MAVEN

MAVEN IDL

Overview

The system of scripts for running the MAVEN IDL Toolkit to generate MATLAB-compatible HDF5 and csv files resides on ITF2 under the ~/tplot_auto directory, which contains the following subdirectories:

- prod - Production folder, where generated HDF5 and csv files are committed into SVN
- working - Working folder, where scripts operate. It contains cache files written by IDL/tplot and other intermediate files
- bin - Python scripts that generate and execute IDL .pro scripts
- idl - IDL modules and configuration files

Most instrument file generators have the same basic process:

- run_INST.py - Kicked off through launchd at specific weekday and time
  - Generates INST.pro file to run covering specified TID range
  - Run IDL with generated script
  - Ready to commit to SVN

Automated execution of scripts is handled through launchd, using an app called LaunchControl. The app generates XML-based configuration files saved in ~/Library/LaunchAgents

IDL is licensed through a license server maintained by SED.

For some background details on the IDL code used to generate these files, see NGIMS_IDL_Requests.docx

Software Updates

Development on the MAVEN IDL Toolkit has mostly settled down, so updates are not as frequent. However, it would be prudent to occasionally check for updates. Unfortunately, this is a manual process. Although the toolkit has an automatic update facility, it does not seem to be triggered when run noninteractively by the automated scripts.

To perform a manual update of the MAVEN Toolkit:

- Start idl (command line or GUI)
- Run the command: mvn_kp_check_version

On occasion I have noticed an update on the SDC Software page, which is not installed by this command. In that case, the recommended manual installation is:

- Uncompress zip file by double-clicking and move new directory into ~/idl (e.g. Toolkit_V2019-09-25_Team)
- cd ~/tplot_auto/idl
- rm Toolkit_Team

- Soft link newly created directory as Toolkit_Team (e.g. ln -s ~/idl/Toolkit_V2019-09-25_Team Toolkit_Team)

If you'd rather not follow this recommendation, the new IDL code must included in the IDL_PATH in the idlstart.pro file of the ~/tplot_auto/idl directory.

The publicly distributed MAVEN Toolkit only covers access to kp (or key parameters) data. There is an add-on used exclusively by the MAVEN Science Team, which is used to generate L2 files, etc. This is updated weekly by downloading the latest maven_sw.zip from the Berkeley site.

- update_maven_sw.py
  - Runs Mondays at 8am

This script unzips the latest IDL code into ~/tplot_auto/idl/maven_sw and keeps a copy of all the past zip files in ~/tplot_auto/idl/history

TIDs

A TID is a unique sequential identifier assigned to NGIMS data. Each orbit that MAVEN makes around Mars is divided into two TIDs. The even-numbered TIDs kick off at the beginning of inbound segment, go through periapsis and end at the start of the outbound segment. The odd-numbered TIDs cover the rest of the orbit. Primary science occurs in the first type of TID where the spacecraft dips to a lower altitude, while the odd TIDs are mostly composed of instrument idle time. Sometimes functional tests and calibrations occur in those TIDs.

The division into TIDs is exclusively a feature of NGIMS data, while the other instruments on MAVEN are generally on continuously and divide their data into day-length CDF files. Since NGIMS is idle for a large portion of the orbit, it's more convenient to compare data with other instruments by selecting the portion of the orbit where we are all taking science measurements. The mapping between TID number and SCLK time is recorded in a file generated by the NGIMS PDL tool and committed to SVN in the mavendata repository:

- spacecraft_data/L0/maven_pdl_TID_table.txt

A special copy of the directory containing this file is checked out under:

- ~/tplot_auto/prod/L0

A script is executed to convert SCLK times into unix time and save them in ~/tplot_auto/working/tidtimes.csv for use by instrument-specific scripts:

- update_tidtimes.py
  - Run daily at noon

STATIC

Most of the instrument data generated by this system is simply for the convenience of the NGIMS science team. However, the HDF5 files containing STATIC data are a necessary step in the NGIMS data processing pipeline, as they contain spacecraft potential data required to calibrate NGIMS science data.

This dependence on STATIC adds some timing constraints. Spacecraft potential does not appear in STATIC data until there are as-run SPICE kernels with spacecraft location. Updated kernels are delivered about a week after new instrument data arrives. STATIC data is then reprocessed after data is 2 weeks, 1 month and 2 months old,

possibly refining the data.

Given that we don't want to continually reprocess NGIMS data, we have chosen to focus on two events: when spacecraft potential first becomes available and when we need to make a PDS delivery.

The python script that generates the S/C potential data at the earliest opportunity is the incremental script:

- run_static_incr.py
  - Runs Sundays at 12:30pm and Wednesdays at 8:30am
  - Looks for most recent sta#####.h5 file where ##### is the TID. To account for partials, the script goes back 25 TIDs and starts processing from there. This is a somewhat arbitrary number that seems to work.
  - Updated and newly generated files are automatically committed to SVN.

There is no error notification when the script has an issue. The maintainer of the script must be on the SVN notification list and look for the commit message expected on Sunday afternoon and Wednesday morning. If no such message is received, the most likely reason is that the SVN tunnel has gone down. From the ITF2 machine, log into mavenioc and restore the tunnel. The script has performed all the necessary 'svn add' operations, and you should only need to run 'svn commit' in the tplothdf/static directory. Also, there is a crude check for an automated process run amuck. The script will not automatically commit STATIC HDF5 files if there are more than 100 new files generated. In this case, an operator should review the output files and manually 'svn commit' if everything is in order.

When a PDS delivery is imminent, a manual run of the run_static.py script is necessary.

1. Determine the date range of the PDS delivery
2. Before you run the script, it's important that the end of this range is at least two months ago. Otherwise, the last scheduled STATIC reprocessing has not finished.
3. Determine the TID range analogous to this date range
4. Edit run_static.py to modify the value assigned to starttid
5. In the LaunchControl app, unload the run_static_incr.py script to avoid conflict with the weekly run by selecting it and from the menu Job/Unload
6. Use the Job menu to load the local.tplot.run_static job
7. Use the Job menu to start the local.tplot.run_static job, which executes the run_static.py script
8. Monitor the progress of the run (e.g. `tail -f sta.log` from tplothdf/static directory). A month of data takes about a day to process.
9. When complete, unload the run_static.py script and load the run_static_incr.py script to reenable the weekly run
10. SVN add and commit operations must be performed manually

In addition to the spacecraft potential, STATIC HDF5 files also include O+, O2+ and H+ densities.

MAG

MAG data generation is one of the more complicated processes. The final product is one csv file per TID, containing measured magnetic field and modeled crustal field at 1-sec resolution. There are two scripts involved, run_mag.py and split_mag.py. The first creates two intermediate files for each day, one with MAG data and the other with crustal field model data. The split_mag script merges the data and then splits them into TID-based files.

- run_mag.py
  - Runs daily at 8am
  - Looks for most recent mag########model.csv, identified by year, month and day. To account for partials, the script goes back 2 TIDs and starts processing from there. The processing ends

on the last day where crustal field model data exists in the archive. Measured magnetic field data is output based on the days determined by the model availability, which is generally two weeks old.
- split_mag.py
  - Runs Mondays at 4:30pm

Generated files are not automatically added or committed to SVN. Manual svn add and commit is carried out upon request by the NGIMS science team.

LPW

LPW HDF5 data generation follows the standard pattern described in the Overview section.

- run_lpw.py
  - Runs Fridays at 10am
  - This script calls 'svn add' but does not automatically commit the new files

Manual commit is carried out upon request by the NGIMS science team. The LPW L2 data lags by at least a month. This process generates empty HDF5 files for the missing data, which are easily identifiable as they have a consistent size of 800 bytes. To avoid committing these empty files, the script only adds files over 1000 bytes to svn.

SWIA

SWIA data generation is similar to MAG. The final product is one csv file per TID, containing the following tplot data structures flattened into their component columns:

- mvn_swim_velocity_mso = Vx,Vy,Vz
- -VxB = Ex,Ey,Ez
- mvn_swim_density = n
- mvn_swim_pressure = p
- mvn_swim_temperature_mso = Tx,Ty,Tz

There are two scripts involved, run_swia.py and split_swia.py. The first creates a SWIA file for each day. The split_swia script merges the data and then splits them into TID-based files.

- run_swia.py
  - Runs daily at 3pm - DISABLED as data did not seem to be useful to team
  - Looks for most recent swi########.csv, identified by year, month and day. To account for partials, the script goes back 2 TIDs and starts processing from there. The processing ends on the last day where SWIA data exists in the archive.
- split_swia.py
  - Runs on Wednesdays at 4:30pm - DISABLED as data did not seem to be useful to team

Generated files are not automatically added or committed to SVN. Manual svn add and commit is carried out upon request

SWEA

SWEA HDF5 data generation follows the standard pattern described in the Overview section.

- run_swea.py
  - Runs Tuesdays at 10:30am - DISABLED as data did not seem to be useful to team
  - Script does not automatically svn commit or add

Manual commit is carried out upon request by the NGIMS science team.

Maintenance Notes

The act of running the MAVEN IDL Toolkit downloads and caches needed files into ~/tplot_auto/working/idl from the Berkeley site. It is populated with day-length files for those that have been loaded and converted to HDF5 or csv format. SPICE kernels are also saved in here. The bulk of the disk space used by this system in the form of the prod (SVN) and working directories resides on the external disk drive. The expectation is that there is plenty of space on ITF2_External and no special cleanup is necessary.

Some scripts have a heuristic for determining where to start processing, but many require periodic updates to the starting TID to avoid unnecessary reprocessing. SWEA, LPW and the tidtimes.csv generator rely on a manual edit of the starttid variable.

MAVEN NGIMS PEF Review

Uplink Bundle Review

First Steps

Determine valid orbit range in Science Week announcement

Download and uncompress the attached wk###.tar.gz

Change directory to sci_wk###

`update_orb.py`

`procpef.py > pef.txt`

Open pef.txt in text editor

Skip down to one before first valid orbit

Standard Science Runs

Look for the following after orbit segment IB_SIDE

```
NGM_LOAD_TBL "d:/ngm/sciperi_1884.bas" 201 "sciperi_1884.bas
NGM_LOAD_SCRIPT "d:/ngm/sci_peri_osion_1886.cfg" UPLOAD_APPE
```

followed by NGM_SCRIPT commands: run, set_peri_time ######, go

Most activites are started in the IB_SIDE segment

Just skim and look for anything off. This is the bulk of what should be in the pef

Under "wizard" commanding, there are now "COMM" and "RELAY" segments that can cause the usual IB_SIDE, PERIAPSE, OB_SIDE and APO segments to be replaced. In these cases, you must verify the timing with the second column of the PEF (aka offset from altitude minimum):

- Verify that script is kicked off around -4350 sec. Relays can delay this time. A standard science script should not be started within 10 minutes of the OB_SIDE, corresponding to a value greater than 556s in the second column

Special Activities

Verify all special activities in the Science Week announcement have been implemented

Wind Scans

For neutral wind scans

```
NGM_LOAD_TBL "d:/ngm/sciwind_1907.bas" 201 "sciwind_1907.bas
```

followed by NGM_SCRIPT command: run

and then a SPECIAL segment with a VM_SPAWN_TAG that refers to a file starting 'ngm_wind_scans_neu'

For ion wind scans

```
NGM_LOAD_TBL "d:/ngm/windion_xxxx.bas" 201 "windion_xxxx.bas
```

followed by NGM_SCRIPT command: run

and then a SPECIAL segment with a VM_SPAWN_TAG that refers to a file starting 'ngm_wind_scans_ion'

High-Speed Argon Scans

This activity bookends the usual science scans with scans focusing exclusively on m/z 40, also extending the range of continuous scanning to an earlier point in the orbit. Although there is no difference between them in the pef commanding, there is a high-altitude version where the spacecraft begins ram pointing earlier in the orbit. For all argon scans, look for this script commanding:

```
NGM_LOAD_TBL "d:/ngm/science_1373.bas" 201 "science_1373.bas
NGM_LOAD_SCRIPT "d:/ngm/sci_arhigh_osion_1905.cfg" UPLOAD_AP
```

followed by NGM_SCRIPT command: run

Note: don't plan two sets of like Argon scans (e.g. two 500km runs) closer to each other than a week (i.e. last orbit of one set occurs < 7 days before first orbit of like set). From a science perspective, it's basically same observation and standard science loses out to a redundant special observation.

OSNB Background

For bi-weekly OSNB backgrounds:

```
NGM_LOAD_TBL "d:/ngm/science_1373.bas" 201 "science_1373.bas
NGM_LOAD_SCRIPT "d:/ngm/sci_loemis_osnb_1538.cfg" UPLOAD_APP
```

followed by NGM_SCRIPT command: run

Note: Unlike most activities, OSNB backgrounds start in the OB_SIDE segment. This activity runs 1h40m.

CPT

For bi-weekly CPT:

```
NGM_LOAD_TBL "d:/ngm/funct_650.bas" 201 "funct_650.bas" 1664
NGM_LOAD_SCRIPT "d:/ngm/functional_cpt_em2_829.cfg" UPLOAD_A
```

followed by NGM_SCRIPT command: run

Note: Unlike most activities, CPTs (aka functionals) start in the APOAPSE segment. This activity runs 1h12m.

Be on the look out for a DTCI toggle in the middle of the CPT:

```
VM_GV_SET_INT GV_NGM_DTCI_CFG 1
VM_GV_SET_INT GV_NGM_DTCI_CFG 0
```

When GV_NGM_DTCI_CFG is set to 1, the interface between the instrument and the spacecraft is suspended. No data that the instrument sends will be recorded. In orbit 21061 and 21149, this happened during a Mass Memory scrub operation on the spacecraft resulting in lost CPT data.

Complications

Relay

Relays will cause NGM_STOP followed by a log of commands including "NGM_MAIN_PWR OFF"

When if comes back, there should be "NGM_MAIN_PWR ON" with several other commands then NGM_GO

At that point, if the orbit is between IB_SIDE and ten minutes prior to OB_SIDE, the normal sciperi script should start, and internally it will resume in the middle to reclaim a portion of the orbit for scanning science data

If the instrument power down occurs soon after a script begins, the sudden on and off of the filament can cause undesirable wear and tear. Please ask for an empty sequence to prevent NGIMS commanding if the delay between script start (EXEC_IMMED 0x676f0000 aka 'go') and NGM_STOP is less than 210 seconds.

DTCI Disable

For some spacecraft activities (e.g. memory scrub), the interface (DTCI) between the spacecraft and NGIMS will be turned off. The NGIMS instrument can continue running during this period where the DTCI is disabled, but no telemetry from the instrument will be recorded. In some cases, commanding may include a DTCI disable in the middle of a script. Be on the lookout for such commanding errors during the bundle review, especially during the OSNB background and CPT scripts.

In the pef, DTCI disable commanding looks like this:

```
NGM_SCRIPT 244 0x21200000 232 EXEC_IMMED 0x7379732e 0x7a6f6e65 0x5f616c65 0x727
b'sys.zone_alert_action=-1'
VM_GV_SET_INT GV_NGM_DTCI_CFG 1
```

In the pef, DTCI enable commanding looks like this:

```
VM_GV_SET_INT GV_NGM_DTCI_CFG 0
NGM_SCRIPT 244 0x21200000 232 EXEC_IMMED 0x7379732e 0x7a6f6e65 0x5f616c65 0x727
b'sys.zone_alert_action=4\x00'
```

Note: the NGM_SCRIPT script line is truncated by procpef.py, so they appear identical. The translation of the hex to ascii that appears below it will include the entire command. From the NGIMS FSW reference: sys.zone_alert_action=-1 ignores zone alerts, while a value of 4 is the default. That is, the instrument will assume it is in a high density zone and go safe upon missing 4 zone alert pings.

Deprecated

Deprecated

MAVEN Get

The MAVEN Get application downloads files from the MAVEN raw database.

Download

maven_get.jar

Usage

```
java -jar maven_get.jar [options]
 -help         print help message and exit
 -version      print version info and exit
 -host <host>  the host name for the XINA Tunnel (default: localhost)
 -port <port>  the post for the XINA Tunnel (default: 41475)
 -ini  <path>  the path to the ini file (default: [cwd]maven_get.ini)
 -path <path>  the path to the downloads directory (default: [cwd]data)
 -appendts     append the entry timestamp to the file name
```

The ini file is loaded to get the starting timestamp. If the file does not exist, all files in the database will be downloaded. The ini file should contain a single section, "maven_get", with a single property, "ts", in the ISO format "yyyy-MM-ddTHH:mm:ss.SSS". For example:

```
[maven_get]
ts=2000-01-01T00:00:00.000
```

All entries with timestamps on or after the start timestamp will be loaded. Files are downloaded to the specified path. If the appendts flag is set, the names will be in the following format, where the timestamp is the timestamp of the entry:

`[yyyy-MM-ddTHH.mm.ss.SSS]_[filename]`

For example:

`2000-01-01T00.00.00.000_somefile.ext`

Entries are processed in order of entry timestamp. If an error occurs while downloading, the process stops and the timestamp of the incomplete entry is recorded to the ini file to resume from that point. If all files download successfully the current time is save to the ini file.

Change Log

1.0.2

- changed saved timestamp to last entry

1.0.1

- added append timestamp option

1.0.0

- initial release