MiniRov Data Management

25Sep2019 Rev 1.0

- Dale is responsible for logging data at sea and transfer to shore.
- Files may be of 'type' CTD, NAV, DVL, ROV
- Dale's landing root directory path is:
- //atlas.shore.mbari.org/ProjectLibrary/901123.ROV1k/Data/Logs
- Files are organized there by Dale to subdirs there by 'type'
- Note that 'ROV_xxx' files land in a subdir called Vehicle
- All files are CSV with header describing column names

Dale moves files to project dir on atlas as described above.

- CTD is considered (by me at least) the critical one. Assuming Dale wont dive without a depth sensor.
- Dale only logs when rov is 'in water' as determined by CTD. (p>? c>?), NAV has been spotty. As of 09/2019 it is expected to be there always.
- ROV is where mbsystem looks for heading and heading is the only field used.
- DVL is where mbsystem looks for altimeter. DVL may not be there on midwater dives. Mbsystem does not currently use anything in DVL besides the altimeter.

```
###    minirov processing   ###
#merge from minirovdivlog database to Expd.Dive table
00 * * * * /u/coredata/minirov/scripts/runMinirovDivelogUploader  >> /u/coredata/minirov/rur

#sweep files from Dales directories to atlas archive and register with database load table
00 16 * * * /u/coredata/minirov/scripts/runMinirovFileArchiver  >> /u/coredata/minirov/runlo

#pick up any newly registered ctd files an upload to 'raw' tables in Expd
02 16 * * * /u/coredata/minirov/scripts/runMinirovCtdUploader  >> /u/coredata/minirov/runlog

#pick up any newly registered NAV,CTD,DVL,ROV files and prep for mb nav editing.
04 16 * * * /u/coredata/minirov/scripts/runMinirovNavUploader  >> /u/coredata/minirov/runlog
```

Preliminaries

- Initial shoreside scripts (/sh and python) are in: /u/coredata/minirov/scripts/
- Above scripts are run by crontab for user coredata on machine coredata
- Above scripts can be run by hand – although the 'runXxxxx' shell script is used to activate

the python virtual environment that is required.

- Once minirov data are 'staged' by the above as either as nav files or as database loaded raw data, they are processed similar to our core ship/rov data by additional scripts on windows task scheduler on machine Draco as user DB_EXPD_TASKEXEC
- Below will describe processing in roughly order of occurrence (although steps are asynchronous)

Merge the minirovdivelog Dive info database to Expd.Dive table

Cronjob on coredata executes script runMinirovDivelogUploader which runs minirovdivelog2expd.py

- Looks for modified entries in MINIROV_DiveLog .Dive
- Merges data from perseus datqabase/table MINIROV_DiveLog .Dive to perseus.Expd.Dive
- Attempts to create a link (a ForeignKey) to ship expedition if it can find one spanning dive start/end time. *** Note:success is highly dependant on correct dive start/end times... if dive doesn't show up with expedition id and you think it should... examine/edit the expedition id with:

```
select * from divesummary
where rovname = 'mini'
and divenumber = ####

select * from dive
where rovname = 'mini'
and divenumber = ####
```

- Action of inserting an Expd.Dive fires a trigger (tgr_Insert_Dive) that creates an initial entry in Expd.DiveSummary with null fields...
- Every night, 'task manager' on Draco (windows machine) runs a bat script that looks for DivesSummaries that are 'dirty' (see Draco TaskScheduler and D:/Expedition/SqlExecScripts/EXPD_UpdataDirtyDiveSummaries.sql

Useful query on database/table: MINIROV_Divelog.Dive

```
SELECT  id, divenumber, launch, recover, isnull(shipname,''), isnull(scientistname,''),snull
FROM dive
WHERE isnull(lastsync,'01-Jan-2019') <  lastupdate
```

Sweep files from Dales directories to atlas archive and register

See: script runMinirovFileArchiver on coredata which executes minirovfilearchiver.py for each flie 'type'

- Sweep Dales directory and cp new files to a 'safe' location on atlas.
- Files are also scanned to see if we can determine start/end times, number of records, and a crude test is done to see if columns contain 'realistic' data.
- Registers the files in a database table on Expd (Table: MinirovLogfiles). Along with start/endtimes etc. from above.
- Files end up on atlas in ShipData/minirov and are write-protected (need IS admin to modify or delete)
- On successful archive the isArchive bit is set in the MinirovLogfiles table and the isBlocked flag is cleared. isBlocked and isProcessed flags are currently read in the query but not used – they are meant to be back compatable with how other core ship/rov data loads are handled
- Also register the file in the type-specific load tracking table. E.g. MinirovRawCtdLoad, MinirovRawNavLoad, MinirovRawRovLoad or MinirovRawDvlLoad (this is where device-specific processing is controlled and tracked)

Useful sample query:

```
SELECT * FROM MinirovLogfiles
WHERE yr = 2019
ORDER BY yrday desc
```

Pick up any newly registered CTD files an upload to 'raw' tables in Expd

Recall that the sweep step above registers the CTD files in table MinirovRawCtdLoad Cron on coredata runs bash script: /u/coredata/minirov/scripts/runMinirovCtdUploader which set python environment and runs: ctdfileloader.py

- Looks at MinirovRawCtdLoad for files with isLoaded = null or 0
- Parses each CTD_ xxx file and does basic QC checks etc.
- loads to MinirovRawCtdData_YYYY base tables
- On success – sets the isLoaded field of the MinirovRawCtdLoad table.
- NOTE: isLoaded field is tied to a trigger, if you manually set to 0 or null, it will cascade delete all data rows using the loadid_fk

Process 'minirov raw' CTD data to final process 1hz and 15sec binned tables

- Every night, 'task manager' on Draco (windows machine) runs a bat script that looks for

MinirovRawCtdLoad table files the are loaded but have isProcessed =0 (see Draco TaskScheduler and D:/Expedition/EXPD_Data_Loads/RovCtd/processRawMinirovCtd.bat

- On success, the core ctd raw 1second and 15sec binned tables are loaded and the isProcessed field of the load table is set.
- Note: Clearing the isProcessed field of the load table will cascade delete dependent row in the processed ctd data tables.
- NOTE: We compute salinity using standard mbari algorithms
- NOTE: O2 QC flags are currently set to suspect.

Pick up any newly registered NAV files

Recall that the sweep step above registers the NAV, CTD,ROV and DVL files in tables MinirovRawNavLoad, MiniroCtdRovLoad, MinirovRawRovLoad, and MinirovRawDvlLoad tables

Cron on coredata runs bash script: /u/coredata/minirov/scripts/runMinirovNavUploader which set python environment and runs: minirov2mbsystem.py

- Looks for entries in MinirovRawNavLoad table with isProcessed field = 0

that also have corresponding `CTD_` and `ROV_` files (starttimes within 15 seconds)

Useful SQL:

```
SELECT * FROM MinirovRawNavLoad
```

- For those files, also check for optional DVL_ files.
- builds a command line to push files thru 'c' program mbminirovnav
- command is ecec'd and output mb165 format file is on atlas in RovNavEdit/year/minirov
- isProcessed field is set on MinirovRawNavLoad

Useful SQL:

```
SELECT  nav.id as id, nav.yr as year,
 nav.path + '/' + nav.filename as nav,
  ctd.path + '/' + ctd.filename as ctd,
   rov.path + '/' + rov.filename as rov
    from  minirovrawctdLoad ctd, minirovrawnavload nav, minirovrawrovload rov
    where isnull(nav.isprocessed,0) = 0
    and isnull(nav.isblocked,0) = 0
    and ctd.yr = 2019
    and ctd.yrday > 200
    and abs(datediff(second,ctd.startdtg, nav.startdtg) ) < 15
    and abs(datediff(second,ctd.startdtg, rov.startdtg) ) < 15
```

Standard nightly processing via task manager on Draco, task EXPD_NightlyMinirovNavLoadToPersus

- runs load_minirov_nav.bat
- which runs UpdateRovNavdata.pl to detect if the edited nav has been produced or is recently modified. Using load status table CleanRovNavLoad
- also runs LoadRovNavdata.pl for both raw and clean nav files found in RovNavEdit/year/minirov . Uses RawRovNavLoad and CleanRovNavLoad staustus table isLoaded field
- Note: isLoaded fields being set to 0 triggers a cascade delete on underlying position data (by loaded_fk)

Note: Videolab personel clean up data raw nav .mb165 format files thru mbsystem and produce the edited file with names like NAV_090517000000edited.txt ie. dropping the .mb165 extention and replacing with “edited.txt” so legacy clean nav load described above code can detect it.

SEE ALSO

Create an Expedition and links to MiniROV dives for non-MBARI ship Expeditions

https://docs.google.com/document/d/1dSRydDU7rBuuZ36AD2fr6xtc4P9axcQMPOvMJxF5m8k/edit#heading=h.gyctmfrfqzu8