**Global Precipitation Mission (GPM)**

**Ground Validation System**

**Validation Network Maintenance Manual**

**August, 2016**

Goddard Space Flight Center

Greenbelt, Maryland 20771

****

**Document History**

| Document Version | Date | Changes |
| --- | --- | --- |
| 1.0 | August, 2016 | Initial document |
|  |  |  |

# Introduction

This document provides a basic set of documentation for the maintenance of the GPM Ground Validation System (GVS) Validation Network (VN). It addresses common activities such as the addition of new ground radar (GR) sites to the VN dataset, the creation and addition of new orbit subset areas, the management of the VN Postgresql database (“gpmgv” database) and the VN file system, and version control of the IDL software in the VN baseline via the Subversion utility.

VN data ingest processes are described in the document “GPM Validation Network Data Acquisition and Preprocessing Software” (see the latest “V\_v” version of the file Documents/GPM/UserGuides/DataAcquisition/GPM\_VN\_Data\_Acquisition\_V\_v.docx).

The publically-exposed parts of the VN file system are described in the documents GPM Validation Network Data Product User’s Guide, Volume 1 – TRMM Data Products (see the latest MMMYYYY-dated version of the file Documents/GPM/UserGuides/DataUsers/Val\_Network\_Users\_Guide\_Vol\_1\_MMMYYYY.docx) and GPM Validation Network Data Product User’s Guide, Volume 2 – GPM Data Products (see the latest MMMYYYY-dated version of the file Documents/GPM/UserGuides/DataUsers/Val\_Network\_Users\_Guide\_Vol\_2\_MMMYYYY.docx). ***It is important that only VN data approved for public exposure and distribution be placed in the public directory structure:***

gpm-validation/data/gpmgv ftp directory on hector.gsfc.nasa.gov; mounted directory /data/gpmgv on ds1-gpmgv and ws1-gpmgv.

All of these directories are one and the same, they are just defined differently on different hosts.

# Adding GR Sites and GPM Subsets

# Site additions

The process of adding a new GR site to the routine data processing stream in the VN involves multiple steps, both internal to and outside the VN system. The first step is to gather the required metadata for the GR site:

1. unique radar site ID (call letters, e.g., KLWX)
2. radar site long name (e.g., Sterling)
3. radar owner (e.g., NWS, DOD, KMA, NASA)
4. site latitude (deg. N), longitude (deg. E), and elevation (meters)
5. state, province, or subregion identifier (made up if needed)

Note that the “unique radar site ID” will be used by the VN file system (data subdirectory name immediately under the /data/gpmgv/gv\_radar/finalQC\_in directory), the VN database, in the CT (coincidence table) site overpass prediction files produced by the PPS, and by the GR Data Acquisition and Quality Control function (i.e., the QC and “data push” steps performed by Jason Pippitt). The VN database requires a radar site ID of between 1 and 15 characters, but a minimum of 4 characters is the practical size. ***The unique radar site ID, latitude, and longitude must be defined identically in the VN, at the PPS, and in the radar QC/data push function.***

The second step is to enter the GR site metadata into the Postgresql ‘gpmgv’ database. There are two database tables that must be populated with the site metadata: the ‘instrument’ table, and the ‘fixed\_instrument\_location’ table. First, determine whether the radar site ID is already defined in these tables. (Note that all radar site IDs in the U.S. and its territories are already defined in these tables, whether or not they are part of the VN.) For instance, for the radar ID ‘KLWX’, do the following steps:

1. log into ds1-gpmgv as user ‘**morris**’ or ‘**gvoper**’
2. start a postgresql interactive query session (*psql* utility) by entering the command ‘**psql gpmgv**’ (without the quotes) at the command line
3. at the **gpmgv=>** prompt, enter the command: “**select instrument\_id, instrument\_type, instrument\_name from instrument where instrument\_id = 'KLWX';**” without the double quotes, but with single quotes around the KLWX, and terminated with the semicolon. Hit Return to run the command.
4. If the radar ID is already defined in the ‘instrument’ table, then the query will return 1 row of information. Verify that the information is correct and proceed to step 5. If the radar ID is not already defined in the ‘instrument’ table, then the query will return 0 rows of information. In this case, skip to Step 7.
5. (Only if instrument\_id is defined in ‘instrument’ table.) At the gpmgv=> prompt, enter the command: “**select \* from fixed\_instrument\_location where instrument\_id = 'KLWX';**” without the double quotes, but with single quotes around the KLWX, and terminated with the semicolon. Hit Return to run the command.
6. If the radar ID is already defined in the ‘fixed\_instrument\_location’ table, then the query will return 1 row of information. Verify that the information is correct and proceed to step 11. If the radar ID is not already defined in the ‘fixed\_instrument\_location’ table, then the query will return 0 rows of information. In this case, skip to Step 9. Note that the instrument\_id (radar ID) must be defined in the ‘instrument’ table before it can be defined in the ‘fixed\_instrument\_location’ table.
7. (If instrument\_id is NOT defined in ‘instrument’ table.) Determine the unique radar ID value, and the radar type, radar long name, and radar owner values to be used for the site. These values will be assigned to the ‘instrument\_id’, ‘instrument\_type’, ‘instrument\_name’ and ‘owner’ table attributes, respectively. For our example, we will use the attribute values ‘WUSA\_TV’, ‘C-band Doppler’, ‘WUSA-TV Weather Radar’, and ‘WUSA-TV’. After having verified that the instrument ID ‘WUSA\_TV’ is not defined (Steps 3-4), then at the gpmgv=> prompt, enter the command: “**INSERT INTO instrument (instrument\_id, instrument\_type, instrument\_name, owner) VALUES (‘WUSA\_TV’, ‘C-band Doppler’, ‘WUSA-TV Weather Radar’, ‘WUSA-TV’);**” without the double quotes, but with single quotes around the attribute text values, and terminated with the semicolon. Hit Return to run the command.
8. If the INSERT command is successful, the database will display a status message “INSERT 0 1”. If success, proceed to Step 9. If not, then an error will result and the INSERT command syntax or values in Step 7 must be fixed and the command repeated. (Optional if successful INSERT: Run the command “**select \* from instrument where instrument\_id = ‘WUSA\_TV’;**” from the *psql* prompt and review the results.)
9. (If instrument\_id is NOT defined in ‘fixed\_instrument\_location’ table.) Gather the radar ID, its install date (if known, or use first day when data is acquired), the state/province and country IDs, latitude (N), longitude (E), and elevation (m). At the gpmgv=> prompt, enter the command: “**INSERT INTO fixed\_instrument\_location (instrument\_id, install\_date, state\_province, country, latitude, longitude, elevation) VALUES (‘WUSA\_TV’, ‘2015-01-01’, ‘MD’, ‘US’, 39.07, -76.90, 25);**” without the double quotes, but with single quotes around the attribute text values, and terminated with the semicolon. Hit Return to run the command.
10. If the INSERT command is successful, the database will display a status message “INSERT 0 1”. If success, proceed to Step 11. If not, then an error will result and the INSERT command syntax or values in Step 10 must be fixed and the command repeated. (Optional if successful INSERT: Run the command “**select \* from fixed\_instrument\_location where instrument\_id = ‘WUSA\_TV’;**” from the *psql* prompt and review the results.)
11. To exit from *psql* and return to the system prompt, run the command “**\q**” from the *psql* prompt.

The complete *psql* session to cover the steps described above is shown in the following:

[morris@ds1-gpmgv ~]$ psql gpmgv

Welcome to psql 8.1.23, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms

\h for help with SQL commands

\? for help with psql commands

\g or terminate with semicolon to execute query

\q to quit

gpmgv=> select instrument\_id, instrument\_type, instrument\_name from instrument where instrument\_id='WUSA\_TV';

instrument\_id | instrument\_type | instrument\_name

---------------+-----------------+-----------------

(0 rows)

gpmgv=> select \* from fixed\_instrument\_location where instrument\_id='WUSA\_TV';

instrument\_id | install\_date | state\_province | country | latitude | longitude | elevation

---------------+--------------+----------------+---------+----------+-----------+-----------

(0 rows)

gpmgv=> INSERT INTO instrument (instrument\_id, instrument\_type, instrument\_name, owner) VALUES ('WUSA\_TV', 'C-band Doppler', 'WUSA-TV Weather Radar', 'WUSA-TV');

INSERT 0 1

gpmgv=> select \* from instrument where instrument\_id='WUSA\_TV';

instrument\_id | instrument\_type | instrument\_name | owner | parent\_child | produces\_data | fixed\_or\_moving | coverage\_type | replaced\_by\_id

---------------+-----------------+-----------------------+---------+--------------+---------------+-----------------+---------------+----------------

WUSA\_TV | C-band Doppler | WUSA-TV Weather Radar | WUSA-TV | N | Y | F | |

(1 row)

gpmgv=> INSERT INTO fixed\_instrument\_location (instrument\_id, install\_date, state\_province, country, latitude, longitude, elevation) VALUES ('WUSA\_TV', '2015-01-01', 'MD', 'US', 39.07, -76.90, 25);

INSERT 0 1

gpmgv=> select \* from fixed\_instrument\_location where instrument\_id='WUSA\_TV';

instrument\_id | install\_date | state\_province | country | latitude | longitude | elevation

---------------+--------------+----------------+---------+----------+-----------+-----------

WUSA\_TV | 2015-01-01 | MD | US | 39.07 | -76.9 | 25

(1 row)

gpmgv=> \q

[morris@ds1-gpmgv ~]$

# Configuring new routine site overpass processing

In order for site overpass data for a new site to be processed automatically by the VN operational scripts, two steps must be taken following the database configuration of section 2.1. First, the PPS personnel must be provided with the site ID (instrument\_id value in the database tables) and latitude and longitude of the new site(s) to be added to the daily CT (coincidence table) file computations. The CT files provide the orbit number, date, time, and closest approach distance and its latitude and longitude for orbits that approach within 750 km or less of a ground radar site. Separate CT files are produced for GPM and for each constellation satellite. CT files are not predictive so they run 2-3 days behind real time.

In general, we ask PPS to configure new site overpasses to be computed for all working satellites in the GPM constellation. Contacts at the PPS for site addition requests currently include:

[patty.mccaughey@nasa.gov](mailto:patty.mccaughey@nasa.gov)

[jon.mitchell@mail.pps.eosdis.nasa.gov](mailto:jon.mitchell@mail.pps.eosdis.nasa.gov)

The generic contact for PPS issues is: [helpdesk@mail.pps.eosdis.nasa.gov](mailto:helpdesk@mail.pps.eosdis.nasa.gov)

Daily CT files are ingested and processed by the VN scripts wgetCTdailies.sh and newCT\_to\_DB.sh, described in the document “GPM Validation Network Data Acquisition and Preprocessing Software” noted in the Introduction section, above. The second step to take to automate the processing of overpass information for the new sites is to modify the patterns in the ‘sed’ command #5 in the newCT\_to\_DB.sh script so that the CT lines for the new site(s) are not excluded from the processing. As that command is getting long and unwieldy, it may be time to consider eliminating that step in the ‘sed’ command sequence in favor of modifying the SQL command files executed by the scripts to reject sites that are not configured in the VN database when attempting to load new site overpass data into the overpass\_event table. No changes need to be made to the wgetCTdailies.sh script to add new stations to the overpass event processing. However, if a new satellite were to be added to the constellation, then the script wgetCTdailies.sh would need to be configured to acquire its CT files.

To validate that the site overpasses for the new site are being processed, there are two places to look. First is to verify that newCT\_to\_DB.sh is extracting and reformatting the site’s overpass information by looking at its output files. The original CT files and the reformatted overpass data in the output files are located in the baseline directories:

/data/gpmgv/coincidence\_tables/YYYY/MM/DD

where YYYY, MM, and DD are the year, month, and day of the daily CT files acquired from PPS. CT files have the naming convention CT.SAT.YYYYMMDD.jjj.txt, where SAT is the satellite ID; YYYYMMDD is the year, month, and day; and jjj is the day-of-year of the site overpass events tabulated in the file. Processed and reformatted overpass data output by newCT\_to\_DB.sh are located in the same directory and have the same naming convention as the original CT files, except that they have a .unl file extension in place of the .txt extension. Following the example in the preceding section, if PPS has added the site to its processing list and if we have configured the scripts to add the site ‘WUSA-TV’ to the CT file processing starting on 2015/01/01, we would do the following:

cd /data/gpmgv/coincidence\_tables/2015/01/01

grep WUSA\_TV CT.\*.20150101.\*.txt

grep WUSA\_TV CT.\*.20150101.\*.unl

The output from the first ‘grep’ command is to validate that the site overpasses are being computed by PPS and included in the CT files. The output from the second ‘grep’ command is to verify that the station is being processed in the newCT\_to\_DB.sh script and written to the ‘.unl’ files. The last step is to verify that the site overpass information is being correctly written to the ‘overpass\_event’ table in the ‘gpmgv’ database. To do that, start a psql session from the command line and run the SQL command shown:

[morris@ds1-gpmgv ~]$ psql gpmgv

psql (8.4.20)

Type "help" for help.

gpmgv=> select \* from overpass\_event where sat\_id='GPM' and radar\_id='WUSA\_TV' order by orbit desc limit 5;

event\_num | sat\_id | orbit | radar\_id | overpass\_time | nearest\_distance

-----------+--------+-------+----------+--------------------------+------------------

230650 | GPM | 4930 | WUSA\_TV | 2015-01-10 08:14:23.7-05 | 131

229501 | GPM | 4893 | WUSA\_TV | 2015-01-07 22:37:19.5-05 | 172

229154 | GPM | 4884 | WUSA\_TV | 2015-01-07 09:16:41.2-05 | 101

228691 | GPM | 4847 | WUSA\_TV | 2015-01-04 23:41:50.6-05 | 110

227902 | GPM | 4807 | WUSA\_TV | 2015-01-02 10:32:13.4-05 | 151

(5 rows)

gpmgv=> \q

[morris@ds1-gpmgv ~]$

The output content is simulated in the query results shown above. Repeat the query for other sat\_id values if desired. Note that only those overpasses within 250 km of the radar site get loaded into the gpmgv database, so the results of the query will not necessarily include all the entries displayed by the ‘grep’ results. Also note that the query results convert the overpass datetime to the local time zone, with the offset from UTC shown at the end of the datetime value (-05, 5 hours behind UTC).

# Adding a new orbit subset

Orbit subsets are named regions defined as a latitude/longitude rectangle and given a unique name. These orbit subsets are used by the PPS to compute subsets of GPM orbit swath data for specific regions (e.g., CONUS, KWAJ, KOREA). These regions may enclose single (KWAJ) or multiple (CONUS, KOREA) ground radars. The rule of thumb is to define a region that extends 2.25 degrees in all directions from the radar lat/lon for a single-radar orbit subset, or 2.25 degrees outside the max and min latitude and longitude bounds for a group of radars to be included in the subset area. These 2.25-degree extensions are roughly based on providing reasonable overlap between the GR scan area within 100-km range and the NS scan of the GPM Ku-band radar. A larger extension would apply if defining a subset area exclusively for the GMI instrument, which covers a much wider swath.

Each VN orbit subset must be configured in the VN database and in the operational scripts that control PPS data ingest. The steps in creating a new orbit subset are:

1. Define a unique, short, descriptive ID for the subset and determine the latitude and longitude boundaries for the region. This ID will be used in the orbit subset file names generated by the PPS and in the VN database and file system.
2. Determine which satellites and which data products we want the PPS to generate for the subset.
3. Determine which ground radars are to be associated with the new orbit subset. If these radars are not defined in the VN Postgresql database ‘gpmgv’, then first add them to the database as described in section 2.1.
4. Add the satellite(s) and subset to the ‘productsubset’ table, and then add the satellite, subset, and associated radar\_id(s) to the ‘siteproductsubset’ table in the ‘gpmgv’ database. For purposes of these two tables, the ‘product’ attribute is simply the satellite ID, not a more specific product type. An example of adding a subset and associated satellites and radar\_ids to these tables is shown below. (NOTE: The radar\_id attribute in this and other tables where radar\_id is an attribute have a “foreign key” relationship to their instrument\_id attribute values defined in the ‘instrument’ table. Unless the ID is first defined in the ‘instrument’ table it will not be possible to use the ID in any table where it needs to be used as the radar\_id attribute.)
5. Configure the ‘*case*’ statements in the operational scripts get\_PPS\_CS\_data.sh and update\_CS\_dirs.sh and the utility script catalog\_PPS\_CS\_leftovers.sh on ds1-gpmgv in the /home/gvoper/scripts directory, to include the new subset/satellite combination(s) for which the PPS is to generate the orbit subset products.
6. Create new subset/ subdirectories for the affected satellite(s) and products(s) under the /data/gpmgv/orbit\_subset directory tree. For this purpose refer to the utility scripts set\_up\_CS\_dirs.sh and set\_up\_KOREA\_CS\_dirs.sh on ds1-gpmgv in the /home/morris/swdev/scripts directory. The directory tree under /data/gpmgv/orbit\_subset is described in detail in the GPM Validation Network Data Product User’s Guide, Volume 2.
7. Provide the satellite ID(s), product type(s), subset ID, and lat/lon boundaries to the PPS for them to add it to the GPM GV subscription. This subscription is handled differently from those created via the PPS STORM web utility.
8. Once the PPS has begun to create the new subset(s), review the daily log files from the script get\_PPS\_CS\_data.sh to make sure the new subset files are being received and processed (files named get\_PPS\_CS\_data.yymmdd.log on ds1-gpmgv in the /data/logs directory, where yymmdd is the 2-digit year, month, and day of the script run). Proper processing involves 3 major steps: (1) identify and download the files of interest listed in the PPS-produced ‘ftp\_url’ files, (2) move the downloaded files into the proper directory in the VN file system, and (3) catalog the files in the ‘orbit\_subset\_product’ table in the ‘gpmgv’ database.

The complete *psql* session to cover step 4 described above is shown below. In this example, we will add a new subset “Brisbane” for the GPM orbit subset dataset, and associate the radar site ID “CP2” to this subset.

[morris@ds1-gpmgv ~]$ psql gpmgv

Welcome to psql 8.1.23, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms

\h for help with SQL commands

\? for help with psql commands

\g or terminate with semicolon to execute query

\q to quit

gpmgv=> insert into productsubset values ('GPM','Brisbane');

INSERT 0 1

gpmgv=> select \* from productsubset where subset='Brisbane';

sat\_id | subset

--------+----------

GPM | Brisbane

(1 row)

gpmgv=> insert into siteproductsubset values( 'GPM','Brisbane','CP2');

INSERT 0 1

gpmgv=> select \* from siteproductsubset where subset='Brisbane';

sat\_id | subset | radar\_id

--------+----------+----------

GPM | Brisbane | CP2

gpmgv=> \q

[morris@ds1-gpmgv ~]$

# Adding a new satellite

The process of adding a new satellite to the constellation is similar to that of adding a new ground radar site. The first step is to gather the satellite and instrument identifiers and owner (agency or country, etc.) for the satellite. The satellite ID must be a unique, new ID, whereas the instrument ID and owner can apply to more than one satellite.

The second step is to enter the satellite metadata into the Postgresql ‘gpmgv’ database. The database table that must be populated with the site metadata is the ‘instrument’ table. First, determine whether the satellite ID is already defined in the table. For instance, for the satellite ID ‘NOAA20’, do the following steps:

1. log into ds1-gpmgv as user ‘**morris**’ or ‘**gvoper**’
2. start a postgresql interactive query session (*psql* utility) by entering the command ‘**psql gpmgv**’ (without the quotes) at the command line
3. at the **gpmgv=>** prompt, enter the command: “**select \* from instrument where instrument\_id = 'NOAA20';**” without the double quotes, but with single quotes around the NOAA19 value, and terminated with the semicolon. Hit Return to run the command. If the satellite is not defined in the table, then no row of data will be returned, as expected for a new satellite.
4. Find a similar satellite already defined in the instrument table, if one exists, or just choose any satellite ID already present in the table. The select its data into a temporary table. At the **gpmgv=>** prompt, enter the command: “**select \* into temp new\_sat\_temp from instrument where instrument\_id = 'NOAA19';**” without the double quotes, but with single quotes around NOAA19, and terminated with the semicolon. Hit Return to run the command.
5. Verify that there is one row of data in the temporary table by running the SQL command “**select \* from new\_sat\_temp;**” without the quotes. One row of data should be returned.
6. Determine which columns of data need to be updated for the new satellite. In this case, we only need to change the instrument\_id from NOAA19 to NOAA20. Update this field in the temporary table via the command: “**update new\_sat\_temp set instrument\_id =’NOAA20’ where instrument\_id = ‘NOAA19’**;”. The status message “UPDATE 1” should be shown if the update is successful. Repeat for as many fields as needed.
7. Verify the contents of the row in the temporary table by running the query “**SELECT \* FROM new\_sat\_temp;**” without the quotes.
8. If the data in the temporary table are OK, then copy the new row of data from the temporary table to the instrument table by running the query “**INSERT INTO instrument SELECT \* FROM new\_sat\_temp;**”. If it succeeds, the message “INSERT 1” will appear.
9. Verify the presence of the new row in the ‘instrument’ table by running the query “**select \* from instrument where instrument\_id = 'NOAA20';**”.
10. Type the command **\q** to exit from *psql*. The temporary table will be automatically deleted.

The satellite instruments themselves are also defined in the instrument table as “children” of the satellites, ***but this is optional***. An example session is shown below, where we define a new scanning instrument ‘QMI’ for a satellite called “QSat1”, where we assume that QSat1 has already been defined in the instrument table as a satellite ID.

[morris@ds1-gpmgv common\_utils]$ psql gpmgv

psql (8.4.20)

Type "help" for help.

gpmgv=> select \* into temp1 from instrument where instrument\_id='TMI';

SELECT

gpmgv=> select \* from temp1;

instrument\_id | instrument\_type | instrument\_name | owner | parent\_child | produces\_data | fixed\_or\_moving | coverage\_type | replaced\_by\_id

---------------+---------------------------+-----------------------+-------+--------------+---------------+-----------------+---------------+----------------

TMI | Scanning Microwave Imager | TRMM Microwave Imager | NASA | C | Y | M | cross track | NA

(1 row)

gpmgv=> update temp1 set instrument\_id='QMI';

UPDATE 1

gpmgv=> update temp1 set instrument\_name='QSat1 Microwave Imager';

UPDATE 1

gpmgv=> select \* from temp1;

instrument\_id | instrument\_type | instrument\_name | owner | parent\_child | produces\_data | fixed\_or\_moving | coverage\_type | replaced\_by\_id

---------------+---------------------------+------------------------+-------+--------------+---------------+-----------------+---------------+----------------

QMI | Scanning Microwave Imager | QSat1 Microwave Imager | NASA | C | Y | M | cross track | NA

(1 row)

gpmgv=> insert into instrument select \* from temp1;

INSERT 1

gpmgv=> \q

After defining the new satellite in the ‘instrument’ table, then we must associate orbit subsets and ground radars to it in the ‘productsubset’ and ‘siteproductsubset’ tables. This has been described in Step 4 in Section 2.2. There are also SQL example commands of how to do this by ‘cloning’ from existing satellite/subset pairs shown in the file /home/morris/swdev/GPM\_DB\_changes.sql.

Once the new satellite ID has been configured in the database, then we need to configure the data directory setup scripts and run them to build the baseline data directories needed to hold data for the satellite. The script /home/morris/swdev/scripts/ set\_up\_CS\_dirs.sh should be edited to configure the new directories. Assuming the new satellite and instrument are only providing sounder or imager products (2A-GPROF and 1C-R-XCAL) of interest to the VN, then edit the red highlighted line as shown below. The assumption is that we are configuring MHS products from a new NOAA20 satellite:

umask 0002

# create the initial directory trees for the GPROF/XCAL instruments/products

#for sat in TRMM/TMI GPM/GMI GCOMW1/AMSR2 F15/SSMIS F16/SSMIS F17/SSMIS F18/SSMIS METOPA/MHS METOPB/MHS NOAA18/MHS NOAA19/MHS **NOAA20/MHS** NPP/ATMS

**for sat in NOAA20/MHS**

do

cd $ORB

Note that satellite/instrument pair has also been added to the commented-out line above the modified line. The line would be used in place of the red line when setting up a complete VN directory tree for all satellites and instruments in a new location. Once the script has been modified, run it and verify that the new directory tree has been created. From the unix prompt run the commands:

[morris@ds1-gpmgv orbit\_subset]$ ls NOAA20/MHS/\*/\*

NOAA20/MHS/1CRXCAL/V04A:

AKradars BrazilRadars CONUS DARW Finland Guam Hawaii KOREA KWAJ SanJuanPR

NOAA20/MHS/2AGPROF/V04A:

AKradars BrazilRadars CONUS DARW Finland Guam Hawaii KOREA KWAJ SanJuanPR

and make sure the directories exist as shown. Note that there may be more subsets than shown above if new subsets have been defined in the interim.

Lastly, configure the ‘*case*’ statements in the operational scripts get\_PPS\_CS\_data.sh and update\_CS\_dirs.sh and the utility script catalog\_PPS\_CS\_leftovers.sh on ds1-gpmgv in the /home/gvoper/scripts directory, to include the new subset/satellite combination(s) for which the PPS is to generate the orbit subset products. This is identical to the process of adding a new subset for an existing satellite as described in Step 5 in Section 2.2. Step 8 in that section describes the process of validating that products for the new satellite are being ingested, cataloged, and moved into their baseline directories in the VN file system.

# Computing historic site overpass and rain event data for a new site

# Site Overpass Data

PPS generates a 7-day prediction of GPM satellite orbit subtrack locations at 1-second intervals on a daily basis in the form of a text file. Thus, there is a 6 day overlap of the prediction data between files of adjacent dates. The daily GT-7 files are acquired from the PPS ftp site by the wget\_GT7\_GPM.sh script, which is run once daily via crontab. The GT-7 filename convention is **GT-7.SAT.yyyymmdd.jjj.txt**, where **SAT** is the ID of the satellite, **yyyymmdd** is the year, month, and day, and **jjj** is the day-of-year of the starting prediction data tabulated in the file. Only GPM predictions are acquired by the VN. The GT-7 files are written outside the public VN file system to the directory **/data/tmp/ground\_track\_7day/YYYY** on ds1-gpmgv, where **YYYY** is the year of the file datestamp **yyyymmdd**.

If wget\_GT7\_GPM.sh is successful at acquiring one or more GT files, it then calls the child script extract\_daily\_predicts.sh to process the GT file(s) into files that contain only a single day’s worth of 1-second subtrack predictions with no overlaps in time coverage. The extracted daily data are written to files named **GPM\_1s\_subpts.yyyymmdd.txt**, where **yyyymmdd** is the year, month, and day of the prediction data tabulated in the file. The **GPM\_1s\_subpts** files are written into the directory **/data/tmp/dailypredict** on ds1-gpmgv, outside the public VN file system. Neither the GT-7 nor the GPM\_1s\_subpts files are used in routine VN operations. However, they are useful in generating historical and/or future site overpass times for sites that are not configured in the PPS CT file production. The IDL function module get\_coincidence\_via\_track() found in the file **/home/morris/swdev/idl/dev/common\_utils/get\_coincidence\_via\_track.pro** provides a capability to compute site overpasses from the GPM\_1s\_subpts files.

Before attempting to manually compute historic site coincidence events from the GPM\_1s\_subpts files, it is best to make sure that the ground radar site metadata have been added to the gpmgv database (Section 2.1) and the site has been associated to an orbit subset (Section 2.2). Once this has been done, then the steps to compute site coincidences for the new site are as follows:

Get the latitude and longitude of the radar from the ‘gpmgv’ database. For instance, for the radar ID ‘KLWX’, do the following steps:

1. log into ds1-gpmgv as user ‘**morris**’ or ‘**gvoper**’
2. start a postgresql interactive query session (*psql* utility) by entering the command ‘**psql gpmgv**’ (without the quotes) at the command line
3. at the **gpmgv=>** prompt, enter the command: “**select instrument\_id, latitude, longitude from instrument where instrument\_id = 'KLWX';**” without the double quotes, but with single quotes around the KLWX, and terminated with the semicolon. Hit Return to run the command. The command and output are shown below:

[morris@ds1-gpmgv ~]$ psql gpmgv

psql (8.4.20)

Type "help" for help.

gpmgv=> select instrument\_id, latitude, longitude from fixed\_instrument\_location where instrument\_id ='KLWX';

instrument\_id | latitude | longitude

---------------+----------+-----------

KLWX | 38.9753 | -77.4778

(1 row)

gpmgv=> \q

[morris@ds1-gpmgv ~]$

1. exit the psql session by typing a ‘\q’ command (without the quotes) as shown above, and start an IDL session by typing ‘idl’ (without quotes) at the command line, e.g.:

[morris@ds1-gpmgv daily\_predict]$ idl

IDL Version 8.5 (linux x86\_64 m64). (c) 2015, Exelis Visual Information Solutions, Inc., a subsidiary of Harris Corporation.

Installation number: 12981.

Licensed for use by: NASA-GSFC Lab for Atmospheres

IDL>

The IDL function get\_coincidence\_via\_track requires 8 input parameters to be specified in order to run correctly. In order of inclusion in the IDL command these are as defined below:

**path** - STRING, directory where the "GPM\_1s\_subpts.\*.txt" files reside.

**start\_date** - STRING, Starting date to compute site overpasses, YYYYMMDD

format

**end\_date** - STRING, Ending date to compute site overpasses, YYYYMMDD format

**siteID** - STRING, site identifier as defined in 'gpmgv' database

(instrument\_id in table fixed\_instrument\_location, or radar\_id

in the gvradar table)

**siteLat** - Latitude of ground radar, deg. N

**siteLon** - Longitude of ground radar, deg. E

**thresh\_dist** - Maximum allowed distance of ground radar from the orbit track.

**outpath** - STRING, directory where the computed site approach data file

"siteID\_Predict.txt" will be created, where siteID is the value

provided as that parameter.

1. Using the siteID, latitude, and longitude from the database query and the known path /data/tmp/dailypredict to the "GPM\_1s\_subpts.\*.txt" files, assemble and run the IDL command for a run of get\_coincidence\_via\_track(). Note that all parameters of STRING type only must be enclosed in single or double quote pairs, and that the parameters must be specified in the order listed above. At the IDL command prompt enter the two lines:

IDL> n = get\_coincidence\_via\_track( '/data/tmp/daily\_predict', $

IDL> '20140515', '20140520', 'KLWX', 38.9753, -77.4778, 250, ‘/tmp’ )

The command has been split into two lines by the IDL continuation character ($) for readability. Assuming there are no errors, the date/times of the GPM overpasses of site KLWX between the dates 2014/05/15 and 2014/05/20 that were within 250 km of the KLWX site will be output to the screen, as follows:

% Compiled module: GET\_COINCIDENCE\_VIA\_TRACK.

% Compiled module: MAP\_2POINTS.

GPM|KLWX|2014-05-16 01:31:54+00|24

GPM|KLWX|2014-05-17 10:20:10+00|176

GPM|KLWX|2014-05-19 00:30:18+00|155

GPM|KLWX|2014-05-20 09:18:24+00|7

Output file = /tmp/KLWX\_Predict.txt

and the same four lines of computed KLWX site overpass data will be written to the file /tmp/KLWX\_Predict.txt, output from the IDL routine.

1. After the function runs the variable “n” which was specified to the left of the = sign will have the value of the number of site overpasses computed in the run. You can see its value immediately after the run completes by running the following command in IDL, followed by the IDL exit command to leave IDL and return to the user command prompt:

IDL> help, n

N INT = 4

IDL> exit

[morris@ds1-gpmgv ~]$

The value of n is 4, as expected from the output to the screen and file. Note that the computed output contains no information on the GPM orbit number, just the date and time (UTC) and distance of the overpass. This output needs to be merged with information on GPM orbits and their start and end times. These orbit parameters are extracted from GPM single-orbit definition files downloaded from the PPS by the script wget\_orbdef\_GPM.sh, which is run daily in the suite of cron-run scripts for the GPM GV operational user account ‘gvoper’. This script loads the orbit number and orbit start and end datetimes to the database table ‘gpm\_orbits’ in the ‘gpmgv’ database.

The SQL commands contained in the file **/home/morris/swdev/scripts/load\_coincidences\_via\_track.sql** are the series of commands needed to merge the overpass times with the orbit numbers and load them to the operational database table ‘overpass\_event’ in the gpmgv database. Before these commands can be run, the third command:

\copy tempovrpass from '/tmp/AU-71\_Predict.txt' with delimiter '|'

contained in the file must be edited to replace the filename /tmp/AU-71\_Predict.txt with the pathname of the file output by the IDL function. In the IDL run in the example above, the output filename was /tmp/KLWX\_Predict.txt, so the line would be edited to become:

\copy tempovrpass from '/tmp/KLWX\_Predict.txt' with delimiter '|'

and the modified file **load\_coincidences\_via\_track.sql** would be saved.

1. Once the editing of **load\_coincidences\_via\_track.sql** is completed, the commands in the file should be run using the psql database utility to load the data to the database. The most direct way to do this is to run psql with the command file sent as its command input. From the unix/linux command line run the two commands:

cd /home/morris/swdev/scripts

echo "\i load\_coincidences\_via\_track.sql "| psql -a -d gpmgv

If all goes well, the output to the screen will have no errors and the diagnostics

insert into overpass\_event(sat\_id, radar\_id, overpass\_time, nearest\_distance, orbit) select sat\_id, radar\_id, overpass\_time, nearest\_distance, orbit from event2load where event\_num is null;

INSERT 0 4

will appear at the end of the output, where 4 is the number of overpass events in the output file (/tmp/KLWX\_Predict.txt) we computed in IDL.

If the following error is reported in the output to the screen:

psql:load\_coincidences\_via\_track.sql:11: ERROR: duplicate key value violates unique constraint "overpass\_event\_sat\_id\_key"

then the likely cause is that two close coincident times at/near 00:00:00 GMT and within the same orbit’s period were inadvertently computed for the radar site, resulting in two entries in the xxx\_Predict.txt file with the same orbit number. In that case open the xxx\_Predict.txt file in an editor and search for the pattern 00:00:00. If such an entry is found and the time of the preceding line is less than the length of a GPM orbit (95 minutes) from the 00 GMT date/time, delete the line with a time of 00:00:00+00 from the file and try the psql command file steps again (step 7, above). There may be more than one such “duplicate” date/time in the file that needs to be removed before the data load successfully.

The example shown above is not valid since the database already contains the orbit event information for the site ‘KLWX’ and the queries are such that any new entries for these orbits will not even be attempted. The data from the file /tmp/KLWX\_Predict.txt will simply be loaded to temporary tables defined in the queries and then safely deleted with no action taken to load them and no errors reported.

1. To validate that the coincident events for the new site have been added to the database, start a ‘psql gpmgv’ session as described above, and enter the query:

select \* from overpass\_event where radar\_id=’KLWX’;

substituting the new radar ID for ‘KLWX’, and check the results.

# Rainy Overpass Events

Once the new overpass event data have been added to the database, and assuming that 2A-Ku or 2A-DPR products for the orbit subset that is associated to the radar ID are already present in the VN dataset and cataloged in the database, there are two more steps to identify and catalog active rain events in the new set of overpasses. The script **/home/morris/swdev/scripts/do\_Missing\_DPR\_Metadata4site.sh** computes raw statistics about the DPR-identified occurrence of rain in a 300x300 km grid of 4 km resolution centered on the ground radar location, and loads this raw information to the ‘event\_meta\_numeric’ table in the ‘gpmgv’ database. The SQL query commands in the file **/home/morris/swdev/scripts/rainCases100kmAddNewEvents.sql** compute the fraction of the ground radar coverage within 100 km range that is covered by the DPR scan swath, and the fractions of these grid samples that are raining of any rain type, convective rain type, and stratiform rain type, where at least 100 samples must have rain of any type detected, and loads this information to the ‘rainy100inside100’ table in the ‘gpmgv’ database. The rainy100inside100 table entries are the basis for the DPR-GR volume matching script to determine which events to include in the volume matching computations.

The **do\_Missing\_DPR\_Metadata4site.sh** script has a set of command line options that can be specified to configure it to run for a certain ground radar site and set of GPM data. The synopsis of the script is:

**do\_Missing\_DPR\_Metadata4site.sh –s SAT\_ID –i INSTRUMENT \**

**–v PPS\_VERSION –a ALGORITHM –g GRSITE –u SUBSET**

where the backslash in the continuation character for the linux shell command, and the options are defined below.

**SAT\_ID** – Always GPM (default is GPM if not specified)

**INSTRUMENT** – Either Ku (default) or DPR

**PPS\_VERSION** – Version of 2A-Ku or 2A-DPR data used (V04A by default)

**ALGORITHM** – Either 2AKu (default) or 2ADPR, and must match INSTRUMENT

**GRSITE** – ID of the ground radar, e.g., KLWX, as defined in the ‘instrument’ and fixed\_instrument\_location tables in the gpmgv database.

**SUBSET** – Orbit subset associated with the GRSITE and SAT\_ID in the ‘siteproductsubset’ table in the database, e.g. CONUS.

The default values for each of these options are defined in the script, so either these default values can be edited in the script to configure the desired dataset to run, or they can be overridden by the command line arguments described in the synopsis. For instance, to run rain event metadata for the KLWX radar in the CONUS subset using the V03B version of the 2A-DPR product, the command would look like:

**do\_Missing\_DPR\_Metadata4site.sh –i DPR –v V03B –a 2ADPR –g KLWX –u CONUS**

If the script runs successfully and populates the metadata into the ‘gpmgv’ database, then the last step is to update the contents of the ‘rainy100inside100’ table in the ‘gpmgv’ database. From the command line, enter the following commands:

cd /home/morris/swdev/scripts

echo "\irainCases100kmAddNewEvents.sql "| psql -a -d gpmgv

on the other hand, if there are errors noted on screen during the running of **do\_Missing\_DPR\_Metadata4site.sh**, they can be checked by reviewing the script’s log file, **/data/logs/meta\_logs/do\_Missing\_DPR\_Metadata.MsgMta.log**.