

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.

1 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.

2 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	

3 Structures and variables

Appendix 1: granule_t structure members

Appendix 2: Size definitions

4 Nagg tool software modules

“Command parser” module

“Get granules” module

“Select granules” module

“Write granules” module

4.1 “Command parser” module

Purpose:

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

Details to come.

4.2 “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.

4.3 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.

4.4 “Write granules” module

Purpose:

To create output files and write granules as directed.

Public Functions:

4.4.1 *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 output 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`.

4.4.2 *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 output 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 granule's 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	N_Granule_Version

		manual execution of a granule.	
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: YYYYMMDDHHMMSSsssss */
#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