NPP Aggregation Tool Components

Albert Cheng

Larry Knox

Elena Pourmal

(DRAFT)

This document describes the components of the nagg tool for aggregating and deaggregating NPP data files. The tool produces a set of NPP data files with the data granules from the original files divided into smaller, larger, or the same size aggregations, according to the specified command line options.

# Introduction

Nagg is a tool for aggregating JPSS data granules in existing files with a different number of granules per file than in the original files, including deaggregation to one granule per file. Future versions will also package and unpackage granules of compatible products.

The purpose of the tool is to facilitate creating aggregations of already downloaded data granules with aggregations of a different size or eventually with a different packaging without requesting and downloading the same data more than once.

# Approach

The nagg tool is intended to rearrange existing data files into new files with different sized aggregations and in the near future in different packaged combinations of compatible products. The tool creates copies of the existing data and updates metadata to reflect the new aggregation. It will also create fill granules with calculated timestamps and fill values for other metadata and for raw data, using existing granules as a pattern. Creating fill values without existing granules is beyond the scope and ability of the tool, but if the majority of the desired data is present, the tool can potentially proceed after the missing data is added from an external source.

Nagg was implemented with several modules to handle different phases of the process. The “Command parser” module processes the options specified on the command line and passes them to the other modules. “Get granules” module produces a table of all the granules in the input files (see Figure 1). “Select granules” module sorts the table, determines the output file names and characteristics, and specifies the writing of the granules to the output files. “Write granules” module uses the HDF5 library to create the output files and write the granules as specified by “Select granules” module according to the JPSS Common Data Format Control Books.

Figure 1: Example Granule Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Granule ID | DPID | GranuleIndex | GranuleVersion | BeginningTime | EndingTime | 8 more  See Appendix 1 |
| NPP001212767892 | REDRO | 0 | A1 | 1422244825812163 | 1422244855612163 |  |
| NPP001212767892 | GCRIO | 0 | A1 | 1422244825812163 | 1422244855612163 |  |
| NPP001212768212 | REDRO | 1 | A1 | 1422244857812163 | 1422244887612163 |  |
| NPP001212768212 | GCRIO | 1 | A1 | 1422244857812163 | 1422244887612163 |  |
| NPP001212768532 | REDRO | 2 | A1 | 1422244889812163 | 1422244919612163 |  |
| NPP001212768532 | GCRIO | 2 | A1 | 1422244889812163 | 1422244919612163 |  |
| NPP001212768852 | REDRO | 3 | A1 | 1422244921812163 | 1422244951612163 |  |
| NPP001212768852 | GCRIO | 3 | A1 | 1422244921812163 | 1422244951612163 |  |
| NPP001212769172 | REDRO | 4 | A1 | 1422244953812163 | 1422244983612163 |  |
| NPP001212769172 | GCRIO | 4 | A1 | 1422244953812163 | 1422244983612163 |  |
| NPP001212769492 | REDRO | 0 | A1 | 1422244985812163 | 1422245015612163 |  |
| NPP001212769492 | GCRIO | 0 | A1 | 1422244985812163 | 1422245015612163 |  |

# Structures and variables

Appendix 1: granule\_t structure members

Appendix 2: Size definitions

# Nagg tool software modules

“Command parser” module

“Get granules” module

“Select granules” module

“Write granules” module

## “Command parser” module

### Purpose:

To read and validate user command line arguments, setting the tool to process designated files accordingly.

Details to come.

## “Get granules” module

### Purpose:

This module reads metadata from the input files and uses it to populate the granule table.

### Public Functions:

4.2.1 nagg\_get\_granules

nagg\_get\_granules(char \*\*file\_list, int number\_of\_files,

char \*\*products\_list, int nproducts, char \*\*geoproduct,

granule\_p\_t \*granule\_info\_p[], int \*number\_of\_granules\_p)

### Parameters:

file\_list list of files containing granules to be added to the granule table

number\_of\_files number of file names in the list

products\_list list of product types for which granules will be written to a file

nproducts number of products types in the list

geoproduct address of variable to return the DPID of the geolocation product

\*granule\_info\_p[] address of the granule table to be populated

\* number\_of\_granules\_p address of variable for number of granules put in the table

### Return values:

0 if successful, -1 otherwise

Description:

The nagg\_get\_granules() function opens and reads the files in the list provided by the command parser, putting the values of attributes necessary for reaggregating the granules in the members of an instance of the granule\_t structure which is added to the granule table. Unless the –g no option is specified or the file is a GEO file, the file specified by the file’s N\_GEO\_Ref attribute will also be opened and read, and its granules added to the granule table.

Error messages will be returned if a file specified is not an HDF5 file, if the file does not exist or cannot be accessed due to insufficient file permissions, or if the file cannot be opened due to an HDF5 failure. The tool will not continue if any of these errors are encountered.

The attributes from which granule information is gathered are attributes of several different objects in the file. Some are attributes of the root group. Others are attributes of the product groups which are subgroups of the /Data\_Products group. The function iterates through all subgroups of /Data\_Products, collecting granule information from the groups and their aggregate and granule datasets.

## Select granules module

### Purpose:

To select granules for writing to a set of output files in aggregations matching those provided by CLASS, indicating fill granules as appropriate.

Details to come.

## “Write granules” module

### Purpose:

To create output files and write granules as directed.

### Public Functions:

* + 1. start\_write

start\_write(const char \*\*outfiles, int noutfiles, const char \*outgeofile,

char \*\*products\_list, int nproducts, const char \*creationdate,

const char \*creationtime, int ngranulesperfile)

### Parameters:

outfiles list of file names to be created for writing an output aggregation

noutfiles number of names in the outfiles list

outgeofile name of the corresponding geo-location file, or null

products\_list list of DPIDs, one for each product. Only one product is supported for this version

nproduct number of DPIDs in the products\_list

creationdate date of creation of the output files (for writing to N\_HDF\_Creation\_Date attribute)

creationtime time of creation of the ouput files (for writing to N\_HDF\_Creation\_Time attribute)

ngranulesperfile number of granules in each aggregation

### Return values:

0 if successful, -1 otherwise

### Description:

The start\_write() function is the first function called when writing an aggregation of granules. For a single product with the corresponding geo granules in a separate file, start\_write() creates the product and geo output files and adds to those files the group structure to which the datasets will be added when granules are written. When multiple products are supported in the future, for the –S nagg tool option, start\_write() will create an output file for each product for each aggregation of granules, plus the geo file if geo granules are aggregated separately. When packaging is supported, start\_write() will create one output file for all products in an aggregation.

The start\_write() function also writes 3 attributes to the root group of the files: N\_GEO\_Ref, for files except the geo file, N\_HDF\_Creation\_Date, and N\_HDF\_Creation\_Time.

* + 1. write\_granules

write\_granules(granule\_p\_t granule, int i\_granule)

### Parameters:

granule pointer a granule\_t structure containing information about a granule in

an input file

i\_granule the index of this granule in the aggregation

### Return values:

0 if successful, -1 otherwise

### Description:

The write\_granules() function is called for each granule selected to be written to an aggregation, and is responsible for writing most of the data and attributes to the new file, whether the values are from the original file or are generated by the nagg tool. The function does the following:

* Selects the output file for the granule’s product.
* Opens the input file specified by granule->file\_in.
* Initializes the ouput file when first called with a non-fill granule.
  + Copies root group attributes
  + Creates group structure in the file, creating product groups in /All\_Data and /Data\_products
  + Copies datasets from file\_in/All\_Data to output\_file/All\_Data, resized for the new aggregation size
  + Copies attributes from the input file’s /Data\_products/<product group> to the output file’s /Data\_products/<product group>
* Copies the /Data\_Products/… dataset for the granule in the input file to the dataset for the granule in the output file. References and metadata that are specific to the new file will be overwritten in subsequent steps.
* Copies the granule’s hyperslab for each dataset in /All\_Data from the input file to the output file creating a region reference to the new location in the granules new /Data\_Products/… dataset
* Creates the /Data\_Products/…Aggr dataset with object references to all the datasets in /All\_Data/< product group>. Copies attributes from the Aggregate dataset in the input file to the Aggregate dataset in the output file.
* Copies values for the Aggregate dataset’s AggregateBeginningDate, AggregateBeginningGranuleID, AggregateBeginningOrbitNumber and AggregateBeginningTime from the first real granule in the aggregation. If fill granules are to have calculated values for time stamps, some or all of these should be copied from the first granule in the aggregation, whether it is real or fill, rather than from the first real granule.
* Increments the value of the last granule written variable.
* [This is the last thing the function does] The index specified in the i\_granule parameter is compared with the index of the last granule previously written to the output file. Granules must be sorted and written in ascending order; therefore if i\_granule is not the next granule after the last previously written granule, the write\_missing\_granules function will be called internally by the module to write all granules between the last written granule and the current granule.

4.4.3 end\_write

### Parameters:

There are no parameters for the end\_write function

### Return values:

0 if successful, -1 otherwise

### Description:

For each output file in the aggregation, the end\_write() function checks to see if the specified number of granules for an aggregation have been written to the file. If not, write\_missing\_granules is called to write the missing granules to the file as fill granules. The file is then closed.

# Appendix 1: granule\_t structure members

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description (from CDFCB Vol V, Table 4.4.4)** | **Source** |
| product\_id | char[] | 5 character DPID | Look up product\_name in table |
| product\_name | char[] | Collection Short Name | Name of group in /Data\_Products |
| granule\_input\_index | Int | index of the granule’s dataset in the input file | Nagg tool |
|  |  | (The rest of these descriptions are the definitions of the attributes in the column to the right. These may need revision.) |  |
| granule\_id | char[] | The unique identifier for each RDR granule  composed of the concatenation of two  components:  (1) The three character satellite identifier  [alias “Platform\_Short\_Name”],  (2) A zero left filled, 12 character number,  specifying the number of tenths of a second  since the first ascending node after launch) | N\_Granule\_ID |
| granule\_version | char[] | Indicates the version number of the granule  that occurs as the result of an automatic  repair of a granule, an IDPS operator  commanded re-execution of a granule, or a  manual execution of a granule. | N\_Granule\_Version |
| granule\_version\_number | Int | /\*granule version number - derived from granule - version: N/A=>-1, An=>n |  |
| granule\_start\_time\_IET | unsigned long long | The time of the beginning of the temporal  range of the data contained in the granule,  expressed in IET. | N\_Beginning\_Time\_IET |
| granule\_end\_time\_IET | unsigned long long | The time of the ending of the temporal range  of data contained in the granule, expressed  in IET. | N\_Ending\_Time\_IET |
| beginning\_date | char[] | Beginning date of the temporal range  (observation date) for a granule. | Beginning\_Date |
| beginning\_time | char[] | Beginning time of the temporal range  (observation time) for a granule. | Beginning\_Time |
| ending\_time | char[] | Ending date of the temporal range  (observation date) for a granule. | Ending\_Time |
| orbit\_number | uint64\_t | The number of the orbit at the start of the  data collection for a data granule. | N\_Beginning\_Orbit\_Number |
| geofile | char \* | Filename of the HDF5 file containing the  related Geolocation information. | N\_GEO\_Ref |
| file\_in | char \* |  | Input file name |

# Appendix 2: Size definitions

/\*Granule macro definitions \*/

#define NAGG\_Product\_Type\_size 63 /\* up to 63 chars long \*/

#define NAGG\_Granule\_ID\_size 15 /\* Satellite 3 bytes, \*/

/\* 10 microsec: 12 bytes \*/

/\* Total 15 bytes \*/

#define NAGG\_GRANVER\_size 15 /\* Granule version info size \*/

#define NAGG\_DATE\_size 8 /\* Granule date info size \*/

#define NAGG\_TIME\_size 14 /\* Granule time info size \*/

#define NAGG\_Granule\_info\_max 7000 /\* Max number of granules managed \*/

#define NAGG\_Product\_list\_max 30 /\* Max number of products requested \*/

#define NAGG\_outputfiles\_max 30 /\* Max number of output file hames \*/

#define NPP\_Product\_max 99 /\* Max number of NPP Products \*/

#define NPP\_Geo\_Location\_max 19 /\* Max number of NPP Geolocations products \*/

#define NAGG\_Granules\_selected\_max 500 /\* Max number of granules selected \*/

/\* to output \*/

#define Product\_DPID 0 /\* DPID column in Product Table\*/

#define Product\_sname 1 /\* short name column in Product Table\*/

#define Product\_lname 2 /\* long name column in Product Table\*/

/\* NPP data product file name struct \*/

#define DPID\_size 5 /\* DPID name size \*/

#define DPID\_NUM\_MAX 30 /\* max number of DPIDs \*/

#define SPACECRAFT\_size 3 /\* Spacecraft ID \*/

#define Data\_date\_size 8 /\* Date: YYYYMMDD \*/

#define Data\_time\_size 7 /\* Time: HHMMSSS \*/

#define Orbit\_number\_size 5 /\* Orbit: nnnnn \*/

#define Creation\_date\_size 20 /\* Creation Date: YYYYMMDDHHMMSSssssss \*/

#define Origin\_size 4 /\* Origin: XXXX \*/

#define Domain\_size 3 /\* Domain: XXX \*/

# Appendix 3: NPOESS Common Terms

**Table 3.5.1-1, NPOESS Data Product Common Terms**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Aggregation | Dereferences (or “points”) to an HDF5 group that contains one or more datasets. These datasets are the individual RDR granules. Granules are ordered temporally. The aggregation can be accessed with the HDF5 reference object.For a detailed explanation of aggregations, see Section 3.5.12, DDS Aggregation Methodology. |
| Attribute | An attribute is a single, named parameter that has one or more values (where more than one value is applicable, the list of values is stored as an array in the NPOESS HDF5 File). |
| Granule\* | Stored purely as an array of bytes (unsigned char) referenced with a single object ID. |
| HDF5 User Block | A subset of metadata attributes stored in the NPOESS HDF5 File. The User Block can be thought of as a “header” on top of the HDF5 file stored as ASCII and is viewable without the need of the HDF5 API. |
| Metadata\* | Attributes that are attached to datasets and groups within the NPOESS HDF5 file which help identify and describe the data. All of the groups and datasets within the HDF5 file, with the exception of the All\_Data hierarchy and the Data\_Products Group, have a set of these attributes. |
| NPOESS Data Product Profile | An XML representation of Granule properties. Each Product Profile describes the contents and properties of a granule (e.g., parameter names, data types, data dimensions, measurement units, which dimension is the aggregation dimension). The NPOESS Data Product Profiles are rendered as tables in the CDFCB–X. NPOESS Data Product Profiles are produced for SDRs, TDRs, EDRs, IPs, and associated Geolocations. |
| NPOESS HDF5 File | An aggregation of one or more data product granules with associated metadata. The file organization is depicted with a UML diagram. The granules within a file are described by the Product Profile. The data within the granule is ordered and presented following the Style Guide. An NPOESS HDF5 file is usually one granule type, although multiple granule types are allowed (e.g., measurement and geolocation granules delivered together or multiple measurements sharing the same geolocation.) Using the HDF5 API, a user can retrieve granules either singly or together. The organization within the HDF5 file can be explained by using the example of a directory tree. Within the file there is a root directory with two sub-directories, these sub-directories are named “All\_Data” and “Data\_Products”. The All\_Data directory contains all of the data that was requested, and the Data\_Products directory contains sub-directories, which help to organize the data, references to allow extraction of the data, and metadata to identify and describe the data. |
| RDR\* | Raw data received from the spacecraft and packaged into HDF5 is referred to as a Raw Data Record (RDR). The data granules composing an RDR are the actual CCSDS application packets from the sensor, and don’t directly map into a set of data arrays. Granules that compose the RDR HDF5 files are aggregated application packets for a given sensor. |
| Style Guide | Section 3.5.4, Data Product Style Guide, constrains the possible choices for how data is stored within a granule: Grid, Swath, and/or Sparse Array. |
| UML Diagram (Class Diagram) | Provides a visual depiction of the NPOESS HDF5 file organization |