value, in ADC units (raw ADC integer values between -128 and +127). Typically $\sim -0.5$ .
adc_rms	float	RMS ADC value, in ADC units (raw ADC integer values between -128 and +127). Should be $\sim 10-20$ .
adc_power	float	mean ADC power, in ADC units squared (raw ADC integer values between -128 and +127, squared). Since mean should be close to zero, this should just be $\text{adc\_rms}^2$ .
pam_atten	int	PAM attenuation setting for this antenna, in dB
pam_power	float	PAM power sensor reading for this antenna pol, in dBm.
pam_voltage	float	PAM voltage sensor reading for this antenna pol, in Volts.
pam_current	float	PAM current sensor reading for this antenna pol, in Amps.
pam_id	string	Serial number of the PAM for this antenna pol.
fem_voltage	float	FEM voltage sensor reading for this antenna pol, in Volts.
fem_current	float	FEM current sensor reading for this antenna pol, in Amps.
fem_id	string	Serial number of the FEM for this antenna pol.
fem_switch	string	Switch state the FEM, one of 'antenna', 'load' or 'noise'.
fem_lna_power	boolean	Power state of the FEM, 'True' means it is powered.
fem_imu_theta	float	IMU-reported theta, in degrees.
fem_imu_phi	float	IMU-reported phi, in degrees.
fem_temp	float	EM temperature sensor reading for this FEM in degrees Celcius.
fft_overflow	boolean	Indicator of an FFT overflow for this antenna pol, set to 'True' if there was an overflow.
eq_coeffs	string	digital EQ coefficients for this antenna, used for keeping the bit occupancy in the correct range. list of floats (one per freq. channel) represented as a string. Note this these are not divided out anywhere in the DSP chain (!).
histogram_bin_centers	string	ADC histogram bin centers, list of ints stored as a string.
histogram_bin_counts	string	ADC histogram bin counts, list of ints stored as a string.

#### 4.5.11 hera\_autos

Antenna autocorrelation statistics (currently just the median over the frequency axis).

Column	Type	Description
<b>time</b>	long	status time in floor(gps seconds)
<b>antenna_number</b>	int	antenna number
<b>antenna_feed_pol</b>	string	antenna feed polarization, either 'e' or 'n'.
measurement_type*	string	Currently can only be 'median'.
value*	float	Measured value.

#### 4.5.12 node\_sensor

Node temperature and humidity sensor readings

Column	Type	Description
<b>time</b>	long	measurement time in floor(gps seconds)
<b>node</b>	int	integer identifying the node
top_sensor_temp	float	temperature of top sensor reported by node in degrees C
middle_sensor_temp	float	temperature of middle sensor reported by node in degrees C
bottom_sensor_temp	float	temperature of bottom sensor reported by node in degrees C
humidity_sensor_temp	float	temperature of humidity sensor reported by node in degrees C
humidity	float	percent humidity measurement reported by node

#### 4.5.13 node\_power\_status

Power status for SNAPs, FEMs and PAMs (monitored by nodes)

Column	Type	Description
<b>time</b>	long	measurement time in floor(gps seconds)
<b>node</b>	int	integer identifying the node
snap_relay_powered*	bool	power status of the snap relay, True = powered
snap0_powered*	bool	power status of the SNAP 0 board, True = powered
snap1_powered*	bool	power status of the SNAP 1 board, True = powered
snap2_powered*	bool	power status of the SNAP 2 board, True = powered
snap3_powered*	bool	power status of the SNAP 3 board, True = powered
fem_powered*	bool	power status of the FEM, True = powered
pam_powered*	bool	power status of the PAM, True = powered

#### 4.5.14 node\_power\_command

Commands issued to change the power status for SNAPs, FEMs and PAMs (via the nodes).

Column	Type	Description
<b>time</b>	long	time the command was sent in floor(gps seconds)
<b>node</b>	int	integer identifying the node commanded
<b>part</b>	string	part commanded, one of 'snap_relay', 'snap0', 'snap1', 'snap2', 'snap3', 'pam', 'fem'.
command*	string	command sent, 'on' or 'off'.

#### 4.5.15 node\_white\_rabbit\_status

Node white rabbit status. There are a duplicate set of columns for the two ports which start with 'port0' and 'port1'. We list them jointly below as 'port[0,1]'.

Column	Type	Description
<b>node_time</b>	long	time of the status reported by the node in floor(gps seconds)
<b>node</b>	int	integer identifying the node.
board_info_str	string	A raw string representing the WR-LEN's response to the 'ver' command. Relevant parts of this string are individually unpacked in other entries.
aliases	string	Hostname aliases of this node's WR-LEN (comma separated if more than one).
ip	string	IP address of this node's WR-LEN.
mode	string	WR-LEN operating mode (eg. "WRC_SLAVE.WR0").
serial	string	Canonical HERA hostname (related to serial number) of this node's WR-LEN.
temperature	float	WR-LEN temperature in degrees Celcius.
build_date	long	Build date of WR-LEN software in floor(gps seconds).
gw_date	long	WR-LEN gateway build date in floor(gps seconds).
gw_version	string	WR-LEN gateway version number.
gw_id	string	WR-LEN gateway ID number.
build_hash	string	WR-LEN build git hash.
manufacture_tag	string	Custom manufacturer tag.
manufacture_device	string	Manufacturer device name designation.
manufacture_date	long	Manufacturer invoice(?) date in floor(gps seconds).
manufacture_serial	string	Manufacturer serial number.
manufacture_vendor	string	Vendor name.
port[0,1]_ad	int	Meaning unknown.
port[0,1]_link_asymmetry_ps	int	Port [0,1] total link asymmetry in picosec.
port[0,1]_manual_phase_ps	int	Not well understood, believed to be Port [0,1] manual phase adjustment in picosec.
port[0,1]_clock_offset_ps	int	Port [0,1] clock offset in picosec.
port[0,1]_cable_rt_delay_ps	int	Port [0,1] cable round trip delay in picosec.
port[0,1]_master_slave_delay_ps	int	Port [0,1] master-slave delay in picosec.
port[0,1]_master_rx_phy_delay_ps	int	Port [0,1] master RX PHY delay in picosec.
port[0,1]_slave_rx_phy_delay_ps	int	Port [0,1] slave RX PHY delay in picosec.
port[0,1]_master_tx_phy_delay_ps	int	Port [0,1] master TX PHY delay in picosec.
port[0,1]_slave_tx_phy_delay_ps	int	Port [0,1] slave TX PHY delay in picosec.
port[0,1]_hd	int	Meaning unknown.
port[0,1]_link	boolean	Port [0,1] link up state.
port[0,1]_lock	boolean	Port [0,1] timing lock state.
port[0,1]_md	int	Meaning unknown.
port[0,1]_rt_time_ps	int	Port [0,1] round-trip time in picosec.
port[0,1]_nsec	int	Meaning unknown.
port[0,1]_packets_received	int	Port [0,1] number of packets received.
port[0,1]_phase_setpoint_ps	int	Port [0,1] phase setpoint in picosec.
port[0,1]_servo_state	string	Port [0,1] servo state.
port[0,1]_sv	int	Meaning unknown.
port[0,1]_sync_source	string	Port [0,1] source of synchronization (either 'wr0' or 'wr1').
port[0,1]_packets_sent	int	Port [0,1] number of packets transmitted.
port[0,1]_update_counter	int	Port [0,1] update counter.
port[0,1]_time	long	Port [0,1] current time in GPS seconds. in floor(gps seconds).

#### 4.5.16 roach\_temperature (deprecated)

Roach (correlator fpga board) temperatures (deprecated 8/2018)

Column	Type	Description
<b>time</b>	long	measurement time in floor(gps seconds)
<b>roach</b>	string	name of roach (correlator fpga board)
ambient_temp	float	ambient temperature reported by the roach in degrees C
inlet_temp	float	inlet temperature reported by the roach in degrees C
oulet_temp	float	oulet temperature reported by the roach in degrees C
fpga_temp	float	fpga temperature reported by the roach in degrees C
ppc_temp	float	ppc temperature reported by the roach in degrees C

## 4.6 QA Info Tables

The QA tables are not all defined yet. Notes on future plans are in section 5.2.

### 4.6.1 metric\_list

List and descriptions of metrics used in antenna or array metrics.

Column	Type	Description
<b>metric</b>	string	name of metric
desc*	string	description of metric

### 4.6.2 ant\_metrics

Antenna metrics, by polarization and obsid. These are metrics, generally generated by hera\_qm, which are keyed to individual antennas. For example, hera\_qm.ant\_metrics will flag individual antennas as bad.

Column	Type	Description
<b>obsid</b>	long integer	observation identifier, foreign key into hera_obs table.
<b>ant</b>	integer	antenna number ( $\geq 0$ )
<b>pol</b>	string	polarization, 'x' or 'y'
<b>metric</b>	string	name of metric, foreign key into metric_list table.
mc_time*	long integer	time report received by M&C in floor(gps seconds)
val*	double	value of metric

### 4.6.3 array\_metrics

Array metrics, by obsid. These are metrics, generally generated by hera\_qm, which are keyed to the overall array. For example, hera\_qm.firstcal\_metrics generates an overall decision whether the firstcal solutions were "good".

Column	Type	Description
<b>obsid</b>	long integer	observation identifier, foreign key into hera_obs table.
<b>metric</b>	string	name of metric, foreign key into metric_list table.
mc_time	long integer	time report received by M&C in floor(gps seconds)
val*	double	value of metric

## 4.7 Site Info Tables

The Site Info tables are not all defined yet.. Notes on future plans are in section 5.3.



#### 4.7.1 weather\_data

Weather data from KAT sensors

Column	Type	Description
<b>time</b>	long	status time in floor(gps seconds)
<b>variable</b>	string	name of weather variable (e.g. wind_speed, wind_direction, temperature)
value*	float	value of the variable at this time

### 4.8 Configuration Management Tables

As described in section 3, there are five tables in the configuration management section of the database: (1) geo\_location, (2) station\_meta, (3) parts\_paper, (4) part\_info, (5) connections. The following tables summarize them with the following key:

#### 4.8.1 geo\_location

Column	Type	Description
<b>station_name*</b>	character varying(64)	Name of position - never changes
<i>station_type_name*</i>	character varying(64)	Type of station
datum	character varying(64)	UTM datum
tile	character varying(64)	UTM tile
northing	double precision	UTM coordinate
easting	double precision	UTM coordinate
elevation	double precision	Elevation
created_gpstime*	BigInt	GPS second of creation.

#### 4.8.2 station\_type

Column	Type	Description
<i>station_type_name*</i>	character varying(64)	Station type name
prefix*	character varying(64)	1-2 letter prefix for part station_name
description	character varying(64)	Short description
plot_marker	character varying(64)	Type of matplotlib marker

#### 4.8.3 parts

Column	Type	Description
<i>hpn*</i>	character varying(64)	HERA part number
<i>hpn_rev*</i>	character varying(32)	HPN revision letter (A-Z)
<i>hptype*</i>	character varying(64)	HPN part type category
<i>manufacturer_number</i>	character varying(64)	Unique serial number for each part
<i>start_gpstime*</i>	BigInt	GPS second when part/rev is activated.
<i>stop_gpstime</i>	BigInt	GPS second when part/rev is de-activated

#### 4.8.4 part\_info

Column	Type	Description
<i>hpn*</i>	character varying(64)	HERA part number
<i>hpn_rev*</i>	character varying(32)	HPN revision letter (A-Z)
<i>posting_gpstime*</i>	BigInt	GPS second information was posted
<i>comment*</i>	character vary- ing(1024)	Comment
<i>library_file</i>	character varying(256)	Librarian filename (how to get it there?)

#### 4.8.5 connections

Column	Type	Description
<i>upstream_part*</i>	character varying(64)	Hera part number of upstream connection
<i>up_part_rev*</i>	character varying(32)	Hera part revision of upstream connection
<i>upstream_output_port*</i>	character varying(64)	Output port on upstream part
<i>downstream_part*</i>	character varying(64)	Hera part number of downstream connection
<i>down_part_rev*</i>	character varying(32)	Hera part revision of downstream connection
<i>downstream_input_port*</i>	character varying(64)	Input port on downstream part
<i>start_gpstime*</i>	BigInt	GPS second when connection started
<i>stop_gpstime</i>	BigInt	GPS second when connection ended

#### 4.8.6 apriori\_antenna

Column	Type	Description
<i>antenna*</i>	text	antenna for which status holds
<i>start_gpstime*</i>	BigInt	GPS second when status change becomes valid
<i>stop_gpstime</i>	BigInt	GPS second at end of status. None for last
<i>status*</i>	text	status enum message, one of 'passed_checks', 'needs_checking', 'known_bad', 'not_connected'

#### 4.8.7 part\_rosetta

Column	Type	Description
hpn	text	HERA part number ( <i>e.g.</i> SNPC000072)
syspn*	text	System-dependent part number ( <i>e.g.</i> heraNode0Snap0)
start_gpstime*	BigInt	GPS second when started
stop_gpstime	BigInt	GPS second when stopped. None for last

#### 4.8.8 cm\_version

Column	Type	Description
update_time*	BigInt	GPS second when version set
git_hash*	character varying(64)	git hash number for version

## 5 Future Plans

### 5.1 Correlator Table plans

The correlator tables are not all defined yet, the following are notes about suggestions and plans for correlator tables. Most of the correlator data will be recorded in a Redis database (a rolling log, ephemeral), that info needs to be grabbed and put in M&Ctables.

**corr\_server\_status:** Correlator version of the server\_status table, see 4.2.1, not yet implemented.

1. correlator on/off? **this is a control**
2. Bit statistics (overflows, ADC clipping, bit statistics after bit selects)
3. correlator network stats (dropped packets)
4. Firmware git hash
5. Engine status
6. Xengine status (might be covered in corr\_server\_status)
7. Walsh on/off **this is a control** (correlator propagates to node)
8. Noise diode **this is a control** (correlator propagates to node)
9. correlator config (walsh patterns; scaling functions for FFT, bit selection)
10. Test mode outputs (results not control) – very notional
  - a. Engine sync test
  - b. Xengine test
  - c. Do at beginning and end of night.
  - d. Analog tests
    1. Noise diode status
    2. Temperature (i2c device)
    3. Walsh switching (on/off control. Make sure bit pattern is known and put into data set.)
11. SNAP information: all info reported through the correlator
  - a. Feed status
  - b. PAM status

12. Node information (from Arduino) (Dave, Jack, Zara, Matt Dexter (mdexter@berkeley.edu), Nima) All node info will be reported through the correlator.
  - a. Clock status info – syncing
  - b. Temperatures (outside + inside, feed?)
  - c. Node M&Csoftware git hash

### 5.1.1 Correlator interfaces complete:

These are done:

1. M&Cinformation the correlator needs to get and write into files
  - a. Antenna positions
2. New info added to correlator files (recorded in hera\_obs table)
  - a. obsid
  - b. duration
3. Node information (from Arduino) (Dave, Jack, Zara, Matt Dexter (mdexter@berkeley.edu), Nima) All node info will be reported through the correlator.
  - a. SNAP power states
  - b. Temperatures in nodes
  - c. Power PAM, FEM status (binary)

## 5.2 QA Future Plans

These are some suggestions for the future, things we might like to see.

1. RTP/online systems
  - a. RFI statistics/info (this might be in ant\_metrics and array\_metrics now)
  - b. Calibration statistics (this might be in ant\_metrics and array\_metrics now)
  - c. LST repeatability
  - d. TBD other things that come up
2. Offline codes (Major work on how to implement this!! Not on the critical path):
  - a. TBD from offline analysis codes

## 5.3 Site Info Future Plans

The following are suggestions for the future, things we might like to see.

1. site power
2. network status

## 5.4 Other Future Ideas

1. Basic ionospheric monitoring
2. RFI monitoring