

NPP Aggregation Tool Components

Albert Cheng
Larry Knox
Elena Pourmal

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 from existing files into new files with a different number of granules per file than in the original files. The tool may be used to create files with larger or smaller aggregations including deaggregation to one granule per file. Future versions will also package granules of compatible products into a single set of files or separate granules in previously packaged files into unpackaged files with granules of one product in each.

The tool facilitates creating aggregations and/or 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 aggregation sizes or different package combinations of compatible products. The tool creates copies of the existing data and updates metadata to reflect the new aggregation. When required it also creates fill granules with calculated timestamps and fill values for other metadata and for raw data, using existing granules as a pattern. For all operations the tool relies only on information available in the original files. It doesn't have access to information used to generate the files.

Nagg has been 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.

3 Nagg Example

How does the nagg tool work? This example uses a simple command to create new files each containing 3-granule aggregations of REDRO granules from all the files with names matching the pattern REDRO*.h5 in the current directory:

```
nagg -n 3 -t REDRO ./REDRO*.h5
```

Each REDRO*.h5 file has an attribute named /N_GEO_Ref whose value is the name of a geo-location file containing the corresponding geo-location granules. If these geo-location files are present, new geo-location files will also be created to match the new 3-granule REDRO files. If the files are not present the tool will fail.

The nagg tool performs these steps to create the new files:

1. Parse the command line flags, their values and file names.
2. Read data from the input files and the corresponding geo-location files to create a table of granule information.
3. Sort the granules by first Granule ID, then DPID, then by GranuleVersion as shown in Figure 1.
4. Select granules aggregations of size specified by the -n flag and identify the aligned boundaries between aggregations according to the Common Data Format Control Books. The beginning and ending files may be partial aggregations depending on the available granules and the particular boundaries for the aggregation size. The tool will create fill granules for any missing granules within the sets of available granules. Preceding and trailing fill granules are not written to the first and to the last file correspondingly.
5. Create files with filenames as specified by the Control Books for each aggregation and copy the existing data and write fill data for any fill granules to the files.

Figure 1: Example of granule table produced by the “Get granules” module and sorted by the “Select granule” module.

Granule ID	DPID	GranuleIndex	GranuleVersion	BeginningTime	EndingTime	More fields 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	
...	

4 Structures and variables

For each granule the tool gets metadata information needed to produce the output files and stores it in a structure shown in Appendix 1: “granule_t structure members”.

Appendix 2: “Size definitions for nagg’s variables” shows miscellaneous variables and their values; some values affect current capabilities of nagg; for example, the tool cannot process more than 500 granules (NAGG_Granules_selected_max) and produce more than 30 output file (NAGG_outputfiles_max).

5 Nagg tool software modules

This section describes the functions of the “Command parser”, “Get granules”, “Select granules” and “Write granules” modules.

5.1 “Command parser” module

Purpose:

To parse the command line options, validate the option values and set the option variables so that the tool may execute according to user request.

Public Functions:

5.1.1 *parse_options*

```
parse_options(int argc, char * const argv[])
```

Parameters:

argc IN: number of elements in argv
 argv IN: the list of command options argument

Return values:

0 if successful, call leave(EXIT_FAILURE) if it encounters irrecoverable errors such as illegal options or bad option values.

Description:

The `parse_options()` function uses the standard `getopt()` function to parse the command options. It will set up the values of the following global variables during its execution.

Option	Global variables	Description
-n	<code>ngranulesperfile</code>	The number of granules per product in each output file. Default is 1.
-t	<code>products_arg</code> <code>nproducts</code>	A link list of products to store in each output file Number of products specified in -t flag.
-d	<code>outDir</code>	Directory name in which output files are generated. Default is NULL (generate files in the current directory).
-O	<code>origin_arg</code>	Origin identifier of 4 characters. Default is "XXXX".
-D	<code>domain_arg</code>	Domain identifier of 3 characters. Default is "XXX".
-g	<code>geofiles_arg</code>	An enum variable representing different geolocation granules selection criterion of "no"(0), "yes" (1), and "strict"(2).
<input_files> ...		
	<code>inputfiles</code>	A link list of input files.
	<code>ninputfiles</code>	Number of elements in <i>inputfiles</i> .

5.2 "Get granules" module

Purpose:

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

Public Functions:

5.2.1 *nagg_get_granules*

```
nagg_get_granules(char **file_list, int number_of_files,
    char **products_list, int nproducts, geolocation_t geofiles_arg,
    char **geoproduct granule_p_t, *granule_info_p[], int *number_of_granules_p)
```

Parameters:

<code>file_list</code>	IN: list of files containing granules to be added to the granule table.
<code>number_of_files</code>	IN: number of file names in the list.
<code>products_list</code>	IN: list of product types for which granules will be written to a file.
<code>nproducts</code>	IN: number of products types in the list.
<code>geofiles_arg</code>	IN: enum value from -g command option (default <code>GEOFILE_YES</code>).
<code>geoproduct</code>	OUT: address of variable to return the DPID of the geolocation product.
<code>*granule_info_p[]</code>	INOUT: address of the granule table to be populated.
<code>*number_of_granules_p</code>	INOUT: 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.

5.3 Select granules module

Purpose:

To select granules from the given `granule_info` table that matches one of the products in the given products list or the geolocation product according to the given number of granules per file.

It returns a list of selected granules, including fill granules, to be written to the output file(s).

Public Functions:

5.3.1 *select_granules*

```
select_granules(granule_p_t granule_info[], int *_gindex, char **products_list,
int nproducts, int total_nproducts, char *geoproduct, granule_p_t
granules_selected[], int ngranulesperfile, int *_granules_remain, int
*_total_granules_file)
```

Parameters:

<code>granule_info</code>	IN: table of granules for selection.
<code>*_gindex</code>	INOUT: index of the next available granule in the <code>granule_info</code> for selection. It reaches the end of the table if <code>_granules_remain</code> is equal to 0.
<code>**products_list</code>	IN: the list of products to match.
<code>nproducts</code>	IN: number of elements in <code>products_list</code> .
<code>total_nproducts</code>	IN: number of products and the geolocation product if wanted.
<code>*geoproduct</code>	IN: geolocation product (NULL if not wanted.)
<code>granules_selected</code>	INOUT: a table of selected granules for output. It is expected that sufficient space has been allocated

```

        for granules_selected to store all granules
        selected.
ngranulesperfile  IN: number of granules of each product per output file.
*_granules_remain INOUT: number of granules in the granule_info table
        available for selection.
*_total_granules_file  OUT: number of granules in the granules_selected
        table.

```

Return values:

Returns SUCCEED (0) if success; FAIL (-1) otherwise.

If return values is FAIL, the values of the OUT or INOUT parameters are undefined.

Description:

The select_granules function selects granules that will fit in the output file according to bucket alignment boundary. The following is a description of the algorithms used.

Nagg algorithm in the calculation of bucket alignment:

Let N be the number of granules requested by the nagg user to reaggregate the NPP product files.

Let Tg be the duration of the first selected granule. (This value is different for different products and is defined in the products table.)

Then Tbucket = N*Tg seconds.

Let An be the n-th bucket since epoch.

Let Asn and Aen be the starting and ending time of An.

Let Gs be the beginning time of the first selected granule.

Then

$$An = \text{floor}(Gs/Tbucket)$$

$$Asn = An*(Tbucket)$$

$$Aen = As + Tbucket$$

How nagg adds fill granules to produced files:

First produced file

For the first file, if the starting time of the first selected granule is bigger than Asn, no fill granules are added before copying existing granules to the new file. This will produce a partial file.

Second to (n-1)-th files

N existing granules per product requested are copied to each of the new files, insert fill granules in place of any missing granules.

Last (n-th) file

Remaining granules per product requested are copied to the last file.

If the ending time of the last granule is less than the ending time of the last bucket, no fill granules are added. This will produce a partial file.

5.4 “Write granules” module

Purpose:

To create output files and write granules as directed.

Public Functions:

5.4.1 *start_write function*

```
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	IN: list of file names to be created for writing an output aggregation
noutfiles	IN: number of names in the outfiles list.
outgeofile	IN: name of the corresponding geo-location file, or null.
products_list	IN: list of DPIDs, one for each product. Only one product is supported for this version.
nproduct	IN: number of DPIDs in the products_list argument.
creationdate	IN: date of creation of the output files (for writing to the N_HDF_Creation_Date attribute)
creationtime	IN: time of creation of the output files (for writing to N_HDF_Creation_Time attribute).
ngranulesperfile	IN: 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. 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.

All of the granules selected for an aggregation will be written to the output files before any granules are selected for the next aggregation. The granules within an aggregation may be written in any order, and typically will be written one to each output file in rotation. The write granules module creates an array of `product_info_t` structures to keep track for each product of output filenames,

input and output file handles, number of granules written, and a pointer to the previously written granule.

```
typedef struct {
    const char dpid[DPID_size+1];
    hid_t infile;
    hid_t outfile;
    const char * outfilename;
    int last_i_granule;
    int granules_written;
    granule_p_t prev_granule;
} product_info_t;
```

A `product_info_t` structure is created and populated for each product and the geofile by the `start_write()` function. The `write_granules()` function will then select the `product_info_t` for each granule that matches its DPID. The `product_info_t` for the separate geolocation file is created last so that its index will always be `nproducts`.

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`. Values for these attributes are generated by `nagg` with the new geo file name and the current time.

5.4.2 *write_granules function*

`write_granules(granule_p_t granule, int i_granule)`

Parameters:

<code>granule</code>	IN: pointer to a <code>granule_t</code> structure containing information about a granule in an input file.
<code>i_granule</code>	IN: 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 `product_info_t` structure matching the granule's product ID (DPID) to find the correct output file.
- Opens the input file specified by `granule->file_in`.
- Initializes the output file when first called with a granule.
 - Copies root group attributes except those written by `start_write()` from the input file to the output file.
 - Creates group structure in the file, creating product groups in `/All_Data` and `/Data_Products`. Product groups in `/All_Data` are named `<productname>_All`; those in `/Data_Products` are named `<productname>`.

- Copies datasets from the /All_Data group in the input file to the /All_Data group in the output file; resizes the datasets for the new aggregation size.
- Copies attributes from the /Data_Products/<productname> group in the input file to the /Data_Products/<productname> group in the output file.
- Copies the /Data_Products/<productname>/<productname>_Gran_n 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 file's /Data_Products/<productname>/<productname>_Gran_n dataset
- Creates the /Data_Products/<productname>/<productname>_Aggr dataset with object references to all the datasets in /All_Data/<productname> 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 granule in the aggregation.
- Increments the value of the variable that keeps track of the number of granules written.

5.4.3 end_write function

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, error status will be returned and an error message displayed.

- Update AggregateEndingDate, AggregateEndingGranuleID, AggregateEndingOrbitNumber and AggregateEndingTime from the last granule in the aggregation.
- Update AggregateNumberGranules with the number of granules written.
- Close the file

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[]	<p>The unique identifier for each RDR granule composed of the concatenation of two components:</p> <p>(1) The three character satellite identifier</p> <p>[alias "Platform_Short_Name"],</p> <p>(2) A zero left filled, 12 character number, specifying the number of tenths of a second since the first ascending node after launch)</p>	<p>Attribute /<Data_Products /<product group> /<product_Gran_n dataset> /N_Granule_ID</p>
granule_version	char[]	<p>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</p>	<p>Attribute /<Data_Products /<product group> /<product_Gran_n dataset> /N_Granule_Version</p>

		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.	Attribute /<Data_Products /<product group> /<product _Gran_n dataset> /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.	Attribute /<Data_Products /<product group> /<product _Gran_n dataset> /N_Ending_Time_IET
beginning_date	char[]	Beginning date of the temporal range (observation date) for a granule.	Attribute /<Data_Products /<product group> /<product _Gran_n dataset> /Beginning_Date
beginning_time	char[]	Beginning time of the temporal range (observation time) for a granule.	Attribute /<Data_Products /<product group> /<product _Gran_n dataset> /Beginning_Time
ending_time	char[]	Ending date of the temporal range (observation date) for a granule.	Attribute /<Data_Products /<product group> /<product _Gran_n dataset> /Ending_Time
orbit_number	uint64_t	The number of the orbit at the start of the data collection for a data granule.	Attribute /<Data_Products /<product group> /<product _Gran_n dataset> /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 for nagg's variables

```

/*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 names */
#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: HHMMSSSS */
#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: Products and GEO Products Tables

Source: Common Data Format Control Book Vol I; Raytheon: INF_CFG.xml

```

/* NPP Products Table */
char *product_table[NPP_Product_max][3] =
{
/* DPID      Short Name      Approximate duration */
"ICALI",    "CrIMSS-CrIS-AVMP-LOS-IR-IP",    31997000,
"ICALM",    "CrIMSS-CrIS-AVMP-LOS-MW-IP",    31997000,
"ICCCR",    "CrIMSS-CrIS-CLOUD-CLEARED-RAD-IP",    31997000,
"ICISE",    "CrIMSS-CrIS-IR-SURF-EMISSIVITY-IP",    31997000,
"ICMSE",    "CrIMSS-CrIS-MW-SURF-EMISSIVITY-IP",    31997000,
"ICSTT",    "CrIMSS-CrIS-SKIN-TEMP-IP",    31997000,
"ICTLI",    "CrIMSS-CrIS-AVTP-LOS-IR-IP",    31997000,
"ICTLM",    "CrIMSS-CrIS-AVTP-LOS-MW-IP",    31997000,
"IICMO",    "VIIRS-CM-IP",    31997000,
"SATMR",    "ATMS-REMAP-SDR",    31997000,
"SATMS",    "ATMS-SDR",    31997000,
"SCRIS",    "CrIS-SDR",    31997000,
"SOMPS",    "OMPS-NP-SDR",    37405000,
"SOMTC",    "OMPS-TC-SDR",    37405000,
"SOMSC",    "OMPS-TC-Cal-SDR",    27000000000UL,
"SOMNC",    "OMPS-NP-Cal-SDR",    27000000000UL,
"SVDNB",    "VIIRS-DNB-SDR",    85350000,
"SVI01",    "VIIRS-I1-SDR",    85350000,
"SVI02",    "VIIRS-I2-SDR",    85350000,
"SVI03",    "VIIRS-I3-SDR",    85350000,
"SVI04",    "VIIRS-I4-SDR",    85350000,
"SVI05",    "VIIRS-I5-SDR",    85350000,
"SVM01",    "VIIRS-M1-SDR",    85350000,
"SVM02",    "VIIRS-M2-SDR",    85350000,
"SVM03",    "VIIRS-M3-SDR",    85350000,
"SVM04",    "VIIRS-M4-SDR",    85350000,
"SVM05",    "VIIRS-M5-SDR",    85350000,
"SVM06",    "VIIRS-M6-SDR",    85350000,
"SVM07",    "VIIRS-M7-SDR",    85350000,
"SVM08",    "VIIRS-M8-SDR",    85350000,
"SVM09",    "VIIRS-M9-SDR",    85350000,
"SVM10",    "VIIRS-M10-SDR",    85350000,
"SVM11",    "VIIRS-M11-SDR",    85350000,
"SVM12",    "VIIRS-M12-SDR",    85350000,
"SVM13",    "VIIRS-M13-SDR",    85350000,
"SVM14",    "VIIRS-M14-SDR",    85350000,
"SVM15",    "VIIRS-M15-SDR",    85350000,
"SVM16",    "VIIRS-M16-SDR",    85350000,
"TATMS",    "ATMS-TDR",    31997000,
"REDRO",    "CrIMSS-EDR",    31997000,
"OOTCO",    "OMPS-TC-EDR",    37405000,
"VAOOO",    "VIIRS-Aeros-EDR",    85350000,

```

"VCBHO",	"VIIRS-CBH-EDR",	85350000,
"VCCLO",	"VIIRS-CCL-EDR",	85350000,
"VCEPO",	"VIIRS-CEPS-EDR",	85350000,
"VCOTO",	"VIIRS-COT-EDR",	85350000,
"VCTHO",	"VIIRS-CTH-EDR",	85350000,
"VCTPO",	"VIIRS-CTP-EDR",	85350000,
"VCTTO",	"VIIRS-CTT-EDR",	85350000,
"VI1BO",	"VIIRS-I1-IMG-EDR",	85350000,
"VI2BO",	"VIIRS-I2-IMG-EDR",	85350000,
"VI3BO",	"VIIRS-I3-IMG-EDR",	85350000,
"VI4BO",	"VIIRS-I4-IMG-EDR",	85350000,
"VI5BO",	"VIIRS-I5-IMG-EDR",	85350000,
"VISTO",	"VIIRS-IST-EDR",	85350000,
"VLSTO",	"VIIRS-LST-EDR",	85350000,
"VM01O",	"VIIRS-M1ST-EDR",	85350000,
"VM02O",	"VIIRS-M2ND-EDR",	85350000,
"VM03O",	"VIIRS-M3RD-EDR",	85350000,
"VM04O",	"VIIRS-M4TH-EDR",	85350000,
"VM05O",	"VIIRS-M5TH-EDR",	85350000,
"VM06O",	"VIIRS-M6TH-EDR",	85350000,
"VNCCO",	"VIIRS-NCC-EDR",	85350000,
"VNHFO",	"VIIRS-NHF-EDR",	85350000,
"VOCCO",	"VIIRS-OCC-EDR",	85350000,
"VISAO",	"VIIRS-SA-EDR",	85350000,
"VSCDO",	"VIIRS-SCD-BINARY-SNOW-FRAC-EDR",	85350000,
"VSCMO",	"VIIRS-SCD-BINARY-SNOW-MAP-EDR",	85350000,
"VSICO",	"VIIRS-SIC-EDR",	85350000,
"VSSTO",	"VIIRS-SST-EDR",	85350000,
"VSTYO",	"VIIRS-ST-EDR",	85350000,
"VSUMO",	"VIIRS-SusMat-EDR",	85350000,
"VIVIO",	"VIIRS-VI-EDR",	85350000,
"REDRS",	"CrIMSS-EDR-SUB",	31997000,
"OOTCS",	"OMPS-TC-EDR-SUB",	37405000,
"VAOOS",	"VIIRS-Aeros-EDR-SUB",	85350000,
"VCBHS",	"VIIRS-CBH-EDR-SUB",	85350000,
"VCCLS",	"VIIRS-CCL-EDR-SUB",	85350000,
"VCEPS",	"VIIRS-CEPS-EDR-SUB",	85350000,
"VCOTS",	"VIIRS-COT-EDR-SUB",	85350000,
"VCTHS",	"VIIRS-CTH-EDR-SUB",	85350000,
"VCTPS",	"VIIRS-CTP-EDR-SUB",	85350000,
"VCTTS",	"VIIRS-CTT-EDR-SUB",	85350000,
"VISTS",	"VIIRS-IST-EDR-SUB",	85350000,
"VLSTS",	"VIIRS-LST-EDR-SUB",	85350000,
"VNCCS",	"VIIRS-NCC-EDR-SUB",	85350000,
"VNHFS",	"VIIRS-NHF-EDR-SUB",	85350000,
"VOCCS",	"VIIRS-OCC-EDR-SUB",	85350000,
"VISAS",	"VIIRS-SA-EDR-SUB",	85350000,
"VSCDS",	"VIIRS-SCD-BINARY-SNOW-FRAC-EDR-SUB",	85350000,
"VSCMS",	"VIIRS-SCD-BINARY-SNOW-MAP-EDR-SUB",	85350000,

```

"VSICS",    "VIIRS-SIC-EDR-SUB",    85350000,
"VSSTS",    "VIIRS-SST-EDR-SUB",    85350000,
"VSTPS",    "VIIRS-ST-EDR-SUB",    85350000,
"VSUMS",    "VIIRS-SusMat-EDR-SUB",  85350000,
"VIVIS",    "VIIRS-VI-EDR-SUB",    85350000,    };

```

```

/* NPP Geolocation Table

```

```

* The source is NPOESS Common Data Format Control Book Volume I, pp 328-9,

```

```

* Table A-8, Geolocation Identifiers.

```

```

*/

```

```

char *geolocation_table[NPP_Geo_Location_max][4] =

```

```

{
"GATMO",    "ATMS-SDR-GEO",    31997000,
"GCRSO",    "CrIS-SDR-GEO",    31997000,
"GAERO",    "VIIRS-Aeros-EDR-GEO", 85350000,
"GCLDO",    "VIIRS-CLD-AGG-GEO", 85350000,
"GDNBO",    "VIIRS-DNB-GEO",    85350000,
"GNCCO",    "VIIRS-NCC-EDR-GEO", 85350000,
"GIGTO",    "VIIRS-IMG-GTM-EDR-GEO", 85350000,
"GIMGO",    "VIIRS-IMG-GEO",    85350000,
"GITCO",    "VIIRS-IMG-GEO-TC", 85350000,
"GMGTO",    "VIIRS-MOD-GTM-EDR-GEO", 85350000,
"GMODO",    "VIIRS-MOD-GEO",    85350000,
"GMTCO",    "VIIRS-MOD-GEO-TC", 85350000,
"GNHFO",    "VIIRS-NHF-EDR-GEO", 85350000,
"GOTCO",    "OMPS-TC-GEO",    37405000,
"GOSCO",    "OMPS-TC-Cal-GEO", 27000000000UL,
"GONPO",    "OMPS-NP-GEO",    37405000,
"GONCO",    "OMPS-NP-Cal-GEO", 27000000000UL,
"GCRIO",    "CrIMSS-EDR-GEO-TC", 31997000,
"GATRO",    "ATMS-REMAP-SDR-GEO", 31997000,
};

```

Appendix 4: 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